Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

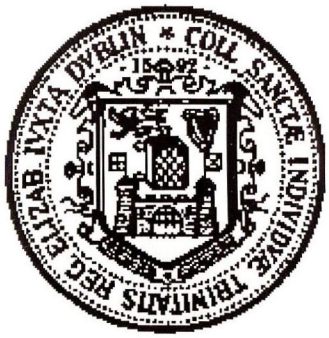
Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Algorithmic Level Low-Power VLSI Design Applied to RGB to HSI Conversion

Andreas Schwarzbacher



Thesis submitted for the degree of Ph.D.

November 2001

Department of Electronic and Electrical Engineering
University of Dublin
Ireland

I, Andreas Schwarzbacher, declare that this thesis is entirely my own work, except where otherwise accredited, and that it has not been submitted for a Degree to any other University or Institution.

I further agree that the Library may lend or copy this thesis upon request:

Signed:.....

Date:05/11/01.....

Acknowledgements

Firstly, I wish to express my gratitude to my supervisor, Dr. Brian Foley, for his support and help throughout this thesis. His advice and guidance was invaluable on many occasions.

I would also like to thank all the staff and Postgrads of the Electronic Engineering Department who have helped me on the way. Special thanks go to Paul Comiskey and Joe Timoney for both technical and spiritual advice all throughout the years.

Furthermore, I want to acknowledge the support of my friends both inside and outside the department Elena Ranguelova, Devendra Kumar, Cecilia Chan and Lorcan Mac Manus.

Additionally I would like to thank the School of Electronics and Telecommunications Engineering at Dublin Institute of Technology for granting me access to their research facilities, especially Chris Cowely and Chris Bruce.

Finally, I wish to thank my parents for the support they have given me over all the years.

Furthermore, I would like to acknowledge the funding received through the IRCARUS2 (DG-XII) initiative for the development of PowerCount.

Abstract

The growing demand for portable applications such as cellular phones, portable digital assistants (PDAs) and notebooks has resulted in a requirement for integrated circuits (ICs) which consume less power while delivering the same performance as non-portable appliances. In addition, the low-power implementation of non-portable circuits has several advantages, notably a marketing advantage in terms of energy efficiency and reduced manufacturing costs because of cheaper packaging.

The focus of this thesis is the application of high-level low-power VLSI design methods to a hardware implementation of Kender's algorithm which converts a camera signal of red, green and blue into a human perception-based code of hue, saturation and intensity. The aim was to consider the circuit implementation of the algorithm on a block-by-block basis in order to identify in each block potential avenues along which power savings can be made, and to produce a power-efficient high-level circuit design targeted to an Application Specific Integrated Circuit (ASIC). The most commonly used approach for power reduction in VLSI circuits is to minimise the supply voltage. However, with ASICs voltage scaling is only applicable within a very limited range. Therefore, this thesis concentrates on the minimisation of the power consumption by reducing the active capacitance of the circuit. This required a high-level power estimation tool capable of assessing the power consumption at the earliest possible design stage, and therefore led to the development a tool that can rapidly measure the active capacitance of a design from a VHDL netlist.

Following the completion of the high-level low-power design, simulation of the implementation was done using a range of real image data. The results from these tests demonstrated that the design introduced no degradation in perceptual quality. Furthermore, the final design showed significant power reduction when compared to a direct ASIC implementation or to a version programmed on a Digital Signal Processing Integrated Circuit (DSP IC).

Table of Contents

List of Figures

List of Tables

1 Introduction

The development of electronics has been fuelled by the invention of the transistor by Bardeen and Shockley over 50 years ago [Rior97]. The initial technical hurdles faced by these pioneers included the development of suitable semiconductor materials and the fabrication of reliable devices. These challenges were quickly solved and were superseded by the demand for circuit integration. This lead to the introduction of the first commercial integrated circuit (IC) in the early 1960s. In 1965 G. E. Moore observed that most integrated circuits had approximately doubled in complexity each year since 1959 [Moor65]. Consequently he formulated Moore's Law, which predicts a doubling of the complexity of ICs every 18 months. The validity of this observation has been demonstrated by the constant increase in integration density over the last three decades. This has lead to the presence of ICs in virtually all consumer appliances. At the same time, computers capable of performing millions of operations per second are already in most households. These computers are based on ICs which themselves contain millions of transistors and operate at frequencies of up to 1GHz. The growth in usage of computing systems has resulted in a demand for portable systems with comparable performance to their non-portable counterparts. However, the inclusion of portability in high performance systems has presented a new challenge. The battery operation time is limited by the power consumption of the system. Also in non-portable systems, reliability and fabrication costs are adversely affected by the power consumption. Therefore, power consumption has now become the third design challenge, in addition to speed and integration complexity. Much research effort continues to be expended in the development of techniques aimed at reducing power consumption at all levels of circuit design. This thesis addresses the challenges of low power design by investigating the implementation of an image processing system. For this purpose, Kender's algorithm for the faster computation of hue [Kender] was chosen as the vehicle to demonstrate the applicability of high-level low power design methods.

Humans process a variety of information, which can be divided into general categories such as speech, images and other data. To provide a human interface, electronic systems have to process data in forms which reflect the human sensory systems of sight and sound. Audio data has a low information content and can therefore be easily processed with current

technology. Video however, has a much higher information content. This places much higher requirements on image processing systems. The most obvious one is the high computational throughput data required. However, when such systems are implemented in integrated form, the area and power consumption become limiting factors in the performance of the system. With the high integration density of current semiconductor technology the area constraint is of lesser importance. The combined effect of high throughput and increased system integration can cause excessive levels of power density resulting in reduced reliability, overheating and premature system failure. To address these issues, the designer is faced with the task of including the power consumption of such systems as a primary design objective. To achieve this, the design community must introduce new methods aimed at reducing the power consumption. This is also the objective of this thesis. To make this work generally applicable, the more computationally demanding problem of video processing was selected as a suitable vehicle for the investigation into low power design. In this way, the results obtained in this thesis may be directly transferred to other, less computationally intensive applications. The model used to study these power characteristics is introduced in the following section.

Video and image signals are usually recorded using the three primary colours of light, red, green and blue (R, G, B). Signal representation in red, green and blue is useful for image recording and image visualisation using a monitor as this is the natural format of all images. However, the RGB representation of images has disadvantages when the image signal requires manual modification. This is because human perception of images is not based on the three primary colours, but rather on the physical perception of three different quantities. These quantities are hue, saturation and intensity. Hue is the pure spectral colour of a pixel (a pixel is the smallest segment of a picture). The saturation is the purity of a colour and indicates how much 'white' a colour contains. The intensity of a colour simply describes its brightness.

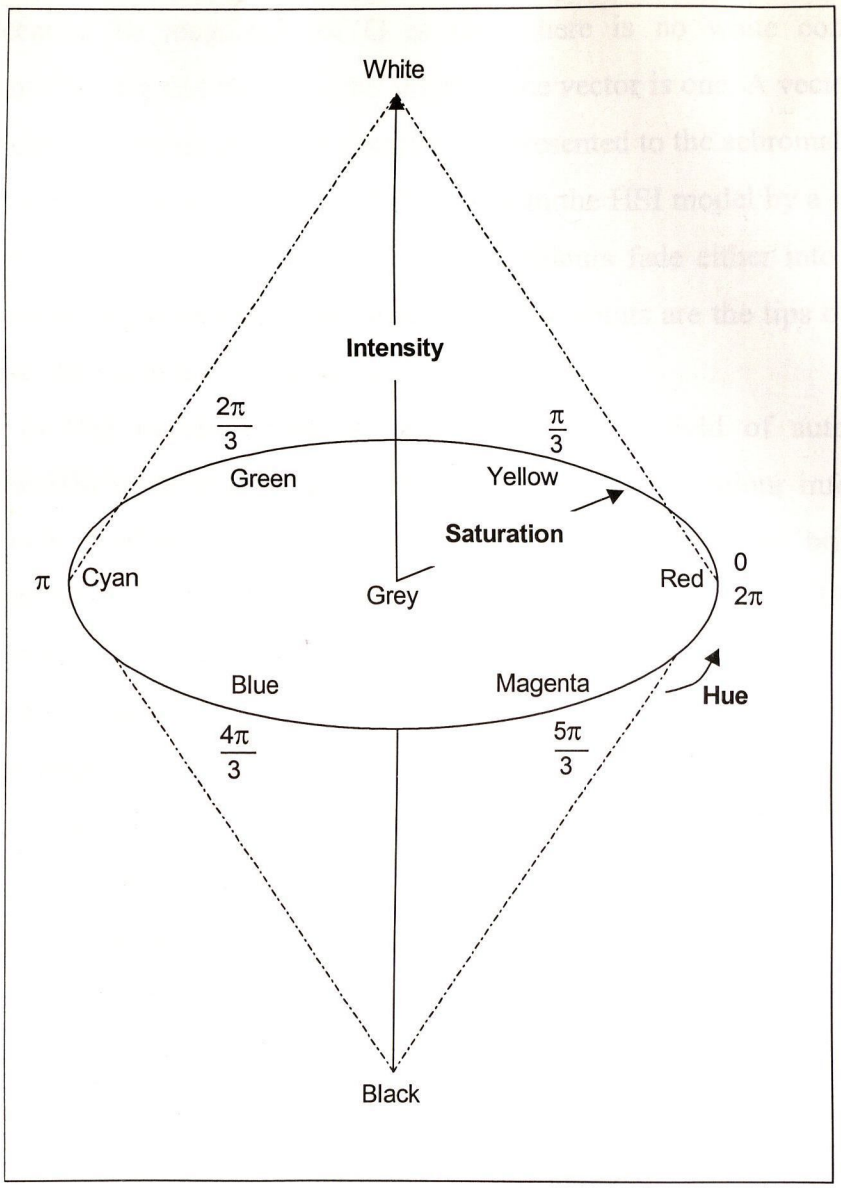


Figure 1.1: The Hue, Saturation, Intensity Model of Human Perception of Colour

Figure 1.1 shows the geometric representation of this hue, saturation and intensity (HSI) model. In this model, hue is represented by an angle on the outer circle. The three primary colours are distributed evenly over 360°. Red is assigned an angle of 0°, while green has an angle of 120° and blue has an angle of 240°. All other colours are found between these points. The achromatic axis, which is also called the grey scale, is found in the centre of the hue circle. This is because all shades of grey are an even mixture of all three primary colours.

The saturation of a colour is defined as the purity of a colour and therefore shows the amount of white contained in that pure spectral colour. To represent hue and saturation, a vector pointing to the pure spectral colour is used, where the magnitude of this vector indicates the saturation. A vector pointing to the perimeter of the circle represents a pure spectral colour. For example, an input signal of R=B and G=0 will result in a pointer to 315°

(Magenta). Because the magnitude of G is zero, there is no white contained in the representation of magenta and therefore the length of the vector is one. A vector with a small magnitude indicates the closeness of the pixel to be represented to the achromatic scale.

The intensity perception of a colour is represented in the HSI model by a second pointer, orthogonal to the hue–saturation vector. Because all colours fade either into pure white or black when the intensity is increased, or decreased, these points are the tips of two triangles with a base equal the magnitude of the saturation.

The RGB to HSI transformation is widely used in the field of automatic pattern recognition. The HSI representation allows the extraction of pure colour information from images taken under various lighting conditions as it is independent of the 'brightness' of an object. Examples of this include the computer assisted detection of cancer [Hama00] or the automatic detection of facial features, such as automatic lip segmentation [Liev99] [Sobo98]. The same principle can be applied to quality control of fruits, where the HSI space is investigated for areas which should not be present in a ideal fruit such as dents or insect stings on the surface of vegetables [Calv00]. Furthermore, after removing the lighting information it is possible to detect hand signals given by a human supervisor by automatically detecting skin areas using hue. Examples for such systems are described in [Wu99] and [Beck98]. Also, in the generation of colour representations for maize crop analysis it is possible to examine the images of maize in more detail using the HSI model than RGB [Ahma96]. The transformation into HSI is also used in the extraction of depth information in stereo colour images. Here, it is possible to extract regions of low and high intensity or low saturation which can then be treated like achromatic regions by the block matching algorithms [Kosc96]. Other applications for the HSI model are in the automatic restoration of paintings, separating cracks in the painting from brush strokes by using their different hue and saturation regions [Giak98], and astronomical image enhancement [Cava99].

The preceding section described the physical phenomenon of light and its human perception. As the RGB to HSI image processing algorithm is be implemented in hardware, a digitised version of the three primary colours is used. Therefore, in the remainder of this thesis the parameters R, G, B are used specifically to indicate digitised inputs of red, green and blue. Furthermore, for the implementation of the RGB to HSI algorithm in hardware, the following specifications have been selected. Firstly, the quantisation of each input signal is eight bits. This results in 24 bit RGB images, which is the standard for high quality image

reproduction. It is used in image recording formats such as BMP, PPM and Sun Raster Format. However, as HSI representation of images plays an important role in special effect creation for film, it was decided to implement both the conversion of images and the real-time conversion of high resolution video into HSI space. There are many resolution standards in today's computer applications, and the 1024 by 1024 pixels resolution was chosen as it is more than three times higher than that of VHS video. At this resolution, the proposed IC must be capable of converting full motion video in real-time. Full motion video is commonly defined as 25 frames per second. To calculate the processing time for each pixel of a frame, the number of pixels in a frame has to be multiplied by the number of frames per second. This results in (25 * 1024 * 1024 =) 26214400 pixels per second or 26.22Mpixels per second. To ensure that this constraint is met, the throughput rate was set to 33Mpixels per second. The input and output resolutions are identical. Therefore, the quantisation of eight bits for R, G and B is transformed into eight bits for H, S and I respectively. The following sections introduce the algorithms used to convert the RGB input into hue, saturation and intensity.

1.1 Kender's Algorithm for Faster Computation of Hue

In order to implement the RGB to HSI transformation, Kender's algorithm for faster computation of hue was chosen [Kender]. Kender's algorithm is shown in Equation 1 below. For easier reference, the different cases of Kender's algorithm are numbered separately.

if ((R > B) and (G > B)) (1.1)

$$hue = \frac{\pi}{3} + \arctan\left(\frac{\sqrt{3} \times (G - R)}{G - B + R - B}\right)$$

else if (G > R) (1.2)

$$hue = \pi + \arctan\left(\frac{\sqrt{3} \times (B - G)}{B - R + G - R}\right)$$

else if (B > G) (1.3)

$$hue = \frac{5 \times \pi}{3} + \arctan\left(\frac{\sqrt{3} \times (R - B)}{R - G + B - G}\right)$$

else if (R > B) (1.4)

hue = 0

else (1.5)

'achromatic'

The first three parts of Kender's algorithm (1.1), (1.2) and (1.3) define the angle of the hue vector of a pixel pointing to the spectral colour. First, the algorithm determines which of the primary colours in a given pixel is the one with the lowest intensity. The amount of white contained in a colour is equal to the smallest input value. This information is not required to calculate the colour angle, because the white information is contained in the saturation of the colour. Now, the area in which the resulting vector will be located is determined. The remaining two colours are then used to compute the exact angle of the colour. On the hue circle, the colour red is defined twice. Firstly, as 0° and than secondly as 360°. In order to account for the second red point the condition (R>B) must be included (1.4). If the value of hue is 360° the resulting vector is set to 0° in order to ensure that each point on the hue circle is only defined once. The last part of the algorithm (1.5) is used to describe all points which are not represented by a colour. This is true for achromatic values. An achromatic pixel is defined as a point, where all three primary colours have the same strength. Therefore this point is not included on the perimeter of the circle but as origin of the hue circle.

1.2 Computation of the Saturation

From (2) the magnitude of the vector pointing to the hue value is then determined leading to a value for saturation.

$$saturation = 1 - \frac{3 \times \min(R,G,B)}{R+G+B} \tag{2}$$

The aim of this calculation is to filter out the white content of the input triplet. As previously described, the smallest magnitude of the primary colours determines the amount of white contained in a given pixel. In order to normalise the hue circle, three times this smallest value is divided by the sum of the input vectors and the result is subtracted from one. This implies that if all three colour inputs have the same input value the length of the pointer is zero, i.e. a

pure achromatic value is applied. If at least one input carries an input value of zero, the saturation is one and therefore a pure spectral colour is applied to the input.

Equation (2) has a singularity at R = G = B = 0. This occurs when the input pixel is black. At this point the saturation is not defined. This singularity can cause problems during implementation. However, as will be shown in this thesis, this singularity is not a disadvantage in terms of low-power design and can actually be used to save additional power.

1.3 Computation of the Intensity

The intensity of a colour is defined as the average value of the three input signals.

$$intensity = \frac{R+G+B}{3} \tag{3}$$

The intensity is generally perceived as the brightness of a pixel. The stronger the average of the input signals the more intense is the perception of the resulting colour. Therefore, the intensity is calculated by summing up the three inputs and dividing it by three.

1.4 Summary

In order to perform the RGB to HSI conversion using Kender's algorithm for faster computation of hue, a number of important mathematical operations are required. These operations include addition, subtraction, multiplication and division with and without fixed constants as well as trigonometric calculations. All of these operations have to be implemented as a design description. This algorithm is therefore an ideal vehicle to show a variety of low-power implementation techniques. Additionally, the simplicity of this algorithm implies an initial direct-form implementation. However, as will be shown in the remainder of this thesis, there are ample opportunities for the implementation of different low-power strategies within this simple algorithm.

1.5 Thesis Overview

The remainder of this thesis is concerned with development of a low-power image processing algorithm. Firstly, a general overview of the different sources of power consumption is provided in Chapter 2. This overview indicates the relative importance of each power

dissipation source. Chapter 3 focuses on the reduction of the dynamic component of power consumption. Several power reduction techniques are evaluated. The utilisation of these techniques requires a high-level power estimation tool, previously unavailable. Therefore, Chapter 4 describes the development of PowerCount, a high-level power estimation tool. A low-power implementation of Kender's algorithm is discussed in Chapter 5. Chapter 6 then presents the implementation of the saturation and hue components of the HSI algorithm, while the features of the RGB to HSI design are evaluated in Chapter 7.

2 Power Dissipation

In the last decade, the reduction of power consumption in IC's has become a primary design goal. This chapter first presents a general overview of the sources of power consumption in IC's and then the effects of supply voltage reduction on power consumption and timing are explored. The results of this exploration are then used to present design techniques for the minimisation of overall power consumption.

2.1 General Equation

Power dissipation in a CMOS circuit is caused by three sources. The dynamic power consumption, the power consumption caused by short-circuit currents and the power consumption due to leakage currents. The equation to calculate the overall power consumption is:

$$P_{total} = P_{dynamic} + P_{short\text{-}circuit} + P_{leakage} \tag{4}$$

In this equation $P_{dynamic}$ represents the switching component of the total power consumption. This occurs each time a power consuming transition is performed. The $P_{short\text{-}circuit}$ term represents the short circuit path which arises when both NMOS and PMOS transistors are switched on and a path is connected directly between supply and ground. The $P_{leakage}$ losses are due to substrate injections and subthreshold effects [Chan92]. To take a closer look at the three components, equation (4) will be investigated more closely.

2.1.1 Dynamic Power Consumption

There are two different methods available to quantify the dynamic power dissipation. The first measures the power transformed into heat. The second represents the power taken out of the supply. In this work, following standard practice the second method was chosen [Chan92] [Chan95b]. The switching or dynamic power consumption occurs because a capacitive load is charged by the supply voltage. The ideal CMOS device only uses switching energy when changing the output value from LOW to HIGH. This is illustrated with the most simple element found in digital design, the inverter.

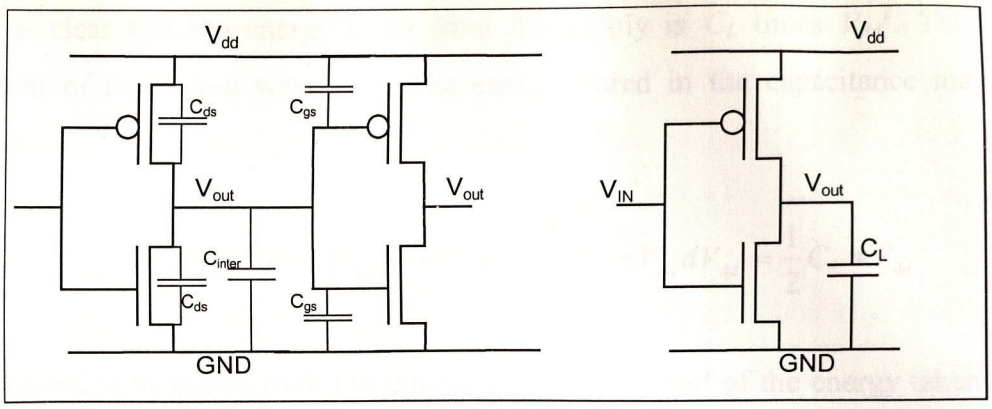


Figure 2.1: The Sources of the Node Capacitance

As seen in Figure 2.1 the total capacitive load C_L is the sum of the output or drain-substrate capacitances C_{ds}, the input or gate-source capacitance C_{gs} and the capacitance of the interconnection C_{inter}. The term node capacitance C_{node} is used synonymously with the load capacitance C_L. To compute the power consumed during an average switching event, the energy taken out of the supply over a particular time must be computed. Only for a LOW to HIGH (or 0 to V_{dd}) transition is a current drawn from the supply. The instantaneous power demand is given.

$$P(t) = \frac{dE}{dt} = i_{supply}(t) \bullet V_{dd} \tag{5}$$

In (5), $i_{supply}(t)$ is the current taken from the power supply with a constant voltage level of the supply voltage V_{dd}. This current can also be expressed as a function of the capacitance which is charged by assuming all $i_{supply}(t)$ is used to charge C_L.

$$i_{supply}(t) = C_L \frac{dV_{dd}}{dt} \tag{6}$$

The energy taken out of the supply can therefore be written as:

$$E_{(0,1)} = \int_0^T P(t)dt = \int_0^T V_{dd} \bullet i_{supply}(t)dt \tag{7}$$

If equation (6) is substituted into equation (5), the energy taken from the power supply can then be expressed as:

$$E_{(0,1)} = V_{dd} \int_0^{V_{dd}} C_L dV_{dd} = C_L \bullet V_{dd}^2 \tag{8}$$

Thus, it is clear that the energy taken from the supply is C_L times V_{dd}^2. This result is independent of the output waveform. The energy stored in the capacitance may also be calculated.

$$E_{cap} = \int_0^T P_{cap}(t)dt = \int_0^T i_{cap}(t) \bullet V_{dd} dt = \int_0^{Vdd} C_L \bullet V_{dd} dV_{dd} = \frac{1}{2} C_L \bullet V_{dd} \tag{9}$$

The conclusion to be drawn from this equation is that only half of the energy taken from the supply is stored by the capacitance. Therefore, the other half is dissipated as heat by the drain source resistance of the PMOS transistor. On each HIGH to LOW change at the input the output changes from LOW to HIGH. The capacitance of the node is then charged to the value of the supply voltage (it is assumed that the swing voltage is equal to the supply voltage). If the output changes from HIGH to LOW the energy stored in the capacitive load is dissipated in the NMOS transistor. Therefore, no power is consumed during this transition.

Equation (8) can be used to derive the general equation for the dynamic power consumption in CMOS VLSI systems. This is done by multiplying the energy consumed in a system by the clock frequency f_{clk}

$$P_{dynamic} = C_{active} V_{dd}^2 f_{clk} \tag{10}$$

In this equation C_{active} is represented by

$$C_{active} = \sum_{k=1}^{m} n_{k(0,1)} C_{node\,k} \tag{11}$$

and is the sum of the active node capacitance of a system with m nodes. The active node capacitance is the physical node capacitance C_{node} multiplied by the number of node switches from LOW to HIGH per cycle $n_{(0,1)}$.

The energy per transition is often used as a quantifier when comparing the power consumption of different implementations of the same design. This has the advantage that the operating frequency is removed from the equation which makes comparison of blocks easier to perform.

$$\text{Energy per transition} = \frac{P_{dynamic}}{f_{clk}} = C_{active} V_{dd}^2 \tag{12}$$

Because of the physical connection of C_L (and C_{active}) to the ground rail no energy is stored or transmitted. All energy is transformed into heat. A different approach is presented in [Athas94] where adiabatic-switching is used to recycle the switching energy during the HIGH to LOW transition. However, such circuits require special components from the power supply to individual single gate.

Often software tools are unable to determine the active capacitance as presented in (11). These tools evaluate the average dynamic power consumption by means of the average node capacitance $C_{average}$ [Nema99]. $C_{average}$ is the total physical capacitance of a design divided by the number of nodes. If the average capacitance is used to determine the dynamic power consumption the active capacitance is called total capacitance C_{total}. Equation (13) is used to determine the total capacitance.

$$C_{total} = \sum_{k=1}^{m} n_k C_{average} \tag{13}$$

In (13), the activity factor, n_k, is the switching probability at the node for random input data. The number of nodes is represented by m. C_{total} is usually estimated within a large margin of error because it assumes that the physical capacitance of each node is constant within the design. The main reason why this value is used is due to the ease of computation and the fact that even the power consumption of large circuits can be estimated within a short time [Nema99].

2.1.2 Short-Circuit Power Consumption

The second term of (4) represents the power consumption caused by a short-circuit current i_{SC}, which flows when both transistors are switched on. Equation (14) describes the short-circuit power dissipation.

$$P_{short-circuit} = i_{SC} V_{dd} \tag{14}$$

Ideally CMOS devices would have an infinitely small rise and fall time between the HIGH and LOW values. Therefore, they should not dissipate any short-circuit power. Real devices need time to charge and discharge the load capacitance. During the time the value of the input voltage lies between the upper threshold voltage (V_{dd}-V_{TP}) and lower threshold voltage (V_{TN}), both transistors are switched on. A path between the supply and ground is formed and a

current flows directly from the supply to the ground. This behaviour is illustrated in Figure 2.2. Therefore, the longer the rise and fall times of the input signal, the larger the short-circuit current. Additionally, it should be noted that during this time the capacitive load is neither significantly charged nor discharged, which leads to an increase in the normal delay for charging and discharging of the node capacitance.

When the input and output signals have equal rise and fall times the short-circuit power consumption is typically between 1% to 2% of the total power consumption [Blair94]. In order to achieve equal input and output rise and fall times, the RC product of the input and output node capacitance must be equal. Achieving this can lead to problems if large loads such as peripheral equipment or variable loads are to be driven. To compensate for that a string of inverters can be used to optimise these delays, as demonstrated by Veendrick [Veen84]. Unfortunately the balancing of the load has to be done manually. Therefore, it is not possible to optimise the loads of all nodes of VLSI circuits. In [Nose00] Nose and Sakurai show that typically around 10% of the total power consumption can be attributed to short-circuit power. Furthermore, Nose and Sakurai argue that this figure will not increase in future, smaller technologies if the ration of the threshold voltage over the supply voltage is kept constant.

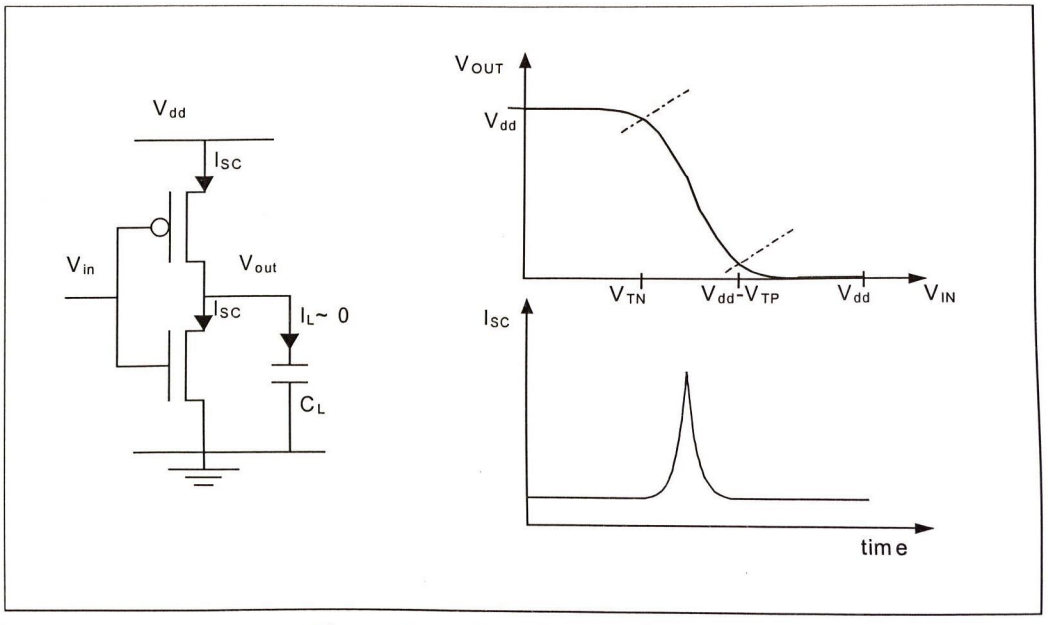


Figure 2.2: Short Circuit Current

2.1.3 Leakage Power Consumption

The computation of the last term of (4), the leakage power, is shown in (15).

$$P_{leakage} = I_{leakage} V_{dd} \tag{15}$$

$I_{leakage}$ represents the current caused by reverse bias currents through parasitic diode structures. The value $I_{leakage}$ is fixed for a given technology and is directly proportional to the area of the circuit. For a 1.2µm technology, a value between 1pA and 5pA per μm^2 is typical [Burd96]. The designer cannot influence this value directly. However, the leakage current is highly temperature sensitive. For this reason it may be possible to limit the leakage current thus optimising the dynamic power dissipation by minimising heat dissipation.

2.1.4 The Complete Equation for the Total Power Consumption

Finally, the complete equation for the total power consumption is given by

$$P_{total} = C_{active} V_{dd}^2 f + i_{SC} V_{dd} + I_{lakage} V_{dd} \tag{16}$$

As shown in the previous sections it is not possible to influence the short-circuit power consumption and consequently the leakage power dissipation at the higher level of the design cycle other than by choosing the proper technology. Therefore the following chapters will focus on mitigating the dynamic power consumption. This is a reasonable approach since the dynamic power consumption is typically more than 90% of the total power consumption [Veen84] [Nose00].

2.2 Reducing the Supply Voltage

The most obvious and most common way to reduce the power consumption is to reduce the supply voltage. The supply voltage is present in all terms of the total power consumption equation (4). In the switching power expression the V_{dd} term is squared if the swing voltage is set to the supply voltage. Therefore, any reduction of V_{dd} would cause a major reduction in the total power consumption. The most important factor in reducing the power consumption of a circuit to a minimum is to reduce the supply voltage as much as possible. This conclusion is supported in most of the referenced literature such as [Chan92], [Liu93], [Malh94], [Wang98], [Sanc99] or [Shiu00].

2.2.1 Delay and Voltage

Figure 2.3 (adapted from [Chan95b]) shows that when the supply voltage is reduced, the normalised delay increases. In IC design a distinction is made between local delay, which is the delay of signal propagation between the transistors on the same chip and the global delay which is the delay between transistors from one chip to another chip. The global delay is only important when the chip specification includes a timing relationship between the chip and other components in the system. In this thesis the term delay is used for the local delay.

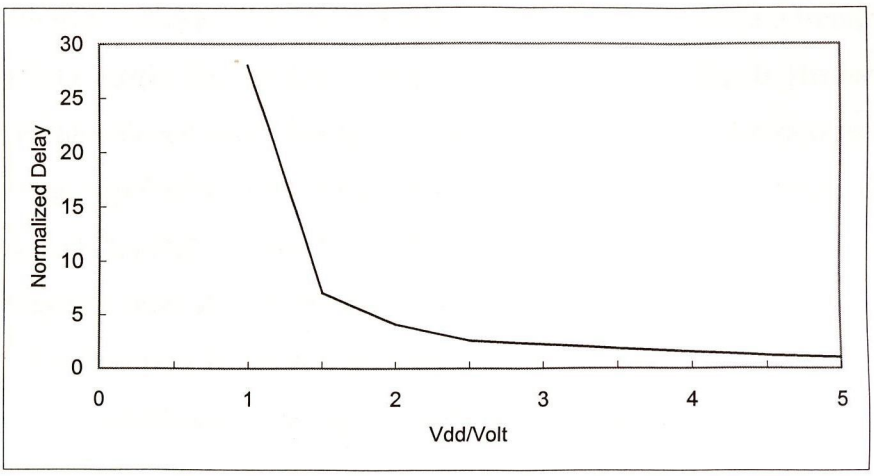


Figure 2.3: Dependence of Delay and Voltage

For most applications a minimal throughput of data is specified and this therefore defines a minimum limit for the supply voltage. The simple first order equation for delay is shown in (17) [Chan92].

$$T_d = \frac{C_L V_{dd}}{I} \tag{17}$$

This equation is suitable for technologies above 1.0μm. For smaller sizes, the saturation of the carrier velocity under higher electric fields, which is not represented in the equation above, becomes significant. The delay becomes increasingly voltage independent as the implementation technology becomes smaller. Therefore, little advantage can be gained by simply reducing the voltage. For a 0.3μm process the critical voltage was found to be 2.43V, at which no further power reductions can be achieved by reducing the supply voltage [Chan92]. If the voltage is lower than this limit the incremental delay is determined and limited by interconnection delays. The equation above may differ from measured values of delay by a factor of up to 20 when channel lengths are below 0.5μm [Liu93]. Unfortunately

the second order equation required to determine delay for these small processes becomes more complex than (17), because now the active length and width of the gates and also other dimensional factors have to be included. For a detailed overview refer to [Liu93].

For cells above 1.0μm a reduction in power by 60% can be achieved with a supply voltage of 3.3V compared to the 5V standard. It is the reason why this supply voltage became a new industry standard. But this new standard also raises problems. Firstly, the I/O interfaces will still have to operate with the 5V standard, while the thinner gate oxide of the chips designed for 3.3V can be damaged at 5V. To avoid this problem, interface chips are designed which can operate with two voltages. The 3.3V supply is used for the functional circuitry and the 5V supply for the I/O ports. This makes design simulation very difficult [Brust93]. Another difficulty is that the reduced speed has to be compensated using architectural techniques such as parallelism and pipelining. This means that these new designs are more complex than traditional designs [Chan92] [Blair94] [Liu93]. Further reductions may be achieved by setting the supply voltage to around 1.5V which is the optimal voltage for most common uses as presented by Chandrakasan [Chan94] and Bellaouar *et al.* [Bell98]. It also seems that this voltage may set a standard supply voltage for RISC chips [Naka94].

2.2.2 Threshold

The threshold voltages (V_T) present a different problem due to the reduction of the supply voltage. For a given process the threshold voltage is defined and therefore sets another limit on supply voltage reduction. To provide a sufficient margin so that an input signal is accepted as HIGH or LOW, there must be a difference between supply voltage and the upper threshold voltage. The exact value is disputed in the literature. In [Blair94] half the supply voltage is suggested as a good rule of thumb while in [Stork95] and [Chan94b], the square root of the supply voltage is offered as a good value for the threshold voltage. The lower limit for the threshold voltage itself should be in the area of 0.3V to provide a good compromise between switching power and leakage power as shown in [Chan94b].

A different problem caused by the subthreshold region is the current which flows when the input signal is in this region. The signal forces both transistors to open which leads to the short-circuit current as presented in Section 2.1.2. To avoid this current, the rise time of the input and output signals should be equal and as fast as possible. Also the region between the

threshold voltages should be as small as possible to decrease the time when both transistors are on.

To optimise the power delay product a supply voltage of three times the threshold voltage was found to give reasonable results [Shim93]. This allows a good noise margin of V_T for both HIGH and LOW levels. Further reduced levels may increase the short-circuit current and therefore the short-circuit power consumption due to unstable states.

An interesting fact is that it is possible to reduce the supply voltage under the sum of the threshold voltages of the NMOS and PMOS transistors. Chandrakasan [Chan94b] presented a low-power chipset that works at 1.1V, where the threshold voltage for the NMOS device is 0.7V and that of the PMOS device is -0.9V. While using such a low supply voltage it is assumed that only one transistor at a time conducts and therefore it is impossible for short-circuit currents to occur. However, this has the drawback of increased propagation delay.

2.2.3 Voltage Scaling

Most systems are designed for maximum throughput of data or for peak performance which occurs infrequently in normal usage [Good98] [Danc00]. Therefore, most computations are finished before the time deadline set by the clock frequency and the internal propagation delay. In such applications the propagation delay can be slowed down by adjusting the supply voltage in such a way that the output data meets the timing restrictions. This is called supply voltage scaling or just-in-time processing, and can effectively reduce the power consumption. After each cycle the critical path is scanned and a voltage is computed at which the operation still meets the time constraints [Good98] [Wei99] [Danc00]. For example if a computation is allowed to take 20ns but needs only 10ns for the actual data to be processed, the voltage can be reduced from 5V down to 2.9V, the value of supply at which the delay doubles (Figure 2.4). Voltage scaling ensures that the circuit is always running at the lowest possible supply voltage. Equation (18) expresses the power saving with this approach for the actual voltage, V_{actual}, in relation to the maximal supply voltage, V_{dd}.

$$n = \frac{P(V_{actual})}{P(V_{dd})} = \left(\frac{V_{actual}}{V_{dd}}\right)^2 \tag{18}$$

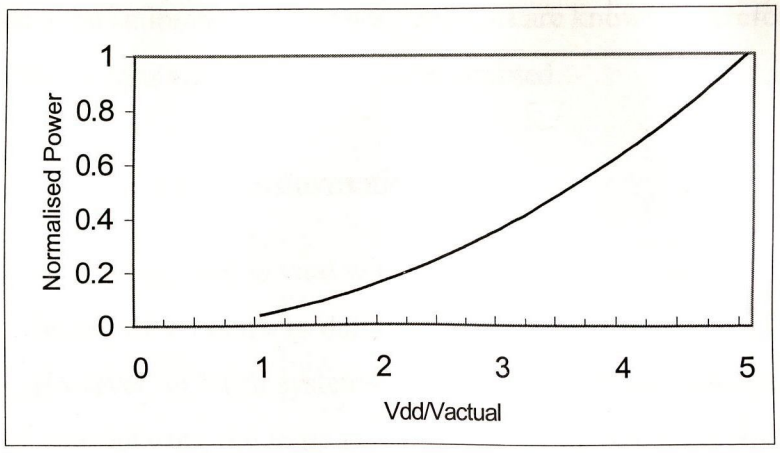


Figure 2.4: Power Consumption and Voltage

The drawbacks of this approach include the additional control logic required by the circuit [Shiu00] and the provision of a supply voltage which can be quickly and dynamically calibrated. A further problem is the environment of the chip. Most applications need a fixed input level, therefore the output buffer of the chip must provide an almost constant output [Sanc99]. This can only be achieved by using adjustable dc to dc converters (DC/DC amplifiers) and additional logic. These amplifiers add additional loss to the power consumption caused by converting the signal level. Therefore, while voltage scaling may be the best way to reduce the total power consumption it is also most difficult to implement. In the next section a compromise between low supply voltage and ease of implementation for a given environment is presented.

2.2.4 Different Voltages on a Single Chip

As shown in the previous section the standard supply voltage can be scaled down without losing performance. However, this voltage scaling approach is difficult to implement because it requires additional logic blocks to overcome subsequent problems such as reduced throughput. Furthermore, a varying voltage might introduce problems if the chip must interact with external applications. The scaling logic can be avoided by using different voltages on a single chip. Usually two voltages are used. One voltage is used to supply the core and the other to power the I/O pads. The lower core voltage should provide a good compromise between low supply voltages and easy to implement design structures. On the other hand the higher voltage at the I/O pad is to ensure that the chip is able to communicate correctly with the peripheral environment [Wang98] [Sanc99]. The two supply voltages can be generated on the chip by using dc/dc converter [Best98] [Jou98]. The advantage of these dc/dc amplifiers is

that they only need to be calibrated once as both voltages are known. Therefore, the additional logic required for continuous voltage scaling can be omitted.

2.3 Critical Path Reducing Transformations

The optimisation of the throughput is vital when attempting to reduce the supply voltage to a minimum. As the timing of a circuit is defined by the critical path it only makes sense to redesign this path. However, in VLSI systems it is often difficult and uneconomical to reduce the critical path by restructuring the logic cells. Therefore, the most common approach is to restructure the functional block, which contains the critical path. This is done using a number of different approaches.

2.3.1 Parallelism of Structures

A simple approach to retain throughput when reducing the clock frequency is to parallelise processes. This is easy to implement but more than doubles the size of the design. Also, special design tools must be used to keep the resulting increase in capacitance to a minimum. Otherwise the capacitance of the larger layout may completely cancel out the improvement due to supply voltage reduction [Mehr97]. Therefore, parallelism of structures is only possible when no or little area restrictions are given. Even when there are no area restrictions the higher chip size will increase the production costs and therefore parallelism may not be suitable for some applications.

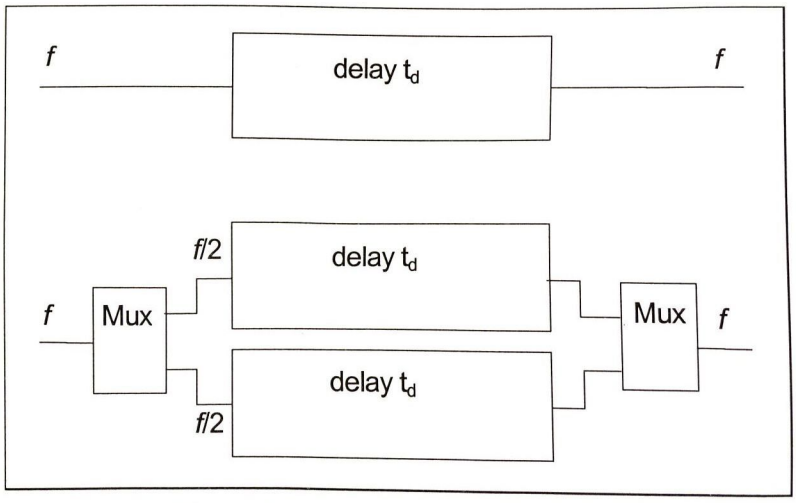


Figure 2.5: Principle of Parallelism

Figure 2.5 illustrates the idea of parallelism. It is assumed that a functional block has a delay time t_d. There is no possibility of decreasing the throughput of the device as the block is already running at a frequency which cannot be reduced. With such a block it is impossible to reduce the supply voltage because of the propagation delay dependence on the voltage. Using duplicate blocks to implement the parallel configuration, the data must now be input to them via a multiplexer. The first piece of data is read into the first block and the second piece of data into the second block. Therefore, the frequency of each block can be reduced by a factor of two, without losing throughput. At the output of the blocks the output data is demultiplexed and the original data rate is reconstructed. Both blocks are now running at half of the original block frequency, therefore the supply voltage can be reduced to a level where the delay doubles [Lyon93]. For an original supply voltage of 5V this happens at approximately 3V as Figure 2.3 illustrates. This approach also has drawbacks. Decreasing voltages must be traded off against the larger chip dimensions [Fren98].

2.3.2 Pipelining

Quite often designs are not optimised for peak performance. A simple way to maximise the throughput of such a design is by using pipelining techniques. Here pipelining latches are used between functional blocks in a way that the longest delay between two latches is smaller than the critical delay [Gonz96] [Good98]. This is illustrated in Figure 2.6.

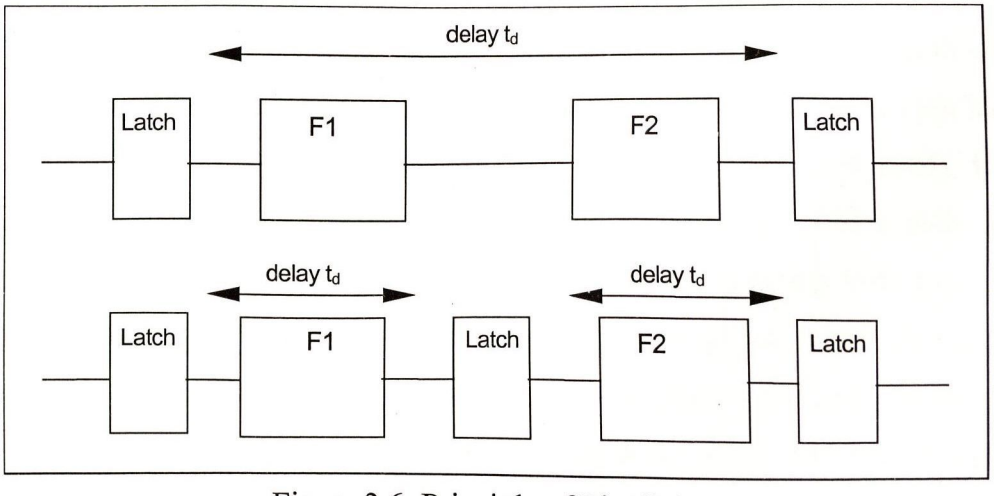


Figure 2.6: Principle of Pipelining

This example shows a circuit which contains two larger functional blocks (F1 and F2) and two latches, one at the input and one at the output of the circuit. With a traditional implementation, both functional blocks are connected in series and the data must run

completely through the circuit before the next piece of data is read into the first block. The time the data needs to run through the circuit is t_d. The second circuit uses the pipelining technique. An additional latch is connected between the functional blocks. When the first functional block has finished the calculation, the data is stored in the latch between the functional blocks and is read into the next block. Now the first functional block is free to perform the next computation. With this technique the throughput of the circuit is doubled. If the throughput is kept constant the circuit can work at half the frequency of the traditional implementation. Similar to parallelism the voltage can be reduced by a factor of two, which will result in power savings of approximately 60% when compared to the 5V supply voltage standard [Fren98]. This technique has the advantage that it requires only a small amount of additional space, when compared to the parallelism technique. It therefore suits applications which are restricted by size. The physical capacitance of the design is also kept nearly constant. The only drawback of this approach is the additional control logic required to control the shifting between the functional blocks and the additional consumption of the pipeline latches.

2.3.3 Pipelining and Parallelism

As shown in the two previous sections parallelism and pipelining of designs can result in major time savings. If there are no restrictions on area then both techniques used together mean that a circuit can be operated at an even lower clock frequency. This is done by pipelining the critical path first. Then the designer must check to see if this path is still the critical path of the design. It would make no sense to speed up any path other than the critical one, because it is only the critical path that restricts the maximum clock frequency. If it is still the critical path then this path should be reduced. If not the new critical path should be identified and its delay minimised. However, if the idea of combining both pipelining and parallelism of structures is used to provide the maximum throughput, it will also combine the drawbacks of both techniques such as the additional control logic and doubling of area [Fren98]. Table 2.1 (taken from [Fren98]) shows the area, power and scaled voltage of a simple structure in comparison to that of pipelined parallel and parallel-pipelined structure.

Architecture	Voltage / V	Area (normalised)	Power (normalised)
Simple	5.0	1.0	1.00
Pipelined	2.9	1.3	0.39
Parallel	2.9	3.4	0.36
Pipelined and parallel	2.0	3.7	0.20

Table 2.1: Effects of Architecture-Based Voltage Scaling

2.3.4 Resource Sharing

In low throughput applications, time multiplexed architectures are often used to minimise the area. However, where module sharing occurs, the resultant throughput is typically increased. If both busses and functional blocks are shared within the modules, then the higher throughput (and therefore the higher switching activity) makes any reduction in supply voltage hard to achieve because of the high clock frequency required to drive such devices. Therefore resource sharing can be seen as serialisation of a design which increases the power consumption [Lyon93].

2.4 Reducing the Voltage Swing

Although the most obvious and effective way to reduce the total power consumption is to minimise the supply voltage, further power savings may be achieved by lowering the swing voltage [Yama96]. In order to investigate the effect of the swing voltage (10) can be written as

$$P_{dynamic} = C_{active} V_{dd} V_{swing} f_{clk} \tag{19}$$

Usually the swing voltage is approximately equal to the supply voltage, but if the swing voltage is V_x volts smaller than the supply voltage, the equation may be rewritten as follows.

$$P_{dynamic} = C_{active} V_{dd} f_{clk} (V_{dd} - V_x) \tag{20}$$

The energy consumed by a single transaction is given by.

$$E_{dynamic} = C_{active} V_{dd} (V_{dd} - V_x) \tag{21)}$$

Now it is clear that the energy is proportional to the capacitive load and that the energy saved is $V_x C_{active}$. Therefore, this approach only makes sense if the load is very large and the supply voltage cannot be reduced any further. This implementation has also several drawbacks. Firstly the noise margin decreases by V_x. If the supply voltage is near the sum of the threshold voltages, then the noise margin may be effectively reduced to zero. Secondly, even for a HIGH level signal the output does not rise to the upper rail and this may cause the next stage not to turn off completely. This would result in a high short-circuit current and large static power consumption. Because of the reduced noise margin special gates are then needed to restore the input signals. These gates require additional devices and lead to extra parasitic capacitance [Zhan00].

Due to the problems stated above voltage swing reduction is generally only useful when driving large loads and when using cell libraries containing cells to restore the noise margin. Therefore, voltage swing reduction should only be used if the supply voltage is already at the minimum value.

2.5 Summary and Conclusions

This Chapter commenced by defining all the sources of power dissipation encountered by an IC designer. These include, in order of increasing significance, leakage, short-circuit and dynamic power dissipation. These components of the total power consumption were then individually discussed. This discussion demonstrated that the most effective way of reducing the overall power consumption can be achieved through a reduction in the supply voltage, as the supply voltage is present in all components of the power consumption.

Where supply voltage reduction is concerned, meaningful results can be achieved by the application of voltage scaling and the use of multiple supply voltages on a single chip. Lowering the power dissipation using these techniques results in an increase in the circuit delay. This delay may be compensated for by the use of architectural transformations. The principle underlying these techniques is based on the reduction of the critical path by the use of pipelining and parallelism. While these techniques are very effective, their application is limited to full custom circuit design. However, as the design activity described in this thesis is targeted to a semicustom design, the scope for voltage scaling is very limited. In the case of the technology library used, the power supply tolerance is ±0.5V. The consequence of this

limitation is that all design efforts must be solely focused on the one remaining component of power consumption, namely the dynamic power dissipation.

3 Dynamic Power Consumption

The previous chapter has shown the limitations of supply voltage reduction as a means of minimising power consumption. In order to achieve significant results, the design focus must be shifted from the circuit technology to a behavioural view of the design. The dynamic power consumption captures the power performance of a design. As dynamic power consumption consumes more than 90% of the total power consumption, reduction of this component is the most effective method of minimising the total power consumption of a design. As shown in Section 2.1.1, dynamic power consumption is caused by charging the node capacitance. Therefore, minimisation of the switching activity and the consequent minimisation of the active capacitance, is the most effective way of lowering the total power consumption.

This chapter presents a comprehensive compendium of techniques described in literature. Each technique is described and evaluated for its effectiveness in reducing dynamic power consumption.

3.1 Adding Additional Logic

There are many possibilities for reducing the switching activity by adding additional logic. One approach is to switch off or power-down unused stages so that transactions are only executed when necessary. This is useful because in synchronous designs the logic between registers is always computing, depending on the present input, even when not performing useful operations. The simplest way to implement this idea is to split the clock signal into different domains and to switch a clock domain off if the functional block, to which it is connected, is not required [Good98]. Such a technique is called a gated clock implementation. This also implies that the load of the clock signal changes which makes the clock generator hard to design. The reason for this is that as the length of the path changes, both capacitive load and the delay time changes. The latter may cause the circuit not to run at the maximum frequency because the clock tree is unbalanced.

A simple way to switch off components with a gated clock logic is by using additional enable logic [Beni00]. These circuits require additional overall control logic and control signal wires, which add switching activity and capacitance to the circuit. A different approach

is to add not only an enable but also *operation ready* signals to the functional stages. These signals are then used to control the enabling of the previous block and the next functional block with a simple gate only. These circuits are called self-timed circuits. Even if the individual blocks require more stages, no global control logic is required to control the maximal throughput. The control wires are kept much shorter due to the fact that only the neighbouring blocks need to be controlled. However, if the path has a high rate of throughput all approaches using additional logic might increase the total power consumption, because the additional logic and capacitance might consume more power than the normal continuous computing path [Niel94].

One of the most vital computational operations is the addition of numbers. There are many different ways to implement adders, but if low-power is the primary goal of the implementation only a subset of approaches are useful. Most adders produce unnecessary operations and glitches (see Section 3.5 for reference).

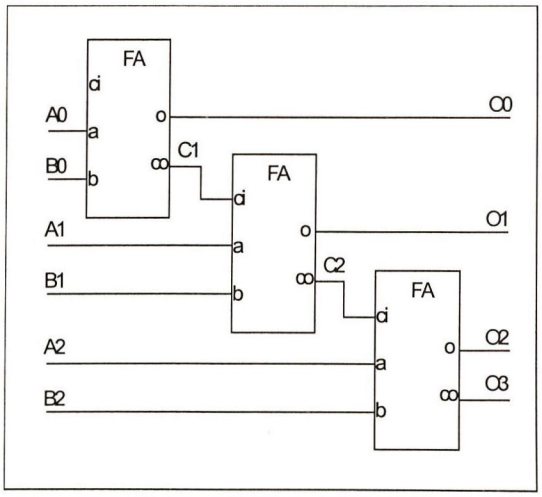


Figure 3.1: Ripple-Carry Adder

The simplest way to implement a large adder is the ripple-carry adder as shown in Figure 3.1. The adder in this example is a three bit adder which performs a(0:2) + b(0:2) = c(0:3). The output signal consists of a 4-bit signal so that the circuit is capable of generating a carry signal. If signals a and b are applied to the input ports all adders start to calculate the output simultaneously. If any adder produces a carry overflow this causes the next adder to start to calculate a new input value. If this new value also produces an overflow the next adder has to calculate a new output. This causes unnecessary calculations before the final result is computed. Without latches at the outputs of the adders, all transactions are transmitted directly into the following stage and so cause power consumption. The worst case activity is

expressed in the equation below. In this equation n is the maximum number of bit calculations and k is the number of full adders.

$$n = \sum_{x=1}^{k} xN \tag{22}$$

For a 8-bit adder implemented using eight 2-bit full adders the maximum number of calculation cycles would be 36. In other words such a 8-bit adder changes output state up to 36 times per calculation, 35 of them being useless power consuming operations. To avoid such behaviour, a carry-look-ahead adder can be used. This calculates firstly the carry's and then connects the correct values to each single input of all adders. For this reason only carry-look-ahead adders (or multi-level carry-look-ahead adders) should be used in low-power circuits. These circuits calculate the carry signals and then start with the adding process. This makes computation not only less power consuming but also faster, because the circuit need only do the calculation in two steps i.e. evaluate the carries and then compute the result.

3.2 Reducing the Number of Nodes

A different approach is to minimise the number of nodes. This technique assumes that the reduction of nodes will not only reduce the total capacitance but also reduce the overall switching. This method may actually produce more switching as presented in [Burd95]. In addition, this method does not address any of the factors influencing the dynamic power consumption. This is the reason why the reduction of nodes is not a valid method of reducing the active capacitance, as it assumes that the power and the number of nodes are directly related while neglecting all other factors.

3.3 Precomputation

Precomputation is already used in traditional designs in order to speed up processes [Mone95]. An example of this is the precomputation of the carry as used in the carry look ahead adder architectures. The goal of precomputation in low-power applications is to reduce the overall active capacitance of a functional block by adding additional local control logic. Assuming a block has to perform the function F1 then the goal of precomputation is to extract a subset of functions F2 as shown in Figure 3.2.

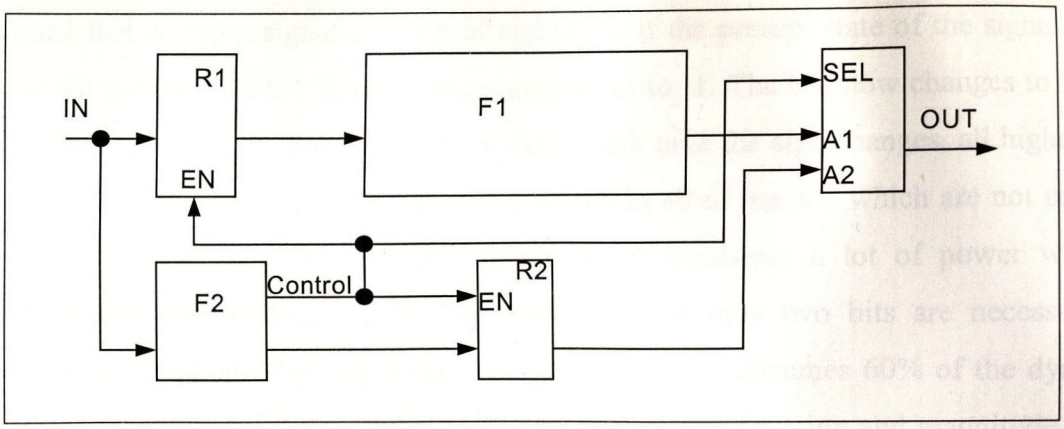


Figure 3.2: The Principle of Precomputation.

As F2 is a subset of the function F1, this allows efficient computation of the result if a certain condition at the input of the functional block is met. The designer has to chose the function in such a way that the switched capacitance of F2 plus the dynamic capacitance of the additional logic is smaller than those of the original computation performed by F1. The designer also has to ensure that the overall active capacitance of the new block, which includes not only F2 but also additional local control logic, is smaller than that of the original design. An example should illustrate this. If the Block F2 plus the additional hardware has an active capacitance of 8% in relation to block F1 then the designer has to be sure that the condition which is checked by F2 occurs more frequently than 8% of the time. For more information on the effects of additional logic refer to section 3.1. It should be noted that precomputation can only be used in paths other than the critical one. This is because the additional functional block F2 adds to the delay of the block at the input of F1. However, if it is deemed to be necessary, it is possible to add an additional register to the input in order to overcome this problem.

3.4 Number Representation

As the total power consumption is highly dependent on the switching activity, in this section the effects caused by the representation of numbers are analysed. Most numbers are represented in two's-complement format. This makes arithmetic processes such as adding and subtraction very easy to execute. Positive values are expressed as a bit integer. Negative values are the positive value which is inverted and a one is added to the result of the inverted value. This means that all higher order bits, which carry no information, represent the sign of the number. If the most significant bits (MSBs) are 0 then the number is positive, otherwise it is negative. This is also the reason why the two's-complement is not recommended for low-power implementations as the following example illustrates.

Assume that an input signal consists of eight bits. If the present state of the signal is +1 the bus is set to "0000 0001". Suppose the signal is set to -1. The bus now changes to "1111 1110". This small example illustrates the problem. Each time the sign changes, all higher bits also perform a change, because the sign is duplicated in all of the bits which are not used to represent a number. Therefore, the two's-complement consumes a lot of power without transmitting any real information. In the example above only two bits are necessary to transmit the information. This implementation unnecessarily consumes 60% of the dynamic power. The best solution for this problem is to split the signal into sign and magnitude. Now only the highest bit carries the sign information and all other bits are used to represent the unsigned number. If the example above is used again, a change from +1 to -1 only causes the highest bit to change from 0 to 1. This example represents best the possibility of power savings because now only 12.5% of the dynamic power is consumed, when compared to the previous example. However, as previous work has shown, the switching probability of signals is highly dependent on the origin of the signal's samples [Chan92]. In [Land94] and [Chan95d] it was shown that music, speech and video signals have a very similar bit level switching probability. They demonstrated that the most significant bits have a switching probability of approximately 0.5 and that of the least significant bits is considerably lower. Figure 3.3 (adapted from [Chan95d]) shows this behaviour for image data. In such a case the use of a sign magnitude representation will have positive effects on the power consumption. Hence, before deciding on which signal representation to use, the properties of the signal should be investigated.

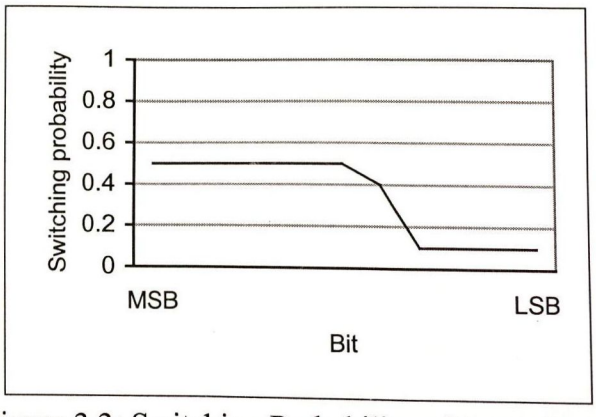


Figure 3.3: Switching Probability of Image Data

The choice of the number system will also cause different activities during computations. If during a computation the sign changes, the same rules apply as shown above. Therefore, it might be useful to compute the sign and magnitude in separate units of a functional block

instead of using two's-complement devices. This requires special devices which are more complex and larger than those used to compute two's-complement. The effect of the higher physical capacity is often smaller than the effect of the reduction in switching activity using sign magnitude representation in highly active paths.

3.5 Minimising Glitching Activity

Due to finite propagation delay through logic blocks (or critical races) the output of a device can have different values during one clock cycle before settling to the correct value. This is called glitching or hazard. These glitches cause this stage and sometimes even other stages to change value and produce unnecessary transitions. These transitions consume dynamic power. This is not necessarily a design error. Only if the design is intended for a low-power application do these glitches become of interest to the designer. Typically these glitches produce around 20% of the total power consumption, which might rise up to 70% of the total power in cases such as combinatorial adders [Naim94]. Power consumption due to glitches is also called toggle power. Figure 3.4 shows a simple example to illustrate glitching activities.

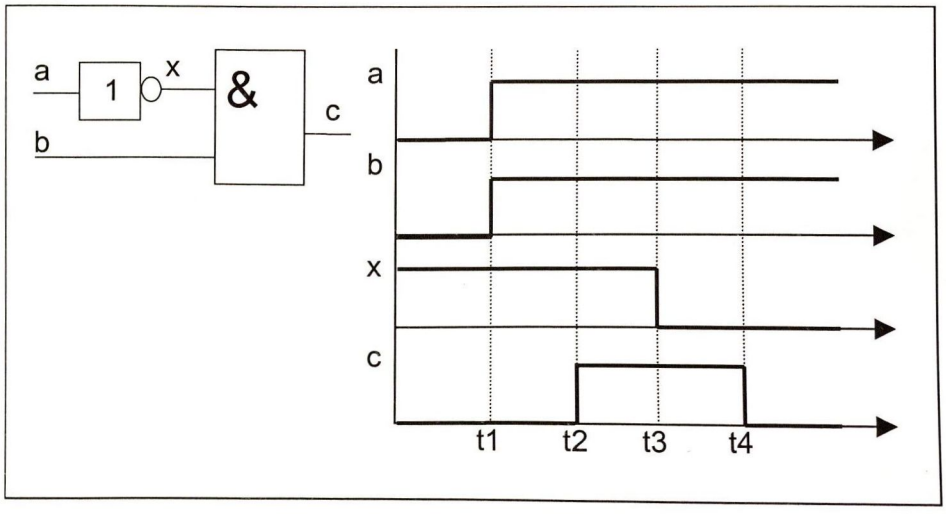


Figure 3.4: Origin of Glitches

The example contains a two-input AND device. The first input is connected to an inverter. The input signal was set to LOW for both inputs and the output c was also set to the LOW level. The output x of the inverter was therefore at the HIGH level. If the input ports are changed to HIGH (at t1) no output will change due to the finite propagation delays. The input b is now HIGH, as is the output x which is still at a HIGH level because of the propagation delay of the inverter. Hence the AND changes value at t2 to a HIGH value. After the inverter propagation delay time, the inverter will switch to a LOW output causing the AND to switch

its output finally to LOW, after its propagation delay. This simple example shows that propagation delays are more critical in low-power applications. One way to avoid such behaviour is to balance all signal paths ensuring that all signals arrive at the same time [Gagh99]. This also ensures that the circuit operates at a maximum frequency. A different example is the carry-look ahead adder already mentioned in section 3.1. Instead of using balanced paths this example uses additional logic to avoid glitches.

The easiest way to avoid glitches is to balance all paths equally. Figure 3.5 illustrates this by comparing a serial design to a tree structure. The serial implementation may enter three unstable states before settling to the correct value. As seen in this figure the tree structure not only has a reduced propagation delay, but is also totally balanced and does therefore not produce any glitches, assuming all gates have the same delay. This example shows the importance of the design structure. While the first example is produced with the statement following statement OUT = A+B+C+D, the tree adder is produced by the syntax OUT = (A+B) + (C+D). Therefore, it is important to keep the synthesised circuit in mind when writing abstract code.

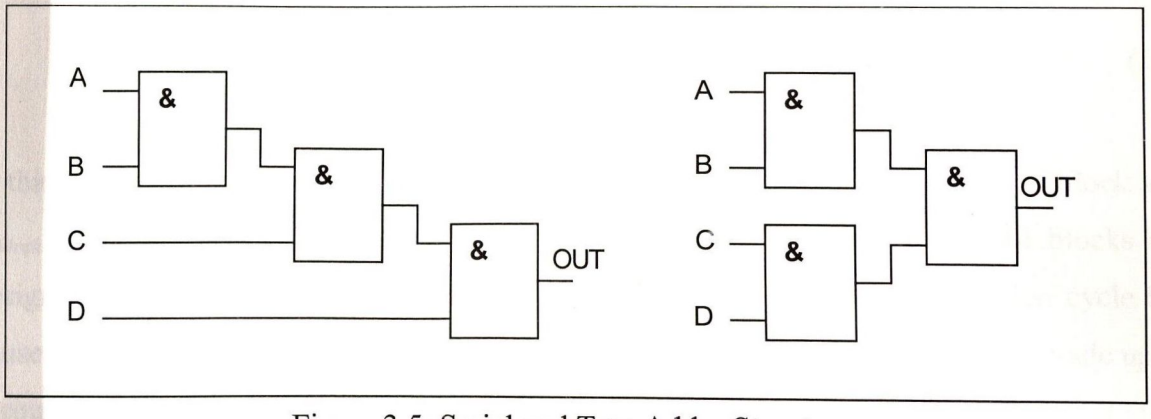


Figure 3.5: Serial and Tree Adder Structure

Finally, it should be noted that even if the tree adder is balanced, in practice there will be still glitching activity due to the fact that the gates themselves are not balanced. These glitches can only be avoided by using glitching free adder modules.

3.6 Additional Capacitance through the use of Latches to Reduce Glitching

The previous section focused on the avoidance of glitches in a design, but often it is not possible to balance all paths to reduce the glitching to a satisfactory level. Cost considerations often make it necessary to use modules which contain several copies of the same functional

block. In such modules the glitching activity is normally "reduced" through the introduction of latches at the output of the design. This, however, only ensures that these glitches do not propagate into the next stages of the design, but does not prevent the occurrence of them [Good98] [Laks99] [Xant99] [Beni00]. This section focuses on the analysis of the additional capacitance brought into a design to reduce the glitching activity.

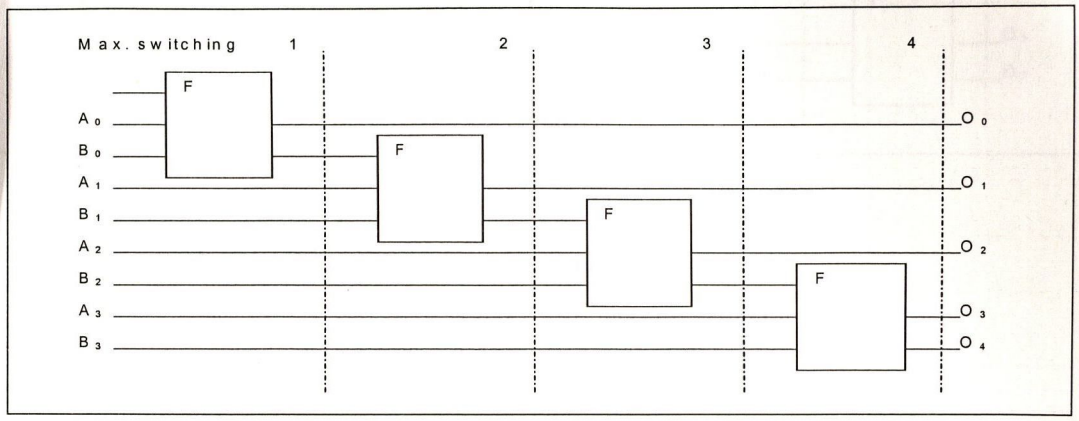


Figure 3.6: Glitching in Cascaded Functional Blocks

The maximum active capacitance of a design can be described as

$$C_{\max} = \sum_{g=1}^{k} g \bullet C_{Block} \tag{23}$$

In this equation k is the number of blocks, g is the glitching activity at the input of block and C_{Block} is the physical capacitance of the module. Often in VLSI designs small blocks are designed and cascaded in order to form larger blocks. This speeds up the design cycle but causes a ripple effect in the circuit. Figure 3.6 is an example of such a design. It is made up of 4 equal smaller design units. If the known values of this design are put into equation (23) it can be written as:

$$C_{\max} = 10 C_{Block} \tag{24}$$

If this figure is compared to the minimum amount of switching required in order to perform the computation, in this case $4C_{Block}$, it can be seen that more than half of the power consumed by this particular design is the result of glitching. For this reason these designs are often pipelined. The same design can be split into two stages separated by a latch. Figure 3.7 shows the overall structure.

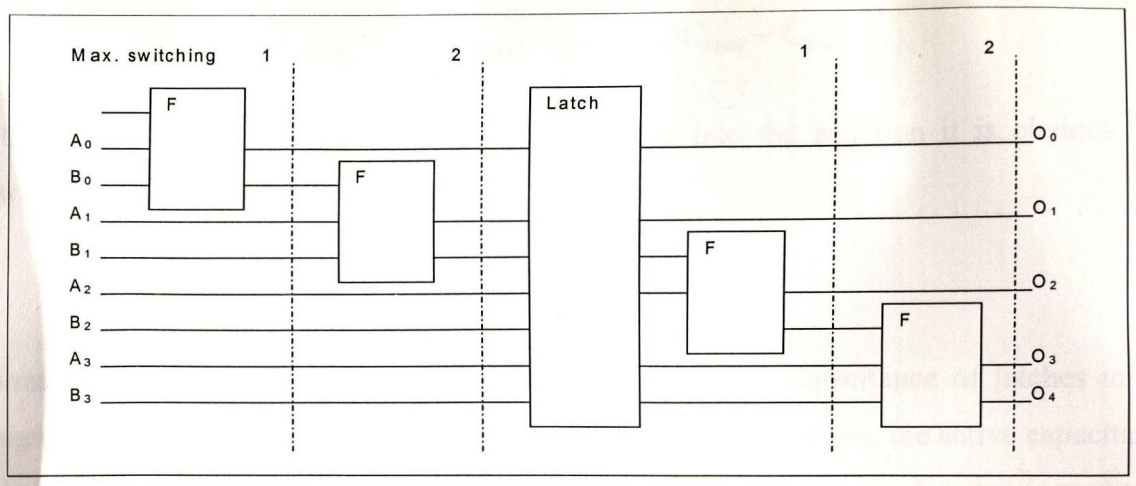


Figure 3.7: Cascaded Structure using one Latch

The active capacitance of such a design can be expressed as

$$C_{\max} = \sum_{g=1}^{K_{stage1}} g \bullet C_{Block} + C_{Latch} + \sum_{g=1}^{K_{stage2}} g \bullet C_{Block} \tag{25}$$

For a symmetrical design structure as shown in Figure 3.7, this equation can be rewritten as follows.

$$C_{\max} = 2\sum_{g=1}^{k} g \bullet C_{Block} + C_{Latch} \tag{26}$$

If now the known values are put into the equation the maximum active capacitance is

$$C_{\max} = 6C_{Block} + C_{Latch} \tag{27}$$

If this is now compared with the result obtained from the unlatched design (24) it can be seen that the glitching is reduced by 40%. In order to determine if the active capacitance of the latched version has a lower capacitance, both implementations have to be compared. By subtracting the capacitance of the latched design from the active capacitance this value can be quantified.

$$C_{saved} = C_{\max - orginal} - C_{\max - latch} \tag{28}$$

By substituting the two variables with the equations derived for the two designs presented in this section (23) (25) the equation can be written as follows.

$$C_{saved} = (g_{Total-orginal} - g_{Total-latch})C_{Block} - C_{Latch} \quad (29)$$

If the known values of the two designs are now put into the equation it is obvious that unwanted switching is reduced if

$$C_{Latch} < 4C_{Block} \quad (30)$$

Normally the designer can look up the values for the active capacitance of latches in the design library data book. Due to the modular concepts of VLSI designs, the active capacitance and glitching behaviour of the functional blocks is determined before the decision is made if a block is going to be latched. In fact, more often it is the case in low-power design that after testing a block, the decision is taken to latch a functional block due to a high active capacitance caused by a high glitching activity.

3.7 Reducing the Switching Activity by the use of Don't Care Terms

The switching activity of Finite State Machines (FSMs) and Look Up Tables (LUTs) can be significantly reduced by the use of '*don't care*' terms. All terms which do not affect the global function of a node should be replaced by a '*don't care*' term since it will guarantee a change in state only if it is essential and therefore produce the lowest possible switching rate and power consumption. The same approach is taken in [Kapa99] to disable a datapath if a *'don't care'* condition is detected.

3.8 Ordering of Operations

If signals of different bit-width are used it is sometimes possible to arrange function blocks in order to reduce operations, device sizes, bus sizes and switching activity. A small example presenting two approaches should illustrate this in Figure 3.8. Three signals (signal *a* of 8-bits, signal *b* of 6-bits and signal *c* of 4-bits) are multiplied using two two-input multipliers connected in series. If the signal *a* is multiplied with the signal *b* and the result is fed into the next multiplier and multiplied with signal *c*, then the bus between the first and second multiplier is 14-bits wide. In the other case signal *c* and *b* are multiplied first and the result is then multiplied with signal *a*. The bus between the first and the second multiplier is only 10-bits wide (four bits of c plus six bits of b).

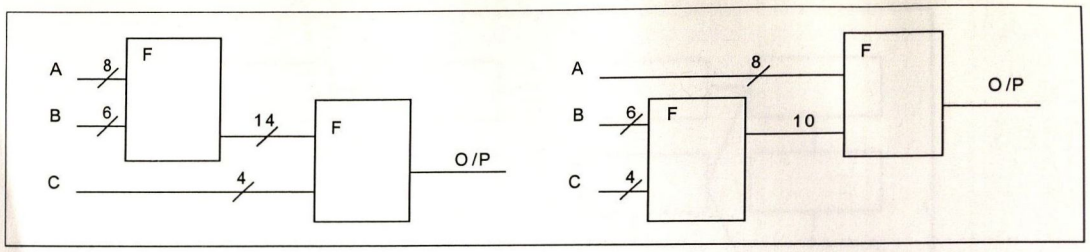


Figure 3.8: Ordering of Operations

Even if the output width in both cases is 18-bits, the second case is preferable. Firstly, it should be noted that both examples are equal in area. Also the capacitive load seen by the previous and following stages is equal, but internally the bus-width in the second example is smaller and therefore the total capacitive load driven by the first stage is smaller. What is most important is that with the smaller bus-size, the switching activity and therefore the active capacitance between both multipliers is reduced and hence this implementation consumes less power. Moreover, ordering of operations cannot only help to reduce the load of busses but can also help to prevent critical races as already explained in Section 3.5.

3.9 Multiplexed Buses

In most designs global buses are used to transmit various items of information between different I/O ports and functional units by using multiplexed buses. The signals are only on the bus for a short period before the multiplexer switches to the next connection. Normally different signals contain completely different information bits such as value or sign. The number of bits needed for the signal might also vary. Therefore, it is very possible that almost the complete information of the bus changes with each switching to another signal. Even if those buses require less area and overall capacitive load, normally fixed local buses (point-to-point buses) provide maximally the same active capacitive load to the driving source as the global bus, because of the reduced switching activity [Mehr97]. If local buses are used it is the usual case that most buses are smaller than one global bus and therefore the active capacitive load of the complete circuit is reduced.

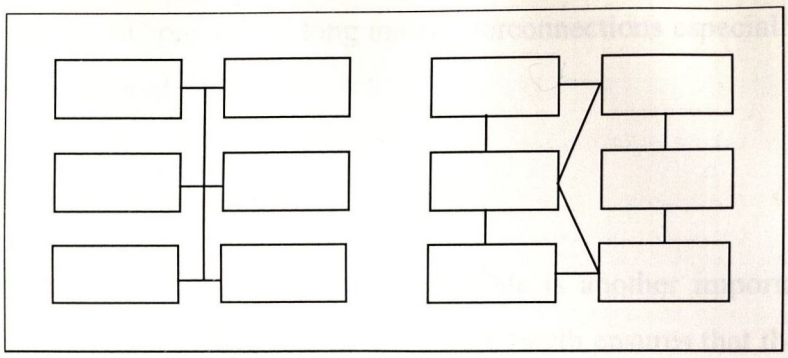


Figure 3.9: Multiplexed vs. Local Buses

For slowly varying signals the switching activity of the local bus is also dramatically reduced when compared to the global one. Even for other signals it is also possible that an unmultiplexed signal will produce less activity when compared to a multiplexed bus. For these reasons, buses in low-power implementations should never be multiplexed and the functional blocks and I/O ports should be as near as possible to the corresponding block in order to keep the buslength as short as possible to minimise the capacitive load of the buses. Figure 3.9 illustrates this. The figure on the right has eight smaller buses with a smaller active capacitance than the global one shown on the right of this figure. It has approximately the same physical capacitance. If only one bus is active the average active capacitance is only 1/8 of the active capacitance of the global bus. On the other hand if all buses are active at the same time the throughput is 8 times higher and can therefore compensate using lower supply voltages.

To implement short point-to-point buses is even more important as processes get smaller and the integration rises. For example, for a 0.1μm VLSI design, buses can consume up to 50% of the total power consumption [Naka94]. Therefore, Nakagome et al. suggest in [Naka94] to use special drivers which are able to reduce the voltage swing on buses in order to lower the power consumption effectively.

3.10 Locality of Reference

The basic idea behind this approach is to use signals locally at the point where they are generated instead of letting them travel over long buses [Mehr97]. This reduces the physical capacitance, and the active capacitance if multiplexed busses are not used. As shown in Figure 3.9 such an approach increases the throughput if signals are processed at the point where they occur and are able to travel over a direct connection. This method also decreases

the delay time since the RC product of long metal interconnections especially on sub-micron processes is a major source of delay.

3.11 Reduction of the Wordlength

To keep the wordlength of signals as short as possible is another important aspect when designing a low-power circuit [Ramp99]. A small wordlength ensures that the width of buses as well as that of the functional blocks can be kept to a minimum size. This keeps the physical capacitance of a design low. Therefore, the designer should always try to keep the signal width small and check if errors such as introduced by truncation instead of rounding are acceptable. Often it is also possible to reduce the wordlength by the use of precomputation techniques.

3.12 Coding of Signals

As discussed in Section 3.4 the level of switching activity is highly dependent on the signal representation. For computational blocks usually signal codes are chosen to allow an efficient computation. Good examples of these codes are the unsigned and the two's complement representations. However, if high-capacitance buses are to be driven then power reductions might be achieved by changing the signal representation before transmission into one that generates a lower switching activity [Yama96] [Ramp99b]. This strategy requires additional hardware to code and decode the signal which can sometimes increase the power consumption of the circuit by a factor that is greater than the power saved by coding the signal. Therefore, a balanced must be struck between the power savings on the bus due to the reduced signal switching activity and increase in power consumption due to the extra hardware.

3.13 Logic Minimising

Traditionally gate libraries use only a small set of gates to implement a given logic function. Therefore, a function is not implemented using an optimal solution, but using one which is achievable with the gates of the target library. This causes the inclusion of unnecessary stages leading to unnecessary switching activity and additional silicon area. As seen in the above

approaches for reduction of power consumption, the area is used in most approaches as a variable, which can be traded off against power and should therefore be used very carefully.

A different approach is presented in [Akita94]. Here it is shown that a larger combination of devices can have a lower switching probability and might therefore consume less power. This is only true if the extra physical capacitance of the larger circuit is balanced against the reduced switching activity. However, new low-power cell libraries are able to reduce the power consumption by approximately 25% compared to traditional libraries [Chan95b]. These libraries contain simpler logic gates and all devices are optimised for power consumption, in a way that traditional libraries have optimised their cells for speed or area.

3.14 Minimising the Number of Operations

Multiplication with fixed coefficients is very common in digital signal processing (DSP) applications and is also used widely in other areas. Traditionally each multiplication is done by one multiplier, but in low-power design a different approach can reduce the switching activity. When using add and shift multipliers it is possible to split them and share subterms. This reduces the number of stages and the switching activity required for a multiplication [Chan95c] [Nguy00]. A simple example should illustrate this:

$$A = sig * 1011 \qquad B = sig * 0111 \tag{31}$$

The same signal (*sig*) is multiplied with two different values in two different terms. The term $sig * 2^0 + sig * 2^1$ is represented in both terms. Instead of using two full multipliers this term is calculated only once. The result is then added with $sig * 2^3$ for term A. To evaluate term B the result is added with $sig * 2^2$. Even this simple example demonstrates that two adders (and the switching activity of both) can be saved without causing a higher throughput of the device.

3.15 Optimisation of Constant Operation

The extensive use of Hardware Description Languages (HDLs) leads to the use of multipurpose functions or design units and intellectual property (IP) blocks of previous projects [Mart99]. By using optimised multiplier structures rather than multi-purpose multipliers the number of operations, the delay through the block or the area, and therefore

the active capacitance, can be effectively reduced [Chan95a]. One example would be DSP applications such as filters, where multiplying with fixed coefficients is often required. These coefficients are normally known before the actual design process begins. Here the use of for-the-task optimised structures can yield significant advantages as the following example illustrates. It compares tests performed on two different divider structures for the intensity path of the RGB-HSI converter. The divider is a 10-bit by three divider the standard version is a multi-purpose divider block, while for the optimised version the inputs in the source file were defined to be a constant three before synthesis.

	Standard	Optimised	Reduction / [%]
Max. Delay / [ns]	32.24	15.4	55
Number of Nets	318	50	84
Area / [µm]	124018	22496	82
C_{active} / [pF]	9.3	2.6	72

Table 3.1: Comparison of Two Divider Structures

Table 3.1 shows that it is possible to reduce the overall power consumption by more than 70% if the frequency of the input data is kept constant [Schw97]. Power savings of more than 90% are achievable if the throughput is set to 16ns. However, these savings have to be balanced against the disadvantage of a longer development cycle.

3.16 Minimising the Capacitive Load

Even if this minimisation of the capacitive load is not as effective as the minimisation of the switching activity, it is possible to reduce the capacitive load in order to decrease the dynamic power consumption. For large technologies the delay on the wires (R_{wire} C_L) is not significant compared to the transistor delays. But if the sizes of the connections shrink with smaller technologies the resistance of the wires increases and hence the $R_{wire}C_L$ product increases. In submicron processes this behaviour can lead to higher communication delays than transistor delay times [Blair94]. Therefore, long connections such as global buses or global control blocks should be avoided. Instead of this, local buses and local control functions should be implemented to reduce communication delays and lower connection capacitances. Functional blocks having a high computational rate should be positioned together in order to keep buses as small as possible. Blocks having a smaller computational rate will then be grouped around

the other blocks to reduce the overall active capacity. This can only be done by using special design tools which are able to analyse the activity of different stages. These tools allow the user to define these blocks and so group more active areas closer together to lower the capacitance caused by interconnection wires. Traditional place and route tools try to optimise the silicon size and try to keep all connections close together in order to fulfil timing constraints.

3.17 Low-Power Libraries

Cell libraries are available which are designed for low-power implementations [Fren98]. These low-power libraries use various approaches to minimise the power consumption, for example, allowing a choice between optimal cells and even blocks, e.g. a carry-look-ahead adder instead of a ripple-carry adder. These libraries also contain more cells to implement the logic using the optimal method avoiding glitching or critical races. Also, the dimensions of the cells are kept as low as possible to reduce the capacitive loads. Furthermore, those libraries allow voltage reduction so the design can be operated at the optimal supply voltage. Some libraries provide also devices which have all the necessary ports to build a self timed circuit without the use of global control logic.

3.18 Summary and Conclusions

This chapter has presented various methods of reducing the dynamic power consumption. All the methods described have focused on the reduction of the active capacitance. The methods can be divided into two main groups. The first group of methods considered was the reduction in physical capacitance, for example the consideration of points of locality. These methods focus on a system level view of the synthesised circuit and include designer knowledge about the routing of the design. Larger bus structures are traded-off for reduced interconnect capacitances and smaller driving gates.

The second group of methods is based on a reduction in power consuming transitions. This group can be further divided into two subgroups. The first subgroup is based on the reduction of non-computational switching while the second subgroup targets a reduction in the overall switching activity. The first subgroup also includes techniques to reduce the glitching of the design. In this thesis, balancing of the signal paths will play a major role in the investigation of the RGB to HSI algorithm.

The second subgroup, which will be used in the implementation of the image conversion algorithm, implements a highly pipelined structure. This avoids the propagation of glitches. This has also the advantage of increasing the throughput of the design to a degree at which the specifications can be fulfilled. Another task will be to reduce the overall switching of the circuit. The choice of number representation will be the first vital design decision to be taken.

As shown in this chapter, the methods of reducing the active capacitance are many fold and often mutually exclusive. However, as a general rule the first design decision to be taken is the selection of the number representation in the system. After this, the selection of power reduction techniques becomes a design dependent issue. For example, the method of reducing the wordlength through truncation will only have a limited application because this also reduces the dynamic range of the signal. However, it was successfully implemented in the RGB to HSI converter and led to power savings of more than 10% in the intensity path. The multiplicity of techniques available is the main reason why the expertise of the designer is the main asset when targeting low-power designs. As a result of this, the designer requires accurate and fast feedback concerning the effects of his design decisions on the power performance. Therefore, the next section will describe the development of such a power estimation tool to compute the power consumption of a design at the highest possible level.

4 PowerCount: A High Level Power Estimation Tool

The investigation presented in the previous two chapters has shown the need for power estimation at the earliest possible stages of the design cycle. To achieve this, a power estimation tool, PowerCount, was developed. This chapter first presents the fundamental concept behind PowerCount and then explains the necessity for the stringent specifications and the reasoning behind the design decisions made. Finally, the operation of the software is detailed and the performance of PowerCount is validated.

Traditionally, high accuracy in power consumption measurements is guaranteed by simulating designs at the layout level. This is usually done by means of SPICE [Spice] simulations. These simulations are based on equation (5). They calculate the power consumption by monitoring the current. While these simulations are accurate, they are slow and can only analyse the design in the final stages. When designing circuits at higher levels, accuracy is not the main concern to industry, where time to market is the most important factor. A further problem is the early bottleneck detection, where it is important to find the 'hot spots' in a design to focus the design efforts. In addition, the possibility of rapidly comparing different versions of a design is of great interest to industry. For all of the above reasons, such a high level power estimation tool is essential to the successful completion of the work described in this thesis. Here the comparison of different implementations will need to be made to verify the different implementation approaches presented.

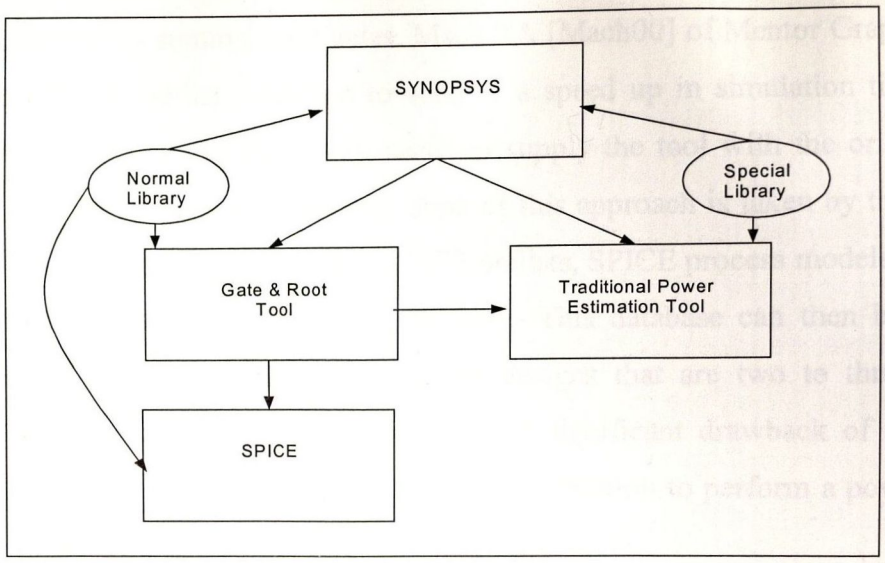


Figure 4.1: Hierarchy of Traditional Power Estimation.

4.1 Current Power Estimation Tools

Approaches to determining the power consumption at the higher levels, such as those taken by Powermill [Pmil98] or IRSIM [Irsim], estimate the power at the register transfer level (RTL) by means of a specially created netlist. While these tools are much quicker and provide an accuracy of 90% and greater, they have the drawback that a special netlist must be generated from a routing tool. They also depend on special design libraries and therefore a particular vendor. Figure 4.1 shows the traditional approach taken to calculate the power consumption of ICs. A totally different approach is taken by Explore. Explore is a behavioural level power estimation tool. This tool estimates the power by means of a flow graph description of the algorithm [Mehr94]. The best possible implementation is then taken from a set of reference data and with this information the power consumption is calculated. This method also has some drawbacks. The designer has to develop and write a special flow graph description of the algorithm. At this phase, the implementation parameters are normally unknown (except perhaps the technology). Therefore, the system must rely on a set of reference information about the algorithm. If no reference data is available, then the power analysis is not possible.

In the late 1990s various commercial power estimation tools emerged. These include MultiSim [Mult00] which has a mixed SPICE/VHDL simulator therefore requiring SPICE models of all components to be simulated. The synthesis tool of MultiSim is limited to FPGA implementations and it only supplies models for these devices. Another tool Power-Cut [Pcut00], has been designed to speed up SPICE simulations, performing a SPICE netlist

component reduction by summarising gates. Mach TA [Mach00] of Mentor Graphics applies this concept of SPICE netlist reduction to achieve a speed up in simulation time of up to 1000. With Mach TA, the designer only needs to supply the tool with the original SPICE netlist to perform the simulation. A modification of this approach is taken by the Star-MBT [Smbt00] cell characterisation tool, where SPICE netlists, SPICE process models, and simple cell pin descriptions are used to create a database. This database can then be used with software such as Star-Sim [Ssim00] to simulate designs that are two to three orders of magnitude faster than SPICE simulations. The most significant drawback of all the tools described here is that they require low-level design information to perform a power analysis. Thus, testing is restricted to the later stages of the design cycle.

Power Tool [Ptoo00] and VeriPower [Vpow00] are both high-level Verilog simulators. The main difference between these tools is that VeriPower has its own library characterisation tool which allows the user to extract the power information from its existing design library, while Power Tool requires already characterised libraries. However, neither of these tools account for the interconnect, unless the user is able to provide them with it. Others include PeakWatcher [Pwat00], a tool to quickly find the 'hot spots' of a design so that the design effort can be more efficiently focused towards the power bottleneck of a circuit. IBM's PowerCalc [Pcal00] essentially uses the same concept as PowerCount. It uses the routing and timing information of either Cadence Verilog-XL [Veri00] or Model Technologies ModelSim/VHDL [Msim00] to obtain an estimate of the dynamic power. The node switching activity is collected during the timing simulation and is then used to calculate the dynamic power consumption.

Only two high-level power estimation tools currently exist which automatically generate an estimate of the interconnect. These are Watt Watcher [Wwat00] and PrimePower [Ppow00]. Watt Watcher is a high-level power estimation tool capable of measuring the power consumption of VHDL and Verilog netlists. It uses native algorithms which estimate the capacitance of the interconnect to achieve estimates within an accuracy of 80%. PrimePower is power estimation tool released by Synopsys and it relies on characterised library information to perform power estimation. This library characterisation can be performed using PowerArc [Parc00]. PrimePower uses additional circuit information such as the interconnect and timing information to perform a high-level power estimation of the average and peak power of a design.

4.2 Power Estimation Technology

Several techniques have been developed to simplify power estimate [Burd94], [Land94], [Tsui95]. All of these papers are based on the same two ideas. Firstly, instead of using the actual value of the physical node capacitance, the average physical capacitance $C_{phy\text{-}avg}$ is used. $C_{phy\text{-}avg}$ is the total physical capacitance of a circuit divided by the number of nodes. The second idea is to replace the node switching activity factor $n_{(0,1)}$ with the switching probability factor p_t of the node. This factor reflects the probability that a power consuming event occurs. The probability switching factor is calculated for a uniformly distributed white noise (UWN) input signal. It is precalculated for each cell and stored together with other library information. After the circuit is compiled, a special netlist is generated and the probabilities are propagated through the design. The average capacitance and average power consumption are then calculated. Equation (32) shows the method used to calculate the average dynamic capacitance.

$$C_{average} = \sum p_t \frac{C_{total}}{n_{nodes}} \tag{32}$$

This value is then used to compute the average dynamic power consumption.

$$P_{average} = C_{average} V_{dd}^2 f_{clk} \tag{33}$$

This method is very fast because the amount of input data required for simulation is small but this method also has its disadvantages. Equation (32) assumes that each node, regardless of its load or output capacitance, has the same node capacitance. This is not true in real designs. Therefore, a path with a high switching activity but low node capacitance or vice versa, will increase the error of the result. Furthermore, it is highly unlikely that the input signal at each node is of a random nature. Internally, due to the connections between the different blocks as well as the distribution of the signal, the switching activity will be correlated with the input signal and will not be random. These methods do not take this factor into account. Therefore, to compensate, techniques which use the probability factor in conjunction with the real node capacitance have been introduced.

Other techniques first simulate the input probability of a module and then overlay this information with the data obtained from a UWN simulation [Burd94]. This method has the disadvantage that a design must be first described at a software level. A simulation with

actual input data is then performed in order to obtain the actual input probabilities at each block. With these probabilities, the input vectors for the overlaying of the probabilities at the RTL are generated before the design can be simulated in small blocks. While the power simulation is fast, the preparation takes a long time. Furthermore, different tools are required for the various levels.

A different method, the Dual Bit Type model, accounts for the input correlation and was proposed by Landman *et .al.* [Land94]. This model relies on a special netlist as well as a particular library and a look-up table to account for the random and correlated parts of the active capacitance. Another drawback is that this model requires calibration for each type of design.

All of these methods rely on special design or technology libraries which include the switching probability for each cell. Many estimation methods also use a zero delay model. These models do not take delay times into account and neglect glitches and hazards. These glitches depend mainly on the implementation of the algorithm and can only be accounted for by simulating with real input data and design information. Finally, it should be noted that these methods also only work for combinational logic. When estimating the power consumption of sequential logic the state probabilities have to be taken into account. Different methods are described in [Tsui95]. To overcome these problems, a new power estimation tool, called PowerCount, was designed.

The next section will show the limitations of using library reference information and therefore argue for a power estimation tool which takes the interconnect into account. Then a brief overview of the high-level design process is given to provide the background required to understand the positioning of such a tool within the design process. At this stage the idea of PowerCount is introduced before the methods of generating an estimate for the active capacitance are introduced. Finally, the operation of PowerCount is shown and evaluated.

4.3 Estimation of the Dynamic Power Consumption using the Library Reference Book

This section shows the importance of including all possible design information into a power estimation. The results of this example are then used to show the factors which have to be taken into account to accurately estimate the power consumption of ICs. For this purpose, a 1bit full adder is used as an example to illustrate how the power consumption is calculated

and the effects of neglecting the influence of the interconnect are shown. Figure 4.2 shows the structure of the 1-bit full adder to be investigated. The reason why a small design like the 1-bit full adder is used is that here it is still possible to manually verify the results of this investigation. The theoretical physical node capacitances, which are required to calculate the active capacitance, are taken from the Library Databook [Es2] of the implementation technology. In this case the ES2 0.7μm standard CMOS library was used.

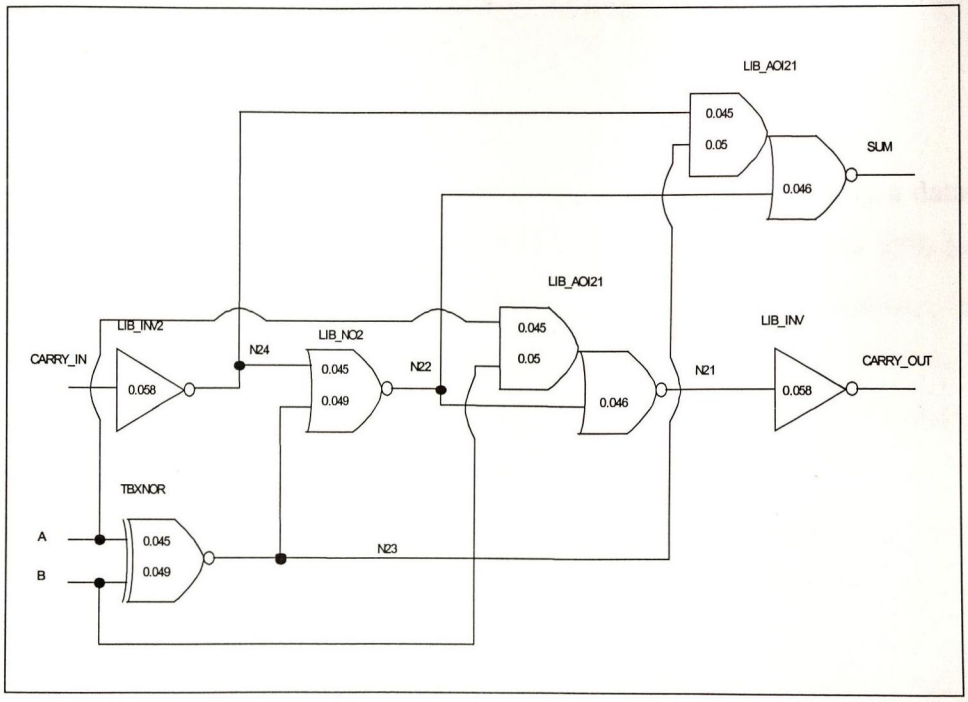


Figure 4.2: Internal Structure of an 1-Bit Full Adder

Table 4.1 contains the physical node capacitances (column 2) as given in the Synopsys netlist report after synthesis, and the physical node capacitance (column 3) of each node, as stated in the Library Databook [Es2]. The true active capacitance (column 6) is calculated using a timing simulation with 600 input vectors and the real node capacitance of column 2 .

	synopsys node cap.(pF)	library node cap. (pF)	deviation true/theor.	library act. Cap (pF)	synopsys act. Cap (pF)	switching probability
N24	0.14	0.095	32.14%	0.023	0.035	0.247
N23	0.14	0.094	32.86%	0.049	0.074	0.525
N22	0.14	0.092	34.29%	0.024	0.036	0.258
N21	0.09	0.058	35.56%	0.020	0.030	0.338
SUM	0.03	Wire	100.00%	0.000	0.014	0.483
CARRY_OUT	0.03	Wire	100.00%	0.000	0.010	0.340
CARRY_IN	0.13	0.058	55.38%	0.015	0.033	0.250
B	0.12	0.076	36.67%	0.020	0.031	0.257
A	0.14	0.086	38.57%	0.023	0.037	0.265
act. Cap				0.173	0.300	
Error	42.30%					

Table 4.1: Power Estimation using a Data Book

As seen in Table 4.1, the error when estimating the power consumption using a data book for a small design, such as the 1bit full adder, with a small number of nodes, is 42% lower. The reason for this is that the capacitances of the interconnect cannot be included in a data book. Synopsys however, estimates the interconnect and provides the user with an estimate of the physical node capacitance including this figure for the interconnect. This estimation is also used for the timing simulation as rise and fall times depend on it.

The capacitances for the output signals SUM and CARRY_OUT cannot be taken from the data book because these nodes do not have a load. The switching probability (column 7), is simulated using UWN input vectors and is used to calculate the active node capacitance using the information provided by Synopsys (column 6) and the Library Databook (column 5) respectively. Even in such a small design, such as the 1bit full adder, containing no busses or other long connections, the error is already above 40%. With increasing design complexity the number of interconnections rises and therefore the relative error will be even larger if only the databook information is to be used. Furthermore, it should be noted that these calculations include the real node activity factors. Techniques using just the probability factors would have an even worse result and could not be used in designs with correlated inputs. Therefore, it has to be concluded that the only way to obtain an good estimate of the active capacitance and therefore achieve accurate power estimation is to use the physical node capacitances, including the interconnect, in conjunction with the node activity factor as computed for a particular input signal.

4.4 The Synopsys System Simulator

Before the operation of PowerCount is explained, the normal design cycle for high-level VLSI development using the Synopsys Design Environment [Synop] will be presented. Figure 4.3 shows the design cycle for generating a routable netlist which can be used to perform real timing simulations.

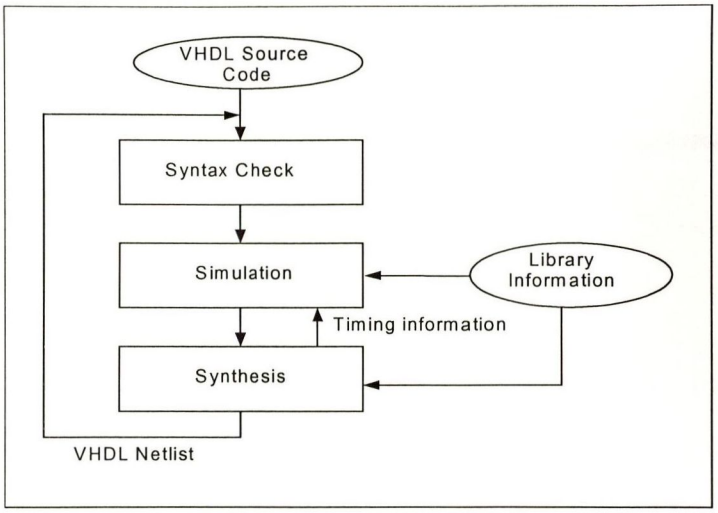


Figure 4.3: Synopsys Simulation Flow

Firstly the abstract source code is written in VHDL. After the syntax is verified, the correct logic operation of the design is tested using the VHDL System Simulator (VSS). Then the design is fed with the optimisation constraints into the Design Compiler (DC). If the design constraints are met the circuit is written out as a VHDL netlist. An additional file, containing timing information for simulation purposes is also generated. The VHDL netlist is fed back into the syntax verification. This is called back-annotation. Then the netlist is used in conjunction with the timing information by the VSS to monitor the real timing behaviour of the circuit. These simulations include rise and fall times as well as glitches and hazards.

4.5 PowerCount

This Section describes the basic concept behind the power estimation tool called PowerCount. The main feature of PowerCount is that it operates as an add-on tool to Synopsys. It uses the normal timing simulation with only one additional file. Figure 4.4 shows the interaction between PowerCount and Synopsys.

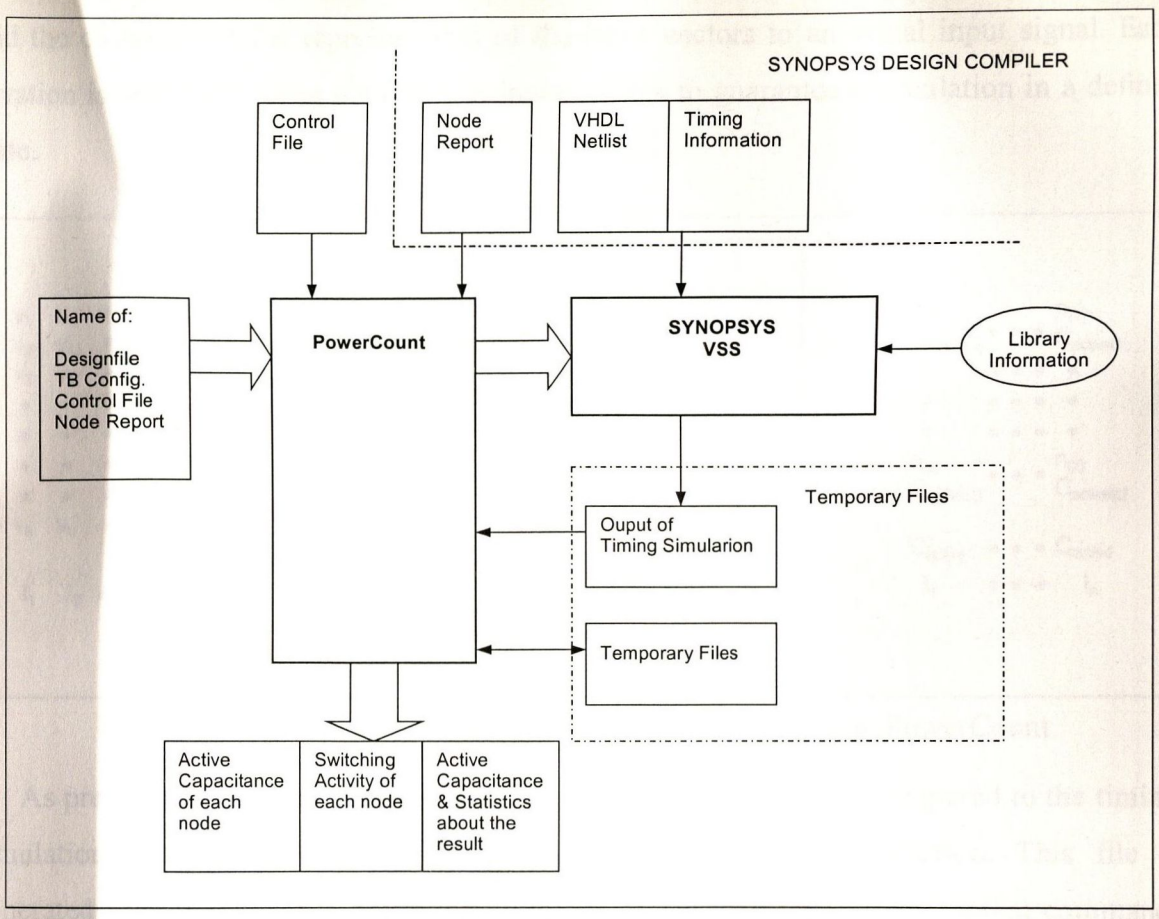


Figure 4.4: PowerCount in the Synopsys Environment

The basic idea behind PowerCount is the computation of the active capacitance as presented in (11). For this purpose, the tool uses nearly the same procedure as used for real timing simulations as explained in the previous section. The only difference is that instead of checking for the proper timing of a design during back-annotation, it automatically monitors the switching activity of each node. This principle is shown in Figure 4.5. PowerCount uses as an input a set of I_n iterations. Each iteration uses a user defined number of input vectors, v_n. Then the VSS is used to calculate the number of power consuming transitions, n, for each of z nodes. The number of power consuming events at each node is then multiplied by the physical capacitance of this node, C_{est}, as estimated by Synopsys DC. Then the sum of all active node capacitances for this iteration is calculated to provide the total active capacitance of the design, C_{active}. Finally, the average active capacitance is calculated by averaging the active capacitances of all iterations. Therefore, the tool calculates the active capacitance with the highest possible accuracy at this level. It uses the Synopsys estimated values of both the physical node capacitance as well as the real switching activity of each node by means of real input data. The accuracy only depends on the accuracy of the information of the Synopsys DC

and the closeness of the representation of the input vectors to an actual input signal. Each iteration is initialised using a number of input vectors to guarantee a simulation in a defined state.

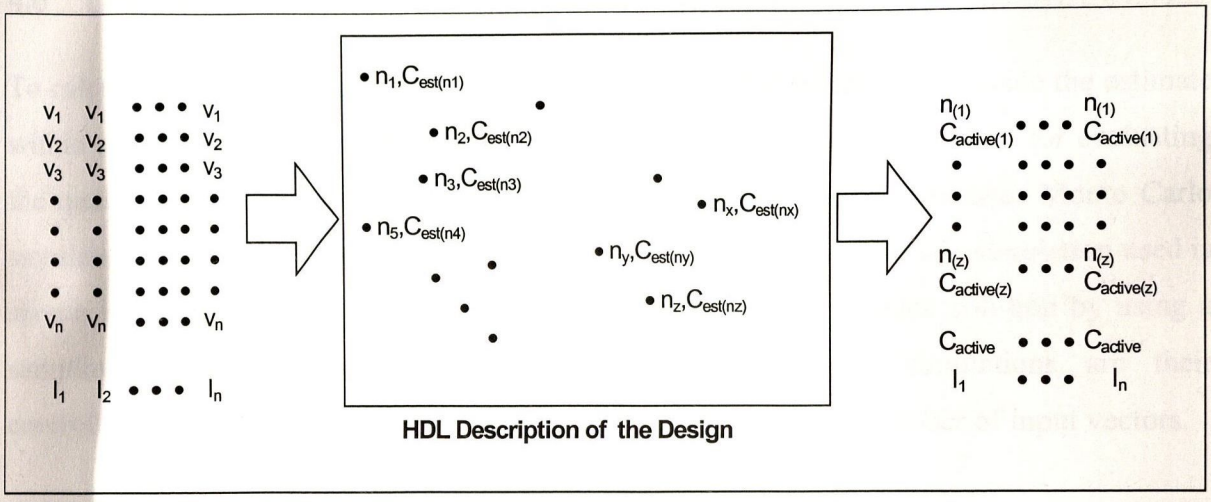


Figure 4.5: General Method of Generating an Estimate Using PowerCount

As previously indicated, there is one additional file required when compared to the timing simulation procedure. This additional file contains the node information. This file is generated after the circuit is optimised by means of the Synopsys report –netlist command. Tests have shown that such a node file with the information of more than 3000 nodes is generated in under 10s on a Sun Microsystems SPARC5 workstation with 128MBytes of RAM. Therefore, this does not significantly slow down the design process. The greatest advantage of PowerCount is that the designer does not need to spend time in preparing a special design netlist. Furthermore, it is not necessary to learn to use another tool, because PowerCount exclusively uses the VSS control language to control the power simulation. All simulation vectors are fed into the system via the normal VHDL testbench, similar to the logic and timing simulations. Therefore, it is possible to use the same test vectors as used for the timing simulations. This has the advantage that the simulation data has to be prepared only once and can be used for all high-level simulations.

PowerCount stores all information in ASCII text files. This ensures that the program runs even on small workstations without large memory resources. Furthermore, ASCII files can be easy ported between software applications on all platforms and can be visually inspected using any standard text viewer. A user-assignable directory for the temporary files makes it possible to store the information locally or on a remote hard disk. PowerCount supplies the

VSS with a 64 bit binary random number as a possible seed for further use within the VHDL testbench.

4.6 Generating the Estimate

To calculate the power consumption of an IC efficiently it is essential to provide the estimate within a short time. To provide a result within a reasonable time, a procedure for evaluating the quality of the estimated active capacitance is required. PowerCount uses Monte Carlo simulations to control the estimation. The idea underlying the Monte Carlo simulation used is shown in Figure 4.6. Monte Carlo simulations provide an approximate solution by using a sampling technique. The advantages of using Monte Carlo simulations are their controllability and their ability to compute an estimate with a small number of input vectors.

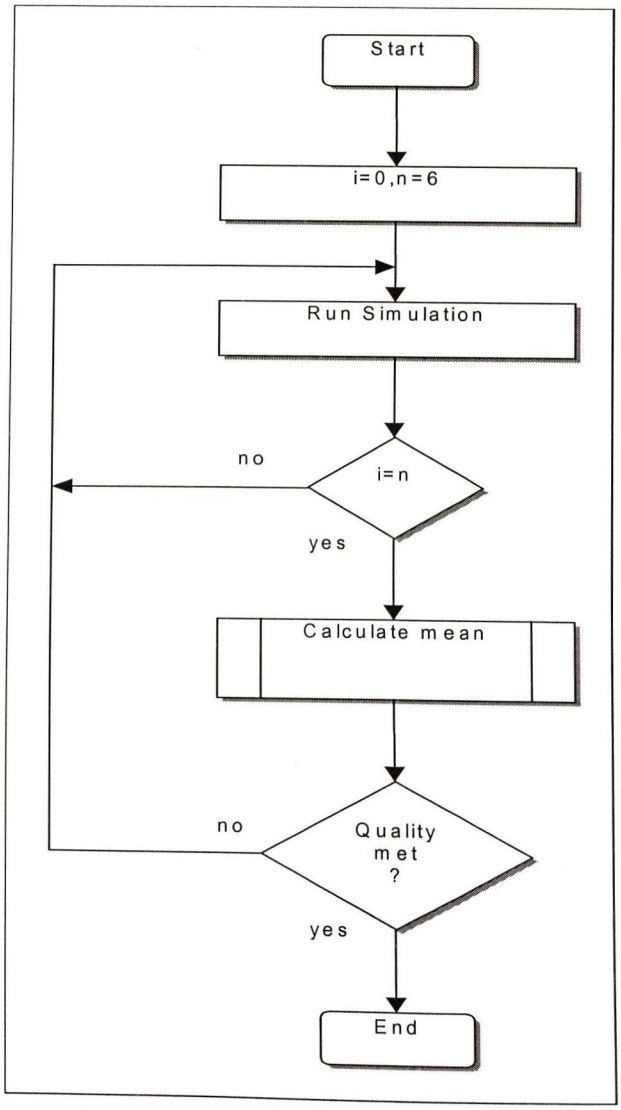


Figure 4.6: Monte Carlo Simulation

For this particular problem of estimating the active capacitance, Monte Carlo simulation is used to compute the average active capacitance using a set of iterations. The samples used to calculate the active node capacitance and the active capacitance are provided by the VSS. As illustrated in Figure 4.5, these samples are taken by iterations with a limited set of input vectors instead of taking the samples in one large set. After six iterations the quality of the estimate is evaluated using a stopping criterion [Papu90]. If the deviation of the estimated active capacitance is within a specified limit, the stopping criterion is met and the simulation terminates. If the stopping criterion is not met, additional iterations are performed until it is. Three different methods of defining a stopping criterion are discussed in the remainder of this section.

4.6.1 Method One

The first feature used to create a stopping criteria is the standard deviation. The following equation is used to determine the standard deviation [Sieg96].

$$\sigma = \sqrt{\frac{\sum \left(C_{active(i)} - \overline{C}_{active}\right)^2}{n_{iterations}}} \tag{34}$$

To determine the quality of the result σ must be related to $\overline{C}_{active}$. The calculation of the percentage error is performed as follows.

$$\varepsilon_{est} = \left| \frac{\sigma}{\overline{C}_{activee}} \right| \cdot 100\% \tag{35}$$

The stopping criterion derived by the standard deviation is noted as

$$\varepsilon_{est} \leq \varepsilon_{defined} \tag{36}$$

The stopping criterion noted in (36) is met as soon as the estimated error, ε_{est}, is either smaller than or equal to the maximum error the estimate can have, $\varepsilon_{defined}$, as defined by the user.

4.6.2 Method Two

The next stopping criterion discussed is derived from the convergence criteria (CC) for a normal distributed population. As shown in [Spie88], the sequence of jointly distributed

random variables $x_1,x_2,...,x_n$ is said to converge almost certainly or converge with probability one if

$$\lim_{n\to\infty} x_n(s) = x(s) \tag{37}$$

In this equation the theoretical actual value μ is assumed to be the mean $\bar{x}$ which is calculated from an infinite amount of samples.

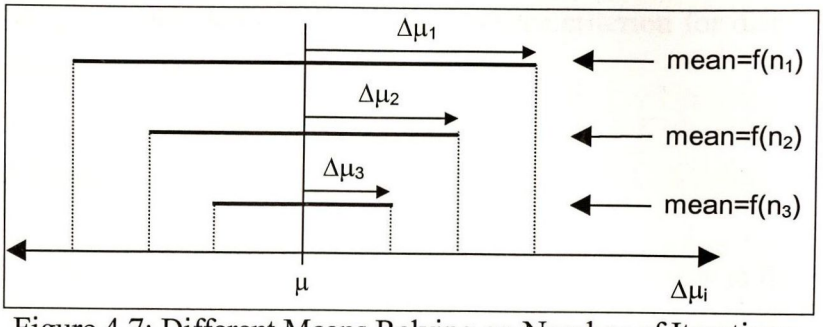


Figure 4.7: Different Means Relying on Number of Iterations

As seen in Figure 4.7, $\Delta\mu$ decreases with an increasing number of iterations, assuming $n_3 > n_2 > n_1$. The larger the number of n, the narrower the confidence level [Schw98]. Assuming $\bar{x}_n$ is almost equal to μ, (35) can be rewritten as

$$\mu \pm \Delta\mu = \bar{x} \tag{38}$$

The absolute deviation $\Delta\mu$ can be calculated by multiplying the inaccuracy factor ε by μ and is denoted as

$$\mu \pm \varepsilon \cdot \mu = \bar{x} \tag{39}$$

When allowing a small deviation it can be assumed that the theoretical value μ is calculated with one more sample than the mean $\bar{x}_n$ after a specified amount of iterations. The actual value μ and the mean $\bar{x}_n$ are then given as

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{40}$$

$$\bar{x} = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} x_{i-1} \tag{41}$$

To formulate the stopping criterion for the Monte Carlo Simulations using the CC, the relative error ε_1 can be defined as

$$\varepsilon_1 = \frac{\left| \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} x_{i-1} - \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \right|}{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i} \cdot 100\% \qquad (42)$$

To define a stopping criterion derived by the convergence criterion for distribution, ε_1 has to be compared to the maximum error as defined by the user.

4.6.3 Method Three

The main problem computing a mean with a small number of samples is the possible spread of the samples. The student's t-distribution is suitable for providing estimates accounting for the sample spread. In this case, tables containing the results of long and extensive computations for critical values are used to calculate the upper and lower limits of a convergence interval for a particular sample size. For each sample size n the critical value, T_α, can be found for a specified confidence level.

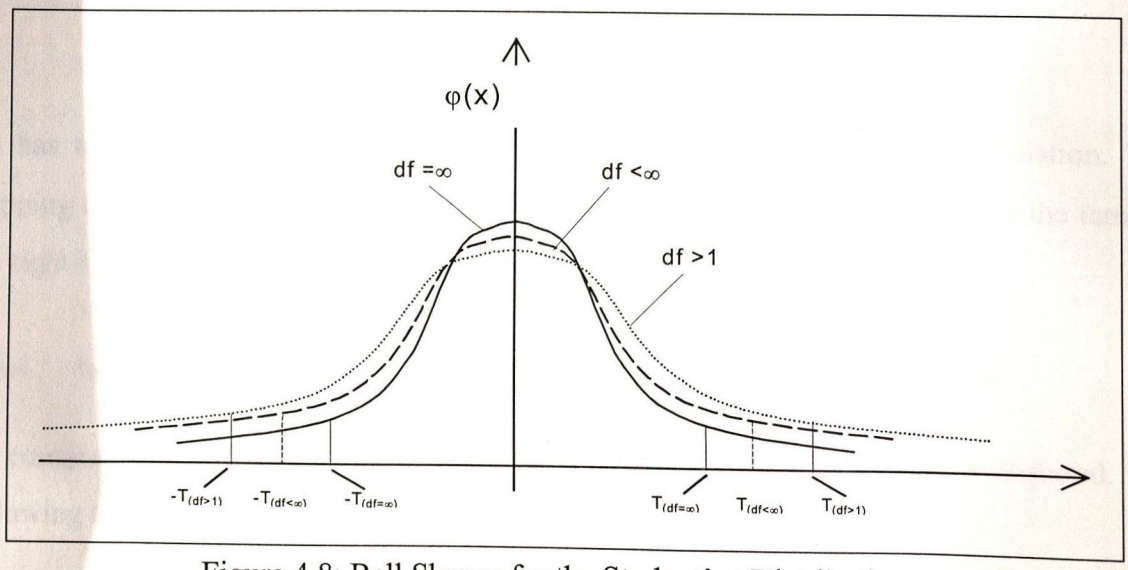


Figure 4.8: Bell Shapes for the Student's t Distributions

As seen in Figure 4.8, the bell shapes are related to different degrees of freedom (df). The df are the number of samples minus one ($df = n - 1$). The lower the degrees of freedom, the broader the spread of the bell shape will be. On the other hand for $df = \infty$ the student's t-distribution is equal to the gaussian distribution. The student's t-distribution can be used to

define a stopping criterion for the Monte Carlo simulations. The maximal accepted deviation, $T_\alpha \cdot \frac{\sigma}{\sqrt{n}}$, from the mean in (41) is substituted by ε_C. Therefore, (38) can be rewritten as

$$\bar{x} = \mu \pm \varepsilon_C \tag{43}$$

Equation (42) represents either the absolute minimum or maximum values for μ. The maximal deviation ε_C is related to the computed mean. In this case ε_C is defined as

$$\varepsilon_C = \left| \bar{x} - \mu \right| \tag{44}$$

The mean is located outside the confidence margins as long as ε_C is bigger than the right-hand term in (43). Equation (43) is then rewritten as

$$\left| \bar{x} - \mu \right| \leq T_\alpha \cdot \frac{\sigma}{\sqrt{n}} \tag{45}$$

These absolute values have to be rewritten as relative values. Therefore, both terms are divided by μ.

$$\frac{\left| \bar{x} - \mu \right|}{\mu} = \varepsilon_{rel} \leq \frac{T_\alpha \cdot \sigma}{\mu \cdot \sqrt{n}} \tag{46}$$

ε_{rel} has to be defined as an acceptable relative error before starting the simulation. The stopping criterion noted in (46) is met as soon as ε_{rel} is either smaller or equal to the term of the right hand side of (46).

4.6.4 Analysing the Stopping Criteria for Suitability in Power Estimation

To compute each of the three stopping criteria a different set of variables is required. The following table gives an overview of the required variables.

Method Three	Method One	Method Two
$\bar{x}$	$\bar{x}$	$\bar{x}$
μ	μ	μ
σ	σ	-
T_α	-	-

Table 4.2: Required Values to compute the Stopping Criteria

As seen in Table 4.2, Method Three requires the most data to calculate the stopping criterion. It requires $\bar{x}$, μ, σ and T_α to define a stopping criterion. The values for a wide range of confidence levels and degrees of freedom have to be stored. Then for the evaluation of the estimate this table must be read. Such a file operation requires extra processing time. However, as will be shown in the next section, the other two methods provide similar results, using less time consuming methods. The stopping criterion derived by Method One needs three values $\bar{x}$, μ and σ. The most time-consuming computation of all variables is the calculation of σ. As shown in (36), σ takes the deviation of each sample into account. Since μ cannot be provided a priori, it is necessary to store each single sample x_i in order to subtract it from μ. Not only is the run time required to calculate σ intensive, but the values of x_i have to be stored in temporary files because of RAM limitations. Such temporary files have to be accessed each time a file is written to or read from. This must be repeated several times. Finally, all those temporary files have to be opened to recover the required values, all of which takes time. Furthermore, the number of these files increases with the number of nodes and can therefore be quite large if complex systems with thousands of nodes are to be simulated.

Method Two requires only $\bar{x}$ and μ. Here, it is not necessary to store each single sample of x_i. Both values are simply calculated by adding the individual estimates. Bearing in mind that $\bar{x}$ is estimated with one iteration less then μ, only two variables are needed to store these values. Therefore, the run time will be faster than using one of the other stopping criteria because no disk access is performed. It should be noted that all the above statements assume that the estimate will be provided after an equal number of iterations regardless of the applied stopping criteria.

4.7 Evaluation of the Stopping Criteria

The objective of developing PowerCount is to provide a fast power estimation tool for high-level circuit testing. However, the speed of PowerCount does not only depend on its environment, such as computer performance or design complexity. Different stopping criteria, used in the Monte Carlo simulations, also influence the speed by determining the number of iterations. To prove which stopping criterion is the most efficient for implementation, four designs are investigated for power and time consumption. Thus the time required to reach the desired accuracy for the mean to converge is measured. The simulations vary in their sum of

input vectors and their stopping criteria. The stopping criteria investigated are the Student's t-distribution, the Standard deviation and the Convergence criterion for distribution, as described in the previous section. However, although the Student's t-distribution was considered as a possible stopping criterion, it was not included in this investigation because of its obvious limitations. The Student's t-distribution is only applicable for simulations with a small number of iterations (section 4.6.3). The design complexity, the sum of input vectors and also the desired convergence influence such a sum of iterations. Hence the number of iterations cannot be pre-determined as would be required by Method Three. The investigated designs are

- 1 Bit Adder
- Divider by 3
- 4 Bit 64 Stage Shifter using Multiplexer
- RGB2HSI Converter

To test PowerCount four designs are to be simulated with the following parameters

- Time base: nano seconds
- Scaling factor: 0.01
- Stopping accuracy: 1%
- Increasing sum of input vectors: 10, 50, 100, 1000
- Each iteration was initialised using 50 input vectors to guarantee a simulation in a defined state.
- Either Method One or Method Two

The time base is the basic time unit used by the VSS while the scaling factor determines how often per time base unit the design is checked for changes. In this case the VSS investigates the design all 0.01 time base units or every 10ps.

Before these four designs are thoroughly investigated the factor distinguishing the different designs is going to get discussed. The distinct characteristics of these designs are their complexities. A contributing factor to the complexity is the number of nodes. This is illustrated using a set of different adder circuits which vary in not only bitwidth and number of inputs but also in basic architecture.

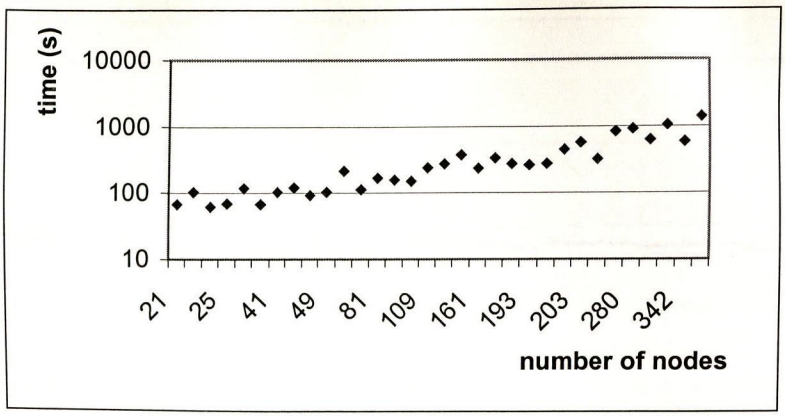


Figure 4.9: Simulation Times of Adder Structures

In Figure 4.9 the simulation times of over thirty different adder structures are plotted. This figure shows how the simulation time rises with an increasing number of nodes. Although, it is not possible to determine the simulation time by only taking the number of nodes into account, it is safe to generally assume that, the greater the number of nodes, the more complex the design, implying that the simulation is more time intensive. It might be expected that the graph would be monotonically increasing in relation to the number of nodes but it is clear that this is not the case and is due to the different level switching activity of the various designs. This means that although a particular design may have more nodes than another, if the level of switching activity is the same in two designs then the simulation time will be similar and thus the graph shape is not monotonically increasing.

Figure 4.10 shows the convergence of the 1-bit full adder which was illustrated in Section 4.1. In this figure it can be seen that after approximately 50 input vectors the estimate is permanently within ±5% of that of the value after 1000 input vectors. Therefore, it was decided to use 100 input vectors as the default value to compute the active capacitance of one iteration.

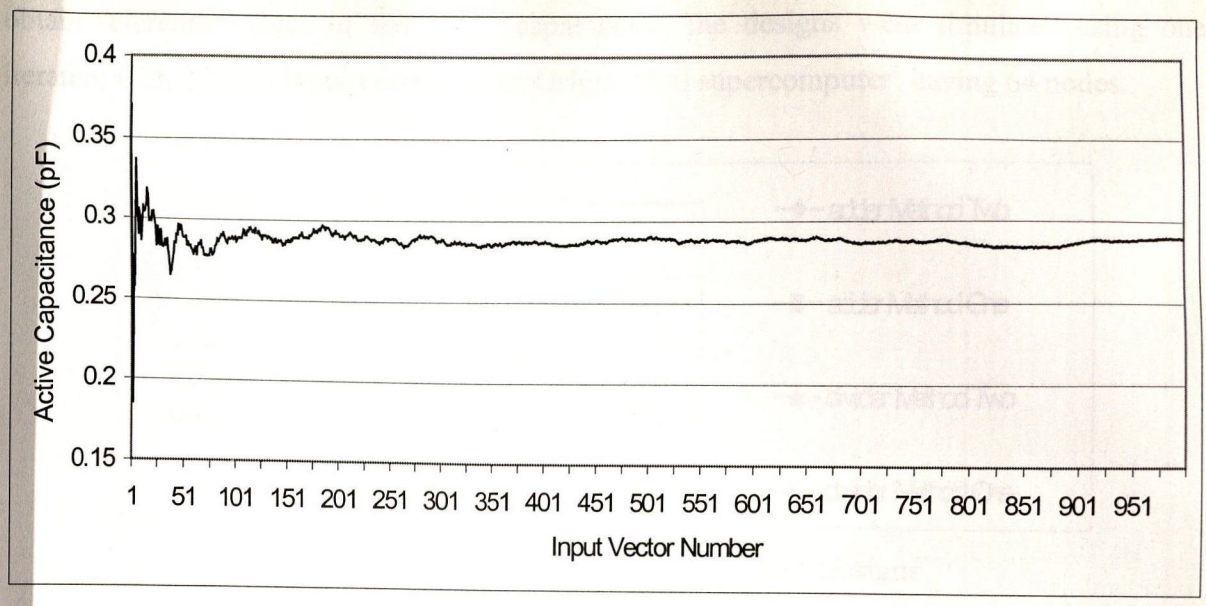


Figure 4.10: Convergence of the Simulation

Figure 4.11 shows the distribution of the active capacitance for 1000 iterations of the 1-bit full adder using 100 input vectors per iteration. As can be seen from this figure the active capacitances of the iterations are gaussian distributed.

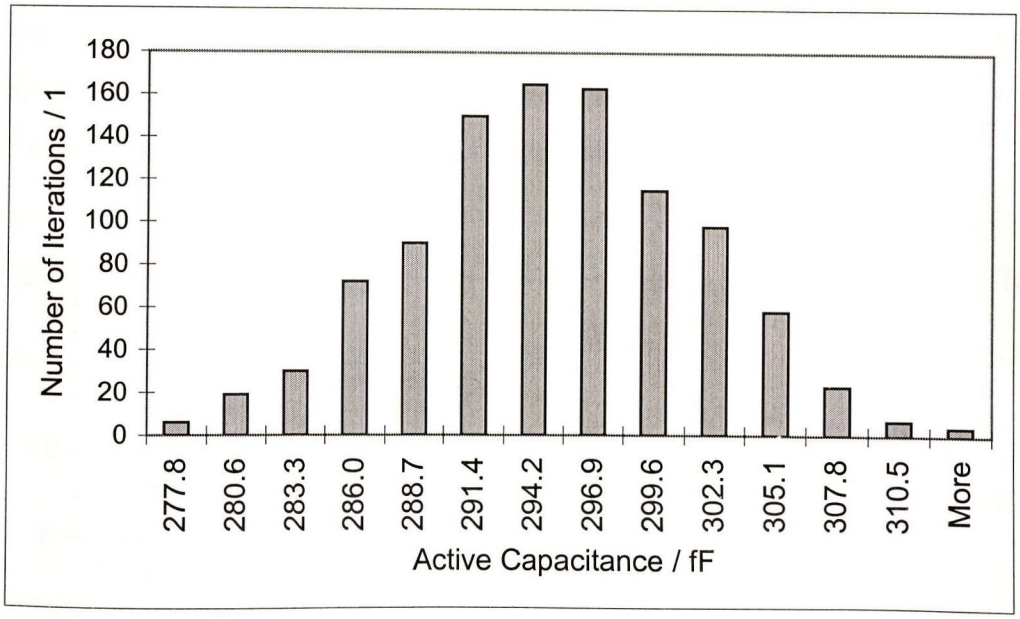


Figure 4.11: Distribution of the Active Capacitance of the Full Adder

4.7.1 Small Designs

The first simulated designs are small designs with a small number of nodes (<70). These designs are a 1-bit Adder and constant divider which divides an 8-bit input by three. Both designs are simulated with 10, 50, 100 and 1000 input vectors on a SUN Sparc5 computer. To

obtain reference values of the active capacitance, the designs were simulated using one iteration with 100000 input vectors on an Origin 2000 supercomputer[1] having 64 nodes.

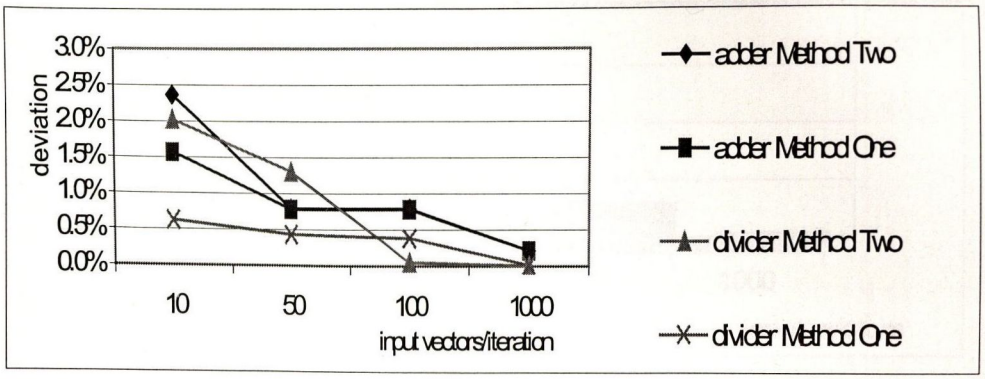


Figure 4.12: Deviations Simulating Small Designs

In Figure 4.12 the deviation of the estimate of the active capacitance in relation to the benchmark value is shown over the sum of input vectors applied. As can be seen from this graph Method One as well as Method Two have similar deviations. Furthermore, it can be seen that the deviation drops initially. For simulations with more than 50 input vectors per iteration the result remains approximately constant and with less than 1.5% more than expectable. Also, Method Two provides a slightly better estimate. However, in order to get simulation it is not enough to focus solely on the deviation of the result. It is also necessary to consider the simulation time to achieve an efficient estimate.

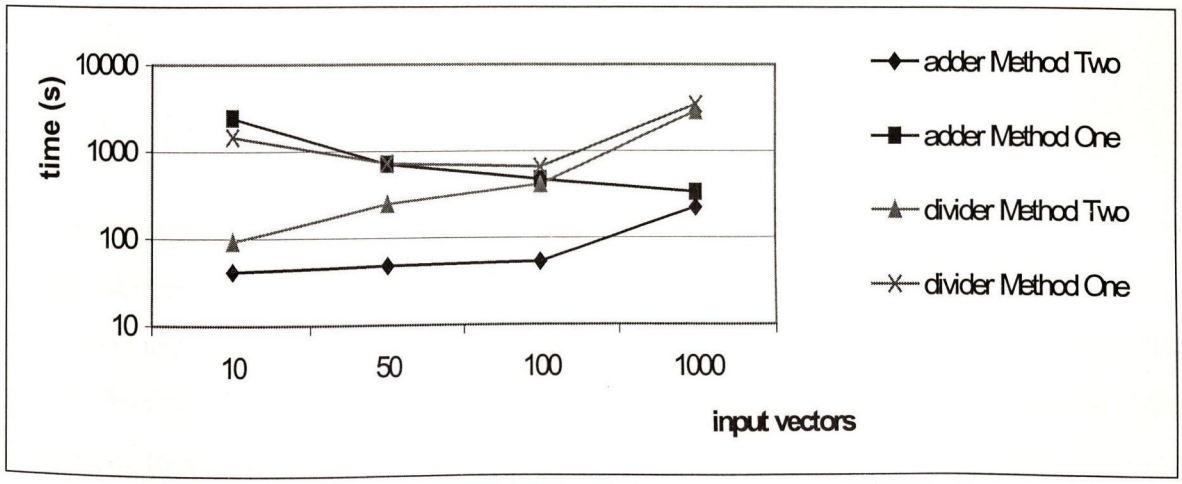


Figure 4.13: Simulation Time for Small Designs

As seen in Figure 4.13, the simulation time using Method Two is always shorter than that of Method One. When simulating designs with a small number of input vectors, >10 and <100,

[1] The high-performance computer was kindly provided by CINECA, Bologna, Italy.

the simulation time as a function of the number of input vectors, has here a minimum. To analyse this the number of iterations required has to be analysed.

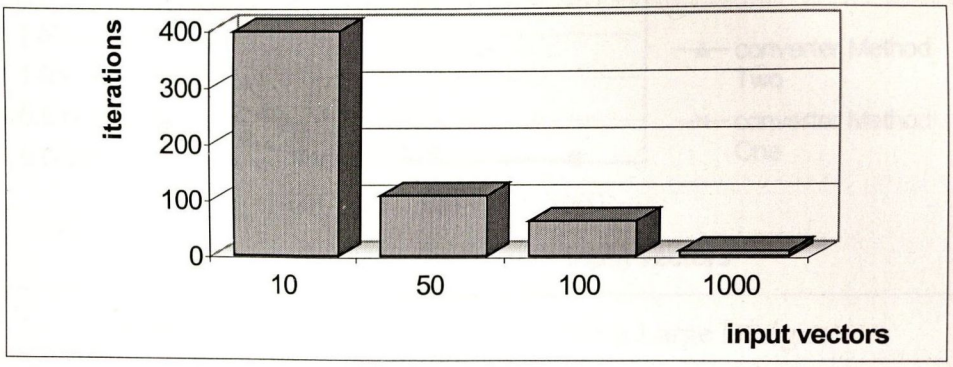


Figure 4.14: Iterations to get an Estimate using the Method One Stopping Criterion

The reason for the higher simulation times for small numbers of input vectors for the standard deviation is shown in Figure 4.14. The greater the number of input vectors, the lower the number of iterations required for the Adder design to converge to a mean. It is less time intensive to increase the number of input vectors. Thus as the number of input vectors increases the simulation becomes faster. However, the simulation is several times faster using Method Two. For example, the simulation time simulating the Adder structure with 50 input vectors and Method Two is 14 times faster. However, as seen in Figure 4.13 for more than 100 input vectors the time required starts to increase again due to the larger amount of input vectors to be simulated. Therefore, it can be concluded that for small designs the optimal input vector size is between 100 and 50 to achieve a fast result.

4.7.2 Large Designs

The next simulated designs are larger circuits with at least 700 nodes. The designs are a 4 bit 64 stage shifter using multiplexers (Mux) and a Converter. The multiplexer is simulated with 10, 50, 100 and 1000 input vectors. The Converter, which is an image processor, is simulated with 10, 50 and 100 vectors on a SUN Sparc5 Workstation. It was not simulated with higher amounts of input vectors as the results of such time consuming simulations were irrelevant having analysed the simulation results using up to 100 input vectors. The reference values for the active capacitance of the two designs were obtained through a long simulation with 100,000 input vectors for the Multiplexer and 10,000 for the Converter on an Origin 2000 Supercomputer.

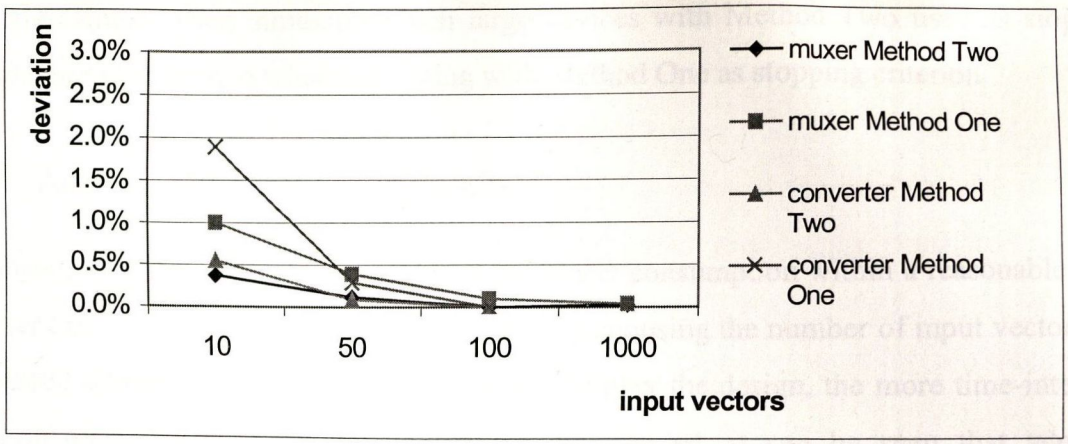


Figure 4.15: Error Deviation Simulating Large Designs

In Figure 4.15, the deviations from the mean over the sum of input vectors are plotted. The graphs represent the deviations from the reference values. The deviations from the mean, computed with Method Two, are already smaller by simulating with 10 input vectors than those obtained using the standard deviation. In these designs the accuracy, which is the reciprocal of the deviation, increases by increasing the sum of input vectors. Also for large designs it can be seen that the deviation drops significantly for more than 50 input vectors. In order to evaluate the stopping criteria, the simulation times are investigated once again.

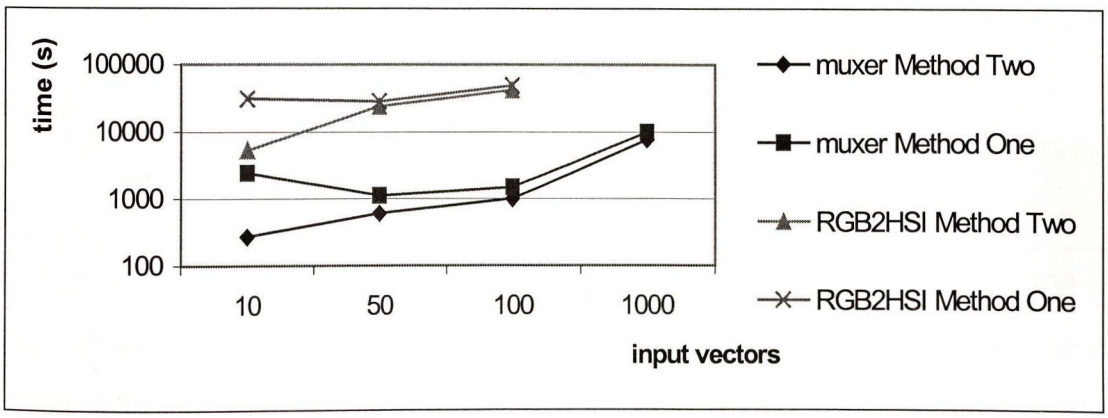


Figure 4.16: Simulation Time for Large Designs

As seen in Figure 4.16, the simulation time for the standard deviation decreases by increasing the number of input vectors from 10 to 50. By increasing the sum of input vectors from 50 to 100, the function course remains nearly constant before rising again for higher input vector sizes. This behaviour is as already described for small designs related to the number of iterations required to meet the stopping criterion. The simulation time using Method Two rises approximately linearly, with the amount of input vectors. The graph for the simulation time of large designs is similar to the graph for small designs (Figure 4.13). In general, the

simulation times, when simulating such large devices with Method Two used as stopping criterion, are smaller than when simulating with Method One as stopping criterion.

4.7.3 Analysis of the Simulation Results

The objective of PowerCount is to estimate the power consumption within a reasonable time. The user can influence the speed of PowerCount by choosing the number of input vectors and the desired accuracy of convergence. The more complex the design, the more time-intensive the simulation. However, in all the designs investigated, it can be seen that adequate estimation time and accuracy are achieved by simulating the system with 50 input vectors and using Method Two as a stopping criterion. Method Two also always requires the least number of iterations to compute the average active capacitance. The stopping criterion derived from Method Two always achieves a faster result than simulation using Method One. The reasons for this have been discussed in section 4.7.

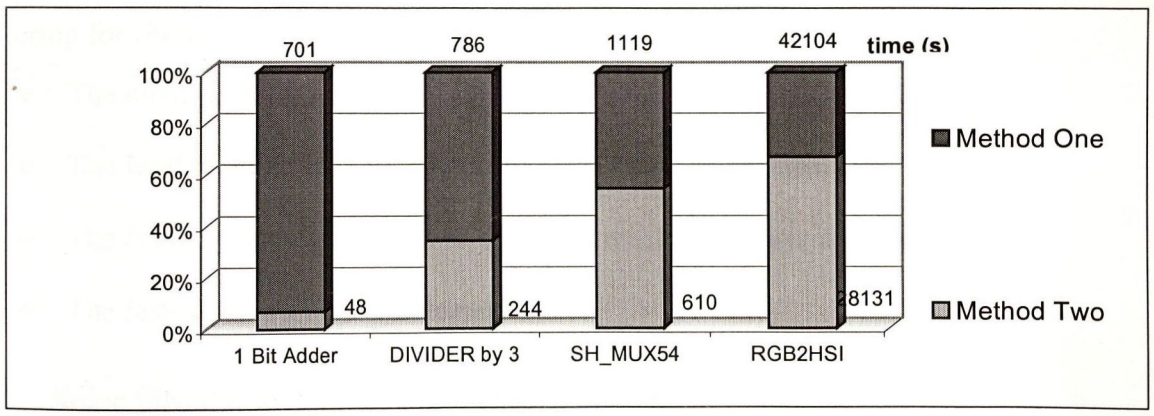


Figure 4.17: Comparison of Running Times of the Investigated Stopping Criteria

Figure 4.17 compares the time of convergence of the two stopping criterion investigated. As has been shown both archive similar results. However, Method Two is always faster. In Figure 4.17 the Method One is set as the reference value to show the time advantage of Method Two in percent. Simulating a rather small design, such as the 1-bit adder with Method Two is 93% faster than simulating with the Method One as stopping criterion. Also, the Divider structure, simulated with Method Two, yields a result that is almost 70% faster than simulating with the other stopping criterion. The SH_MUX54 simulated with Method Two as stopping criterion requires 45% less simulation time than Method One. Finally the Converter which can be assumed to be a complex design, (> 1000 nodes), also needs 33% less simulation time when simulated with Method Two.

Simulations of complex designs such as SH_MUX54 or the Converter converge after the minimum amount of iterations to estimate the active capacitance. Thus, the run time of the tool is now the only factor that determines the simulation time. As seen in equations (18) and (19), it is necessary to store every single computed value when simulating with Method One as the stopping criterion. This requires temporary files for every single node. Every single calculated value for each iteration has to be saved in a separate temporary file. In the worst case, all the temporary files have to be examined several times in order to compute the stopping criterion. Finally, all the temporary files are deleted.

Calculating the stopping criterion using Method Two has an additional advantage. Bearing in mind (52), the mean value $\bar{x}$ and the actual value μ are computed by accumulating the values and dividing their sum by the number of iterations; it is not necessary to generate additional temporary files. The value for the actual value μ and the $\bar{x}$ mean can be easily stored in two variables. This method therefore requires much less computation and storage performance. Therefore, it is also less time intensive. Method Two is chosen as a stopping criterion for Monte Carlo simulations because of its obvious advantages:

- The most accurate results are achieved although less iterations are required
- The least requirements for computation and storage performance
- The fastest stopping criterion to achieve a result in a reasonable time
- The fastest algorithm to implement

4.8 Spice Simulations

To validate the operational correctness of PowerCount, layouts of three designs were generated and the power consumption was determined using Spice simulations. To generate the layouts, the VHDL netlists of three designs were converted into Verilog netlists using X-HDL [Xhdl00]. The Microwind [Micr00] layout tool was then used to generate the physical layout of the design and to verify the correct operation. Figure 4.18 shows the layout of the 1-bit full adder using the ES2 0.7μm technology adapted throughout this thesis.

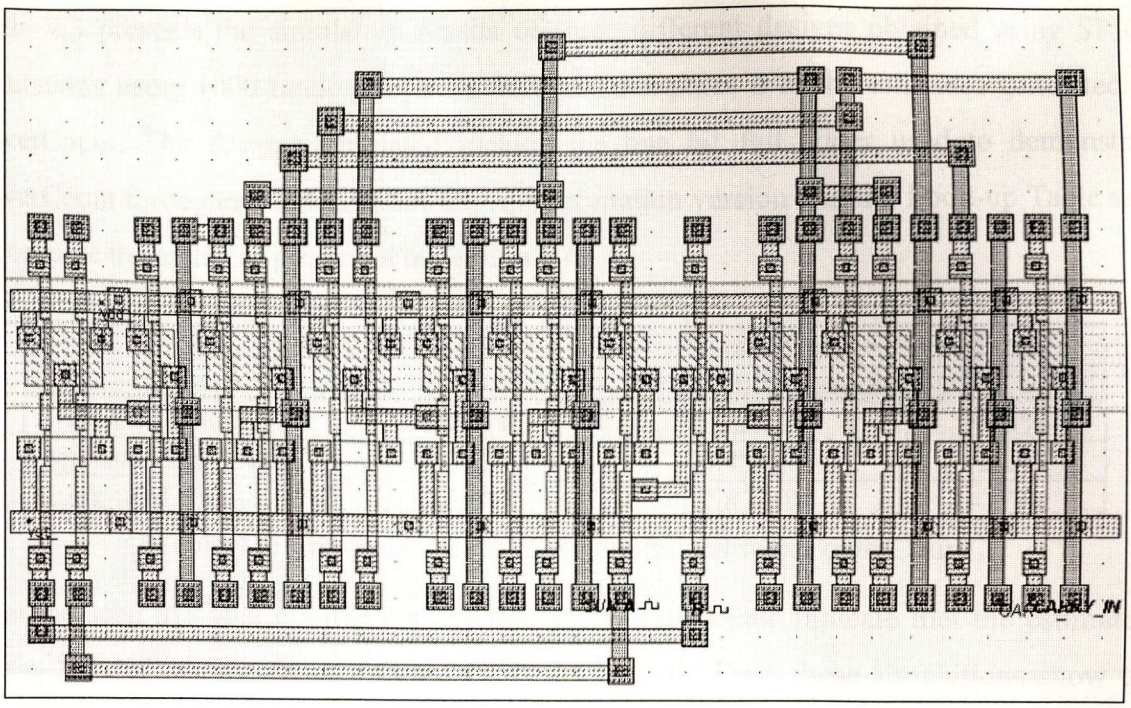


Figure 4.18: Layout of the 1-Bit Full Adder

Having laid out the circuit, a SPICE netlist was generated to perform a power analysis. Figure 4.19 shows the current simulation of the 1-bit full adder with 1000 random input vectors which were applied at a frequency of 30MHz and a supply voltage of 5V. As can be seen, the graph converges to a value of approximately 41μA after 10μs.

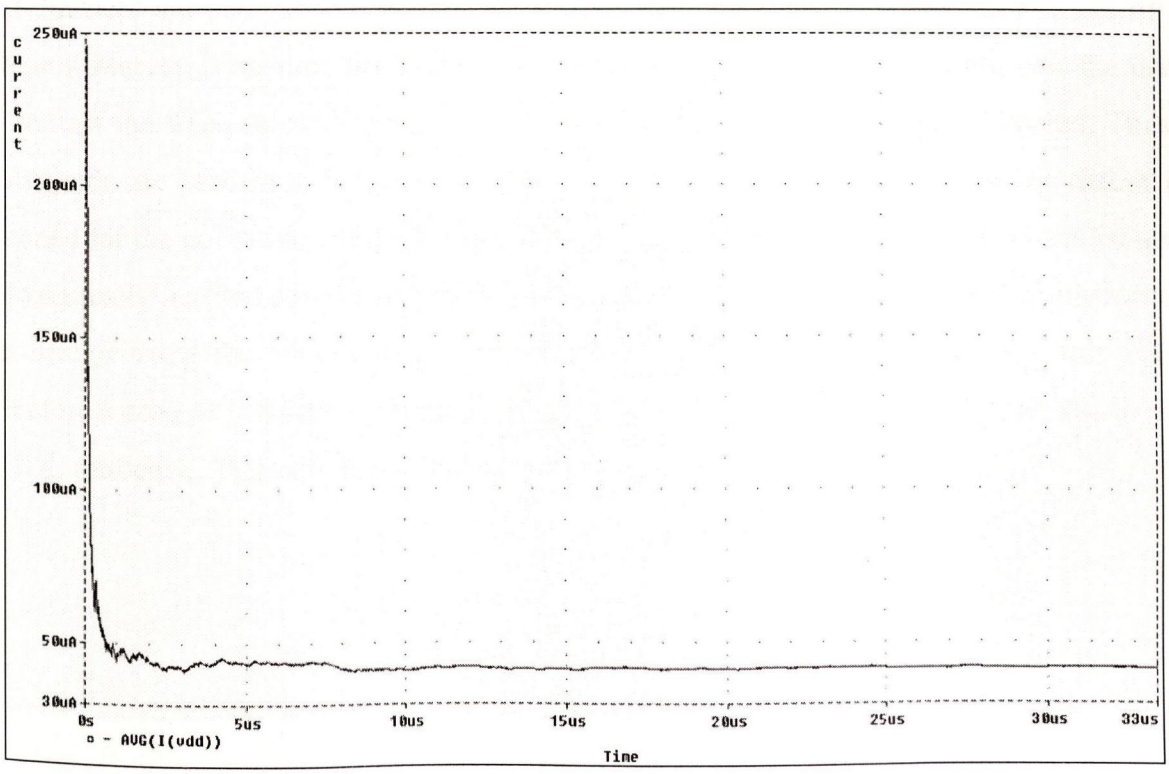


Figure 4.19: Current Simulation of the 1-Bit Full Adder

Table 4.3 presents the simulation results of three different designs obtained using SPICE simulations using 1000 random input vectors and compares it to the estimates generated by PowerCount. The designs simulated include the one bit full adder used to demonstrate PowerCount throughout this chapter, the Approximation version and the Look-up Table used to compute the arctan as presented in Section 5.4.

Design	SPICE Power Consumption	PowerCount Power Consumption	Error
1-Bit Full Adder	0.188mW	0.221mW	+18%
Approximate Arctan	1.53mW	1.49mW	-2.6%
Look-up Table	3.45mW	2.87mW	-17%

Table 4.3: Comparison of SPICE Simulations with PowerCount

It can be seen in Table 4.3 that the errors of the designs tested indicate that the estimate is within 20% of that of obtained from SPICE simulations. Thus, these simulations show that PowerCount gives sufficient results for the design work undertaken in this thesis.

4.9 Theory of Operation

This section shows the general cycle of a power estimation using PowerCount. Figure 4.20 illustrates the operation of the program. After starting, PowerCount checks for a valid licence. The licences are node locked, but can be stored in a single file. This enables the use of a single fileserver. If no valid licence is available the program terminates and prompts the user to contact the VLSI research group. Next a set of UNIX system commands is created. These commands are used to start Synopsys or to generate temporary folders. Then the netlist is scanned for the nodenames of the design. These nodenames are then stored in a linked list and automatically inserted into the controlfile. The controlfile is used to control the simulation of the design using the VSS. It is not possible to hand over a random seed into the VSS, therefore a seed of 128 bits is generated and included into the controlfile for further use in the VHDL testbench. This ends the initialisation process.

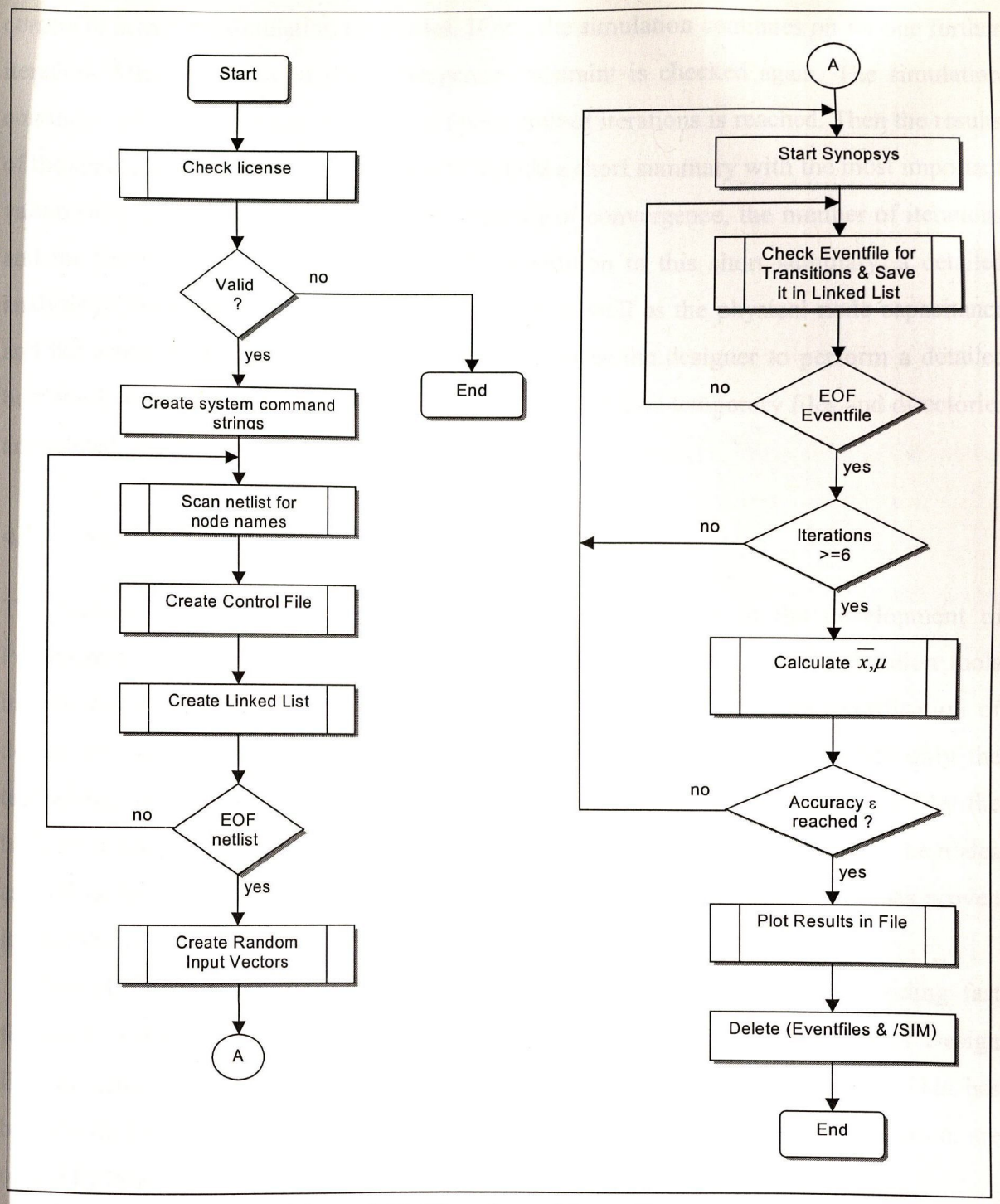


Figure 4.20: Flowchart of PowerCount

Next VSS is started and a file with all events at all nodes to be monitored is generated. This file is called the eventfile. At the end of the simulation, the eventfile is analysed and the number of transitions at each node is stored. Then, the next simulation is started and the event file is analysed again. This is repeated six times. After the sixth time PowerCount calculates the average active capacitance according to (11). Furthermore, μ is calculated and the convergence criteria is used to determine the confidence of the result. If the convergence

constraint is met the simulation terminates. If not, the simulation continues on for one further iteration. After this iteration the convergence constraint is checked again. The simulation continues until the constraint is met or an upper limit of iterations is reached. Then the results of the simulations are printed. These results include a short summary with the most important values such as the active capacitance, the accuracy of convergence, the number of iterations and the time required to achieve the result. In addition to this short summary, a detailed analysis of the number of transitions at each node, as well as the physical node capacitance and the active node capacitance, is plotted. This enables the designer to perform a detailed bottleneck analysis of a design. After the results are plotted, all temporary files and directories are deleted and the program terminates.

4.10 Summary and Conclusions

The requirement for fast feedback to the designer has resulted in the development of PowerCount, a novel power estimation tool. Evaluation of existing power estimation tools has indicated limitations with regard availability of technologies and early verification of design decisions. PowerCount overcomes these limitations by incorporating not only the technology library information, but also the additional design information provided by the high-level design environment. This extra design data comprises the interconnect of the nodes as well as the switching activity at those nodes. Extensive usage of PowerCount has proven its usefulness in a variety of low-power design projects.

PowerCount has advantages over traditional power estimation tools by providing fast feedback to the designer. PowerCount operates as an add-on tool to the Synopsys Design Environment, which is the system of choice of the majority of ASIC designers. This has benefits for the designer because design files, previously generated for timing simulation, are reused by PowerCount without modification.

PowerCount operates at the VHDL netlist level. Working as an add-on tool to Synopsys it does not require any special libraries. It can be easily controlled via a command line interface, making use of standard default values of parameters such as the number of input vectors. PowerCount is not limited by design size. PowerCount uses the native Synopsys control language. Therefore, the designer does not need to learn another language to control the power estimation. Furthermore, PowerCount uses the same library information as used by Synopsys for synthesis and timing simulations. For this reason, whenever the technology

library changes, PowerCount will automatically use these new libraries. Therefore, it is easy for the designer to become accustomed with this new tool in a familiar environment.

PowerCount has been tested extensively and no bugs are known. It has been developed so as to be error tolerant towards such errors as the absence of a netlist and access denial to the Synopsys Simulator (i.e. if all licenses are in use). However, if an error occurs which PowerCount cannot overcome, a file is created containing error information. Using this file, the user can quickly determine the source of error.

The basic concept of PowerCount is based on the use of the node capacitances which include an estimate of the interconnect capacitance. Furthermore, PowerCount computes the switching activity factor for each node. To generate a fast estimate, Monte Carlo simulation has been incorporated into PowerCount. The advantage of Monte Carlo simulation is that it provides a fast result with a definable confidence interval.

Extensive benchmarking has been carried out. This has enabled the incorporation of default values for the desired confidence interval, the number of input vectors and the number of iterations. These values have been verified by the use of PowerCount in several design projects. In conclusion, PowerCount has demonstrated its usefulness to the designer, by providing fast power estimation at an early stage of the design cycle.

Having described the background to power consumption and the development of a novel power estimation tool, the remainder of this thesis focuses on the implementation of an image processing algorithm. For this purpose, the RGB to HSI algorithm is decomposed into two functional units. The next Chapter describes the implementation of Kender's algorithm for faster computation of hue, while Chapter 6 deals with the implementation of the saturation and intensity path.

5 The Hue Algorithm

The objective of this thesis is the application of low-power design techniques at the algorithmic level of a design. For this purpose, Kender's algorithm of faster computation of hue was chosen [Kender]. This algorithm provides a number of rigorous design challenges such as the implementation of trigonometric functions, multiplication by fixed coefficients and fast divisions.

This Chapter deals with the implementation of Kender's Algorithm, which is the most complex part of the RGB to HSI conversion. To provide a more detailed, functional analysis the algorithm is decomposed into a number of blocks. Low-power implementations for each block are proposed and evaluated. Kender's algorithm (1) is shown once again, to provide a basis for comparison of the implementations discussed in this Chapter.

if ((R > B) and (G > B)) (1.1)

$$hue = \frac{\pi}{3} + \arctan\left(\frac{\sqrt{3} \times (G - R)}{G - B + R - B}\right)$$

else if (G > R) (1.2)

$$hue = \pi + \arctan\left(\frac{\sqrt{3} \times (B - G)}{B - R + G - R}\right)$$

else if (B > G) (1.3)

$$hue = \frac{5 \times \pi}{3} + \arctan\left(\frac{\sqrt{3} \times (R - B)}{R - G + B - G}\right)$$

else if (R > B) (1.4)

$$hue = 0$$

else (1.5)

'achromatic'

Firstly, the partitioning of the hue algorithm is explained. Figure 5.1 presents the breakdown of the hue algorithm into functional blocks. These blocks contain parts of the algorithm which can be implemented separately with little or no effect on the power consumption of the other blocks in the overall structure. In the following sections, each block is described and various implementations are compared. The first block performs the comparisons necessary to determine which of the five cases of Kender's algorithm is true. The second stage calculates the divisor and dividend required to compute the argument of the arctan. The third stage performs the division. The fourth module computes the arctan and the final stage adds the coefficient to the result of the arctan. An additional stage for delaying control information is also included in the bock diagram.

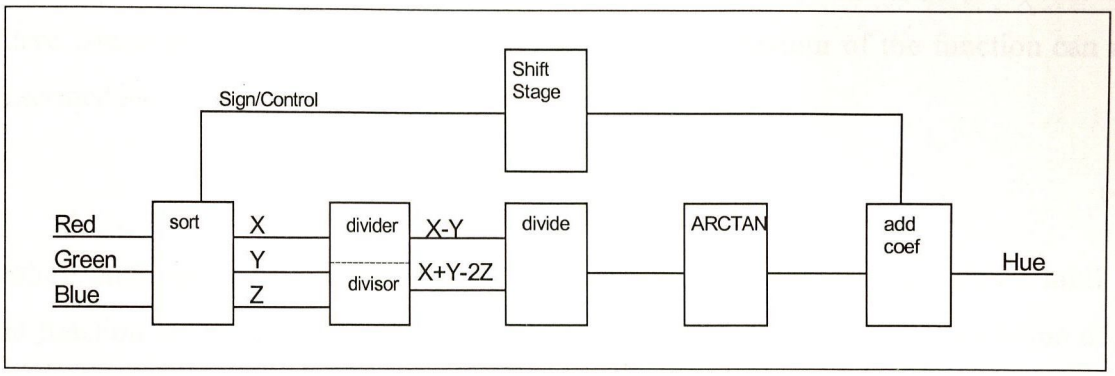


Figure 5.1: The Breakdown of the Hue Algorithm into Components

5.1 Comparing the Input Vectors

The first module of the hue algorithm performs the task of detecting which part of Kender's algorithm for the faster computation of hue is true. Figure 5.2 shows the block diagram of this module. The control signal is not included in this figure as it varies depending on the implementation.

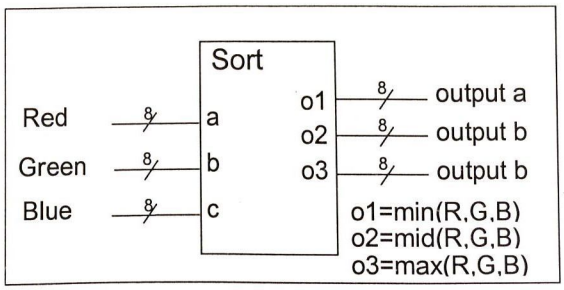


Figure 5.2: The Block Diagram of the Comparison Module

5.1.1 The Sorting Algorithm

As shown in section 3.4, it is most desirable to implement designs using unsigned arithmetic. The key to the implementation of an unsigned version of Kender's algorithm is the understanding of the argument of the arctan function. The arctan function appears in the first three parts, (1.1), (1.2) and (1.3), of the hue algorithm and can generally be written as follows:

$$\arctan\left(\frac{\sqrt{3}\times(X-Y)}{X-Z+Y-Z}\right) \tag{47}$$

In this equation Z represents the smallest of the three input signals. Therefore, it can be seen that the argument of the arctan function only becomes negative if the term *(X-Y)* becomes negative. Since the arctan function is an odd function, the behaviour of the function can also be described as:

$$\arctan(x) = -\arctan(-x) \tag{48}$$

Therefore, the sign can be excluded from the subtraction of X and Y and stored until the arctan function is computed. It is then used as the sign of the result of the computation of the arctan. In traditional designs, the determination of the sign is achieved by the use of a sign-magnitude subtractor (Figure 5.3). As can be seen from this discussion, the operation *(X-Y)* plays a vital part in the design decision of how to implement the sorting algorithm. Therefore, the dividend of the argument of the arctan will be included in this section.

The first implementation of this stage uses a sign magnitude adder. These adders are traditionally built using a comparator which is connected to the input of an adder. This, however, has the disadvantage that all the input bit signals have to be compared in order to determine the larger number. Then the smaller number is subtracted from the larger one. Therefore, a different approach is presented in this section.

The first two inputs, R and G, are subtracted using a two's complement subtractor. Now only the MSB has to be checked. If it is 0 the difference is positive and no further computation is required. If the MSB is 1 the result is negative. Now it is inverted and a ONE is added to give the positive equivalent. This procedure has the advantage that only one bit has to be checked. Furthermore, only for half of the output range is an additional computation is required. Figure 5.3 shows the principle approach to this implementation.

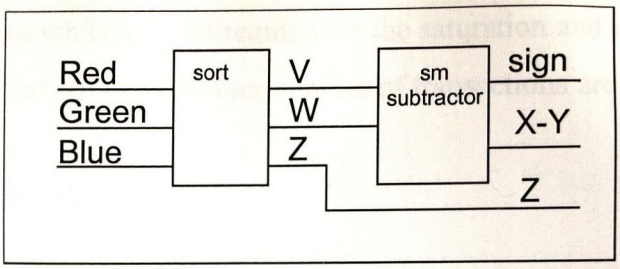


Figure 5.3: The Traditional Implementation Performing the Sign Detection

A different method is shown in Figure 5.4. Here two comparison stages sort the three input words according to their magnitude and then subtract the second largest signal from the largest one.

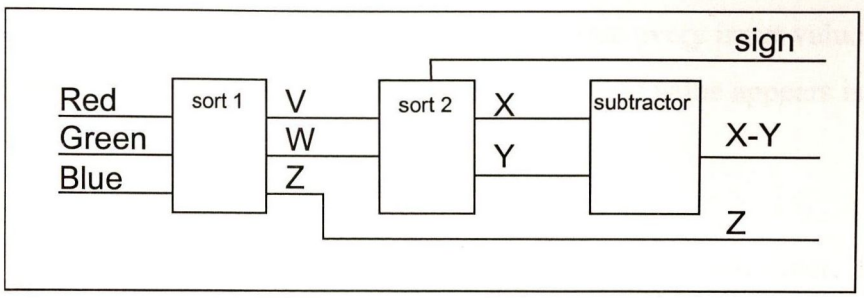


Figure 5.4: Extracting the Sign using the Comparators

This has the advantage that an ordinary unsigned subtractor can be used instead of a larger and more complex sign magnitude subtractor. A further advantage is that the sign is extracted at the earliest possible stage and therefore it is possible to reduce the signal bus size by one bit. This results in a smaller subtractor in the next stage. As the traditional approach requires a smaller comparator and the second proposed method requires a smaller subtraction stage, both implementations were considered. The results of both circuits are shown in Table 5.1.

	Area	Timing	Active Capacitance
Sorting	0.391mm^2	16ns	12.4pF
S-M Subtractor	0.404 mm^2	16ns	13.3pF

Table 5.1: Key Features of the Comparison Module

As seen in Table 5.1 the Sorting algorithm is the preferable implementation as the active capacitance is 7.3% smaller than that of the sign magnitude subtractor.

If one of the last two cases of Kender's algorithm is true, then the value of hue needs not to be computed because it is already determined. Therefore, only the control bus may change

value. In this case, stages which are not required in the saturation and intensity algorithms are disabled. This ensures that only a minimum number of transactions are performed.

5.1.2 The Encoder Block

The second part of the Sort block is the encoder block. The function of this block is to determine which of the 5 cases of Kender's algorithm is to be applied to the data. The output of this block is then send to the control bus. The order in which the 5 different cases are examined is based on the probabilities of occurrence associated with them. The calculation of these probabilities is founded on the following reasoning: Knowing that each of the three input signals R,G,B has 256 possible values and assuming that every input value has the same probability [Schw99], then the probability that a particular input value appears is

$$P_{\text{fixed input value}} = 1/256 = 0.391\% \tag{49}$$

An achromatic value appears if all three input values are equal to each other. There are 256 different possibilities leading to the achromatic case.

$$P_{\text{case 5}} = P_{\text{Achromatic}} = 256 * P_{\text{fixed input value}} * P_{\text{fixed input value}} * P_{\text{fixed input value}} = 0.00153\ \% \tag{50}$$

The probability of any of the 4 remaining Kender-cases is therefore

$$P_{\text{Case 1 to 4}} = P_{\text{Case 0; 1/3*}\pi\text{; }\pi\text{; 5/3*}\pi} = 1 - P_{\text{Achromatic}} = 99.99\ \% \tag{51}$$

For the cases 1 to 4 the following rules are valid

Condition	Kender case
Green > Red and Blue ≥ Red	2
Blue > Green and Red ≥ Green	3
Red > Blue and Green ≥ Blue	1 or 4

Table 5.2: Conditions for Kendercases 1 to 4

The three conditions in the next equation have the same probabilities:

$$P_{\text{Case 2}} = P_{\text{Case 3}} = P_{\text{Case 1}} + P_{\text{Case 4}} = P_{\text{case 1 to 4}} / 3 = 33.3\ \% \tag{52}$$

If the third condition (Red > Blue and Green ≥ Blue) is true, Kender-case 4 only appears if the value of green is equal to the value of blue.

$$P_{Green = Blue} = P_{Red = Blue} = P_{Red = Green} = 256 * P_{fixed\ input\ value} * P_{fixed\ input\ value} = 0.391\ \% \quad (53)$$

$$P_{Red \neq Blue} = 1 - P_{Red = Blue} = 99.6\ \% \quad (54)$$

$$P_{Red > Blue} = (P_{Red \neq Blue} / 2) = 49.8\ \% \quad (55)$$

$$P_{Case\ 4} = P_{Red > Blue} * P_{Green = Blue} = 0.195\ \% \quad (56)$$

$$P_{Case\ 1} = P_{Case\ 1} + P_{Case\ 4} - P_{Case\ 4} = 33.1\ \% \quad (57)$$

In Table 5.3, the probabilities of the individual Kender cases are shown:

Kender case	Probability
1	≈ 33.1 %
2	≈ 33.3 %
3	≈ 33.3%
4	≈ 0.195 %
5	≈ 0.0015 %

Table 5.3: Probabilities of the Kender-cases

As seen in Table 5.3, the probabilities for the first three cases are approximately equal to each other. The last two cases have a significantly lower probability. This leads to the conclusion that the Kender-cases should be investigated directly in the order of appearance in (1). Furthermore, analysis has shown that the transmission of a positive sign has the same probability as the case that a negative sign is transmitted [Schw99].

5.2 Computing the Argument of the Arctan

This block computes the dividend and divisor of the argument of the arctan. Because of this, the block is split into two sections. The first section computes the dividend of the argument of the arctan according to equation (58) and the second deals with the computing of the divisor equation (59) of the argument of the arctan. The computing of the dividend is closely linked to the implementation of the comparator structure. Because of this, the discussion of this topic

has been included in the previous section. However, due to implementation concerns, the actual implementation was placed in the same stage as the computation of the divider of the arctan. Therefore, this is included in the block diagram (Figure 5.5) of this module without further discussion, and this section focuses on the implementation of the divisor of the argument of the arctan. Figure 5.5 shows the block diagram of this module.

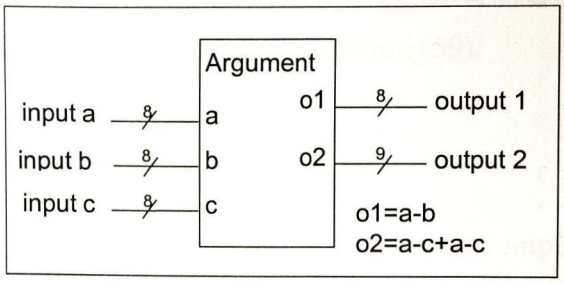


Figure 5.5: The Block Diagram for Computation the Argument of the Arctan

The divisor of the argument of the arctan requires, if implemented directly as described in equation (47), three modules, namely one adder and two subtractors.

$$divisor.arctan=(X-Z)+(Y-Z) \tag{58}$$

Such an equation can be implemented using a tree structure. This can be achieved by first computing the two sub-terms *(X-Z)* and *(Y-Z)*. Then the addition of the two sub-terms is performed. This structure is balanced and has therefore a low glitching activity. However, the divisor can also be expressed in a general form as follows

$$divisor.arctan=X+Y-2Z \tag{59}$$

This equation can be implemented with only two combinatorial modules. The multiplication by two is a simple shift operation and does not require any logic elements. Table 5.5 presents the key measures of both implementations of the computation of the divider of the argument of the arctan.

	Area/mm^2	Number of Nets	Timing/ns	Active Capacitance/pF
Tree Implementation	0.13	106	13.64	1.75
Small Implementation	0.098	85	13.41	4.68

Table 5.4: Key Features of the Computation of the Divisor for the Arctan

Usually, larger structures have higher power consumption than smaller designs. However, as seen in Table 5.4, structures such as adders or subtractors do have a large glitching activity. As already described previously in Chapter 3 the glitching activity can be reduced by balancing all paths of a module. The tree structure used is such a fully balanced design. This is the reason why in this case the larger design has a lower active capacitance. Therefore, for the hue implementation the tree structure is going to be used, despite the fact that it is 32% larger than the smaller implementation based on equation (59).

5.3 The Divider Structure

The next module in the hue path is a divider. This divider computes the argument of the arctan. Figure 5.6 shows the block diagram of the divider module indicating the input and output ports.

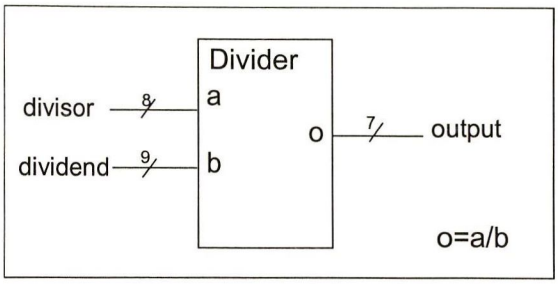


Figure 5.6: The Block Diagram of the Divider Module

As seen in (1), the divisor contains a multiplication by $\sqrt{3}$. A discussion of this factor will not be included in this section as this is a scaling factor for the arctan function. Therefore, this multiplication will be considered in detail in the next section, which describes the implementation of the arctan function. Before the actual implementation of the divider is described, the general background of a binary divider will be explained.

When dividing binary numbers, the same principle applies as when dividing decimal numbers. First the dividend is compared with the divisor. If the dividend is smaller than the divisor, the MSB is zero. Then the dividend is shifted by one bit to the left and the comparison is performed again to compute the (MSB-1). If the divisor is larger than the dividend, the dividend is subtracted from the divisor and the corresponding output bit is set to one. Then the resulting value is shifted. Therefore, each divider stage has to contain a comparator as well as a subtractor. Due to this highly repetitive procedure, dividers are often implemented in a loop procedure in order to minimise the area. However, as outlined

previously, the design has specification requirements, regarding the pixel throughput, of 30Mpixels per second. This necessitates the use of a combinational structure. Furthermore, such structures have the disadvantage that they compute the result each time even if the input does not change. In order to avoid these power consuming transitions these dividers require additional logic to detect equal input signals and so disable the block.

If the input range of the arctan given by (47) is investigated, it can be shown that to fulfil the condition that Z is either the smallest value of the three variables or jointly the smallest value with Y, then the value of the argument prior to multiplication by $\sqrt{3}$ must be less than or equal to one, i.e.

$$\begin{array}{ll} \text{if} & \dfrac{X-Y}{X+Y-2Z} \le 1 \\ \text{then} & X-Y \le X+Y-2Z \\ \text{rearanging gives} & Z \le Y \end{array} \qquad (60)$$

By introducing a detector into the device it is possible to use an intelligent divider structure. Such a divider can disable all following stages of the divider if a one is detected at the input. Taking this into account, the structure of the next stage connected to the divider can also make use of this by checking first if the input is one. This can be done by an iterative investigation from the most significant bit down to the least significant bit. One of the consequences of such a design is the implementation of a pipeline stage after the first computation block. This increases the area and overall capacitance. However, a reduced glitching in the remaining stages reduces the power consumption of the additional logic. In Table 5.5, the best standard implementation is compared with three "intelligent" designs. These three designs take into account that the maximum input value is one. The only difference between them is the method of detection of a one. In design 1, a comparison is performed to determine whether or not the divisor is larger or equal to the dividend, in order to detect a one. During normal operation the divisor can be only as large as the dividend. Therefore, this comparison is purely a safety measure. Design 2 uses a comparison to detect whether or not the dividend is smaller than the divisor. This gives the same safety as design 1 but can be synthesised using less logic. The last design implemented (design 3) uses a comparison to determine if the divisor is equal to the dividend. This is also a simple operation, requiring approximately the same amount of logic as the smaller comparison. However, such an implementation does not have the safety margins the two previous designs

had. But under normal operation conditions this will not result in any problems, as shown during simulations.

	Area/mm^2	Number of Nets	Timing/ns	Active Capacitance/pF
Best Standard	0.431	289	13.10	32.09
Intelligent Design 1	0.852	483	12.55	31.84
Intelligent Design 2	0.425	276	18.80	31.85
Intelligent Design 3	0.422	276	10.63	29.78

Table 5.5: Key Features of an Intelligent Divider Module

As seen in Table 5.5, the difference between all implementations with respect to the power consumption is small. However, the area and timing comparison shows that design 3 has the best overall performance as well as the lowest power consumption. Therefore, this design is used in the overall implementation.

5.4 The Arctan

The arctan function is an odd function. This also halves the range of the arctan function which has to be calculated. A problem encountered in computing the arctan function is the representation of the factor $\sqrt{3}$ which is present in all arctan terms of the hue algorithm. Since it is not possible to represent this value with an accuracy of 100%, an implementation would either contain a significant error or would be unreasonably large. Large busses not only have the drawback of higher switching but also lead to longer interconnection lengths and larger functional units. This is because the subsequent units have to compute more bits. For these reasons alternative implementations of the arctan function were investigated. This section describes four different possibilities for implementing the mathematical function arctan in VLSI. Figure 5.7 shows the block diagram of the module for computing the arctan.

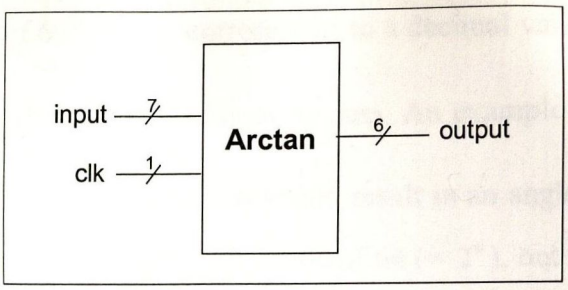


Figure 5.7: Block diagram of the Arctan

Before presenting the implementation of the arctan function, the numerical relationship between the input argument code and the output function code is established. The input argument of all versions has a bitwidth of 7 bits, where the most significant bit corresponds to a decimal value of 1. The six least significant bits represent the digits after the decimal point. As has been shown in the previous section, the maximum input into this stage is one. Therefore, each changing bit-position within the six least significant bits corresponds to a difference of the input value of $\frac{1}{2^6} = 0.015625$. For example, a binary input value of 1000111 corresponds to 1.109375 decimal. The output of the block has a bitwidth of 6 bits. The output function range is limited to 60° because of the three equivalent sections describing the hue space, as shown in Figure 5.8.

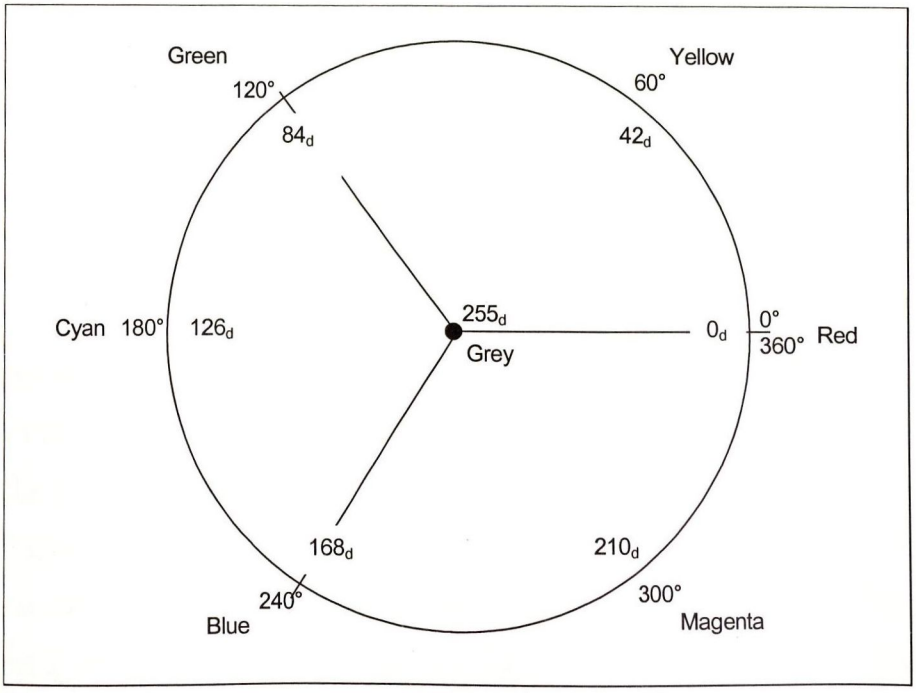


Figure 5.8: The Hue Circle

In Figure 5.8 various colours and their corresponding angles are shown on the outer perimeter of the hue circle. Inside the circle the decimal values for an output range of 8-bits are shown.

As can be seen an angle of 60° or $\frac{1}{3}.\pi$ corresponds to a decimal value of 42. This results in a resolution of hue of 1.43 degrees ($\frac{60°}{42}$) for every step. An example illustrates this idea. The binary number 100011 is 35 decimal, which would result in an angle of 50° (= $35 \cdot 1.4286^{\circ}$). With 6 output bits, it is possible to get a resolution of 64 (= 2^6), but the output cannot exceed 74°, because of the restriction of the input definition. Therefore, the highest output value is 52 (= $\frac{74^{\circ}}{1.4286^{\circ}}$).

5.4.1 The CORDIC Algorithm

The COrdinate Rotation Digital Computer (CORDIC) algorithm is traditionally used for implementing trigonometric functions in hardware. Volder first introduced the CORDIC Algorithm in 1959 [Vold59]. This algorithm uses only shift steps and addition operations to calculate most mathematical functions such as multiplication, division, addition, subtraction and trigonometric operations. The basic idea of CORDIC is to take a vector, given by (x,y) or (x,z), and drive it towards zero using a series of additions and subtractions steps. The steps required to drive the vector to zero correspond to the result of a mathematical function. Volder's CORDIC algorithm uses three variables (x, y, z) and is based upon the equations (61).

$$\begin{aligned} x_{n+1} &= x_n + d_n y_n 2^{-n} \\ y_{n+1} &= y_n - d_n x_n 2^{-n} \\ z_{n+1} &= z_n + d_n \arctan 2^{-n} \end{aligned} \tag{61}$$

In all implementations of (61), the values for arctan 2^{-n} are precomputed and stored where n is the index of the iteration (0,1, 2, 3,.... N). Thus, this set of equations is repeatedly executed N times until the result converges to the required accuracy. The term d_n is chosen to drive either y or z closer towards zero and can assume a value of either +1 or -1. The CORDIC algorithm can operate in two modes depending on the mathematical function required. These are known as the vectoring mode (VM) and the rotation mode (RM). For $n \to \infty$ this leads to the results summarised in Figure 5.9. As can be seen in this figure the arctan function is part of the VM and therefore only this mode will be discussed in the remainder of this section.

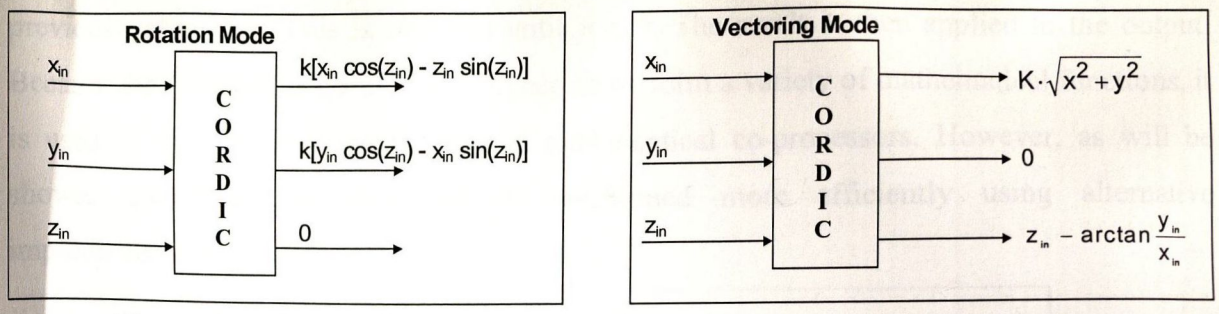
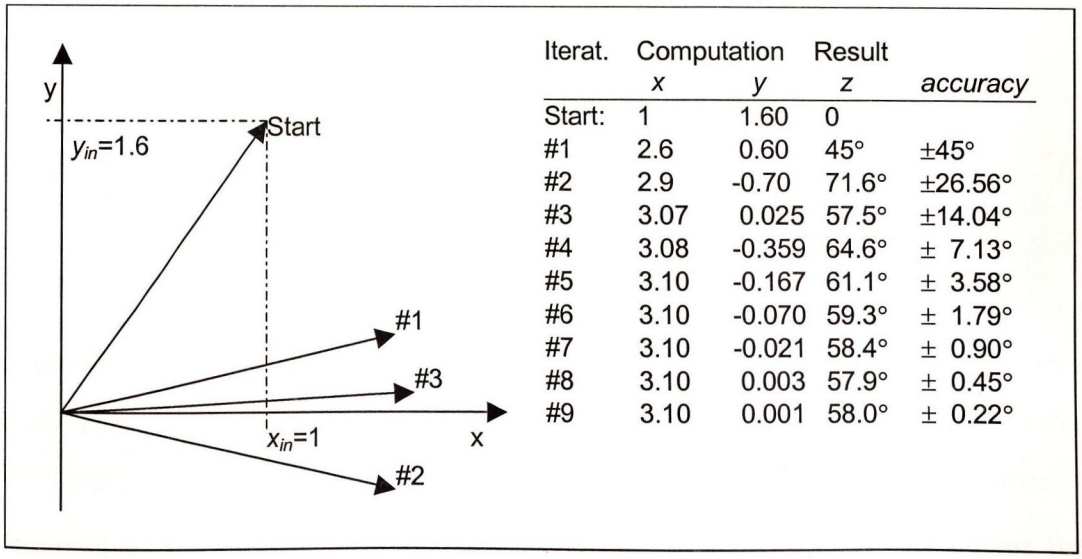


Figure 5.9: The Rotation and the Vectoring Mode of the CORDIC Algorithm

In the VM, the co-ordinate components of a vector (x,z) are fed into the CORDIC algorithm and the magnitude and angular argument of the original vector are computed. The value of d_n in (61) is then chosen to drive y toward zero. If y_n is greater than zero, then d_n=+1 and conversely, if y_{in} is less than zero, the value of d_n will be -1. If y is zero, the result will be applied directly to the output. To calculate the arctan(y), the input variables have to be set to x_{in} = 1, z_{in} = 0 and y_{in} will then become the argument of the arctan. By way of an example, taking y_{in} =1.6, the computation of arctan(1.6) is illustrated in Figure 5.10. For clarity only the first three iteration steps are included in the diagram, however the whole computation is shown on the right hand side of the figure, and convergence to an accuracy of better than one degree is achieved after 8 iterations, producing the desired result of 58°.



Iterat.	Computation x	y	Result z	*accuracy*
Start:	1	1.60	0	
#1	2.6	0.60	45°	±45°
#2	2.9	-0.70	71.6°	±26.56°
#3	3.07	0.025	57.5°	±14.04°
#4	3.08	-0.359	64.6°	± 7.13°
#5	3.10	-0.167	61.1°	± 3.58°
#6	3.10	-0.070	59.3°	± 1.79°
#7	3.10	-0.021	58.4°	± 0.90°
#8	3.10	0.003	57.9°	± 0.45°
#9	3.10	0.001	58.0°	± 0.22°

Figure 5.10: Computation of the Arctan using the CORDIC Technique

It is necessary that the angular increments of the vector rotation are computed in a decreasing order, e.g. for a range of $\pm\pi/2$, the magnitude of the first rotation step is $\pm\pi/4$. Therefore, the value of y is driven closer to zero with each rotation step.

Figure 5.11 shows a flowchart of the VM. In the first step the shifted values of x and y are stored in b and c. Then, an angle is either added or subtracted from the input depending on the

previous value of y. This is repeated until $y = 0$. The result is then applied to the output. Because the CORDIC algorithm is designed to perform a variety of mathematical functions, it is used in most pocket calculators and mathematical co-processors. However, as will be shown, individual functions can be performed more efficiently using alternative implementations.

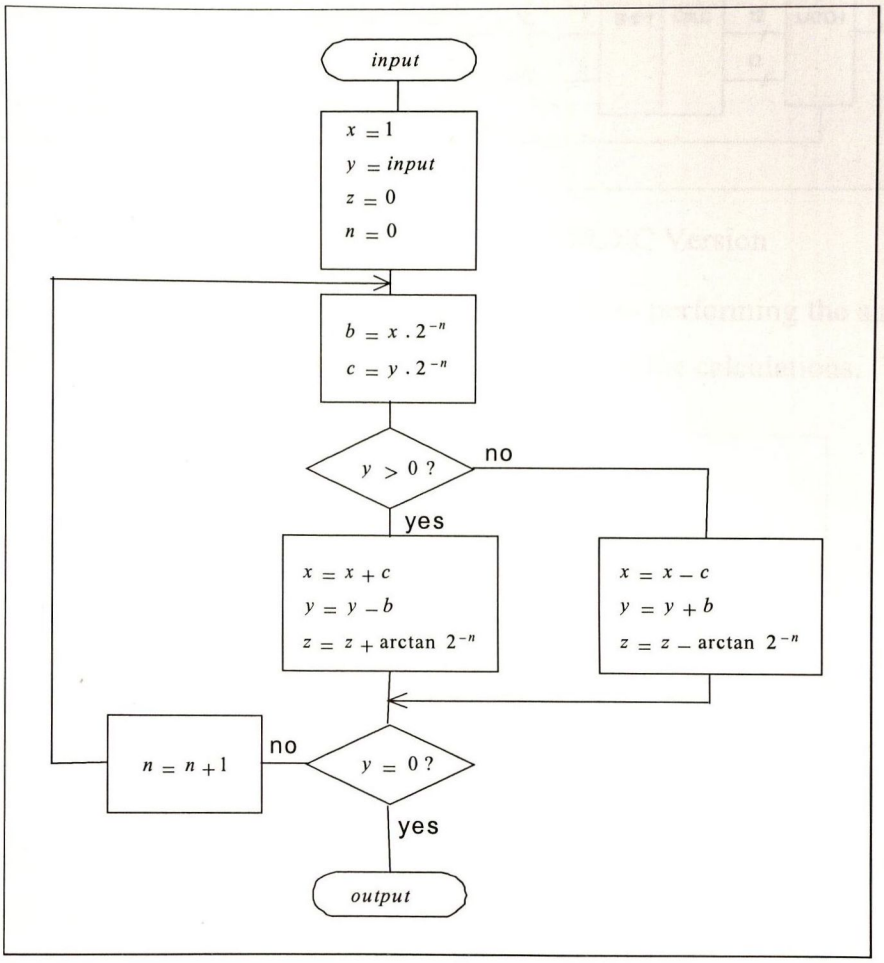


Figure 5.11: Flowchart for the Vectoring Mode

The arctan implementation using the CORDIC algorithm computes the arctan of an input with 7 bits and presents the result with 6 bits at the output. This version is clock-controlled and handles the input vector as a positive number (there is no sign bit). The input is normalised as the argument of the division of the variables is multiplied with $\sqrt{3}$. This means that the output then shows the result of the arctan($\sqrt{3} \cdot inputvalue$). The maximum input value is 1111111, which corresponds to 1.984375. This restricts the output to a value of 74° (= arctan ($\sqrt{3} \cdot 1.984375$)). However, as the highest input value is one the maximum output value is 60°.

The design consists of 10 combinatorial stages and 11 pipeline stages. The input value will be applied to the first latch. With every positive clock event, the flip-flop transmits its

input value to the following stage. This ensures that there will be no new input value during the calculation process. The block diagram of the complete design is shown in Figure 5.12.

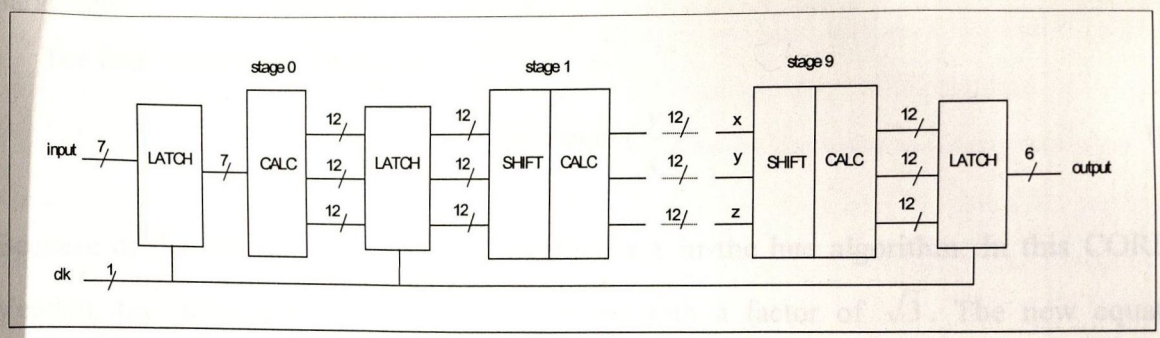


Figure 5.12: The Structure of the CORDIC Version

The following figures show the block diagrams of the modules performing the shift operation and the calculations. Figure 5.13 shows the module performing the calculations.

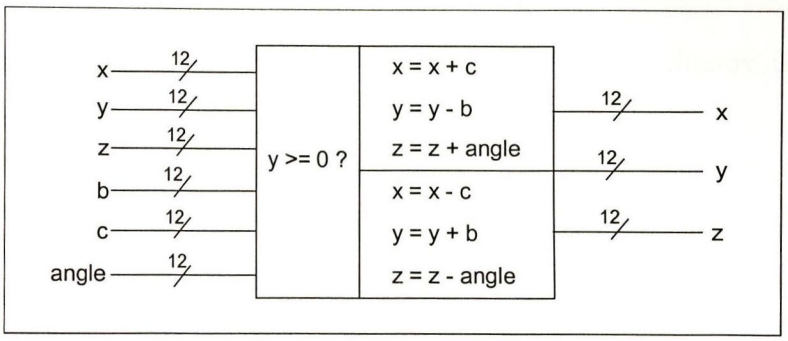


Figure 5.13: The Calculation Stage

The shifted values of x and y will be used by the calculation stage as b and c. The number of shifted bits will be defined by the applied value of n as shown in Figure 5.14.

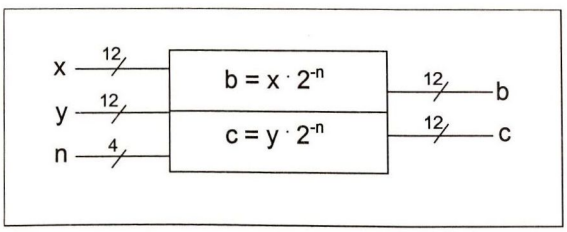


Figure 5.14: The Shift Stage

In stage 0, the input range of 7 bits will be sign-extended to 12 bits. The most significant bit represents the sign, the next bit is used as an overflow, 7 bits for the input and 3 bits as the least significant bits. The sign bit allows it to calculate positive and negative results. With 2 bits in stage 9 it is possible to get a value of b which is not equal to 0, when the x value is shifted by 9 bit to the right.

The binary code for the x value in stage 0 is chosen to be 000100100111 because of the final equation for z from the CORDIC Algorithm, which will be explained in the following paragraph.

The final expression for of z is

$$z = \arctan\left(\frac{y}{x}\right) \tag{62}$$

Because of the normalised input, y is divided by x in the hue algorithm. In this CORDIC Version, the input will be multiplied internally with a factor of $\sqrt{3}$. The new equation becomes

$$z = \arctan\left(\frac{\sqrt{3} \cdot y}{x}\right), \tag{63}$$

where $\frac{\sqrt{3}}{x}$ is a constant, as the input to the arctan module is already the result of $\frac{y}{x}$. Therefore, the value of x is 1.

$$z = \arctan\left(\sqrt{3} \cdot y\right) \tag{64}$$

To implement this constant, the factor will be represented by 0.57735 ($= \frac{1}{\sqrt{3}}$). This number will be described with 10 bits. The value of one bit step is 0.000977 (2^{-10}). By dividing the constant value by the step value, the decimal code will be the result 591. The number 591 represented in binary code is 000100100111.

In stage 0 it is not necessary to implement a shift operation because here the values of b and c are equal to the value of x and y. However, there must be a process in stage 0 which increases the input value received from the first latch to achieve the internal bitwidth. This is performed by adding 2 bits before and 3 bits after the binary code for the value of y. Every stage contains the binary code for the fixed angle, which will be applied to the calculation stage. This fixed angle will be added or subtracted to the previous value of z. Also, the stages contain the factor which decides how many times the x and y value will be shifted in the shift stage. The results of the shift stage are returned as b and c. The last latch has only to process the value of z, which is 12 bits wide. The use of 12 bits is necessary, because of the inclusion of 10 stages. In this lookup table, the fixed angle for stage 0 needs 10 bits plus 2 bits for the

sign. It also uses a bit that ensures that no overflow occurs. Bits 9 down to 4 will be transferred to the output with the next positive clock event.

As described in this section, the bitwidth of x, y and z as well as the number of stages have been chosen so that the simulation has a result corresponding to the next possible value. This value has a smaller deviation. Also in this configuration the order of output numbers will decrease monotonically without using a higher number than used before, when the input will be counted down from 1111111 down to 0. The result of this implementation is calculated after 10 clock cycles.

As previously described, the values of the angles used for the calculation of z must be precalculated and stored. For these precomputed values, decimal 42 corresponds to an angle of 60°. The internal bitwidth of z is 12 bits as described previously. Therefore, the binary code for an angle of 60° can be defined by $42 \cdot 2^4$ (4 extra bits), which is 672 decimal. By dividing 60° by 672, the result is a step value of 0.089286°. The code for the fixed angle will be calculated by dividing the fixed angle by the calculated bit value. After this, the decimal result is transferred into binary code. The table of the fixed angles used in the different stages is shown in Table 5.6.

	Fixed angle	Binary code	Stage number
Arctan 2^0	45	000111111000	0
Arctan 2^{-1}	26.565	000100101010	1
Arctan 2^{-2}	14.036	000010011101	2
Arctan 2^{-3}	7.125	000001010000	3
Arctan 2^{-4}	3.576	000000100111	4
Arctan 2^{-5}	1.7899	000000010100	5
Arctan 2^{-6}	0.8952	000000001010	6
Arctan 2^{-7}	0.4476	000000000101	7
Arctan 2^{-8}	0.2238	000000000011	8

Table 5.6: Table of the Fixed Angles for the CORDIC Version

5.4.2 The Lookup Table

The second proposed technique to implement the arctan function is the use of a Look-Up Table (LUT). LUT's are simple Read Only Memory (ROM) storing devices. They contain only one set of data, in this case one number for each input address. These numbers represent the result of the process and the result has to be calculated for each possible input value before implementing the LUT. LUT's are very fast compared to an algorithmic implementation, as all possible output values are already calculated. Traditionally, LUT's are used in high-speed applications where other implementations would be too slow. But they will also prove to be very well suited for use in low-power applications because no dynamic power is consumed after the address bits are applied.

Unfortunately LUT's also have disadvantages. For most applications they are larger than the normal algorithmic implementation due to the fact that all possible output values have to be stored. A simple example should illustrate this. A simple 8 by 8 bit multiplier requires 65536 address spaces, each containing 16 bits and this would result in a memory of more than 1.5 Mbits. For this reason LUT's can only be used for applications with a small number of addresses and a limited width of the output data. This excludes most computations.

Normal memory devices consume large amounts of power due to precharging. Therefore, special circuits as presented in [Athas94] can be used in order to reduce the power up to 75%, depending on the minimal allowable swing voltage at the bitlines.

Because today's digital RGB standard uses a quantisation of the input signal into 8 unsigned bits for each colour, the hue output is chosen to also have 8 valid bits which results in a possible output range of 0 to 255. Due to Kender's algorithm for the computation of hue, one number has to be reserved for indicating an achromatic pixel. This leaves a possible range of 254 for describing the hue space. The algorithm splits the output range into three separate units, each of the same range (Figure 1.1). Therefore, the output range is split into three parts, each containing 84 values. This gives a total of 252 plus two values for the singularities if hue is 360° or achromatic. Because it is not possible to use two values, the dynamic range of hue is reduced by 0.68%.

To determine the necessary number of addresses and output bits for the LUT, Kender's algorithm has to be investigated once again. The first three sorting terms of the hue algorithm, (1.1)(1.2)(1.3), ensure that the hue space is unique for all three cases and that each of the ranges spread from -60° to +60° are added to a coefficient depending of the case chosen. This

results in an integer range of 84 values when transferred to the digital hue space. Because the arctan is an odd function the number of values can be reduced by a factor of two since only the positive (or negative) values have to be stored. Therefore, 42 values require an address bus of 6 bits. The unused values could be replaced by don't care terms, which would result in a smaller circuit. It is also possible to use these values to provide sufficient accuracy and also to implement the necessary rounding by storing the previously rounded result for each address.

In (63) the maximum value of the argument of the function is $\sqrt{3}$. Since the factor $\sqrt{3}$ was only needed to fulfil the requirements of the computation of the arctan function, it can now be replaced by the maximum number of addresses contained in the LUT. This replacement by an integer number guarantees not only the smallest possible bus, but also a minimum amount of switching activity on the bus without an additional error. The input bitwidth is 7 bits wide, and the output will be represented with 6 bits. The precalculated output value has been chosen so that the input value appeared to be multiplied with $\sqrt{3}$.

$$output = \max.output \cdot \arctan\left(\frac{\sqrt{3}(X-Y)}{X+Y-2Z}\right) \tag{65}$$

5.4.3 The Modified Lookup Table

This design is based on the same idea as the LUT described in the previous section. The only difference is that the first 22 output values are not precalculated and stored. The values of the input for this range will be assigned directly to the output for input values from 0000000 to 0010110. This is possible because of equation (66).

$$\arctan(x) \approx x \quad ; x<0.5 \tag{66}$$

With 7 bits at the input and 6 bits at the output, the difference in the resolution is 1.5957. This will be balanced with the factor $\sqrt{3}$ (1.732) at the input. Therefore, the direct assignment from the input to the output is valid. For example, an input value of 0011011 corresponds to a value of 0.1719 decimal. Multiplied with the factor $\sqrt{3}$, the value becomes 0.2977. This results in an angle of 16.58° for the arctan of 0.2977. With the definition of the output (60° corresponds to 42 decimal) the output value is 12 decimal. This corresponds to the binary code 1100 ($\frac{16.58^\circ \cdot 42}{60^\circ}$). This implementation technique has the advantage over the first LUT

that the number of stored values can be reduced by a factor of 37%. This was achieved by introducing a comparator into the circuit. Therefore, the additional capacitance of the comparator has to be balanced against the smaller memory device.

5.4.4 Approximating the Arctan

The approximating of the arctan is another possibility for implementing the arctan into VLSI. The idea is to divide the input range into different parts and describe these sections separately with simple functions. In this case it was possible to split the arctan into four parts and describe them using linear approximations. The input values range from 0 to 1.98438 using a quantisation of 0.015625. The output is defined so that 60° corresponds to a decimal value 42. The Approximate Version is divided into four sections as shown in Figure 5.15.

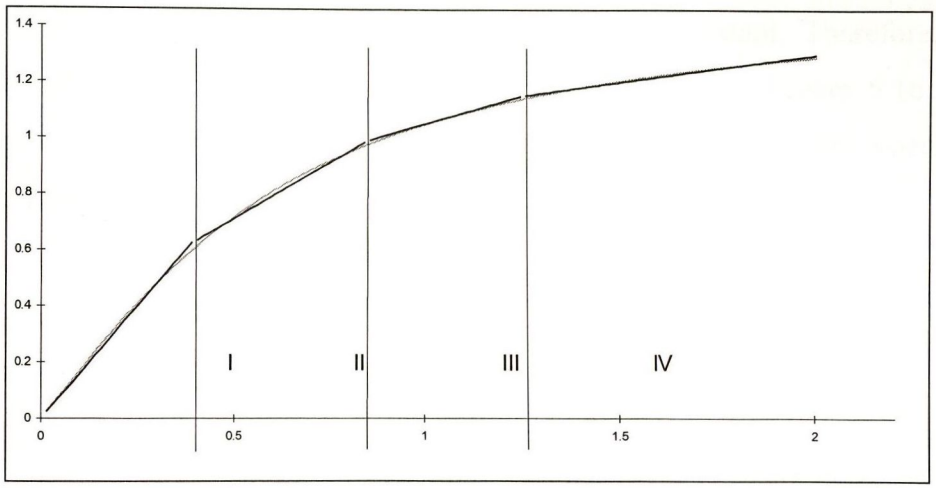


Figure 5.15 The four Sections of the Approximate Version

Part I based on the fact that for small input values the equation

$$\arctan(x) \approx x \tag{67}$$

is valid. In this case the result appears at the output, because of the balance between the factor $\sqrt{3}$ and the condition that 60° corresponds to 42 decimal. Therefore, the same binary code appears at the output with factor 1.596 depending on the different resolutions at the input and output.

Parts II to IV are based on simple linear equations. All equations use a constant factor, which is a multiple of 2^{-n}. This factor is multiplied by the input value and a constant is added. Therefore, it is easy to implement this equation in hardware by using shift options and one addition. The equations for the different sections of the algorithm are shown below and are

only optimised for this particular input range and the function arctan $(\sqrt{3} \bullet x)$ with the additional condition that the result appears in the form that was described previously.

I.	0 : 0.39063	$\arctan(\sqrt{3} \cdot x) = x$	(68)
II.	0.40625 : 0.84375	$\arctan(\sqrt{3} \cdot x) = 2^{-1} x + 0.31$	(69)
III.	0.84938 : 1.25	$\arctan(\sqrt{3} \cdot x) = 2^{-2} x + 0.65$	(70)
IV.	1.26563 –:1.984375	$\arctan(\sqrt{3} \cdot x) = 2^{-3} x + 0.9$	(71)

This particular approximation was optimised for the implementation into hardware for low-power applications. As can be seen, the last 3 parts follow the same idea that the input signal is multiplied by a factor 2^{-n}. This operation can be performed using only shift operations, which consume little power. Then the result is added to a constant. Therefore, only one comparator and one adder are needed to perform the computation. Figure 5.16 shows the characteristic of the approximate version in comparison with the original function arctan, including the effects of the finite word length.

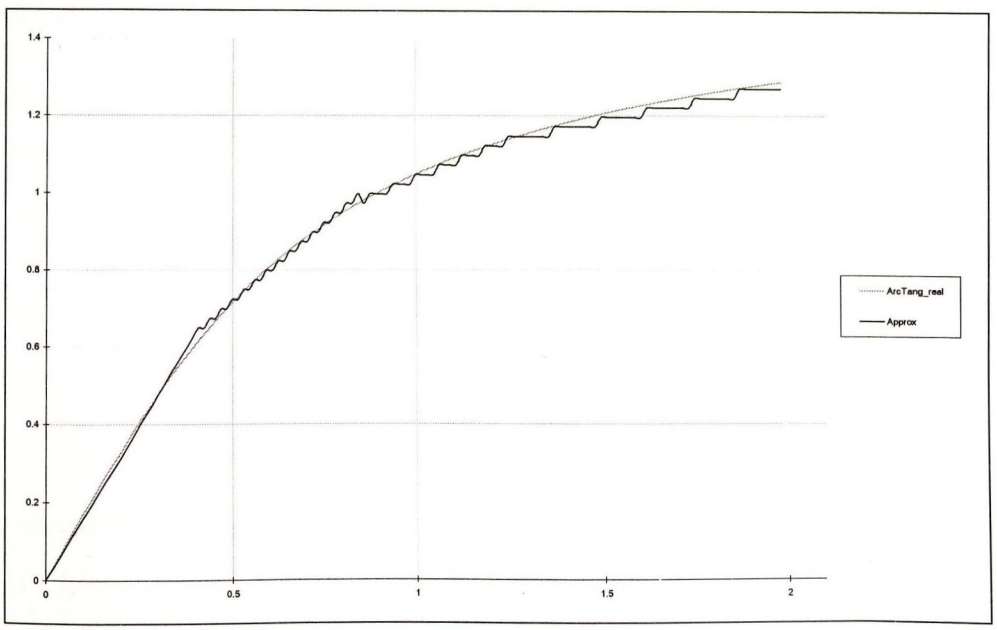


Figure 5.16: Characteristic of the Approximate Version and the Arctan

5.4.5 Features of the Different Implementation of the Arctan

Table 5.7 shows the characteristics of the different solutions, representing the most important properties of each implementation.

Version	Number of nodes	Timing/ns	Active Capacitance/pF	Total area / mm^2
CORDIC	1010	9	70.1	2.69
Lookup Table	140	2	3.82	0.187
Modified LUT	90	2	2.95	0.126
Approximate	69	2	1.99	0.100

Table 5.7: Characteristics of the Different Implementations

Another important point in a comparison of the four implementations is the error deviation. Firstly the absolute error is presented. Then the errors are presented in degrees. This makes a comparison of the different functions and their quality easier to perform.

Absolute Deviations

The Lookup Table is the version which has an optimised error deviation. Every single precalculated output value has been chosen to have the least deviation. The accuracy is presented in absolute values in Figure 5.17. Here it can be seen that at no point a deviation of more than 0.8 degree occurs.

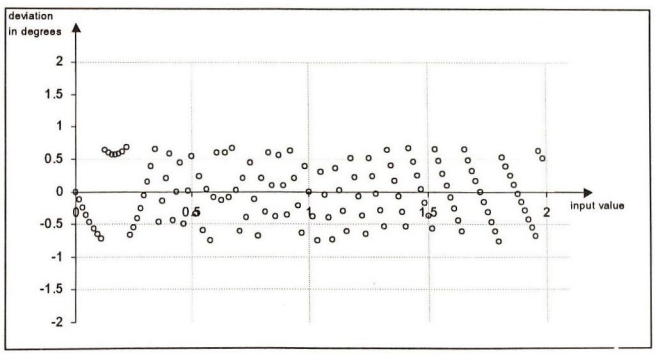


Figure 5.17: Absolute Deviation of the LUT

Next the modified LUT is presented in Figure 5.18. Here the only difference to the LUT in Figure 5.17 is that for small input values the input is directly connected to the output. This results in a slightly higher deviation for the lower input range. However, the maximum deviation is still within 0.8 degrees.

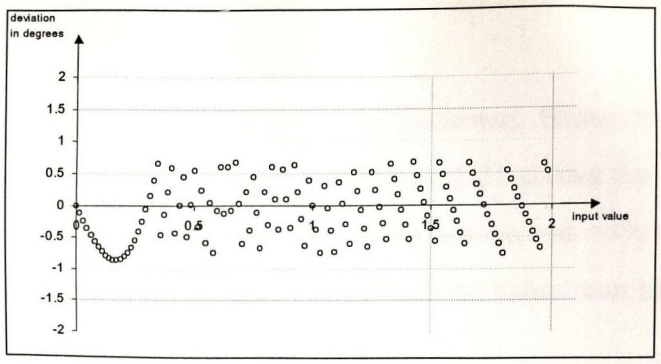


Figure 5.18: Absolute Deviation of the Modified LUT

Figure 5.19 shows that the maximum deviation of the CORDIC algorithm is 1.2 degrees. Therefore, the CORDIC algorithm is not as accurate as the LUT.

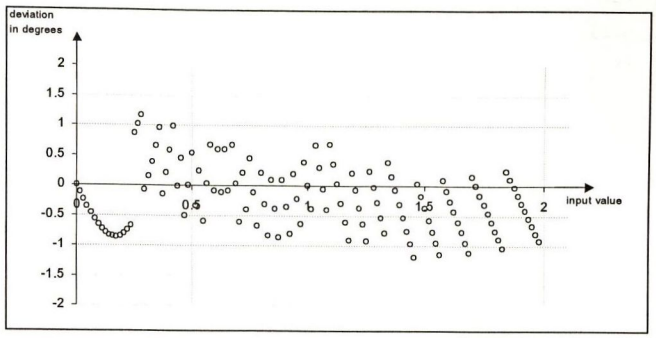


Figure 5.19: Absolute Deviation of the CORDIC Algorithm

As seen in Figure 5.20 the maximum error of the Approximation Algorithm is 2 degrees. This is at least 0.8 degrees higher than for every other implementation. Therefore, this algorithm was investigated using real image data. The results in Chapter 7 confirmed that such an error does not give a visible error.

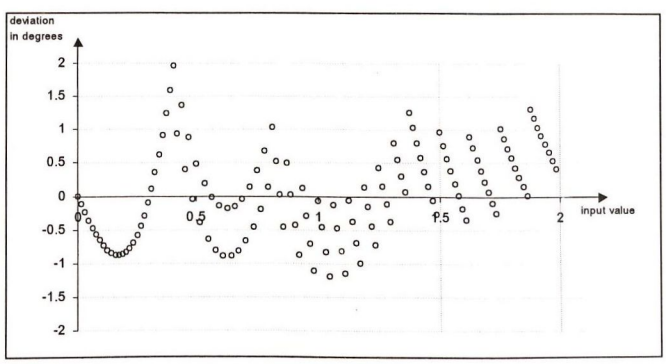


Figure 5.20: Absolute Deviation of the Approximation Algorithm

The results presented show that the look up tables have the smallest error deviation. This was also expected as in this case the results of the arctan are stored in the best possible manner. Therefore, the LUT can be seen as the benchmark for the other implementations. To enable a better comparison, the errors are presented in the next section as percentages.

Percentage Error

The four implementations of the arctan are again presented. However, now the percentage error compared to the theoretical value is given. Figure 5.21 shows the error deviation of the LUT in degree. Here it can be seen that the error does not exceed +4% and -8% respectively. Because the LUT uses the most accurate information these values can be seen as the standard against which the other designs may be measured.

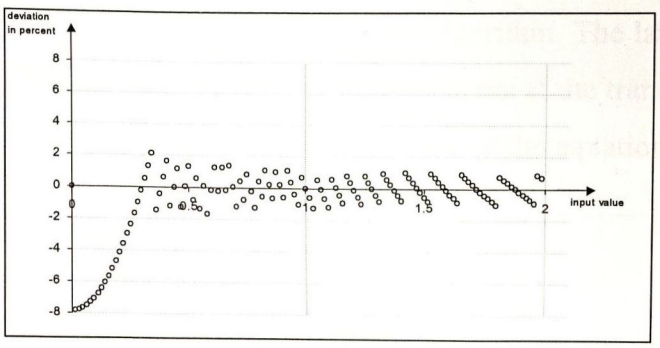


Figure 5.21: Error Deviation of the LUT

In comparison to the Lookup Table, the Modified LUT version has just 6 values, which deviate from the theoretical possible values of an implementation with the given output ranges. This is due to the direct assignment for small input values. There errors however increase the positive error range to nearly 6%.

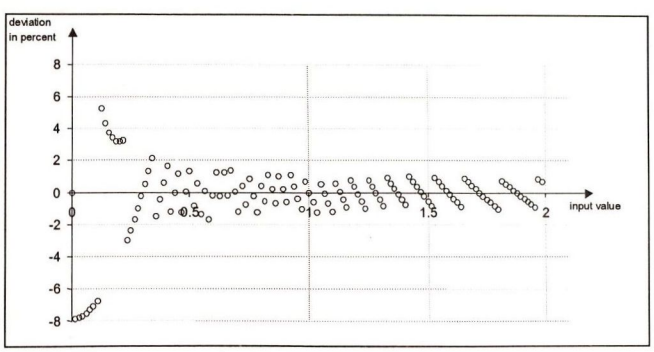


Figure 5.22: Error Deviation of the Modified LUT

The CORDIC algorithm achieves a good result in comparison to the LUT, which has the least error deviation. As seen in Figure 5.23, most of the output values have at most a deviation of ±2%. Furthermore, the error may be further reduced by using more internal bits.

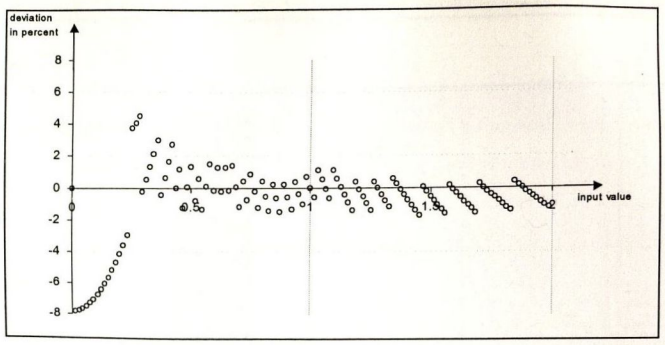


Figure 5.23: Error Deviation of the CORDIC Algorithm

Figure 5.24 shows the behaviour of the approximation algorithm. The largest deviation of the output value, when compared to the theoretical values, occurs at the transition of the different equations. The highest errors occur at the transition points of the equations.

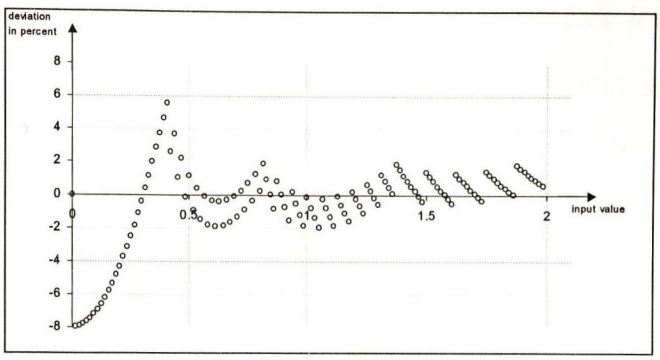


Figure 5.24: Error Deviation of the Approximation Algorithm

If the error deviation of the Approximation algorithm is compared with the error deviation of the modified LUT, it can be seen that the maximum deviation of 8% is also never exceeded. Because this looks like a large error the algorithms were investigated using images. It has been shown, however, that these errors are not visible in a transformed image.

Next, the power consumption of the different implementations of the arctan function is compared. Figure 5.25 shows the power consumption for the four different solutions at 10MHz.

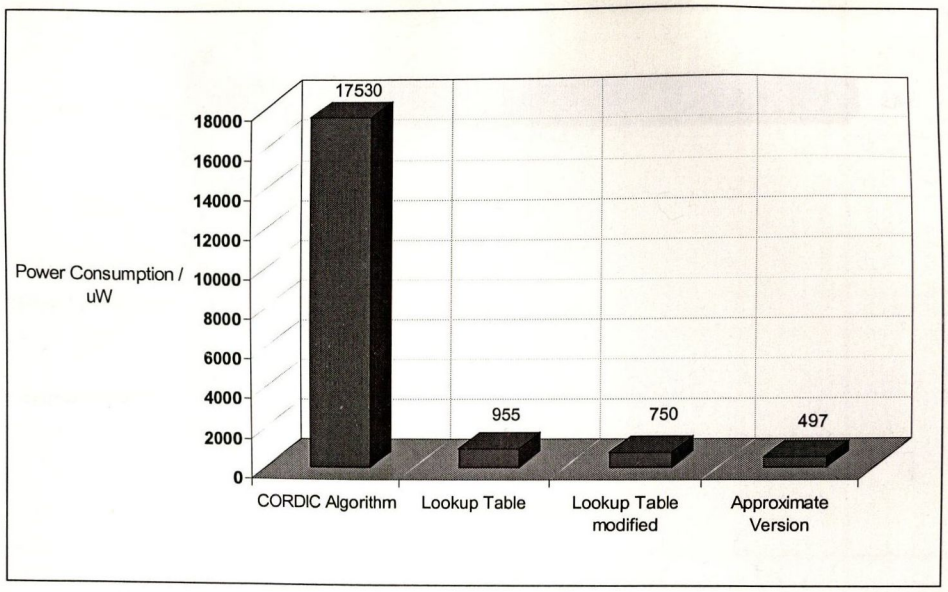


Figure 5.25: The Power Consumption of the Arctan Implementations

In comparison to the Lookup Table, the Modified Lookup Table and the Approximate Version, the CORDIC Version requires approximately 25 times more power. The Lookup Table needs 5.5%, the Modified Lookup Table needs 4.3% and the Approximate Version only 2.8% of the power required by the design using the CORDIC algorithm. Therefore, in this case the CORDIC Version is ruled out for implementation. The Modified Lookup Table requires 78.5% of the power which is necessary for the normal Lookup Table. The reason for this is that the first 23 of the 128 input values (from 0000000 to 0010110) will be assigned directly to the output. Theoretically, with the Modified Lookup Table a saving of 18% is possible. However, the Synopsys Design Compiler optimised this modified version in such a way that there is a saving of 21.5%. The Approximate Version needs only 66.3% of the power of the implementation of the Lookup Table and therefore, this version is the optimum with respect to power consumption.

Area is in many designs also a constraint as it determines the fabrication cost. Figure 5.26 shows the required area in mm^2 for the four implemented versions of the arctan.

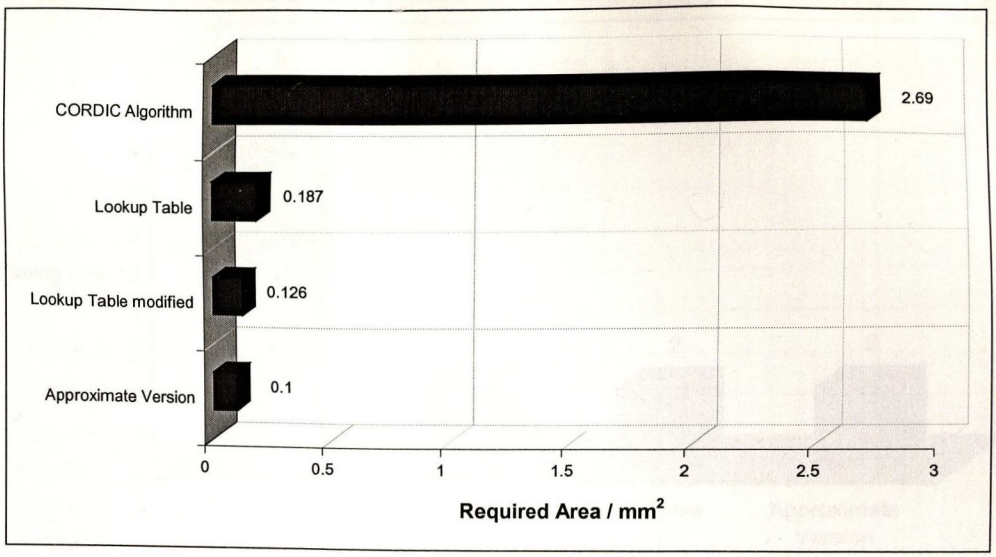


Figure 5.26: The Required Area

The area requirements behave very similarly to the requirements of the power consumption. Usually, the more area required the more power is consumed. The CORDIC Version requires approximately 25 times more area than the other three versions. The Lookup Table needs 7%, the Modified Lookup Table needs 4.7% and the Approximate Version needs only 3.7%, when compared to the power required of the CORDIC implementation. The version which uses the CORDIC Algorithm is a very large design compared with the other three solutions. Thus, it is not practical to implement it into a microchip. The Modified Lookup Table needs less area than the normal Lookup Table does. Similarly to the power consumption, the best result with respect to the required area will be produced by the Approximate Version. This version needs 53.5% of the area required by the Lookup Table and 79.4% of the area required by the Modified Lookup Table.

The time which a design needs to compute the arctan is shown in Figure 5.27. All designs achieve the timing constraint of 33ns. The CORDIC algorithm requires the longest computation time of 9ns. This was exactly as anticipated because of the large calculation process of this algorithm in comparison to the other implementations. The Approximation algorithm, the Lookup Table and the Modified Lookup Table require approximately 80% less time to calculate the arctan than the CORDIC Algorithm.

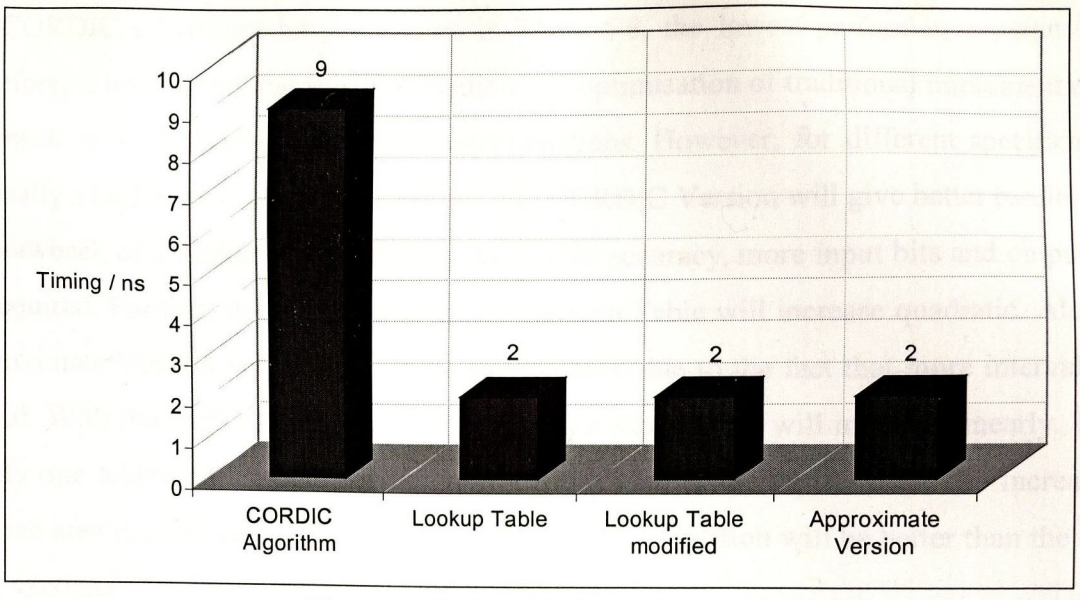


Figure 5.27: The Timing Behaviour

Therefore, the highest operating frequency for the Lookup Table, the Approximate Version and the Modified Lookup Table is a frequency of 100MHz. It should also be mentioned that the CORDIC algorithm requires nine clock cycles to compute the result in 9ns, because of the pipeline stages which were implemented. These pipeline stages will add an additional overhead to the design in the form of more pipeline stages in other parts of the HSI algorithm.

This Section has described different methods of implementing the arctan function. These functions were investigated with respect to their main features. Under the given conditions the Approximation Version is the most attractive solution. It has the least power consumption and requires the smallest area. With respect to the timing behaviour, there is no difference in both Lookup Table Versions. The accuracy of all versions does not exceed 8% deviation from the theoretical value for each possible input value and in terms of bit deviation not more than 3 bits. Therefore even the Approximate Version does not contribute a significant error to the result.

	The Approximate Version compared to the CORDIC Algorithm
Power Consumption	25 times less power
Required Area	25 times less area
Timing Behaviour	80% less time for the calculation

Table 5.8: Comparison between the Approximation Version and the CORDIC Algorithm

The CORDIC Algorithm has, as shown in Table 5.8, the lowest performance parameters. Therefore, it was shown that the investigation and optimisation of traditional implementations can result in significantly improved implementations. However, for different specification, especially a higher requirement in accuracy, the CORDIC Version will give better results with the drawback of a further increased area. For higher accuracy, more input bits and output bits are required. For this, the expenditure for the Lookup Table will increase quadratic. Also the Approximate Version will become more complicated, due to the fact that more intervals are needed. With the CORDIC Algorithm however, the expenditure will increase linearly. There is only one additional computing-step for each extra bit at the input. Thus, the increase in required area is small and at a certain point, the CORDIC Version will be better than the other three versions.

5.5 Adding the Coefficient

The last module in the hue line has two main purposes. The first is the decoding of the control signal and the second is the adding of the coefficient as defined in (1.1) to (1.5). Therefore, the block diagram uses two input signals as shown in Figure 5.28. The 8-bit hue signal is available at the output of the block.

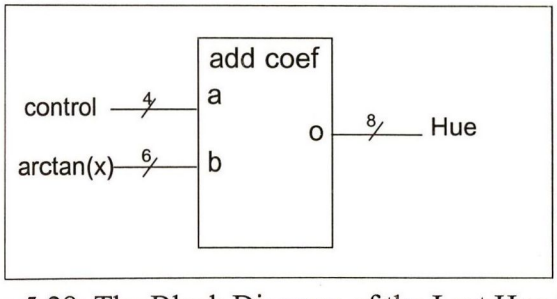


Figure 5.28: The Block Diagram of the Last Hue Stage

The task of the decoder block is to extract the information about the Kender case as well as the sign of the argument of the arctan which is provided by the previous stage. Two different decoders were implemented. The first is a 3-bit decoder, which uses the smallest bit size possible. Here all three inputs have to be monitored to compute the result. The second version is a 7 bit, *active one* bus. Here only one bit changes value each time the input changes. As seen in Table 5.9, the difference in power consumption between the 3 bit and the 7-bit version is very small.

Block	Max. time delay ns	Area mm^2	Nodes n	Power Consumption mW
PRES 3-bit	4.05	0.0552	45	1.37
PRES 7-bit	7.81	0.0677	58	1.46

Table 5.9: Feature of the Decoder Block

At first this result might seem to be surprising as the 7-bit version requires a decoding of more than double the amount of bits compared to the 3-bit version. However, as only one bit changes in the 7-bit version, an iterative algorithm can be used to compute the control signal corresponding to the input value. This is done by introducing don't care statements into the high-level code. In terms of delay and area requirements the 3-bit version is superior to the 7-bit decoder. The area consumption is, at $0.055mm^2$, 18% smaller than the 7-bit decoder and at 4.05ns, is 48% faster. The poorer timing performance of the 7-bit decoder cannot be seen as a real disadvantage because the timing constrains of 33ns are easily met by both implementations. Furthermore, the small difference in area does not make one or the other version superior. At this stage a detailed analysis is not possible as the power consumption of the control line is not yet included in the discussion.

The adder block of the last hue stage is responsible for adding the corresponding coefficient to the argument of the arctan according to the Kender case as selected by stage one. If one of the last two cases of Kender's algorithm (1.4) or (1.5) is true the output signal is known and the value is applied directly to the output. Therefore, the different values of the coefficients have to be stored. The sign information is computed separately. Therefore the adders suggested by Kender's algorithm have to be replaced by adder-subtractor modules. Table 5.10 shows the performance parameters of the complete CAL block. Version CAL3 uses the 7-bit decoder while CAL4 uses the 3-bit decoder block.

Block	Max. time delay ns	Area mm^2	Nodes N	Power Consumption mW
CAL 3	17.59	0.0910	73	1.98
CAL 4	12.84	0.0786	60	1.59

Table 5.10: Results of CAL-block

When analysing the features of the CAL block it is obvious that, with respect to the area requirements as well as power consumption, the features of the decoder used in this block are important. However, the adder structure has the strongest influence in this case. Therefore, the version using the 3-bit decoder is preferable to the 7-bit state changing optimised version, if only this stage is investigated. However, as the control line is directly connected to the decoder, the implementation of the control line is discussed in the next Section.

5.6 Control Line

As already described in section 5.1, a control signal is needed to tell the last stage which of the five cases of Kender's algorithm is true. To reduce the power consumption caused by glitching and to meet the timing constraints of 33ns, the hue algorithm was implemented using a highly pipelined structure. Depending on the implementation of the hue algorithm, up to 14 pipeline stages were used. This resulted in the need to delay the control signal by a corresponding number of cycles. For the further investigations a delay of 7 cycles was used. This is because it is the best number of cycles for the implementation of the hue branch in respect of the power consumption. Figure 5.29 shows the block diagram of the control line.

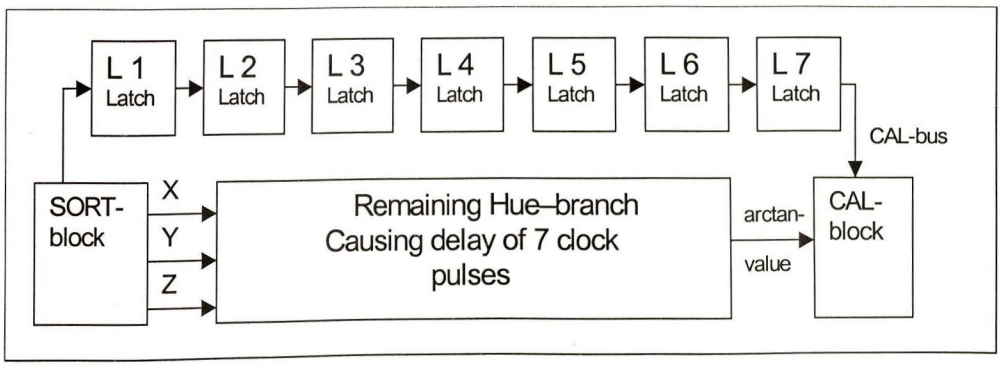


Figure 5.29: Structure of the CAL-bus

In addition to the number of cycles, the coding style of the signal to be transmitted is of direct relevance to the power consumed. Therefore, the control bus has been investigated for bus widths of 3, 4 and 7 bits.

5.6.1 Implementation of the Control-Bus

Firstly the 3 bit-implementation of the control bus will be explained. This bus uses the smallest possible bus width to encode the eight possible combinations. Table 5.11 shows the

coding of the 3-bit bus. Here the coding has the advantage of a small bus width with the disadvantage of a higher switching activity than larger buses.

Kender-case	Sign of the 2nd summand	CAL-bus
Achromatic	Do not care	111
0	Do not care	110
5/3*π	+	001
5/3*π	-	000
π	+	010
π	-	100
π/3	+	101
π/3	-	011

Table 5.11: Coding of the 3-bit CAL-bus

In Table 5.12, below the coding of the 4-bit version of the control bus is shown.

Kender-case	Sign of the 2nd summand	Output of Kender-Bus
Achromatic	Do not care	0110
0	Do not care	0101
5/3*π	+	0000
5/3*π	-	0001
π	+	0100
π	-	0010
π/3	+	0011
π/3	-	1000

Table 5.12: Coding of the 4-bit CAL-bus

In the 7-bit implementation, every Kender case is represented with a '1' in the binary code. This coding style has the advantage that only one bit changes from LOW to HIGH if a change in the Kender case appears. Therefore, with each change in the signal exactly one power consuming transaction occurs. This implementation has the disadvantage of being the largest

silicon structure as well as containing a higher amount of devices to be implemented. In Table 5.13 the coding of the 7-bit bus is shown.

Kender-case	Sign of the 2nd summand	Output Kender-Bus
Achromatic	Do not care	1000000
0	Do not care	0100000
5/3*π	+	0001000
5/3*π	-	0010000
π	+	0000000
π	-	0000001
π/3	+	0000010
π/3	-	0000100

Table 5.13: Coding of the 7-bit CAL-bus

In the following table, the results of the different implementations of the control bus are shown. In the table, the number of nodes, time delay, area and power consumption of the different implementations of the CAL-bus are shown.

Implementation of CAL-bus	Nodes	Time delay ns	Area mm^2	Power Consumption mW
3 bit	25	2.02	0.0544	1.04
4 bit	33	2.02	0.0716	1.30
7 bit	57	2.02	0.1231	1.93

Table 5.14: Results of the CAL-bus Structures

A plot of the results for area is presented in the following diagram.

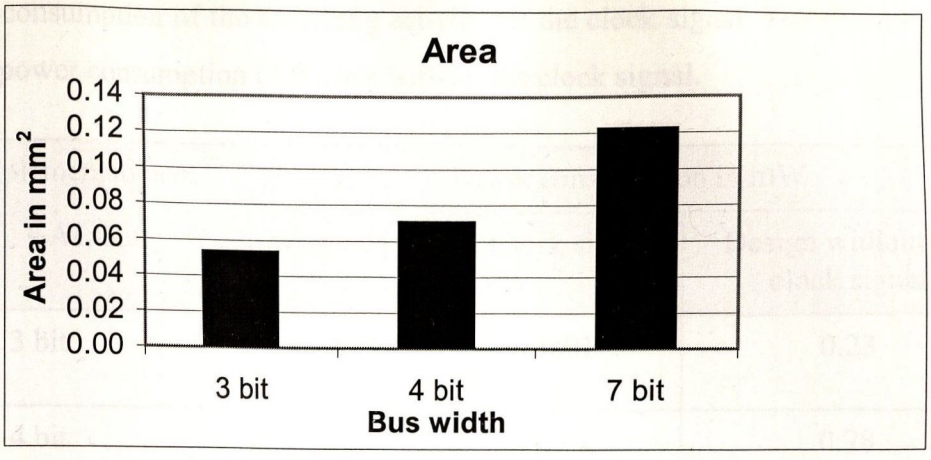


Figure 5.30: Area of the CAL-bus in mm^2

The area requirements for the control bus depend only on the bus width. With a higher bus width the area is increased significantly. Furthermore, the diagram shows that the implementation with registers needs over 20% less area than the other two variants.

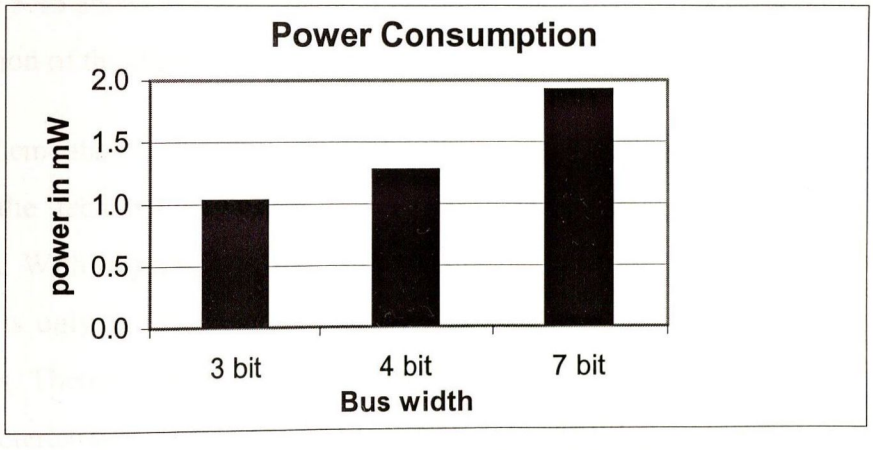


Figure 5.31: Power Consumption of the CAL-bus

The analysis of Figure 5.31 shows that the power consumption depends on the bus width. With a higher bus width the power consumption increases appreciably. The implementations with 4 bits need about 20% more power than implementations with 3 bits. The implementations with 7 bits have about 70% higher power consumption than those with 3 bits. Furthermore, the diagram shows that the implementations with registers have over 45% less power consumption than the other two variants. These results are in contrast to the theory, where the 7-bit version should have a smaller active capacitance than the 3-bit version, due to the reduced switching activity. Therefore, the individual components of the power consumption are investigated separately in the following table. In this table the power consumption of the whole bus is spilt into the two main components. The first component is

the power consumption of the switching activity for the clock signal. The second component shows the power consumption of the bus without the clock signal.

Implementation of CAL-bus	Power consumption in mW		
	Whole design	Clock signal	Design without the clock signal
3 bit	1.04	0.81	0.23
4 bit	1.30	1.02	0.28
7 bit	1.93	1.68	0.25

Table 5.15: Components of the Power Consumption of the CAL Bus

The power which is used by the switching activity of the clock signal depends on the bus width. Table 5.15 shows that the saving in power in the implementation with registers is due to the reduction of the share of the power consumption of the clock signal.

The implementation of the bus with registers instead of flip-flops, both of which are defined in the technology library, is the best solution in relation to area and power consumption. With respect to the number of nodes there is almost no difference. In the time delay, there is only an insignificant difference of 0.35ns to the disadvantage of the variant with registers. Therefore, the implementation of the bus with registers has significantly better design characteristics. The disadvantage of the register variant is that it is more time costly to implement than the variant using the if-inquiry to generate flip-flops. This is because the designer has to connect the registers manually. The solution with the if-inquiry is more elegant than the solution with D-flip-flops. Using the if-inquiry Synopsis converts the VHDL-code automatically into a design consisting of connected D-flip-flops. The advantage of the register variant in relation to power consumption is that the latches need less power for the switching activity of clock signals. The reason for this is that for the same number of bits Synopsis needs less standard cells.

5.6.2 Physical Structure of the Delay Line

Delay lines are usually realised using shift registers. Shift registers are simple latches, which rotate the input information each cycle by one bit. They are mainly used in order to preserve

information for a fixed number of cycles. Shift registers are easy to construct using D-type latches. Figure 5.32 shows such a simple shift register.

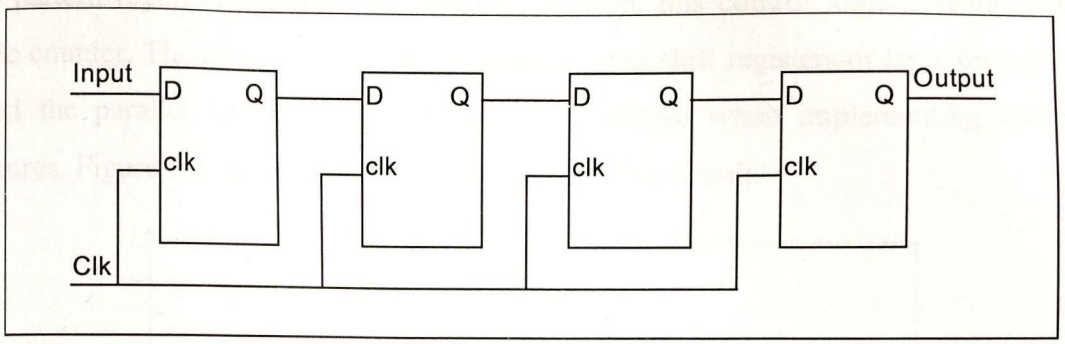


Figure 5.32 Four Stage Shift Register.

In the figure above the data lines of the D-latches are connected in series and a parallel clock signal is applied to the clock input of each stage. This causes the next stage to change each time if a new signal and a clock event is applied to the input. The worst case situation would be an alternating input signal. Therefore, the active capacitance of such registers is

$$C_{shift} = T \times \sum_{k=1}^{m} n_{(0,1)} C_{reg_k} \tag{72}$$

In this equation T is equal to the number of clock cycles during which the circuit is operating, m is the number of bits in the shift register and $n_{(0,1)}$ is the number of LOW to HIGH transitions per clock cycle. For a uniform white noise input signal this equation can be rewritten as:

$$C_{shift} = T \times m \times 0.25 \times C_{reg} \tag{73}$$

In this equation it is assumed that the active capacitance of each node of each stage is equal to the same node capacitance of the corresponding node in the other stages. This is reasonable as this design can be synthesised as a uniform structure. It should be also noted that the LOW to HIGH switching probability for uniform white noise is 0.25. As can be seen in (73) the shift register consumes power in each of the stages. However, as the signal is only needed in the last stage after T cycles, this means that each signal consumes unnecessary power in T-1 stages. Therefore, a different approach was investigated.

Here, instead of using a serial implementation for the stages, the input data is demultiplexed before it is stored in latches. This has the advantage that the data is preserved in one cell rather than rotated through all registers, which causes switching each time the data

is transferred to the next cell. The trade-off of this design is the increased amount of control logic necessary to realise the read and write control of the overall circuit. In circuits where a fixed pattern regulates the read and write mechanism, this control logic is realised with a simple counter. Therefore, it will be shown that for long shift registers or large bit sizes to be shifted the parallel approach is the one to be preferred when implementing low-power structures. Figure 5.33 shows the general design of such a circuit.

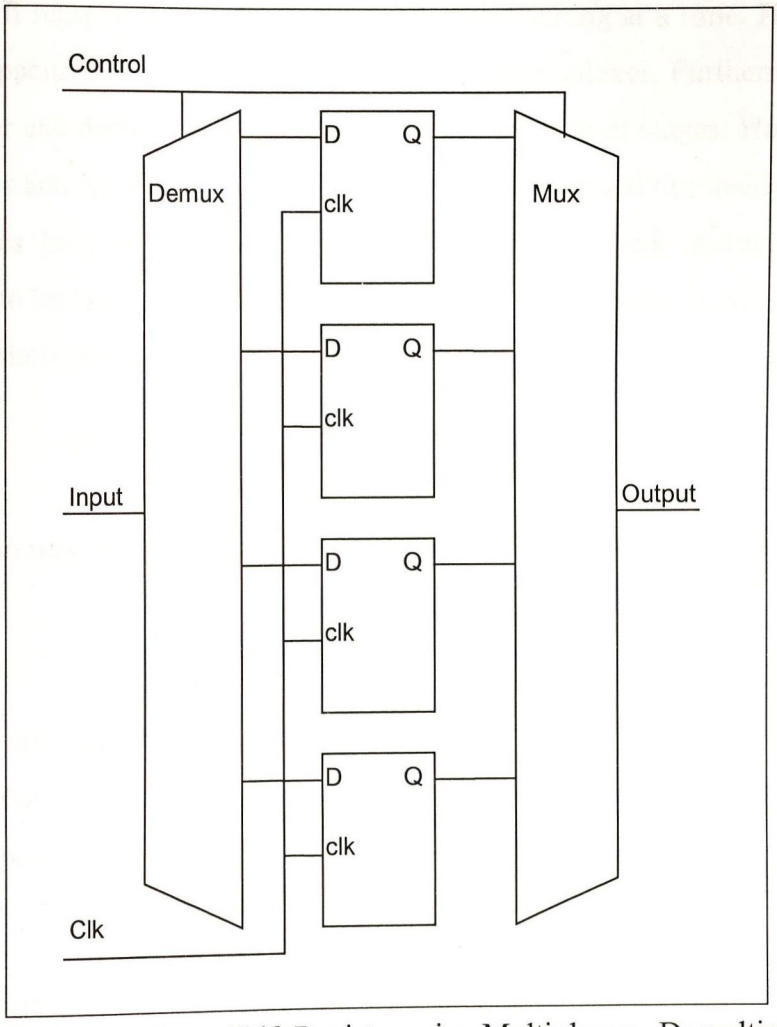


Figure 5.33: Four Stage Shift Register using Multiplexer - Demultiplexer.

The active capacitance of such a circuit can be generally written as follows:

$$C_{sm} = \sum_{k=1}^{m} n_{(0,1)} C_{reg(k)} + n_{(0,1)} C_{mux} + n_{(0,1)} C_{demux} \tag{74}$$

Since only one register per clock cycle can change stage the equation can be rewritten as:

$$C_{sm} = n_{(0,1)} C_{reg(k)} + n_{(0,1)} C_{mux} + n_{(0,1)} C_{demux} \tag{75}$$

In order to compare both approaches for implementing shift registers the number of power consuming events is once again substituted by the probability for a power consuming event to occur for a UWN input stimulus.

$$C_{sm} = 0.25\left(C_{reg} + C_{mux} + C_{demux}\right) \tag{76}$$

As seen this equation has a constant active capacitance. It is therefore independent of the length of the shift register. It will only have one latch switching at a time. However, there is an additional capacitance due to the multiplexer and demultiplexer. Furthermore, the size of these multiplexer and demultiplexer increases with the number of stages. However, only one stage per cycle is active, which results in a constant switching activity inside these modules. But larger stages have a larger interconnect. Therefore, a slight increase in the active capacitance has to be expected which is not reflected in this equation. In order to compare the two implementations the equations (73) and (76) are set equal.

$$T \times m \times 0.25 \times C_{reg} = T \times 0.25\left(C_{reg} + C_{mux} + C_{demux}\right) \tag{77}$$

Equation (77) can now be written as:

$$\left(m-1\right) \times C_{reg} = C_{mux} + C_{demux} \tag{78}$$

Now it is easily seen that if the physical capacitance of the multiplexer and demultiplexer is lower than the total capacitance of the number of stages minus one, for the traditional shift register, the proposed approach will save active capacitance and will have a reduced power consumption.

Before comparing both general approaches different implementations of the multiplexer and demultiplexer are investigated with respect to power consumption, in order to decide on the most favourable implementation. Firstly, different demultiplexer implementations are investigated. The first design used a library demultiplexer supplied with the software library. The second idea was to use a custom demultiplexer using AND gates. The third design proposed is a gated clock implementation. Here only the clock of the reading stage is enabled via a simple AND gate. The results are shown in Table 5.16.

Implementation	Library Demux	Custom Demux	Gated Clock
Active Capacitance	180pF	140pF	46,5pF

Table 5.16: Comparison Between the Different Demultiplexer Implementation

As can be easily seen in the comparison of the different demultiplexer ideas the clock enable proved to be by far the most efficient implementation. Therefore, all further designs were implemented using this design. Figure 5.34 shows the principle of this method.

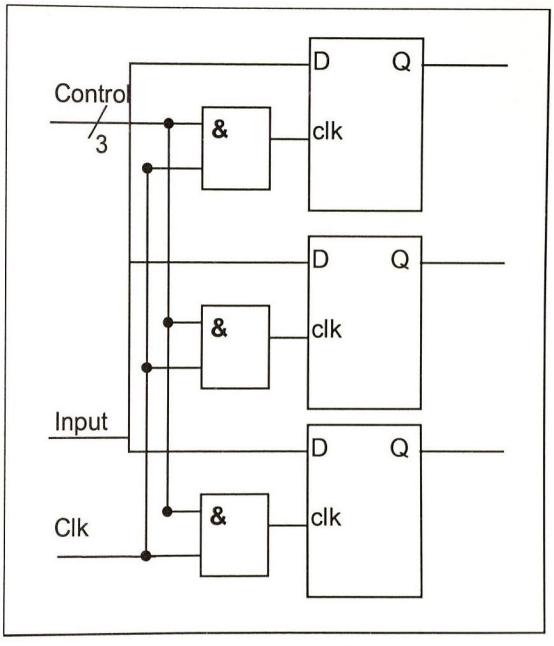


Figure 5.34: Implementation of the Demultiplexer

Here simple AND gates are used to drive the clock input of the latches. The main advantage of this method is the reduced physical capacitance of the inputs of an AND gate in comparison with a latch. The other input of the AND (control) is used to activate the AND gate enabling the latch. With such a design it is important to have no glitching activity in either the clock and the control signal. Otherwise the AND gates might be active at the wrong time and a new input will be applied to a latch, which should have been disabled. In the case of this design, this is true. If glitching is to be expected, then additional logic for removing the short pulses must be included in the design.

In order to validate equation (76), three traditional shift registers as discussed in the introduction to this chapter were implemented. The three designs were 4 bit, 8 bit and 16 bit wide shift registers. The shift length of these shift registers was varied from 4 to 128 stages.

When the active capacitance is plotted over the number of shift stages (Figure 5.35) the result is linear. This behaviour was expected, because of equation (76).

The shift register using a demultiplexer and multiplexer was implemented in the same way and the results of the simulations were plotted together with the results of the shift register in Figure 5.35. As can be seen for these designs, the active capacitance is slightly higher for small designs than the traditional approach. This was expected due to the additional capacitance provided by the multiplexer and demultiplexer. But after 8 stages the proposed method reduces the active capacitance significantly. For even larger implementations the advantage becomes even larger.

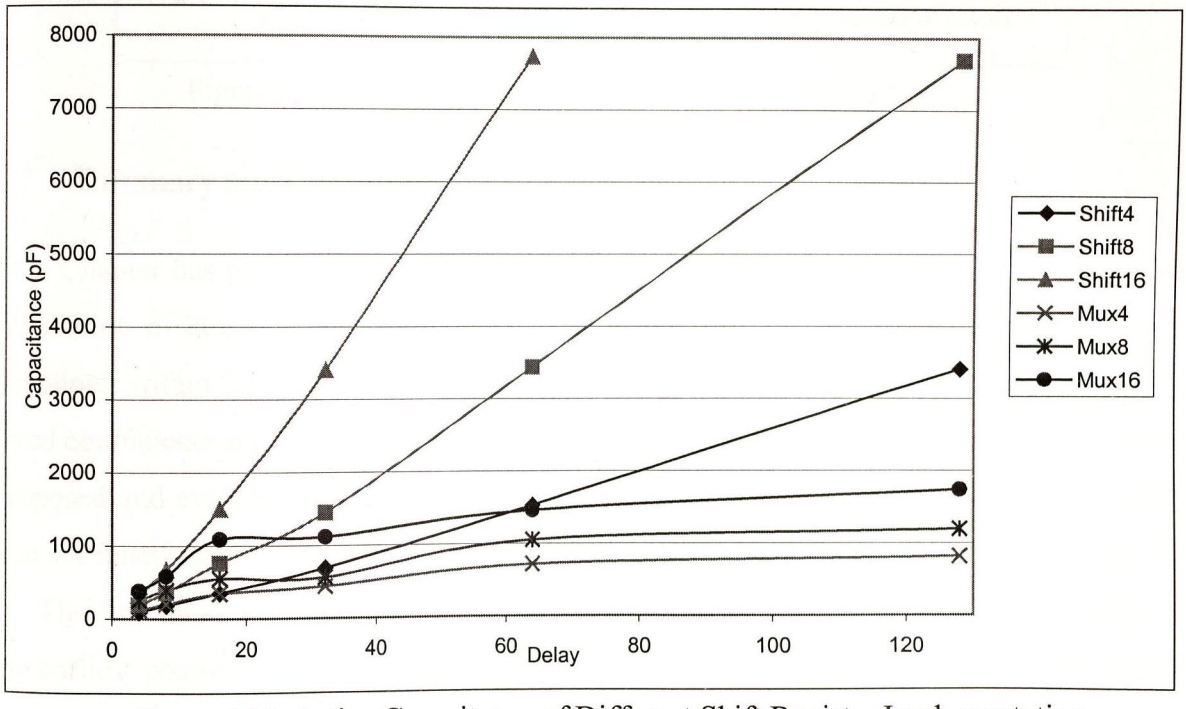


Figure 5.35: Active Capacitance of Different Shift Register Implementation.

The superior power consumption of the multiplexer method however comes at the cost of increased logic as shown in Figure 5.36. In this figure it can be seen that even for small designs like the 4-bit 4 stage shifter the area increases by 74% for the multiplexed implementation. For a 16-bit 8 stage shift register the increase in area rises to more than 100%. Therefore, the high power savings of the multiplexed version have always to be balanced against the higher area requirements.

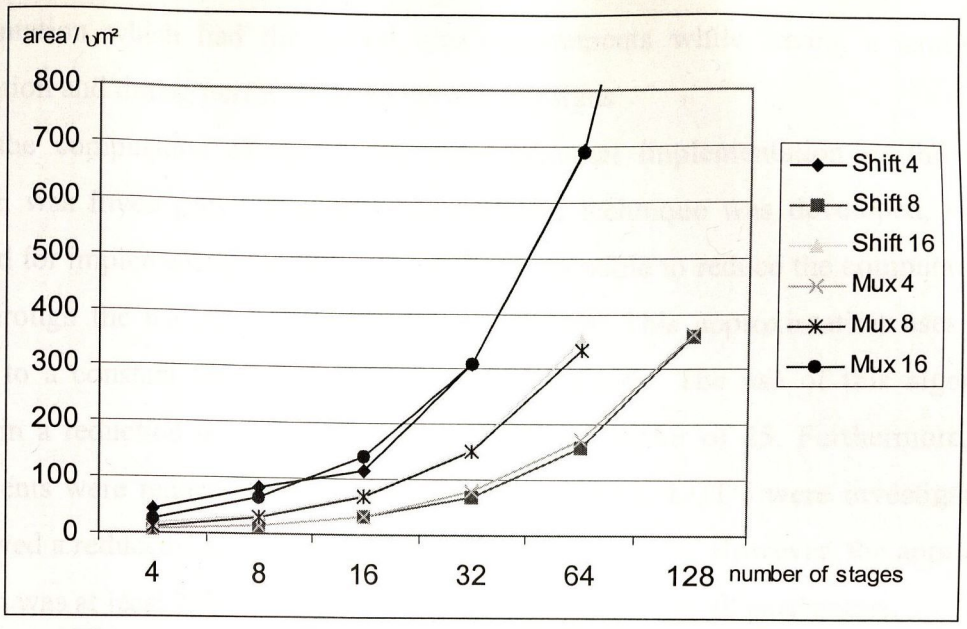


Figure 5.36: Area Requirements of the Different Shift Registers

5.7 Summary and Conclusions

This Chapter has presented a low-power implementation of Kender's Algorithm for the fast calculation of hue. The algorithm was chosen because of the extensive design challenges contained within it. These challenges, including trigonometric functions, multiplication by fixed coefficients and fast divisions, were addressed individually. A number of solutions were proposed and evaluated. To enable such a detailed investigation, the algorithm was divided into six functional blocks. Each block was implemented independently of the other blocks.

The first and most critical aspect of the implementation was the exclusion of the sign at the earliest possible stage. This then enabled an unsigned implementation, which consumes less power when compared to a signed implementation. Furthermore, this exclusion of the sign made it possible to restrict the computation of the arctan to positive input values only. This exclusion of the sign was possible by implementing a two stage comparison of the input signals. Theses two stages were implemented according to the probabilities of the inputs.

To compute the arguments of the arctan, a stage containing adders and dividers was implemented. It was shown that a larger tree implementation had a power advantage of a factor of three over a smaller, unbalanced implementation. The next computational stage required a division of the results of the previous stages. Four alternative divider designs were suggested. While all designs had a similar power consumption, the timing and area performance varied considerably. Therefore, it was possible to show that there was one

implementation which had the lowest area requirements while having a similar power consumption and timing performance to the other designs.

For the computation of the arctan, the traditional implementation of the CORDIC algorithm was investigated. A novel approximation technique was developed, which was optimised for implementation into hardware. It was possible to reduce the computation of the arctan through the use of an approximation technique. This approximation uses only one addition to a constant factor and only one shift operation. The use of this algorithm has resulted in a reduction in the power consumption by a factor of 25. Furthermore, the area requirements were reduced by a similar factor. In addition, LUT's were investigated. They also showed a reduced power consumption of approximately 25. However, the approximation algorithm was at least 20% better than the LUT's with respect to all parameters.

The delay stages were also investigated with regard to different coding styles. It was found that theoretically superior approaches such as the "one-hot" coding had a higher active capacitance when compared to theoretically inferior codes. A detailed analysis of the designs showed that this was due to the fact that in those designs, the clock was the dominant factor in power consumption. However, the coding must be evaluated together with an analysis of the control subsystem. In this subsystem, the control signal had to be delayed by seven cycles. This is usually achieved using a shift register. However, through the use of multiplexers, it was possible to reduce switching significantly. This lead to a reduced active capacitance. The next Chapter will present the implementation of the saturation and intensity paths of the design.

6 The Saturation and Intensity Algorithm

The previous chapter has presented a low-power implementation of the hue component of the RGB to HSI algorithm. This chapter will focus on a joint investigation of the remaining two components of the HSI algorithm. This combination of saturation and intensity has advantages over an implementation of the individual components.

This chapter first presents the basic design decisions and evaluates them. Four implementations of the combined saturation and intensity paths are presented and are evaluated for area, speed and power consumption. Finally, possibilities for improving the accuracy of the intensity algorithm, without compromising the power consumption, are presented.

6.1 Implementation Considerations of the Saturation and Intensity Algorithm

The sum of the three input values Red, Green and Blue is used in both algorithms (2) and (3). Therefore, this term must be calculated only once.

The divisor of the argument of the arctan function can be written as follows:

$$Divisor = X+Y-2Z \tag{79}$$

In this equation, the smallest of the three input values (R, G, B) is represented by Z. This value has been already extracted in the very first stage of the implementation of the hue algorithm. Therefore, it is now possible to use the term in equation (1) and add $3Z$ to it. This has the advantage that only one term must be added. It also gives a reduction in the number of pipeline stages. Due to the sorting algorithm used, the smallest value Z is already available at the input of the saturation algorithm. This operation is shown in (80).

$$R+G+B = X+Y-2Z+3Z \tag{80}$$

This operation would normally require three adders. However, operation with fixed numbers can often be simplified. In this case the multiplication by three can be split into $(Z+2Z)$ as shown in (81).

$$R+G+B = X+Y-2Z+(Z+2Z) \tag{81}$$

As demonstrated in (82), the constants in this equation can be expressed as powers of two.

$$R+G+B = X+Y-2Z+(2^0 Z+2^1 Z) \tag{82}$$

Because of (82) only one adder is required to perform the addition of *3Z*. The necessary multiplication by one and two are simple shift operations, which do not require any logic elements. Therefore, it is possible to reduce the number of adders required for the addition of R, G, B from 5 to 4. This has the advantage that a balanced tree adder structure can be used instead of an unbalanced adder design.

As described in Chapter 1, the saturation is the magnitude of the pointer to the pure spectral colour. If all three input signals have the same magnitude the resulting colour is achromatic. Therefore, the saturation is not defined and the computation of this path is disabled using a gated clock approach. To calculate an 8-bit output value of the saturation equation (2) is investigated. The maximum magnitude of the saturation as defined in (2) is one. Therefore, for an 8-bit implementation (2) has to be multiplied by 255.

$$Saturation = 255 - \frac{255 \times 3 \times \min(R,G,B)}{\sum R,G,B} \tag{83}$$

Such an implementation would require an 8-bit multiplier to realise this equation. However, if the numerator is multiplied by 256 the multiplier can be replaced by a shift operation in the divider. This would not require additional logic and power. Because the multiplication by three is not performed yet the maximum output of the saturation can be only close to one. Therefore, a possible overflow is prevented. After describing general approaches to implement the saturation and intensity part of the circuit, four different implementation possibilities will be presented.

6.2 Direct Implementation

The first implementation of the saturation algorithm uses a direct implementation of the saturation and intensity algorithm as illustrated in Figure 6.1.

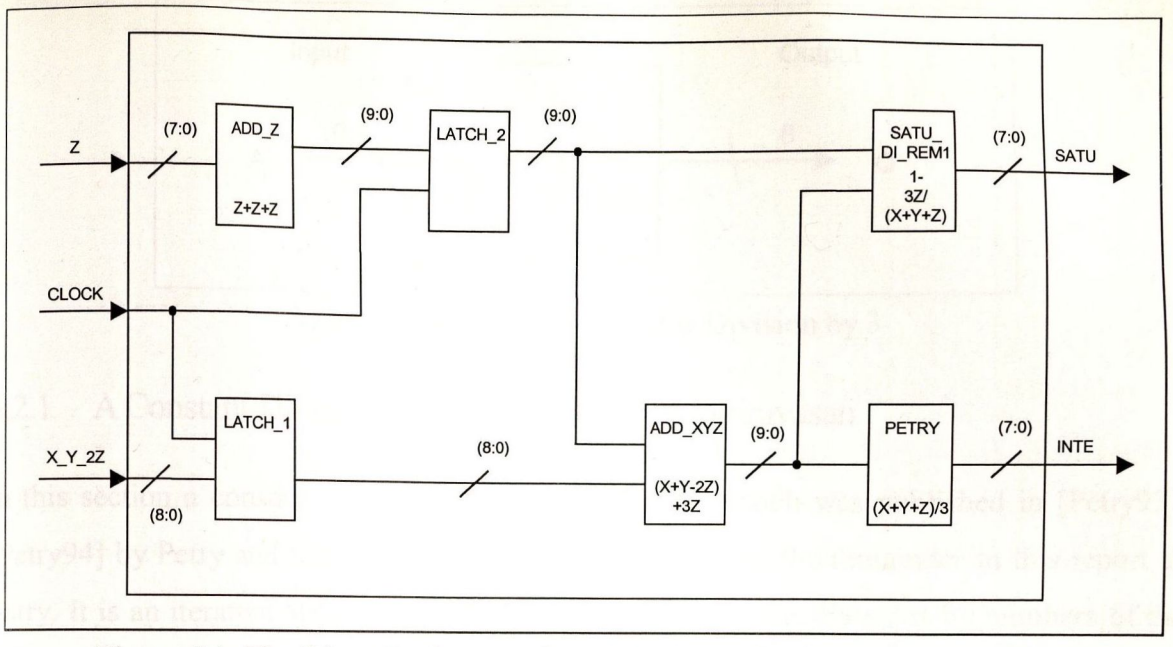


Figure 6.1: The Direct Implementation of the Saturation/Intensity Algorithm

The equations required to solve the saturation and intensity are shown in (84).

$$Saturation = 1 - \frac{3 \times \min(R,G,B)}{\sum R,G,B}$$

$$Intensity = \frac{\sum R,G,B}{3} \tag{84}$$

In this implementation, the term ΣR,G,B can be used in both the saturation and intensity parts. The second advantage is the constant division by 3. Such a divider can be implemented using an optimised structure. Various designs have been suggested in the past. In the following sections six approaches are presented. The notation below is used to describe the algorithms analysed:

Expression	Quotient	Divisor	Dividend	Remainder	Quotient bits	Number o/p bits	Number i/p bits
Notation	Q	D	A	R	q	m	l

Table 6.1: Notations for Describing the Analysed Algorithms

Figure 6.2 shows the block diagram for the implementation of the divider by 3. This design uses an input bitwidth of 10 bit for the dividend A and an output bitwidth of 8 bit for the quotient Q.

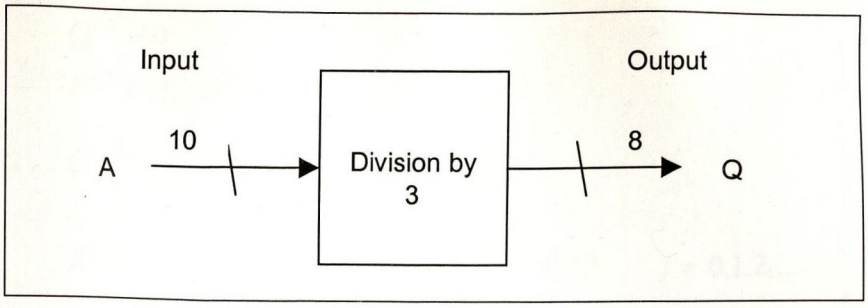


Figure 6.2: Block Diagram of the Division by 3

6.2.1 A Constant-Division Algorithm by Petry and Srinivasan

In this section a constant-division algorithm is analysed which was published in [Petry93], [Petry94] by Petry and Srinivasan. Therefore, it is referred in the remainder in this report as Petry. It is an iterative algorithm, which has been developed for divisions by numbers of the form 2^n+1 and 2^n-1. It represents these terms as 2^n-h, where h is either +1 or -1. The computation of the quotient can be generally expressed as follows:

$$Q = \sum_{i=1}^{m} (h)^{i-1} 2^{-in} A \tag{85}$$

As illustrated in Figure 6.3, it is possible to describe (85) as an array of grouped dividend bits. The equation must be rewritten to simplify the implementation. Shifts and additions can now be used to solve the quotient Q, where Q consists of quotient bits q_i and the remainder R.

$$\begin{array}{llllllllll}
 & h^0 & * & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & . & a_0 \\
+ & h^1 & * & & a_{m-1} & a_{m-2} & \cdots & a_2 & . & a_1 \\
+ & h^2 & * & & & a_{m-1} & \cdots & a_3 & . & a_2 \\
 & \vdots & & & & & \cdots & \vdots & & \vdots \\
+ & h^{m-2} & * & & & & & a_{m-1} & . & a_{m-2} \\
+ & h^{m-1} & * & & & & & & . & a_{m-1} \\
\hline
 & & & q_{m-2} & q_{m-3} & q_{m-4} & \cdots & q_0 & . & R
\end{array}$$

Figure 6.3: Iterative Division by 2^n+1 and 2^n-1

According to the formulas described in [Petry93], the partial quotient Q_i and the partial remainder R_i can be expressed as follows

$$
\begin{aligned}
Q^{(0)} &= 0 \\
R^{(0)} &= A_1^{(0)} A_2^{(0)} \cdots A_k^{(0)} \\
Q^{(j+1)} &= Q^{(j)} + \sum_{i=1}^{k_j-1} h^{i-1} A_1^{(j)} A_2^{(j)} \cdots A_{k_j i}^{(j)} \\
R^{(j+1)} &= \sum_{i=1}^{k_j-1} h^{i-1} A_{k_j-i+1}^{(j)} = A_1^{(j+1)} A_2^{(j+1)} \cdots A_{k_j+1}^{(j+1)} \qquad j = 0,1,2,...
\end{aligned}
\tag{86}
$$

Here, k is a number of n-bit digits and l is the bitwidth. For such successive iterations the partial remainder is used as input and the algorithm stops when the partial remainder R consists of a single bit group or digit.

Equations (86) have been analysed and optimised for carrying out the division by 3. It can be seen from (85) that the use of $h=1$ is preferable when performing this division in order to avoid an alternating series. The latter has a larger power consumption than a non-alternating one. Moreover, the dividend A is independent of m and the equation can be simplified to

$$Q = A\sum_{i=1}^{m} 2^{-in} \tag{87}$$

Now the divisor 3 is equal to the form $2^n - 1$. Therefore, n is equal to 2 and m depends only on the bitwidth of the dividend A. Furthermore, the input bitwidth is 10. Therefore, m has to be set to 10. Hence, m is now independent of the bitwidth and the equation can be expressed as follows

$$Q = A\sum_{i=1}^{10} 2^{-2i} \tag{88}$$

This constant division algorithm as presented in [Petry93] contains a higher accuracy than required. For an accuracy of one bit only 4 terms instead of 10 are required. The smallest term is now equal to 2^{-8}. For this reason the maximum deviation can be written as:

$$Maximum\ deviation = \frac{256}{2^8} = 1\,\text{bit}$$

As described above, a deviation of 1 bit occurs while the quotient ranges from 0 to 255. Now it is possible to reduce the number of terms required for the division by 3, from 10 to 4. The

main reasons for a term reduction will be explained in the following section. The optimised formula can be generalised as:

$$Q = A\sum_{i=1}^{4} 2^{-2i} = A*(2^{-2} + 2^{-4} + 2^{-6} + 2^{-8}) \tag{89}$$

In the equation above it can be seen that the division by 3 is reducible to a multiplication by a value which is a close approximation of 1/3. To explain this an example is given. In this example the dividend A is equal to 168 and m is equal to 4 and then the following holds:

$$A = 168_{10} = 0010101000_2 = (2^7 + 2^5 + 2^3)$$
$$Q = (2^7 + 2^5 + 2^3)*(2^{-2} + 2^{-4} + 2^{-6} + 2^{-8})$$
$$Q = 2^5 + 2^4 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-5}$$
$$Q = 110111{,}11001_2$$
$$Q = 55{,}78125_{10} \approx 56_{10}$$

Figure 6.4: Example for a Constant Division by 3

As illustrated in Figure 6.5, only shifts and additions are used. For this reason the algorithm is very simple to implement.

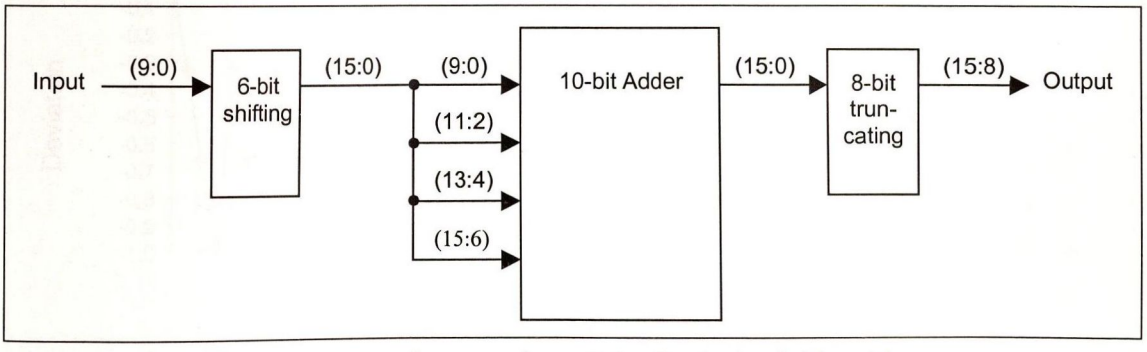


Figure 6.5: Implementation of the Optimised Algorithm

As shown in Figure 6.4, the quotient contains several digits after the decimal point. If truncated the quotient will be 55 instead of 56. Therefore, it is necessary to investigate the accuracy. Figure 6.6 shows the accuracy of the optimised constant-division by Petry and Srinivasan in comparison to general division by 3.

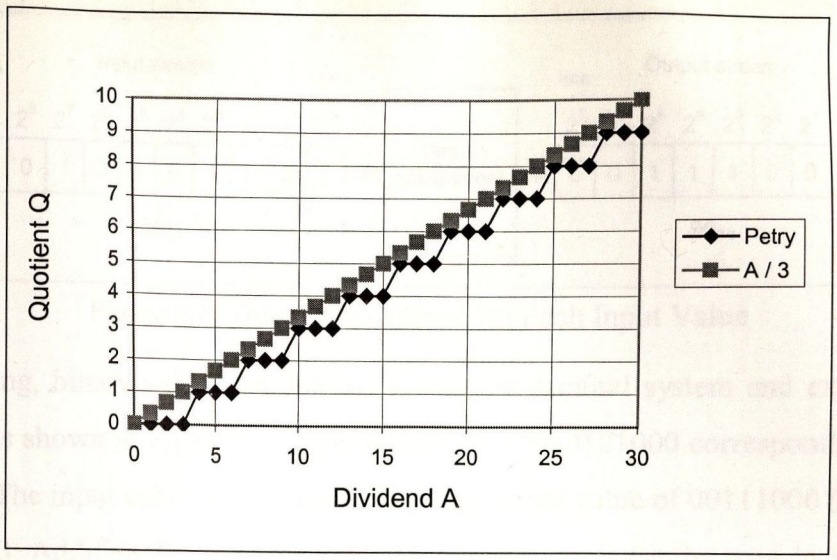


Figure 6.6: The Accuracy of the Petry Algorithm

It can be seen that the deviation using Petry has a maximum deviation of –1 bit in comparison to the theoretical division by 3. Because of this small deviation of 1 bit, it is possible to use the optimised algorithm for the implementation of the RGB to HSI algorithm.

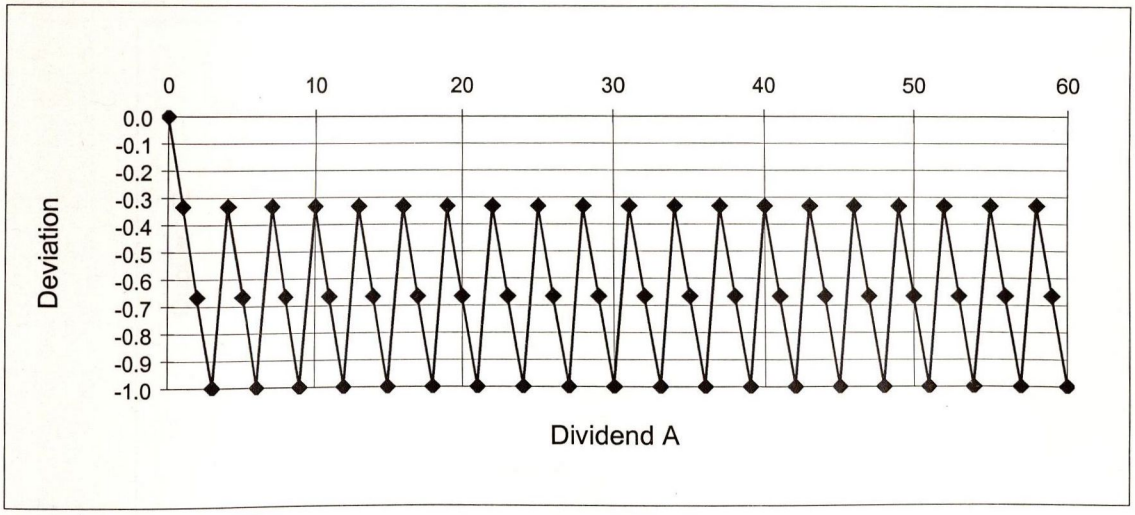


Figure 6.7: The Error Deviation of Petry

6.2.2 The Lookup Tables

To satisfy the requirements of the division by 3, it is not necessary to use reprogrammable integrated circuits. As previously explained for the arctan function, a LUT can be used to store the result for all possible input values. In the following example the value of the output for an input value divided by 3, is illustrated.

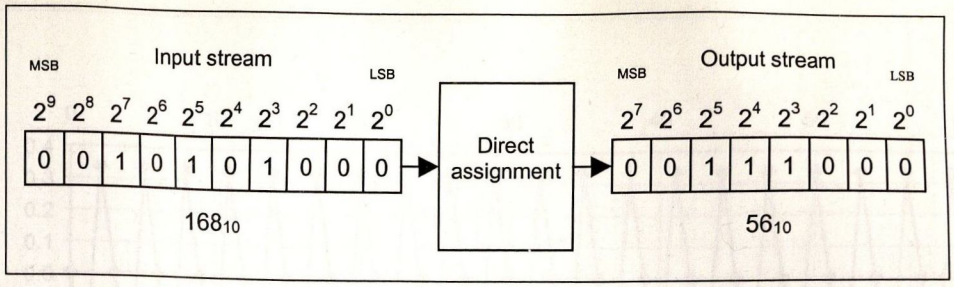


Figure 6.8: Direct Assignment for Each Input Value

In the following, binary values are converted into the decimal system and expressed in the form 168_{10}. As shown in Figure 6.8, an input value of 0010101000 corresponds to a value of 168 decimal. The input value will be assigned to the output value of 00111000 (56_{10}) which is a division by 3. Additionally, the accuracy was investigated. It can be seen in Figure 6.9 that positive and negative errors occur in the output value. The ordinary division graph lies between these errors and represents the right non-truncated calculation. The deviation of the LUT ranges from 1/3 to –1/3 as shown in Figure 6.10. The maximum error of the output values will have an absolute error of 1/3.

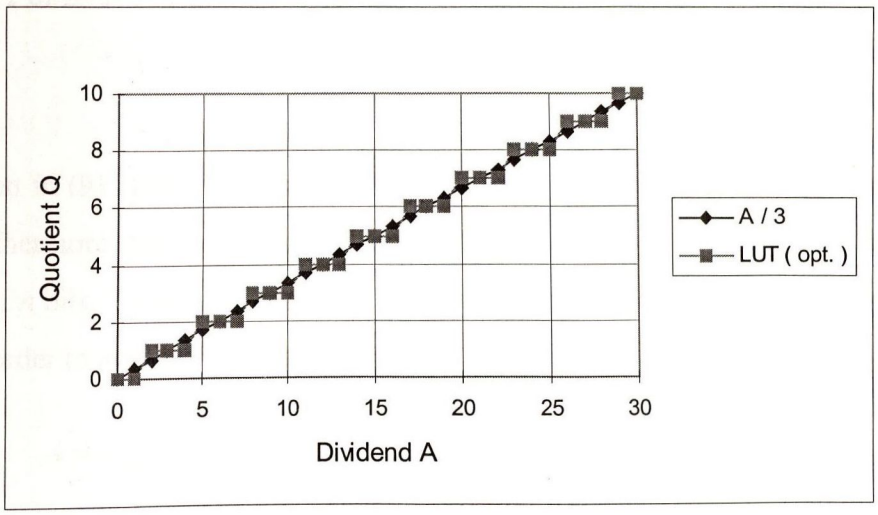


Figure 6.9: The Accuracy of the LUT

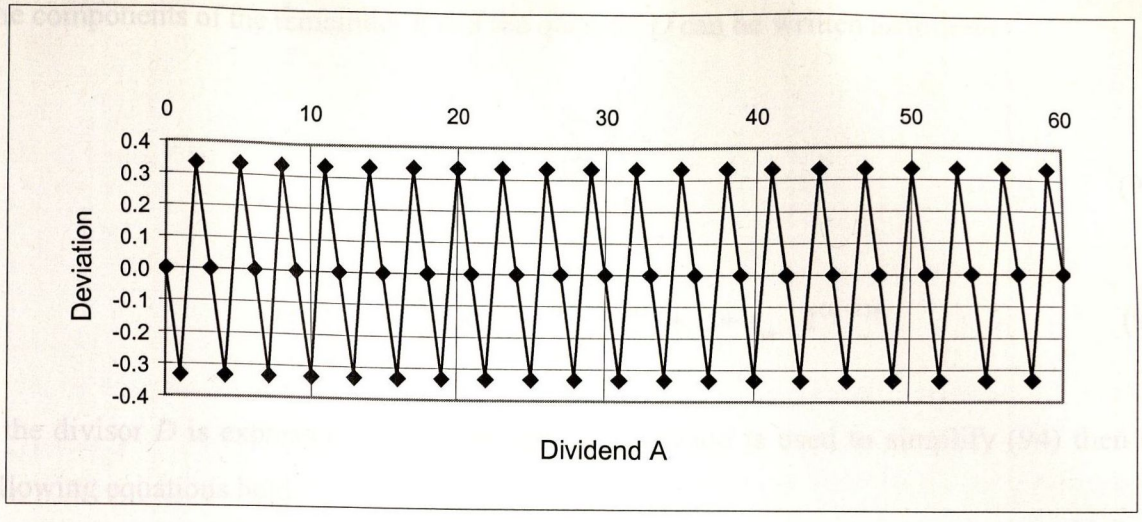


Figure 6.10: The Error Deviation of the LUT

6.2.3 An RNS based Division Architecture for Constant Divisors

This section presents the Residue Number System (RNS) based divider algorithm as published in [Albe97]. The architecture of the RNS either can be used for constant divisors of the form 2^n+1 or 2^n-1. The dividend A can be written as follows:

$$A = a_{N-1}a_{N-2}....a_2a_1a_0 \tag{90}$$

It can be seen in (91) that the dividend is a positive N-bit number where the number of bits $m=N/n$. Furthermore, the dividend A can be written as a number of m digits where each digit A_i consists of n bits. Therefore, it may be necessary to append zero bits to the most significant bits of A in order to present $N=nm$ as follows:

$$A = A_{m-1}A_{m-2}....A_1A_0 = A_{m-1}2^{n(m-1)}+....+A_2 2^{2n} + A_1 2^n + A_0 2^0 \tag{91}$$

Taking $A=DQ+R$ and replacing the value of A in (92) the equation can be written as:

$$\frac{A}{D} = \frac{A_0 2^0 + A_1 2^n + A_2 2^{2n}+....+A_{m-1}2^{n(m-1)}}{2^n - 1} \tag{92}$$

Since $2^{2n} = 2^n * 2^n$ and $2^n = (2^n -1) +1$, (93) can be simplified by using $2^n -1$ as shown in (94):

$$\frac{A}{D} = \frac{\sum_{i=0}^{m-1} A_i}{2^n - 1} + \sum_{i=1}^{m-1} A_i + \sum_{i=2}^{m-1} A_i 2^n + \sum_{i=3}^{m-1} A_i 2^{2n} + ... + A_{m-1} 2^{(m-2)n} \tag{93}$$

The components of the remainder R and the quotient Q can be written as follows:

$$R = \frac{\sum_{i=0}^{m-1} A_i}{2^n - 1} \tag{94}$$

$$Q = \sum_{i=1}^{m-1} A_i + \sum_{i=2}^{m-1} A_i 2^n + \sum_{i=3}^{m-1} A_i 2^{2n} + \ldots + A_{m-1} 2^{(m-2)n} \tag{95}$$

If the divisor D is expressed as 2^n+1 and the same method is used to simplify (94) then the following equations hold:

$$R = \frac{\sum_{i=0}^{m-1} (-1)^i A_i}{2^n - 1} \tag{96}$$

$$Q = \sum_{i=1}^{m-1} (-1)^{i+1} A_i + \sum_{i=2}^{m-1} (-1)^i A_i 2^n + \sum_{i=3}^{m-1} (-1)^{i+1} A_i 2^{2n} + \ldots + A_{m-1} 2^{(m-2)n} \tag{97}$$

A careful analysis reveals that it is useful to focus on the form 2^n-1 for the divisor to avoid alternating series. Therefore, it is only necessary to solve the quotient Q as shown in section 6.2. The following example describes division by 3 using an input value of 168_{10}. Therefore, the divisor $D=3$ can be expressed as:

$$3 = 2^n - 1 \Rightarrow n = 2 \tag{98}$$

As shown in section 6.2.1, it is desirable to implement the division using a non-alternating structure. It is possible to use the RNS based algorithm for the decimal system computation but this algorithm is based on the binary system computation.

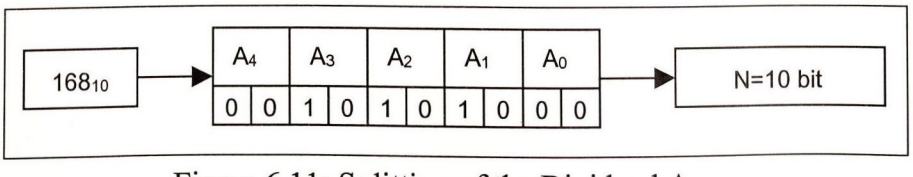


Figure 6.11: Splitting of the Dividend A

Firstly, the input value must be analysed and split into different bits as illustrated in Figure 6.11. This analysis is necessary for pre-calculating m which can be written as:

$$m = \frac{N}{n} = 5 \tag{99}$$

Next, (95) must be modified as follows:

$$Q = \sum_{i=1}^{4} A_i + \sum_{i=2}^{4} A_i * 2^2 + \sum_{i=3}^{4} A_i * 2^4 + \sum_{i=4}^{4} A_i * 2^6 \tag{100}$$

It can be seen in (100) that it is not necessary to compute the digit A_0. Only the digits A_1-A_4 are involved in the computation. They are calculated as follows:

$$A_1 = 1*2^1 + 0*2^0 = 2$$
$$A_2 = 1*2^1 + 0*2^0 = 2$$
$$A_3 = 1*2^1 + 0*2^0 = 2$$
$$A_4 = 0*2^1 + 0*2^0 = 0$$

Figure 6.12 Computation of the Digits A_1-A_4

A defined shift and addition of the digits A_1-A_4 is computed as:

$$Q = (2+2+2+0)*2^0 + (2+2+0)*2^2 + (2+0)*2^4 + (0)*2^6 = 54_{10} \tag{101}$$

The result of a division by 3 with an established bit deviation of 2 can be written as:

$$Q = 54_{10} \Rightarrow 00110110_2 \tag{102}$$

As seen in Figure 6.13, the division by 3 is also reducible to a combination of shifts and adders. The input bitwidth of 10 bit is split into digits of 2 bits.

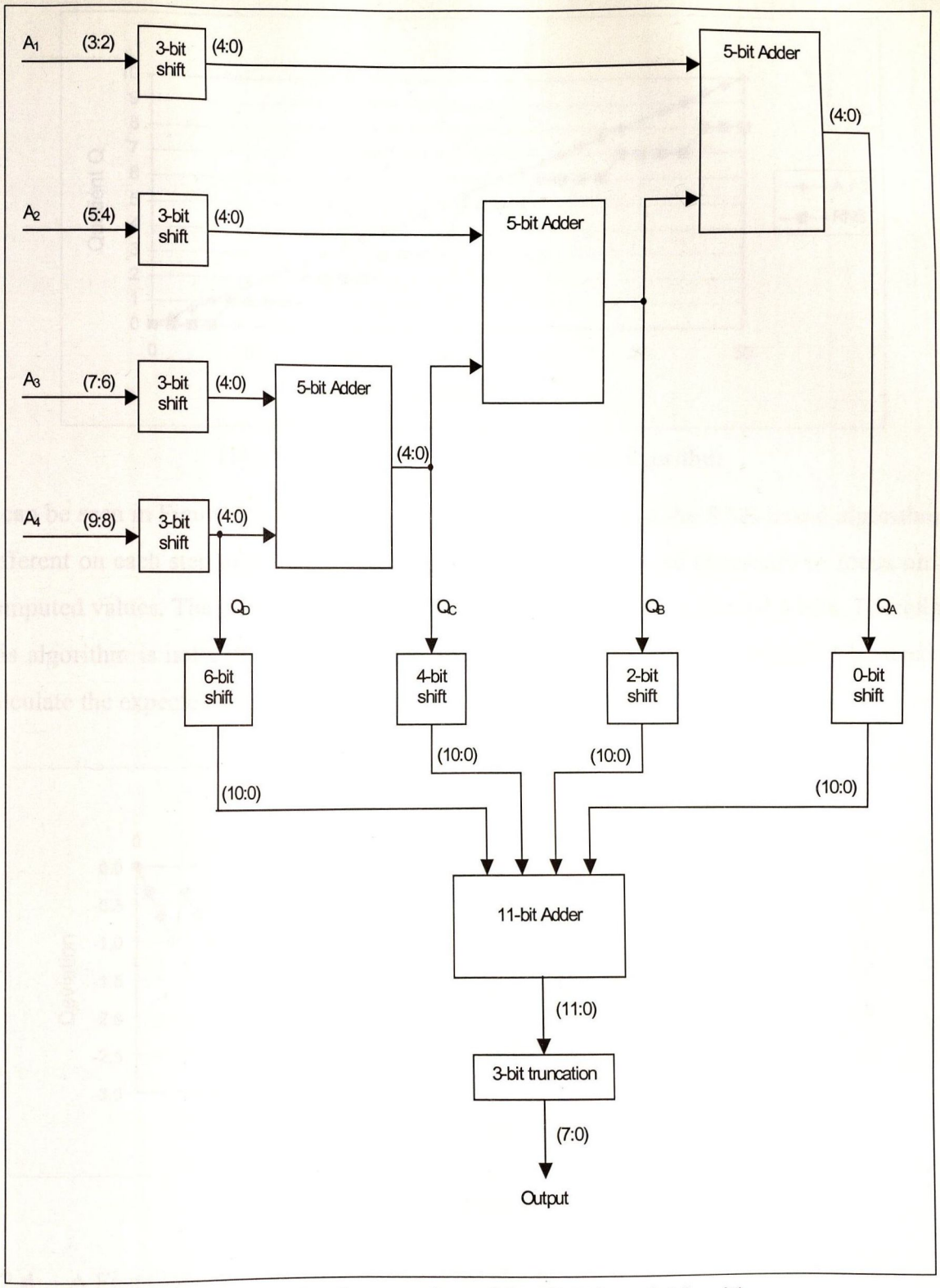


Figure 6.13: Implementation of the RNS based Algorithm

Figure 6.14 shows the deviation between ordinary division by 3 and the RNS-based algorithm.

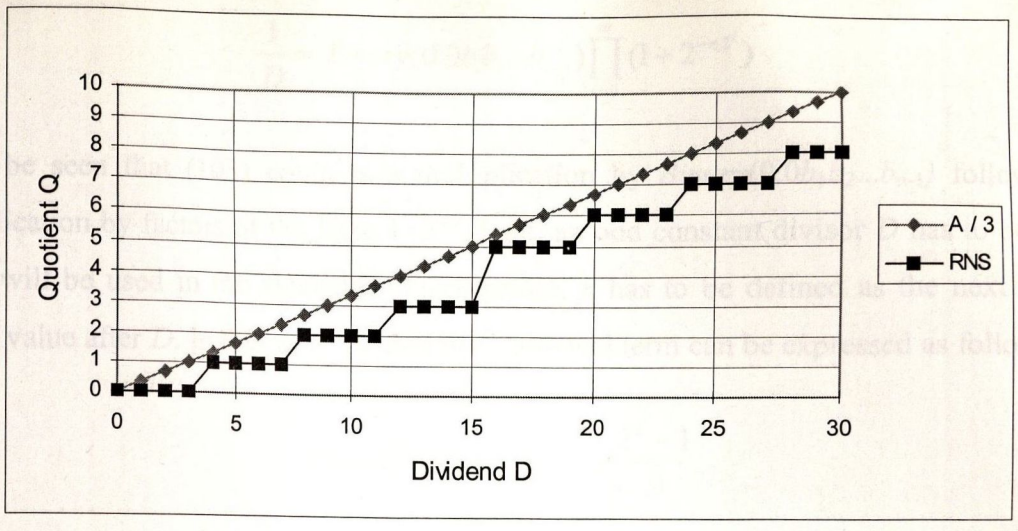


Figure 6.14: The Accuracy of the RNS algorithm

It can be seen in Figure 6.14 and Figure 6.15 that the deviation of the RNS-based algorithm is different on each step of the computation. For this reason it is also necessary to focus on all computed values. The presented structure contains a maximum deviation of 4 bits. Therefore, this algorithm is not useful when performing the division by 3 and can thus not be used to calculate the expected result.

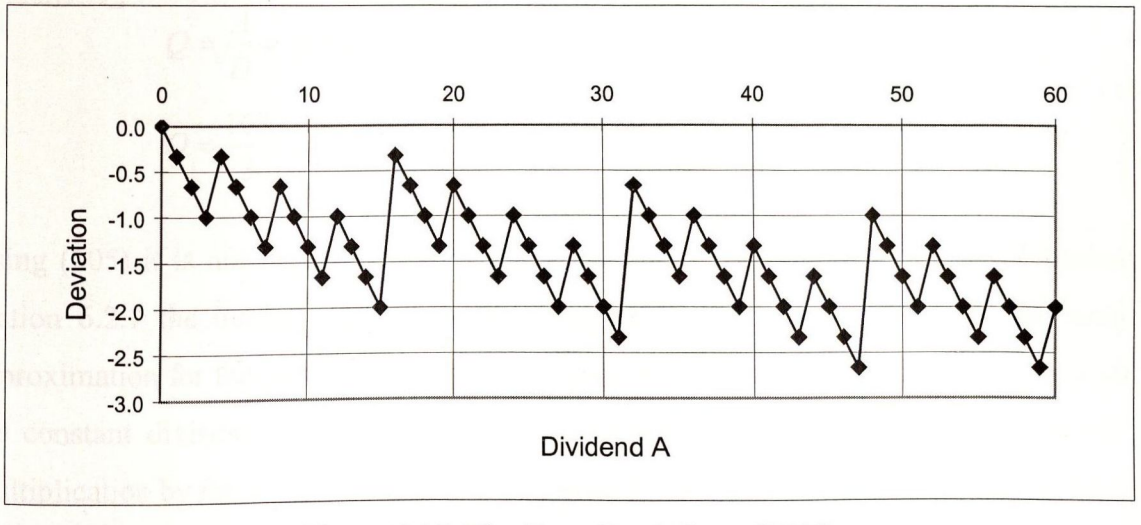


Figure 6.15: The Error Deviation of RNS

6.2.4 A Fast Constant Division Routine by Shuo-Yen Robert Li

In this section another constant-division routine will be shown which was presented in [Li85]. By using *Euler`s function* and *Fermat`s Little theorem* this division algorithm can be generalised as a multiplication with the reciprocal number of the divisor D. Equation (103) shows that the multiplication is an approximation to $1/D$.

$$\frac{1}{D} = Binary(0.0b_1b_2...b_{n-1})\prod_{i=0}^{\infty}(1+2^{-n2^i}) \tag{103}$$

It can be seen that (103) contains a multiplication by $Binary(0.0b_1b_2...b_{n-1})$ followed by multiplication by factors of the form $1+2^{-m}$. First, an odd constant divisor D has to be found which will be used in the algorithm. Furthermore, n has to be defined as the next smaller integer value after D. In this way, the first mathematical term can be expressed as follows:

$$Binary(0.0b_1b_2...b_{n-1}) = \frac{2^n - 1}{D} \tag{104}$$

As shown in (104), it is necessary to determine the bitwidth of the binary expression on the right-hand side of the equation depending on D and n. The form $1+2^{-m}$ can be expressed as an approximation of a infinite product series as shown in (104). A small example illustrates the use of the division routine. In the following routine the divisor D is equal to 3 and the dividend A is equal to 168. First n must be defined. In this case n is equal to 2. According to (104), $b_1 = 1_2$ and the approximated product of the infinite series can be expressed as follows:

$$\begin{aligned} Q &= \frac{A}{D} = A * Binary(0.0b_1) * \prod_{i=0}^{k}(1+2^{-n2^i}) \quad k = 0,1,2 \\ Q &= \frac{168}{3} = 168 * 2^{-2} * \left(1+2^{-2*2^0}\right)\left(1+2^{-2*2^1}\right)\left(1+2^{-2*2^2}\right) = 55.9991 \end{aligned} \tag{105}$$

Using (105) it is not necessary to solve all products of the infinite series. As described in section 6.2.1 the implementation of the algorithm by Petry and Srinivasan also uses an approximation for the divisor D. The only difference is the expression for $1/D$. In the case of the constant division routine by Li a multiplication by $Binary(0.0b_1b_2...b_{n-1})$ followed by multiplication by factors of the form $1+2^{-m}$ is used to express this term. On the other hand the constant division algorithm by Petry and Srinivasan is expressed as a weighted sum of the value 1/3.

Figure 6.16 shows the accuracy of the fast constant-division by Shuo-Yen Robert Li in comparison to general division by 3.

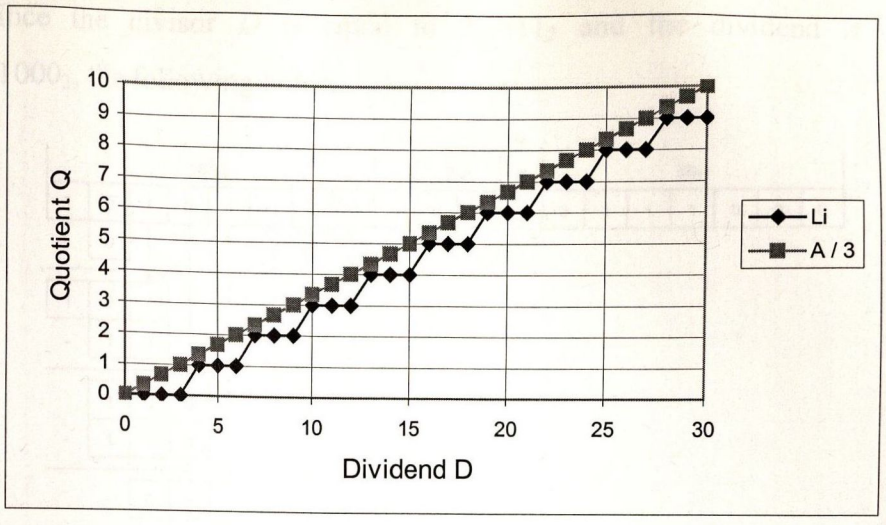


Figure 6.16: The Accuracy of the Li Algorithm

From a comparison of Figure 6.6 and Figure 6.16 it can be seen that the constant-division has the same accuracy and deviation as the algorithm of Petry and Srinivasan as described in section 6.2.1.

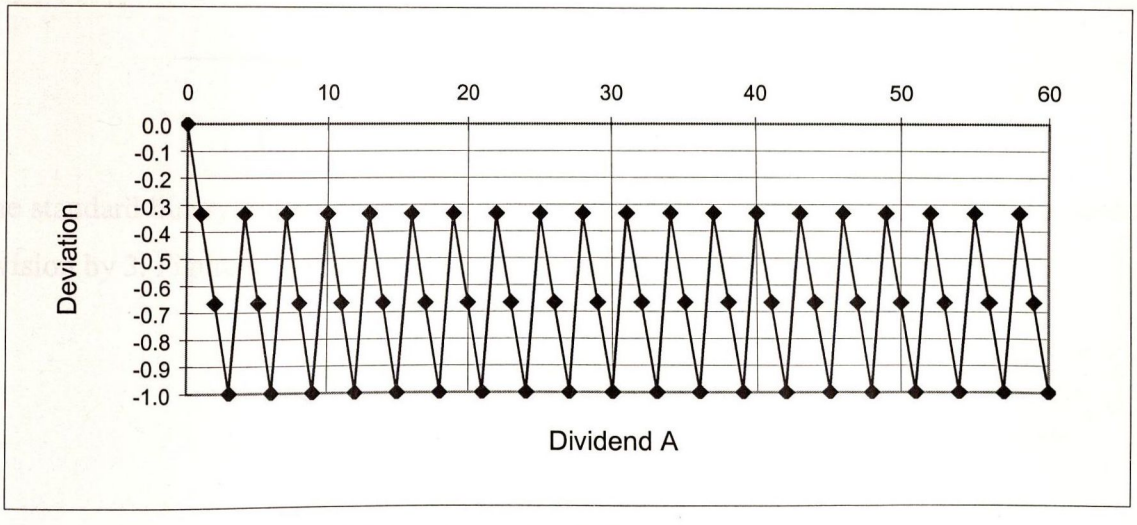


Figure 6.17: The Error Deviation of the Li Algorithm

6.2.5 The Standard Binary Divider

In the following section another way to perform the binary division will be described. Binary division is basically a procedure to determine how many times the divisor D divides the dividend A resulting in the quotient Q. At each step in the process, the divisor D either divides into a group of bits or it does not. Therefore, the quotient either is a 1 or a 0. Moreover, the divisor divides a group of bits when the divisor has a value less than or equal to the value of those bits. The example in Figure 6.18 shows the procedure for binary

division. Since the divisor D is equal to $3_{10}=11_2$ and the dividend A is equal to $168_{10}=10101000_2$, the following holds:

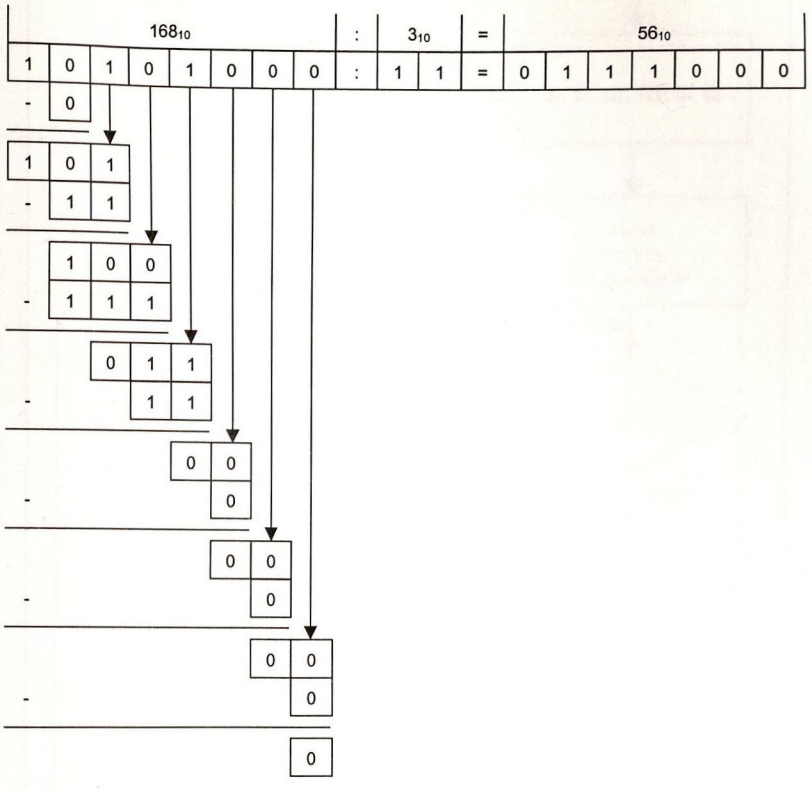


Figure 6.18: Example of a Standard Division by 3

The standard binary divider (SBD) has been analysed and optimised with respect to constant division by 3. Figure 6.19 shows the implementation of the SBD.

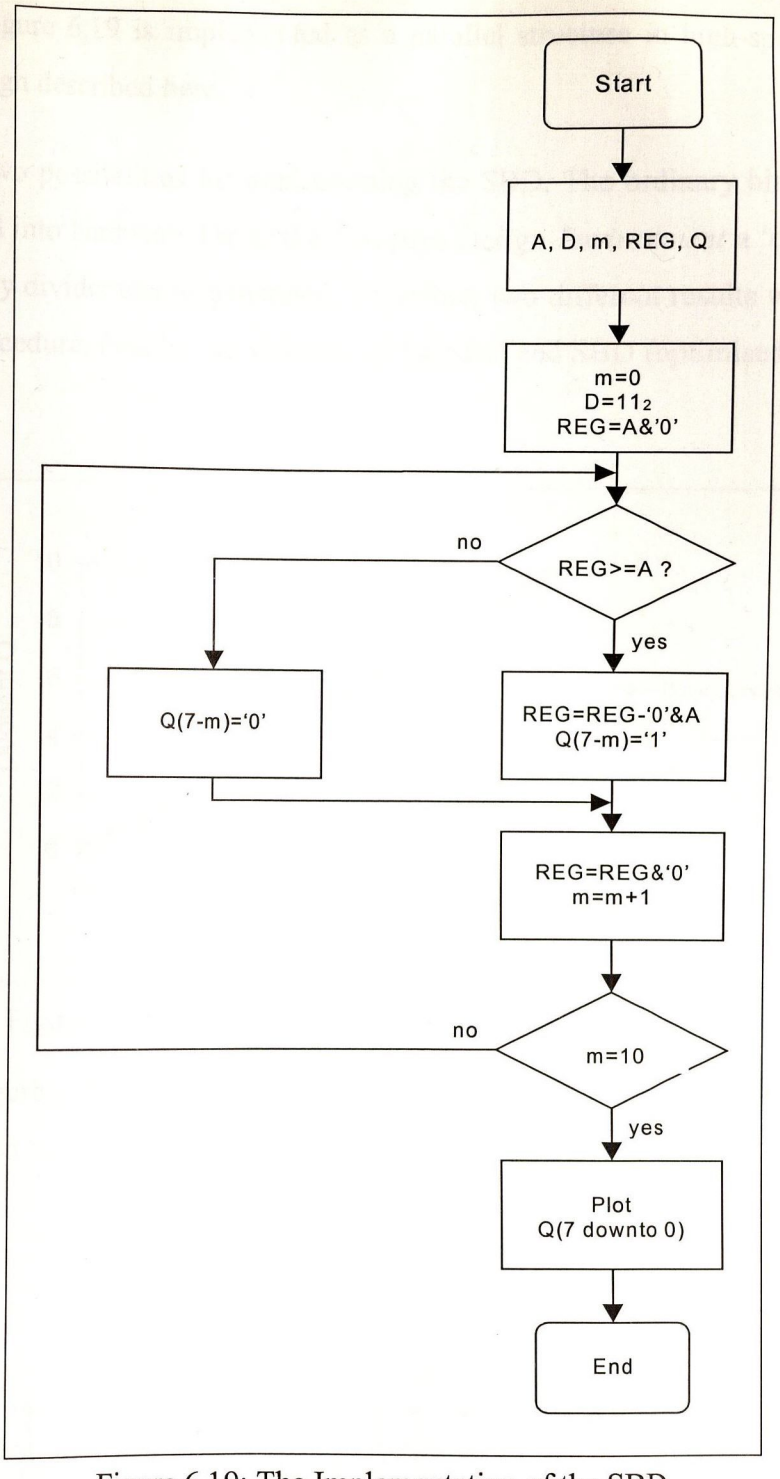


Figure 6.19: The Implementation of the SBD

It can be seen that there is the additional signal REG for solving the quotient Q. First there will be a query. If REG is smaller or equal to the dividend a subtraction will be carried out and the bit of the quotient will be assigned the value '1'. Otherwise the quotient bit will be assigned a value of '0'. The cycle runs as long as the bitwidth m is smaller than 10. After the process the quotient will be available at the output. It should be noted that the serial structure

suggested in Figure 6.19 is implemented as a parallel structure in high-speed applications, such as the design described here.

There are two possibilities for implementing the SBD. The ordinary binary division can be implemented into hardware. Using the *Synopsys Design Environment* a 'division by three' optimised binary divider can be generated. Therefore, two different results will be shown for the division procedure. Finally, the accuracy of the SBD and SBD (optimised) is illustrated in the Figure 6.20.

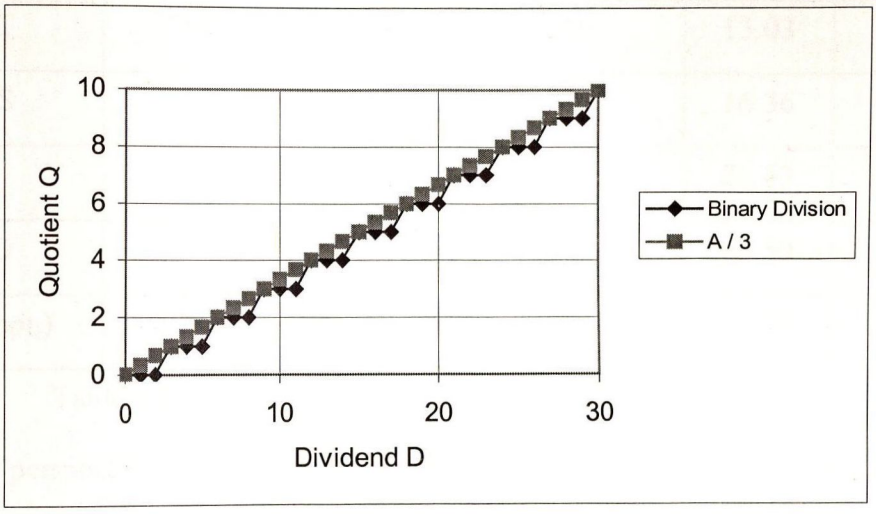


Figure 6.20: The Accuracy of the SBD and SBD (optimised)

As shown in Figure 6.20 the SBD and the SBD (optimised) have exact the same accuracy as truncated division by 3. The computed negative deviations are 0, -1/3 and –2/3 and are shown in Figure 6.21.

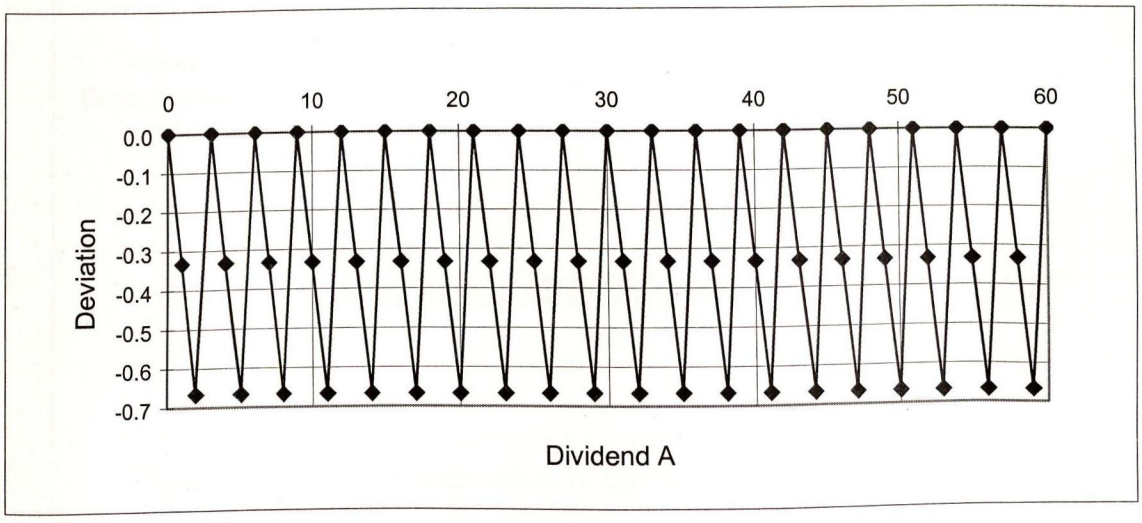


Figure 6.21: The Error Deviation of the SBD and SBD (optimised)

6.2.6 Features of the Divider Algorithms

Table 6.2 shows the characteristics of the different solutions, which were obtained by the Synopsys Design Environment and represent the most important features of each solution.

Version	Active Capacitance / pF	Number of nets	Power Consumption / mW	Max. Delay / ns	Total area / μm^2
Petry	1.95	63	1.63	21.43	93.10
LUT	7.91	548	6.59	13.03	727.99
RNS	1.94	50	1.62	16.36	68.25
Li	1.95	63	1.63	21.43	93.10
SBD	36.64	235	30.53	64.50	307.08
SBD (opt.)	2.68	48	2.23	22.79	56.88

Table 6.2: Characteristics of the Constant Divider Structures

For a clearer perspective, the results of the table above are presented graphically. Figure 6.22 shows the power consumption for the different algorithms.

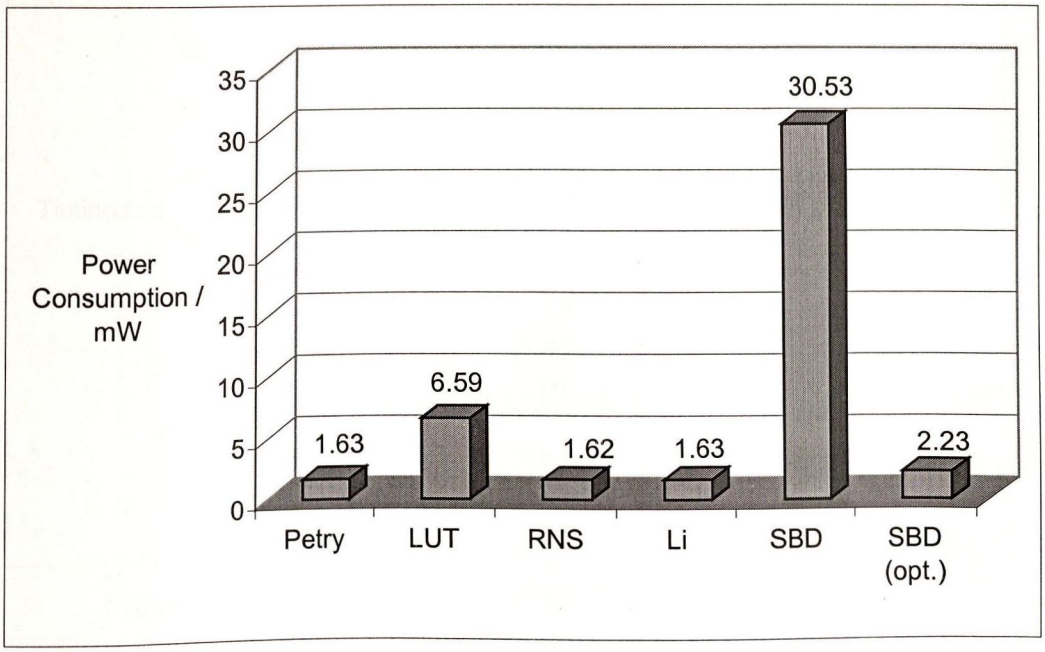


Figure 6.22: Power Consumption of the Constant Divider Structures

In comparison to the algorithm by Petry and Srinivasan, the RNS based algorithm and the fast constant division routine by Shuo-Yen Robert Li, the SBD requires approximately 95% more power. The SBD needs the most power at 30.53mW. Moreover, the SBD (optimised) needs only 7% of the power of the SBD. The LUT requires 6.59mW, which is 21% of the power of the SBD. Therefore, the SBD and the LUT are not useful for implementation because of the larger power consumption.

The timing behaviour of the different algorithms is shown in Figure 6.23. To perform the division by three it is necessary to compute the quotient Q in t_{limit}= 30ns. As illustrated in Figure 6.23 the SBD cannot be used to perform this division, because of a time of 64.5ns. The reason for this is that 10 comparator stages are included in this design. All other implementations can be used to compute the result in time. Using a simple structure and no computation process the Lookup Table has the fastest timing behaviour with 13.03ns. The algorithm by Petry and Srinivasan and the fast constant division routine by Li need the same time because of the same implementation structure with adders. The RNS based algorithm requires approximately 25% and the SBD (optimised) needs 35% of the time of the slowest structure, the SBD.

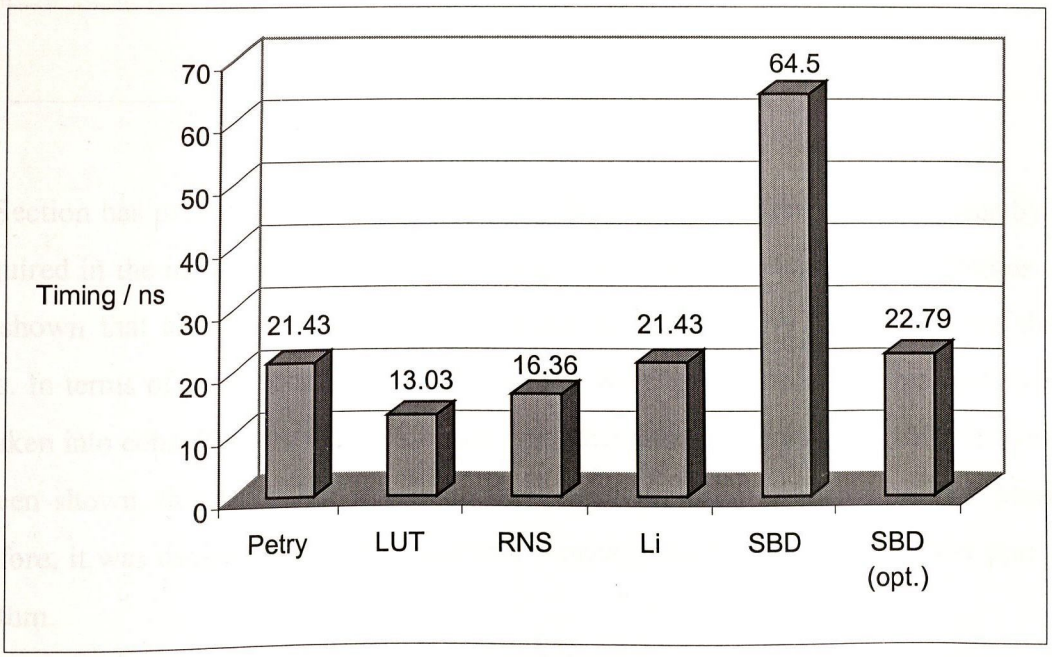


Figure 6.23: Timing Behaviour of the Constant Divider Algorithm

Figure 6.24 shows the required area in μm^2 of the implemented solutions. It can be seen that the Lookup Table requires the most area because all possible output values have to be stored. For this reason this solution does not represent a useful to implementation. As illustrated in

Figure 6.24, the SBD needs 42.2% of the area of the LUT which requires an area of approximately 728μm^2. The algorithm by Petry and Srinivasan and the fast constant division routine by Shuo-Yen Robert Li need the same area because they use the same implementation. Both structures require an area of 12.8% of the area of the Lookup Table. Using the RNS-based algorithm it is possible to perform the division by three with an area of 9.4% of the LUT. The best solution in respect to area is the SBD (optimised) which only needs 7.8% of the area of the LUT.

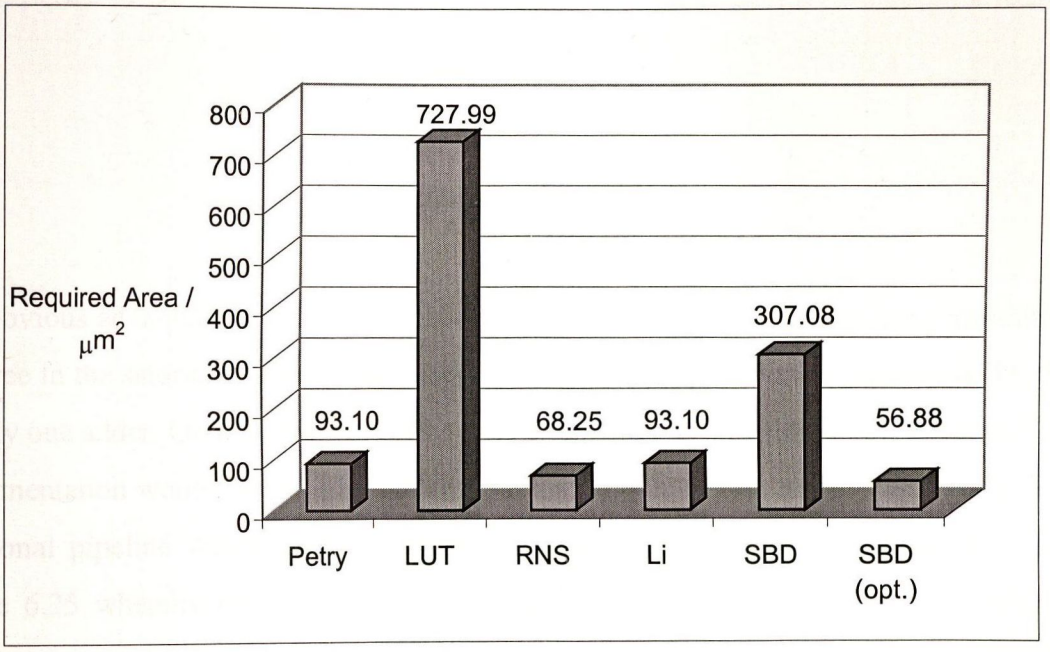


Figure 6.24: The Area Requirements

This Section has presented a detailed analysis of the implementation of the divider by three, as required in the intensity part of the algorithm. The analysis consists of six designs. It has been shown that algorithms which are optimised for division by constants gave the best results. In terms of power consumption, the results were quite similar. If area and speed are also taken into consideration, the RNS based implementation appears preferable. However, as has been shown, this algorithm is less accurate than the other implementations presented. Therefore, it was decided to use the algorithm proposed by Petry in the intensity part of the algorithm.

The LUT is, as expected, the fastest and largest implementation. However, as the implementation of Petry meets the timing requirements with smaller area and power consumption the LUT was not taken into consideration. This section also showed the advantages of using optimised designs, as opposed to standard modules. Standard modules

are frequently used in industry to shorten the design cycle. However, as was shown in this section, the use of optimised designs resulted in an improvement in area, speed and power of at least three.

6.3 Second Implementation of the Saturation/Intensity Algorithm

The second implementation of the saturation/intensity algorithm is based on use of the intensity output in the saturation algorithm (107).

$$Intensity = \frac{\sum R,G,B}{3}$$
$$Saturation = 1 - \frac{\min(R,G,B)}{Intensity} \tag{106}$$

The obvious advantage is the simplification of the multiplication. In this case, multiplication by three in the saturation was not necessary. This multiplication can be performed by the use of only one adder. On the other hand this implementation also has its disadvantages. Such an implementation would increase the maximum path length by one stage. This results in three additional pipeline stages each containing an 8-bit latch. This new structure is shown in Figure 6.25 whereby the additional blocks are emphasised. Therefore, the reduced power consumption in the saturation part of the RGB to HSI algorithm has to be balanced with the overall increase in the other paths.

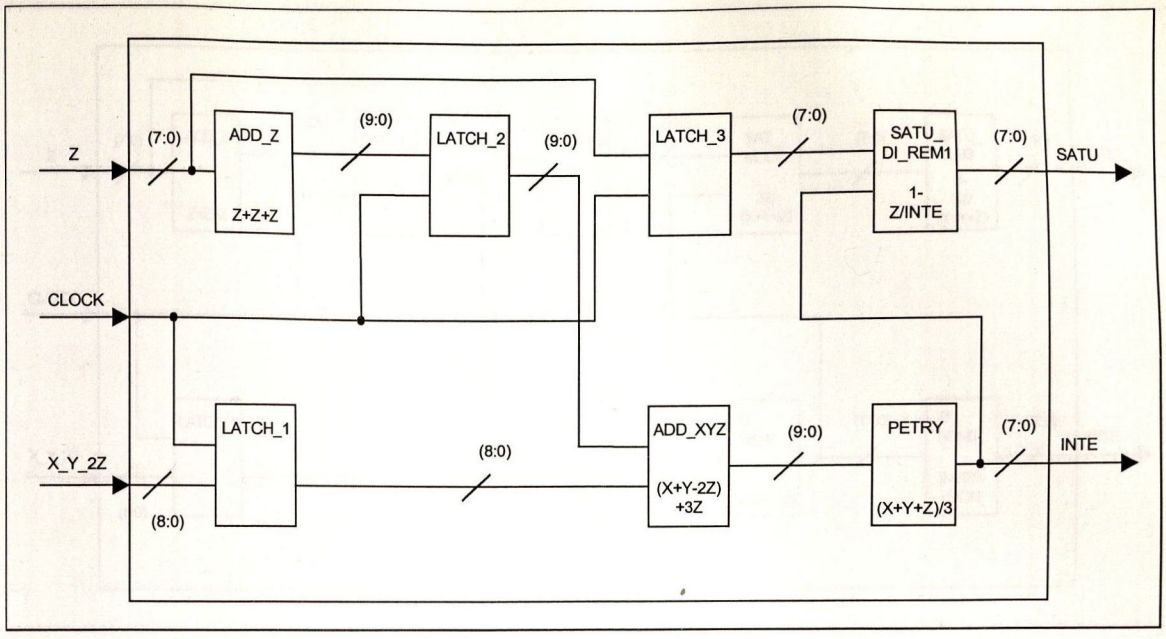


Figure 6.25: Modified RGB to HSI Algorithm

6.4 Third Implementation of the Saturation/Intensity Algorithm

As shown in Section 6.3, the amount of logic required is not necessarily reflected by the equation. Therefore a third approach was implemented and its features were investigated.

$$Intensity = \frac{1}{\frac{3}{\sum R,G,B}} \qquad (107)$$

$$Saturation = 1 - \frac{3 \times \min(R,G,B)}{\sum R,G,B}$$

In this approach, the term three divided by the sum of the input values is calculated as required in the saturation algorithm. This term is then inverted to form the intensity. As seen in Figure 6.26, such an implementation will add no additional stages to the overall design even if the equation is more complex than (107).

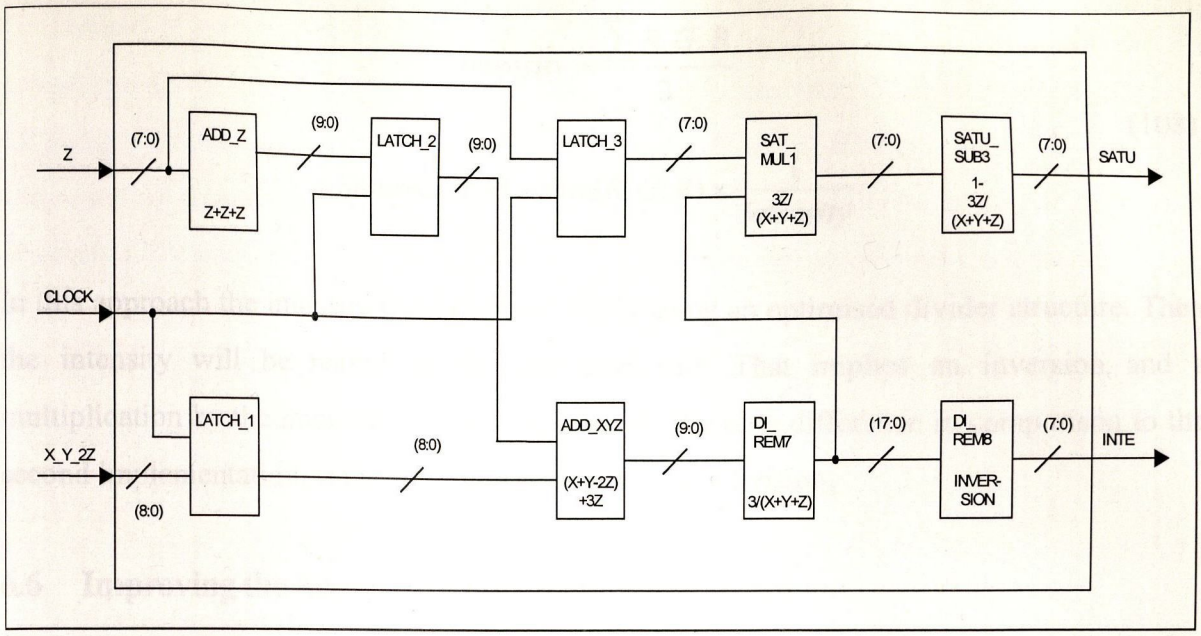


Figure 6.26: Third Implementation of the Saturation/Intensity Algorithm

However, this implementation poses a problem if all three input signals are zero. In this case the term divided by the sum of Red, Green and Blue performs a division by zero. Therefore, a detection of this case is required to overcome this problem.

6.5 Fourth Implementation

Another way to implement the saturation and intensity algorithm is illustrated in Figure 6.27.

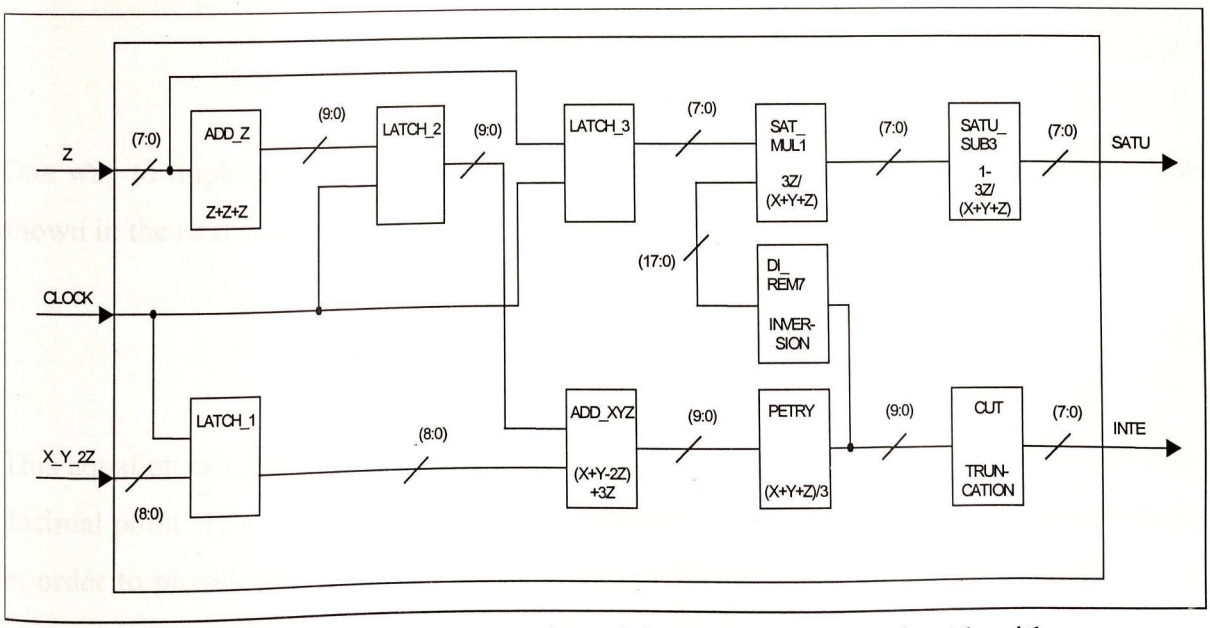


Figure 6.27: Fourth Implementation of the Saturation/Intensity Algorithm

This implementation of the saturation and intensity algorithm uses the following formulae:

$$Intensity = \frac{\sum R,G,B}{3}$$
$$Saturation = 1 - \min(R,G,B) \times \frac{1}{Intensity} \tag{108}$$

In this approach the intensity will be solved firstly using an optimised divider structure. Then the intensity will be reused in the saturation part. That implies an inversion and a multiplication by the minimum value of R, G or B. The only difference in comparison to the second implementation, is the multiplication instead of a division.

6.6 Improving the Accuracy of the Intensity Algorithm

The Intensity Algorithm as presented does not contain any rounding function. This causes a maximum error of -0.5 when compared to the theoretical value, which can be described as shown in the following equation.

$$max.\ error\ \ Intensity = Intensity_{Theory} - 0.5 \tag{109}$$

In order to improve the intensity algorithm, the implementation of a rounding function is investigated.

$$Intensity = \mathrm{Rnd}\left[\frac{R+G+B}{3}\right] \tag{110}$$

One way to implement this function is to add 0.5 to the function and truncate the result as shown in the next equation.

$$Intensity = \mathrm{Abs}\left[\frac{R+G+B}{3} + 0.5\right] \tag{111}$$

This equation can only be used if the accuracy of the division includes several digits after the decimal point. This implies that the division algorithm has to be expanded by several stages in order to provide the additional digits. Those additional stages consume additional power proportional to the amount of the extra stages.

Instead of adding the 0.5 to the result of the division, the rounding factor can also be transferred to the addition of the divisor. This has several advantages. Firstly, the result of the division is already rounded and the natural truncation of the result therefore already includes the rounding without any additional stages. Secondly, the summing of the three input signals is originally unsymmetrical. This means that no balanced adder structure can be applied in order avoid glitching. By adding a fourth figure, a balanced adder can be used and glitching can be avoided. Unfortunately this method also has its disadvantages. Mainly, the addition of 1.5 expands the divisor by one additional bit. This bit increases the size of the divider and therefore the power consumption.

$$Intensity = \mathrm{Abs}\left[\frac{R+G+B+1.5}{3}\right] \tag{112}$$

6.6.1 Modifying the Divider Structure

Therefore, a different approach is presented. All divider structures must be initialised. This is done by loading all stages, not used for storing the divisor, with 0. These 0's represent the digits after the decimal point. Instead of loading these digits with 0 these registers are now loaded with ones. Now the equation can be written as follows. In this equation i represents the number of registers loaded with 1.

$$Intensity = \mathrm{Abs}\left[\frac{R+G+B+\sum_{i=1}^{k}\frac{1}{2^i}}{3}\right] \tag{113}$$

By writing the rounding term independent of the division the effect of this operation becomes clear. This is done in (115).

$$Intensity = \mathrm{Abs}\left[\frac{R+G+B}{3}+\frac{1}{3}\sum_{i=1}^{k}\frac{1}{2^i}\right] \tag{114}$$

Since the divisor term (R+G+B) is already present on a bus an additional adder could not be used in order to avoid glitching through balancing of paths. The maximum value of the rounding term is now 1/3. Therefore the equation can be written as.

$$Intensity \approx \mathrm{Abs}\left[\frac{R+G+B}{3}+0.3\right] \tag{115}$$

Using this equation the maximum error of the intensity algorithm compared to (110) is now:

$$Max.\ err.\ Intensity = Intensity_{Theoy} - 0.2 \tag{116}$$

Even if there is still a maximum error of -0.2, when this value is compared with the original error of -0.5 a considerable improvement has been made. Again it should be stressed that this improvement has been achieved without any additional logic or switching. In other words the method improves the result without causing additional power consumption.

6.6.2 Replacing the LSB by ONE

Instead of modifying the divider it is also possible to drop the LSB of the sum of Red, Green and Blue and replace it by a constant ONE at the input of the divider. This can mathematically be expressed as:

If (R+G+B) even (117)

$$Intensity = \frac{R+G+B+1}{3}$$

else

$$Intensity = \frac{R+G+B}{3}$$

As seen, this computation satisfies the accuracy requirements of the intensity algorithm only for even numbers. Odd numbers are still the absolute value of the result. Nevertheless, this improves the accuracy of the output by 33% without any additional logic. Furthermore, it is possible to reduce the number of bits to be transmitted and stored in pipelining stages by one bit down to 7 bits. Due to the quadratic impact on power consumption in pipelining stages this results in a reduction in power consumption of 20% in the pipelining stages alone. Further reduction in power consumption is possible through the declaration of the LSB to be ONE. This enables the designer to optimise the divider structure and results in a less logic.

6.7 Results of the Saturation-Intensity Path

Table 6.3 shows the characteristics of the different solutions which were obtained by the Synopsys Design Environment and represents the most important features of each implementation.

Version	Active Capacitance / pF	Number of nets	Power Consumption / mW	Max. Delay / ns	Total area / μm^2
Direct	73.18	516	60.98	119.01	1021.74
Second	76.24	452	63.53	126.63	889.47
Third	323.27	1613	269.36	342.79	3960.34
Fourth	118.44	1064	98.69	209.22	2426.29

Table 6.3: Characteristics of the Saturation-Intensity Implementations

6.7.1 The Power Consumption

Figure 6.28 shows the power consumption for the different versions implemented.

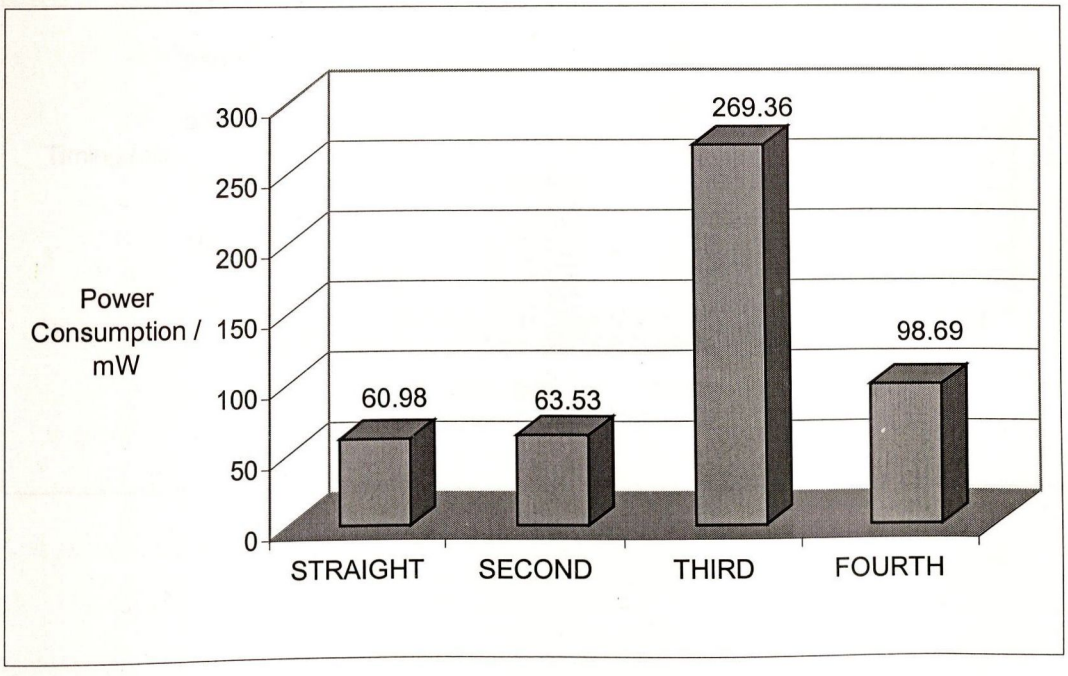


Figure 6.28: The Power Consumption of the SI Algorithms

As illustrated in Figure 6.28, the third implementation requires 269.36mW, which is the most power intensive. The reason for this high power consumption are the large divider stages

required. The direct and the second implementations need approximately 23% of the power of the third implementation. The fourth implementation uses one additional divider when compared to the first two designs. Therefore, this implementation requires 36% of the power of the third implementation.

6.7.2 Timing Behaviour

In these particular saturation and intensity modules the same factors contributing to the power consumption also influence the timing and in fact also the area of the circuit. Therefore, these factors are not mentioned again in this section. As illustrated in Figure 6.29, the third implementation needs 342.79ns, which is the most time to solve the saturation and intensity. The direct implementation only requires 34.7% and the second solution needs approximately 37% of the time of the third implementation. The fourth implementation requires 209.22ns, which is 61% of the third implementation.

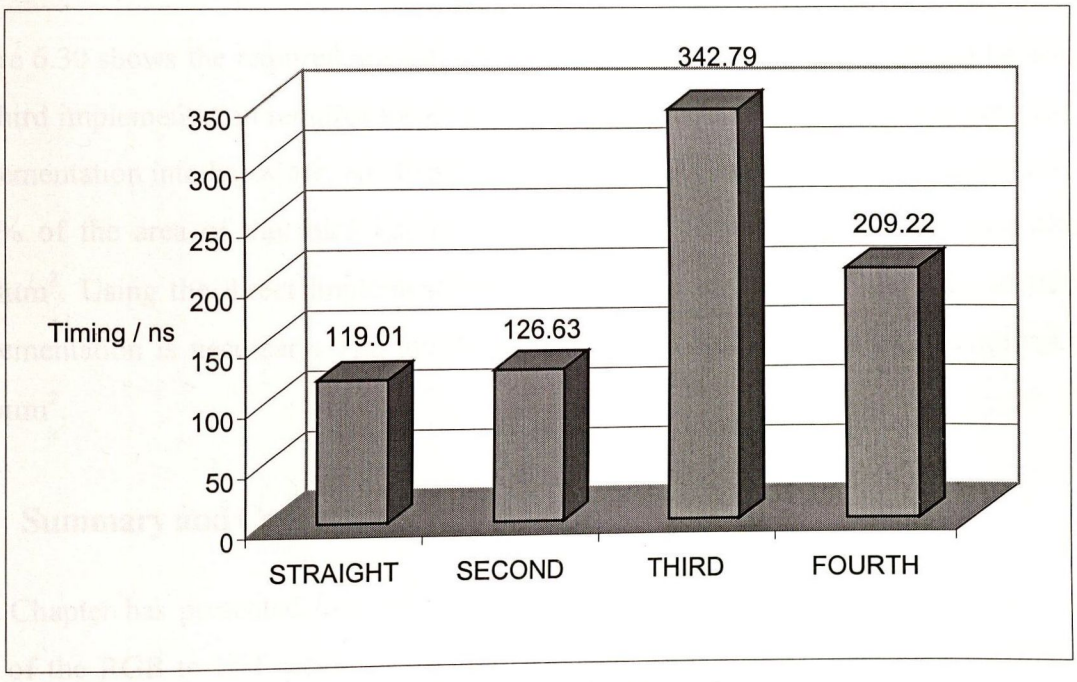


Figure 6.29: The Timing Behaviour

6.7.3 Required Area

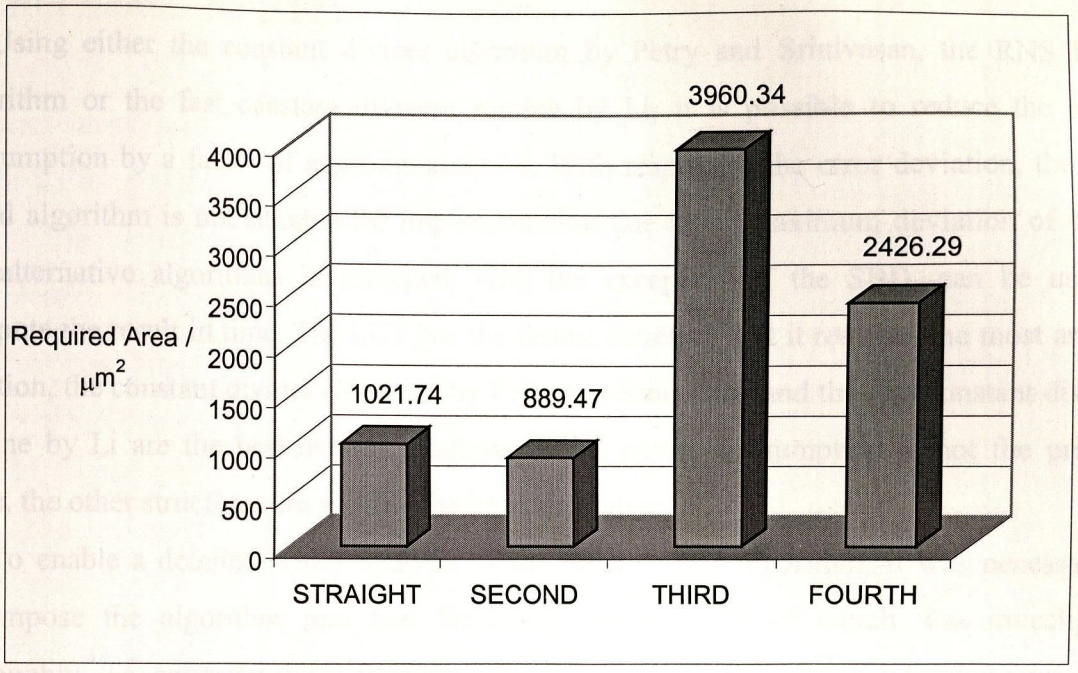


Figure 6.30: The Required Area of the SI Algorithm

Figure 6.30 shows the required area in μm^2 of the solutions implemented. It can be seen that the third implementation requires the most area. For this reason this solution is not useful for implementation into hardware. As illustrated in Figure 6.30, the second implementation needs 22.5% of the area of the third implementation, which requires an area of approximately $3960\mu m^2$. Using the direct implementation only an area of 25.8% of the area of the third implementation is necessary. The fourth implementation requires an area of approximately $2426\mu m^2$.

6.8 Summary and Conclusions

This Chapter has presented four alternative implementations of the saturation and intensity path of the RGB to HSI converter. Here, it has been shown that the direct implementation yields the best results with respect to power consumption and timing behaviour. Using non-optimised constant divider structures, the third implementation requires the most power, time and area. For this reason this approach is not useful for implementation. The direct implementation needs approximately 23% of the power and 35% of the time of the third implementation. The direct implementation requires an area of 26% of the third implementation. The rewriting of the equations of the saturation and intensity algorithm did

not yield a reduction of the power consumption, speed and area of the second, third and fourth implementations.

Using either the constant divider algorithm by Petry and Srinivasan, the RNS based algorithm or the fast constant division routine by Li, it is possible to reduce the power consumption by a factor of approximately 19. With respect to the error deviation, the RNS based algorithm is not suitable for implementation due to its maximum deviation of 4 bits. All alternative algorithms investigated, with the exception of the SBD, can be used to compute the result in time. The LUT has the fastest structure, but it requires the most area. In addition, the constant divider algorithm by Petry and Srinivasan and the fast constant division routine by Li are the best overall solutions. When power consumption is not the primary issue, the other structures are suitable for implementation.

To enable a detailed power analysis of the RGB to HSI algorithm, it was necessary to decompose the algorithm into two functional blocks, each of which was investigated thoroughly. To optimise these functional blocks, they were further subdivided into basic computational elements. These elements were in turn investigated for power consumption, in addition to area, speed and where appropriate, accuracy. This work resulted in a subset of low-power components, which were used as the building blocks for the RGB to HSI converter. The next Chapter presents a comprehensive overview of the electronic and image features of the system, including results obtained by processing real image data.

7 Performance of the RGB to HSI Converter

The previous chapters have developed the implementation of the individual blocks of a low-power RGB to HSI converter. This chapter will now present the overall performance of the design. This presentation consists of two sections. The first section describes the hardware specification of the low-power converter, including a detailed power breakdown of each image component. In addition, the design is compared to a direct implementation with no low-power features.

The second section investigates the image performance of the design using real image data. This investigation consists of two stages. Firstly, a graphical analysis of images produced by the RGB to HSI converter is performed. This analysis comprises the visual investigation of both the converted image and the individual HSI components. Secondly, a statistical analysis of the distribution of individual bit errors is undertaken. Finally, these results are then used to suggest design modifications to satisfy alternate specifications.

7.1 Circuit Performance

This section will present the features of the implementation of the image processing algorithm. Table 7.1 summarises the features of the algorithm as presented throughout the previous chapters.

Features of the RGB to HSI Converter	
Technology	ES2 07μm (industrial)
Supply Voltage	5V (±0.5V)
Input Signals	Red, Green, Blue 8-bit unsigned Clock
Output Signals	Hue, Saturation, Intensity 8-bit unsigned
Throughput	33Mpixels / cycle
Operating Frequency	33MHz
Area	4.19mm^2
Number of Pipeline Stages	5
Output Signal Deviation	Between -2 and +1 bit (-0.78% to 0.39%)
Active Capacitance	187.9pF

Maximum Settling Time	18ns
Maximum Throughput	50Mpixels / cycle
Maximum Operating Frequency	50MHz
Average Dynamic Power Consumption	140mW (@30MHz)

Table 7.1: Features of the RGB to HSI Converter

As can be seen from the table, the throughput can be improved by a factor of up to 1.6 better than the minimum specification for this project of 1200 by 1200 pixels at 25 frames per second, as set out in Section 1, Chapter 1. Thus, with this design it is possible to convert images with a resolution of up to 1600 by 1200 pixels at 25 frames per second. However, a drawback is that the power consumption at this higher throughput will be 39% greater, i.e. 226mW, in comparison to that for the 1200 by 1200 pixel resolution. Therefore, for the lowest possible power dissipation the design should be always be run at the minimum allowable operating speed.

It is worth noticing that if the sum of all the active capacitances of the individual blocks listed through the thesis is calculated the result will be 121.5pF. From Table 7.1 it can be seen that the active capacitance of the overall circuit is 187pF. This is a difference of +35% and can be explained by the additional physical capacitance of the block interconnect. This however does not change any of the statements of the individual blocks made in the previous chapters, it merely increases the overall capacitance of the final circuit. Figure 7.1 shows the detailed breakdown of the individual components of the active capacitance. In this figure it can be seen that, with the optimised constant divider, the intensity has the lowest power consumption. If the intensity and hue subsystems are investigated further, it can be seen that the large divider structures present in both paths have the highest proportion of active capacitance, followed directly by the comparator in the hue path. To minimise the power consumption of these blocks, particular interest should be paid to the layout of these stages.

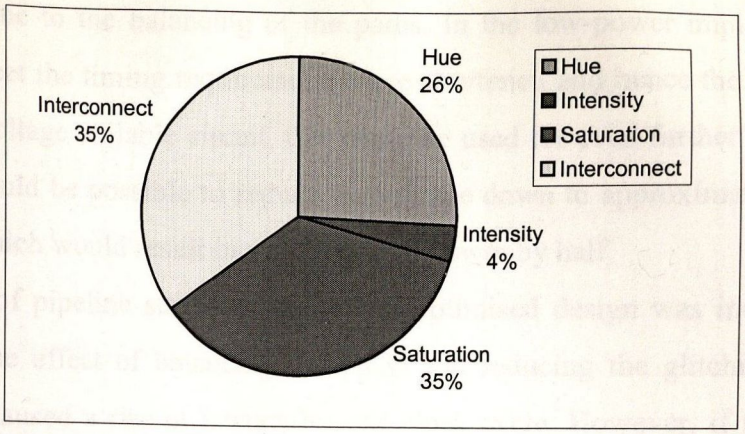


Figure 7.1: Breakdown of the Active Capacitance of the RGB to HSI Converter

7.1.1 Comparison with a Direct Implementation

The RGB to HSI algorithm was implemented in two different ways. In addition to the low-power implementation which was described in previous chapters, a second implementation was also designed. This second design was described using native VHDL operations to implement the various functions. Furthermore, the circuit was synthesised using design constraints to meet only the timing requirements. Therefore, this circuit was not designed to meet any low-power constraints. The second implementation is referred to as the direct implementation of the RGB to HSI algorithm. The results of this direct implementation are shown in Table 7.2.

Features of the RGB to HSI Converter (Direct Implementation)	
Area	$3.7mm^2$
Number of Pipeline Stages	4
Output Signal Deviation	Between -1 and +1 bit (0.39%)
Active Capacitance	298.9pF
Maximum Settling Time	36ns
Maximum Throughput	27.7Mpixels / cycle
Maximum Operating Frequency	27.7MHz
Average Dynamic Power Consumption	202mW (@30MHz)

Table 7.2: Features of a Direct Implementation of the RGB to HSI Converter

As can be seen from a comparison of Table 7.1 and Table 7.2, the power consumption of the direct implementation is 37% higher than that of the optimised implementation. Additionally, the maximum throughput, and therefore the maximum computational image resolution, was increased in the low-power version by a factor of 1.8. This increased

performance is due to the balancing of the paths. In the low-power implementation, paths which did not meet the timing requirements were shortened and hence the overall delay was decreased. In a voltage scalable circuit, this could be used for even further power reductions. In this case it would be possible to reduce the voltage down to approximately 3.5V in a full custom design which would result in a reduction in power by half.

The number of pipeline stages of the power optimised design was increased by one to four. This had the effect of balancing the paths and reducing the glitching in the divider structures. This caused a rise in latency by one clock cycle. However, if the latency of the power optimised design of 90ns is compared to that of the direct implementation, it can be seen that the additional pipeline stage did in fact reduce the latency by 54ns.

The only feature of the low-power implementation which has not improved is the area. However, this was a result which was anticipated. As has been shown in the introduction of this thesis, the factor most often traded-off for a reduced power consumption is the area. The increase of 13% is very reasonable if it is compared to the reduced power consumption of 37% and increased performance of nearly 93%.

While this direct implementation of the circuit does not perform as well as the optimised design, it has the advantage of a much faster design development cycle. Therefore, if fast time to market is of the uppermost importance to the designer, not all power saving features should be implemented. Thus, the use of gated clocks and the replacement of trigonometric functions by the approximation algorithm is one method of effectively reducing the power consumption in a time efficient manner.

7.1.2 Comparison with a DSP

The RGB to HSI transformation can also be implemented on a DSP chip such as the TI TMS320C6211. This chip has a maximum performance of around 1GOPS [TI98] and a power consumption of 1.1W [Cast99]. To implement Kender's algorithm, a minimum of 19 operations are required. Also, if an RGB signal with a resolution of 1024 by 1024 pixels is to be transformed at a frame rate of 25, this results in 78,643,200bytes/second. Therefore, implementation of the transformation of this RGB signal using Kender's algorithm would result in 489MOPS which is half of the available processing power of the TMS320C6211. On the TMS320 however, the operation would consume 1.1W which is 7.9 times that of the proposed image conversion circuit. Furthermore, the maximum resolution of the RGB to HSI

converter is 50Mwords/second which corresponds to a resolution of 1920 by 1080 pixel at 25 frames per second and a power consumption of 226mW. The same resolution would require 2.84GOPS on a TSM320C6211. Therefore, to perform such an operation three boards are required, giving a total power consumption of 3.3W which is nearly 15 times that of the RGB to HSI circuit presented in this thesis.

7.2 Image Quality Performance

Having discussed the circuit features of the RGB to HSI converter circuit, the implemented algorithm will now be investigated with respect to its image converting properties. For this purpose, a C program was written which simulates the behaviour of the hardware. The accuracy of the hardware implementation is compared to a numerical implementation of the algorithm which calculates HSI using double precision. This program was required because of the large amounts of image data to be compared. Such a simulation would require days in a VHDL simulator for a single picture.

On the next pages two pictures are used to illustrate the functionality of the algorithm. These pictures have a size of 600 by 600 pixels and a resolution of 24 bits. Both the original images have been taken from [Uscs]. To compare the images the original picture and the converted image are shown in Figure 7.2 and Figure 7.3. Due to the fact that there is no visible difference between these images two further pictures are included into the analysis. Firstly, a subtraction picture is shown. This is the subtraction of the new picture from the original. In order to show positive as well as negative deviations, the default background of these pictures is set to 50% grey. Here the first patterns appear. These however only show that there is a deviation between the original and the transformed image. To provide a better method of analysis, a colour map of the individual components of hue, saturation and intensity was also included for the different pictures. Here the variation of individual bits from the expected value is shown using different colours. The values of this colour map are shown in Table 7.3.

Errors	<-5	-5	-3	-2	-1	0	1	2	3	5	>5
Colour	Purple	dark blue	Light blue	dark green	Light green	white	Yellow	Light orange	dark orange	Brown	red

Table 7.3: Colour Map Index for the Analysis of the HSI Algorithm

The colour map index is used for an analysis of the images for hue, saturation and intensity separately to make the errors more visible. It can now be seen that the deviations are in fact very small. Furthermore, it appears that most of the pixels in all the colour maps have a deviation from the theoretical value. The colour map for the intensity part shows the largest inaccuracies. Here it appears that nearly all pixels are inaccurate.

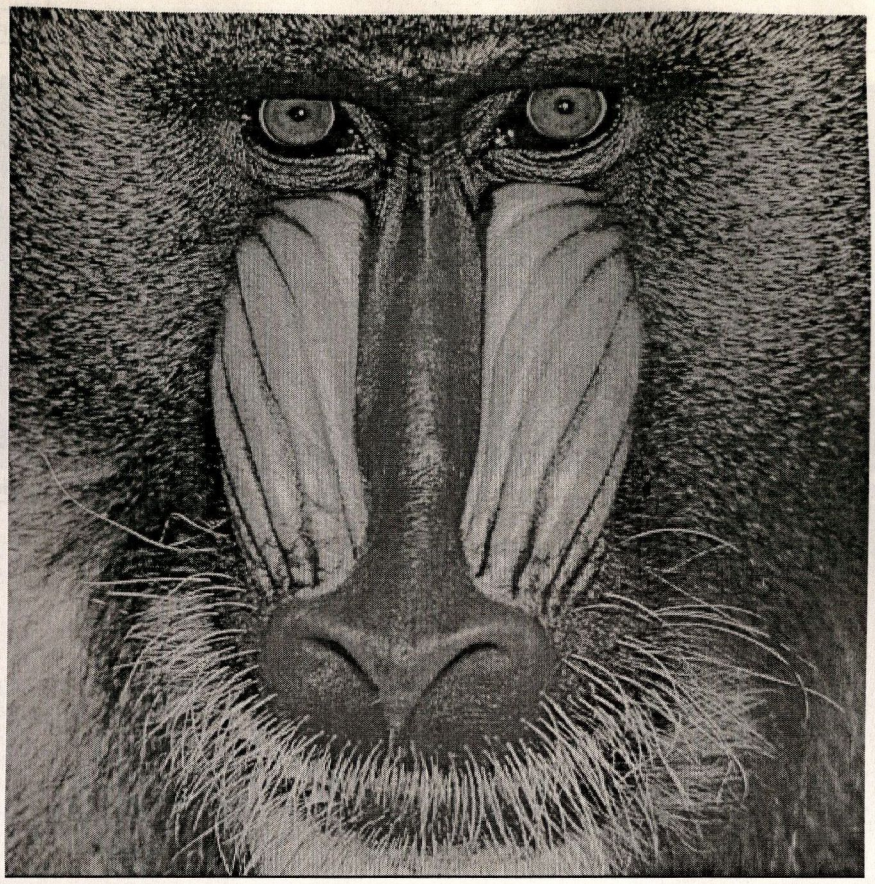


Figure 7.2: The Original Baboon Image

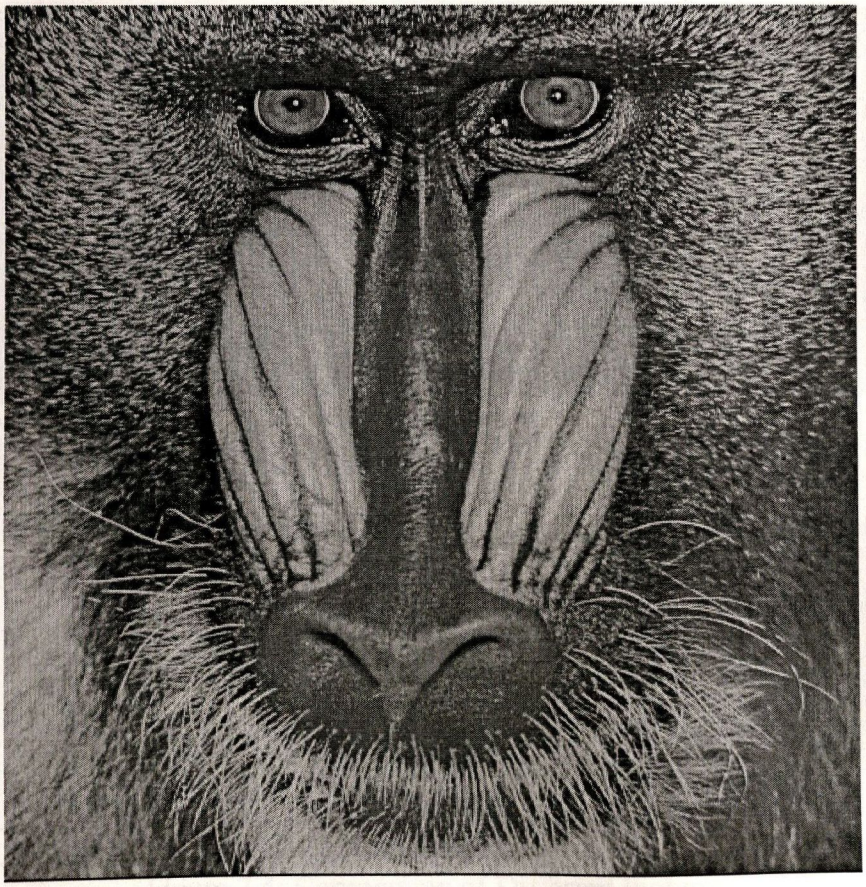


Figure 7.3: The Transformed Baboon Image

Analysis of the Images Pepper and Baboon

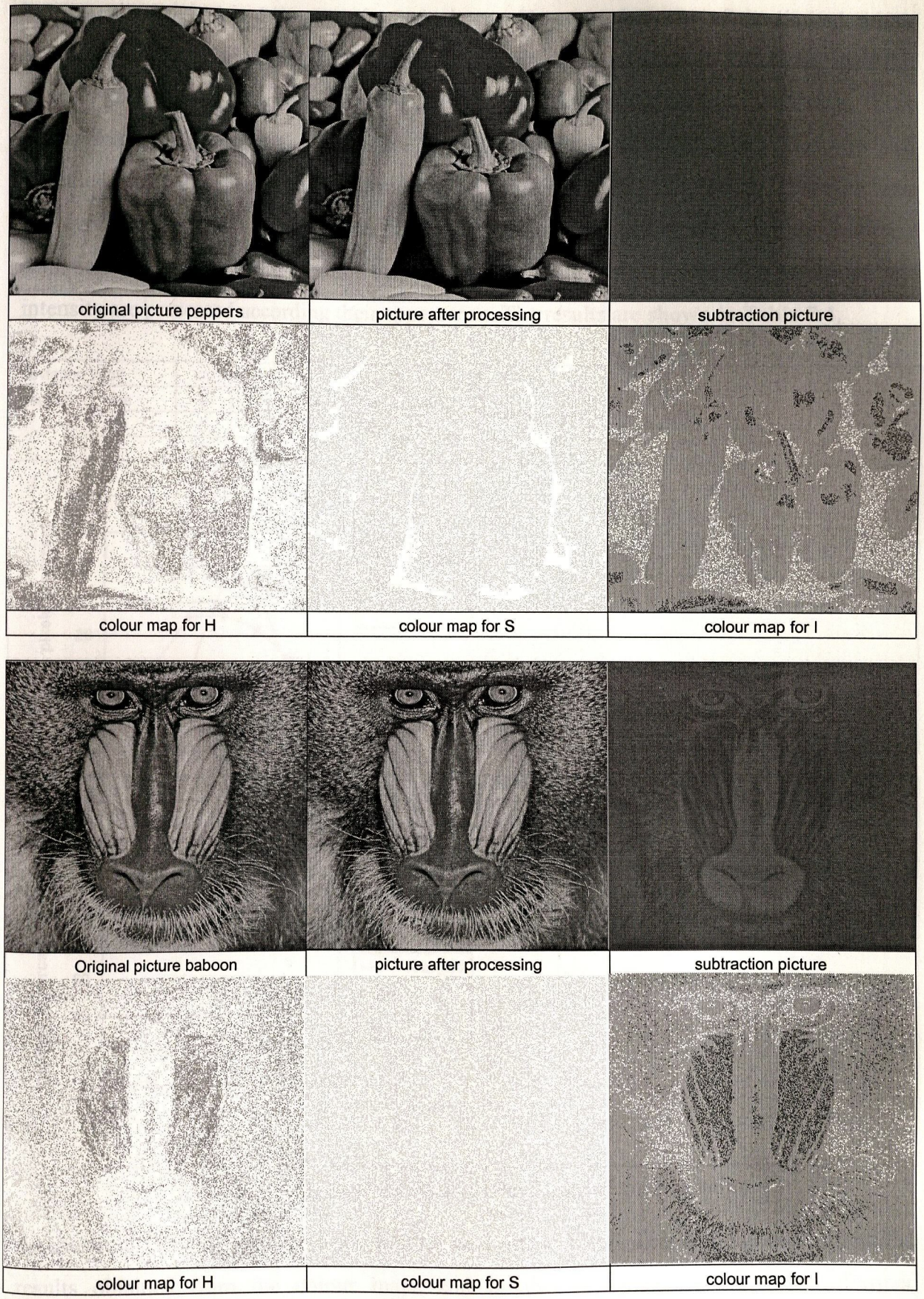
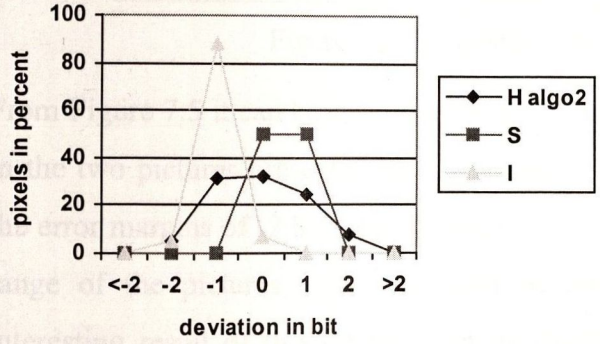


Figure 7.4: Comparison of Different Pictures

As seen in Figure 7.4, the algorithm is operating correctly as there is no visible difference between the original and the transformed picture. However, because of the limited colour spectrum of today's printers and monitors, this is not a conclusive result. Therefore, again the pictures are analysed for errors in the hue, intensity and saturation using a colour map. These colour maps suggest that the errors are limited between -2 bits and 2 bit. This was also expected from the theoretical implementation as presented in this thesis. These pictures however contain 360000 pixels, so that individual errors may not be visible. For this reason, a statistical analysis of these pictures was undertaken. Here the errors of hue, saturation and intensity are presented according their appearance. The results are shown in Table 7.4.

deviation in bit		<-2	-2	-1	0	1	2	>2	
pepper	Hue	0	5	31	32	25	8	0	%
	Sat	0	0	0	50	50	0	0	%
	Int	0	5	88	7	0	0	0	%



deviation in bit		<-2	-2	-1	0	1	2	>2	
baboon	Hue	0	3	35	26	31	4	0	%
	Sat	0	0	0	50	50	0	0	%
	Int	0	6	87	7	0	0	0	%

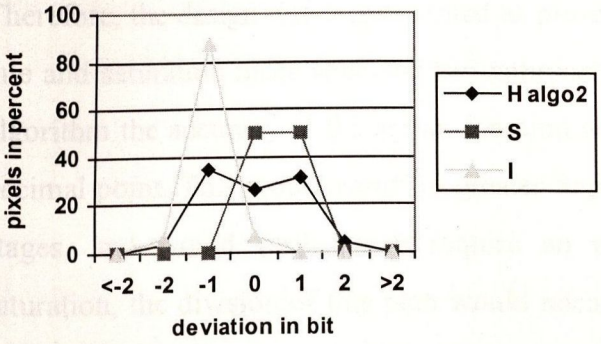


Table 7.4: Graphical Analysis of the Errors of the Algorithm

As shown in Table 7.4, the errors are limited to -2 bits and 2 bit. This is in agreement with the results obtained from the colour maps and is also supported by the theory. Further

investigations using a wide range of pictures has been carried out. The result of a simulation of 242 images, taken from [tcd00], containing more than 84.6M pixels is shown in Figure 7.5.

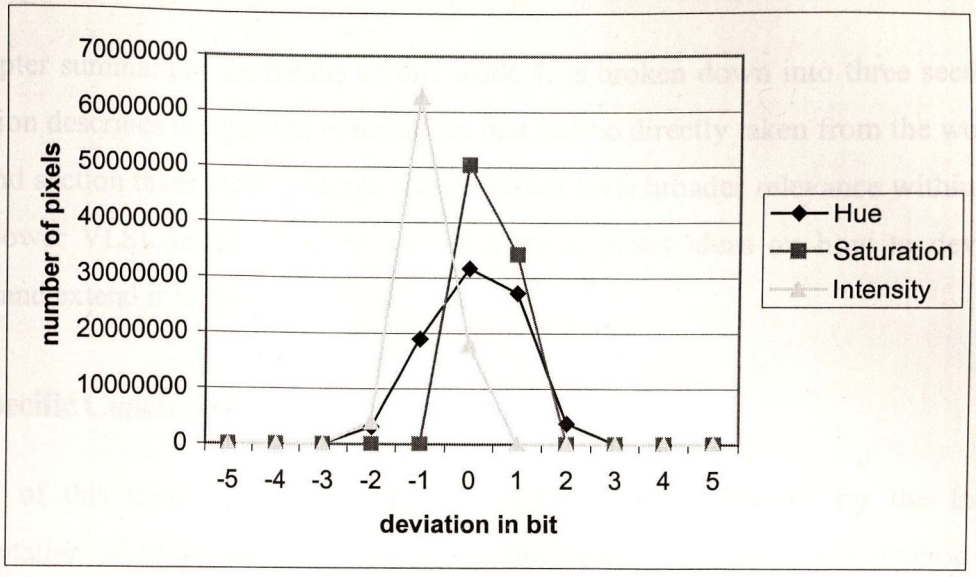


Figure 7.5: Analysis of the Errors of the Algorithm

From Figure 7.5 it can be seen that the errors are similarly distributed to the errors contained in the two pictures presented in this section. Furthermore, this investigation has shown that the error margins of -2 bits and +2 bit are never crossed. This results in terms of the dynamic range of the pictures in a maximum percentage error of -0.78% to +0.78%. Another interesting result of this investigation is that all pictures investigated have a deviation of minus one for approximately 80% of the pixels in the intensity. It would therefore be possible to add one to the result in order to lower this deviation to 20% of the pixels. This however would add an 8-bit adder to the design without improving the visual perception of the images. Therefore, the design was implemented as presented in Section 6.6. To improve the results of hue and saturation more sophisticated approaches would be required. In the case of the hue algorithm the accuracy of the arctan function would need to be increased to one bit after the decimal point. This would result in greater logic in the arctan stage as well as the following stages, and would furthermore require an additional rounding stage. To improve the saturation, the division of this path would need to be more accurate and therefore larger and more power consuming. However, the accuracy achieved is more than sufficient for human inspection and has no noticeable loss in image quality. Therefore, the algorithm presented is implemented as described because of the advantages in power consumption and computational throughput.

8 Conclusions

This chapter summarises the results of this work. It is broken down into three sections. The first section describes the specific conclusions that can be directly taken from the work, while the second section takes these findings and describes their broader relevance within the field of low-power VLSI design. The last section then proposes ideas on how to develop this research and extend it for future projects.

8.1 Specific Conclusions

The aim of this thesis was to investigate system level approaches for the low-power implementation of computationally intensive algorithms. The traditional method for low power IC design is directed towards reducing the supply voltage. However, this approach was not applicable to the project goal as the design had to be mapped into an ASIC library and for ASICs voltage scaling is only possible within a very limited range. The next significant quantity influencing the power consumption is the active capacitance of the design. Therefore, it was intended here to focus on applying techniques to reduce this quantity. For this purpose, a particular implementation of Kender's algorithm for faster computation of hue was chosen to explore the potential usefulness of a variety of methods for reducing the power consumption at the initial stages of the VLSI design cycle. This algorithm was chosen purely as a vehicle for the investigation and it is acknowledged that algorithmic decisions also impact on the low power design issue.

The initial investigation showed the need for a novel CAD tool capable of estimating the power consumption at the earliest possible design stage. Available tools had the disadvantage that they had not been fully incorporated into the standard design cycle or that they tested the design at a late stage in the circuit development. Therefore, a power estimation tool, PowerCount, was developed to rapidly measure the active capacitance of a design from a VHDL netlist. This tool offers the advantage of being fully incorporated into the Synopsys design cycle, as well as using the most accurate available information at that stage. Furthermore, PowerCount uses real timing simulations without making any generalisation when computing the real node activity factor. A Monte Carlo approach guarantees fast and reliable results while using only small sets of input vectors. With this tool it is possible to

simulate large designs within a matter of hours and to make reasonable estimates of the power consumption.

Following the development of this tool, the specific case of the implementation of Kender's RGB to HSI algorithm in the form of a low power circuit was then considered. The aim was to examine this implementation on a block-by-block basis in order to identify potential avenues along which power savings could be made. To enable such a detailed analysis, the design was split into the three paths, studying the computation of hue, saturation and intensity as individual operations. These paths were again subdivided into smaller blocks, each containing their own set of implementation problems. These smallest blocks were investigated separately to find ways of reducing their power consumption.

In the hue path, the main task was to implement the alternating function of the arctan using unsigned arithmetic. This was achieved through the use of sign detection in the first stage of the design, resulting in reduced logic for the remaining stages. A second task was the implementation of the arctan function itself. The standard implementation for all trigonometric functions is the CORDIC algorithm. However, when the CORDIC algorithm was compared with three alternative algorithms, it was found to require 25 times more power than the most efficient alternative. Furthermore, by using any of the alternatives other design features such as the maximum computational time and the area could be significantly reduced. Therefore, it appears that the primary strength of the CORDIC algorithm is in the area of mathematical multiprocessors rather than single function implementations. Following this task, the investigation was turned towards the control pipeline which was shown to have a significant power inefficiency. Firstly, different coding styles were applied to the block. However, the result was unsatisfactory as theoretically superior codes produced a higher power consumption than standard approaches. Detailed analysis showed that the greater power consumption of these reduced-power codes was in fact caused by a larger clock network. Therefore, an alternative to the traditional shift register implementation was developed. Here it was possible to demonstrate that, in general, for small designs, such as that of the control bus, power savings of up to 30% could be achieved and for larger shift registers this figure can increase even further.

The first design decision for the saturation path was to reuse terms already computed in the hue path. This resulted in reduced logic and less pipeline stages in the saturation path. Moreover, it was then possible to use balanced structures to compute a proportion of the mathematical operations. Four different implementations of the saturation block were

considered and all implementations demonstrated their own particular advantages. However, the direct implementation of the saturation showed the best overall power performance. This marked the difference between the software-optimised and hardware-optimised algorithms as while a direct implementation appeared to be mathematically more complex, the block diagram showed that it had the smallest number of functional blocks.

The last path to be implemented was the intensity algorithm. As in the case of the saturation algorithm, it was possible to reuse terms previously calculated. Therefore, only one division by three had to be implemented. To build such a constant divider various mathematical algorithms were selected from the literature as, to date, there have been no efforts undertaken to compare their power consumption. It was recognised that several of these divider algorithms could be implemented with both alternating and non-alternating signs but, as shown, alternating implementations consumed more power. Therefore, the investigation was restricted to the non-alternating versions. From the simulations, the algorithm proposed by Petry was shown to give the best power to area-speed performance and it was therefore decided to use this algorithm in the implementation. Lastly, the accuracy of the intensity was also investigated. It was found to be possible to replace the least significant bit by a constant ONE at the input of the intensity path. This resulted in smaller logic and reduced power consumption while actually improving the accuracy by 33%.

Finally, the performance of the implemented design was investigated using digital images. Despite the fact that the potential sources of the error were known, it was desired to demonstrate that this algorithm produced images that were perceptually indistinguishable from the original. This was done empirically and it was found that the maximum errors of between -2 bits and +2 bits did not appear to have any influence on the perceptual quality of the images.

In summary then, a comparison of the approach presented in this thesis and a direct implementation of the RGB to HSI algorithm showed that a significant power saving of 37% could be made. Also, the computational throughput of the circuit was improved by a factor of 1.8. This was because in the low-power version of the implementation a path balancing approach was used which resulted in a maximum path length of 18ns. The only drawback of this low-power implementation is the increase in required area by 13%. This is due to the more complex logic needed as well as the additional pipeline stages used to balance the path length. A theoretical comparison to an implementation of the algorithm on a TI TMS320C6211 DSP board showed that the low power implementation is approximately eight

times more power efficient at 26Mwords/second and nearly 15 times more efficient at 50Mwords/second.

8.2 General Conclusions

One can identify several key stages in the design process depending on the particular point of view. For instance, considering a problem-solving exercise, e.g. the design of an automated, visual, fruit defect detection system, there will be several stages of thought. One possible solution then could consist of the following stages:

{1} A high-level algorithm development e.g. the design of signal processing techniques to analyse the colour data to detect defects.

{2} Optimisation of each part of the high-level process e.g. assuming HSI space is required – how to generate the RGB to HSI conversion?

{3} Choice of implementation vehicle e.g. a C program on a PC with an analog to digital (A/D) card, or a C program on DSP with an A/D card, or an implementation on an IC.

{4} Once implementation vehicle choices are made, it is possible then to carry out further optimisation with respect to other design criteria such as speed or data rate.

In relation to the implementation of the RGB to HSI conversion algorithm, this thesis was concerned with the design choices around low power criteria once the decision was taken to use an IC as the implementation vehicle. It was clear at the outset that decisions belonging to stages {1} and {2} must effect low power criteria in IC design. For example, if one found that the use of luminance (Y) only was sufficient to solve the problem as described by {1}, then clearly the design of a Y processing chip will have a lower power consumption than a HSI processing chip. However, it was not the purpose of this work to examine the impact of such high level decisions on low power design. Specifically, in order to reveal low power design issues a simple task but with a demanding data rate was chosen.

The methodology utilised in this work for investigating the RGB to HSI conversion consisted of three main stages. Firstly, the functionality as defined by the system specification was examined for power-reduction potential. The second stage gave consideration to the selection of the most suitable number system with which to implement the circuit. Lastly, the

third stage analysed how best to divide the algorithm into sub-blocks and then determine which of the standard techniques for sub-block implementation would produce the optimum low-power performance. Regarding power estimation, a novel tool was presented in the thesis that offered the distinct advantage of being fully incorporated into the Synopsys design cycle.

The methodology was validated in the final results and therefore confirmed the use of this staged procedure in a VLSI design environment. It appears that it is more profitable, in term of power savings, to work from the generalities of the design towards the particulars of the implementation. This is also important to realise when a constraint such as the time to market impinges on the project. In this situation, the designer must be conscious of their time management and therefore should focus their efforts on the power bottlenecks of the system caused by power consumption. Working directly on the algorithm specification, significant power savings can be made without having to study the details of the circuit design. This means that at this high level fast, feedback about the power consumption is required. Therefore, it is advantageous if the power estimation tool is incorporated into the design environment. This sort of requirement was anticipated when developing PowerCount and it is justified by the recent appearance on the market of similar tools which can be incorporated into the design environment.

8.3 Future Work

Two possible roads for future work may be considered. The first would be to use the whole RGB to HSI algorithm and to optimise it at the lower levels of VLSI design. It would be important then to investigate the physical implementation of individual structures. Due to the large number of basic structures used by the algorithm, this could be a time-consuming task if a fully optimised design is to be produced. A second possible direction would be to examine the applicability of the power reduction techniques presented in this work to other areas of VLSI design in particular adaptive filter design. An investigation of a mixture of computationally intensive structures and medium-to-low performance designs would give a good overview of the suitability of these techniques in various application domains. At the high end of the design scale, particular interest should be paid to the possibility of the direct application of the techniques suggested in this thesis to computationally intensive structures. With such structures, low power designs are an essential aid to overcome heat dissipation problems and also help to enable the development of portable real-time computing equipment. At the lower end of the performance scale, an investigation into the use of the

proposed techniques could focus on determining the optimum balance between the silicon area and the power consumption. The particular relevance of such research for industry would lie in the fact that lower power consumption allows the use of cheaper packaging because of reduced heat dissipation, while a larger die area increases the fabrication costs.

References

[Ahma96] I. S. Ahmad and J. F. Reid, "Evaluation of colour representations for maize images," *Jour. of Agricultural Engineering Research,* Vol. 63, PT. 3, pp. 185–195, 1996.

[Akita94] J. Akita and K. Asada, "A method for reducing power consumption of CMOS logic based on signal transition probability," *EDAC-ETC Euro-ASIC'94,* pp. 420-424, February 1994.

[Albe97] B. Al-Besher, A.Bouridane and A. S. Ashur, "An RNS-based division architecture for constant divisors of the form 2^n+1 and 2^n-1," *Irish Signals & Systems Conference*, June 1997.

[Athas94] W. C. Athas and L. Svensson, "Low-power digital systems based on adiabatic-switching principles," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, December 1994.

[Beck98] M. Becker, E. Kefalea, E. Mael, M. Pagel, J. Triesch, J.C. Vorbrueggen, S. Zadel, and C. v.d.Malsburg, "GripSee: A Robot for Visually-Guided Grasping," *Proceedings of ICANN International Conference on Artificial Neural Networks,* Swoevde, Sweden, Sept. 1998.

[Bell98] A. Bellaouar, A. Fridi, M. I. Elmasry and K. Itoh, "Supply voltage scaling for temperature insensitive CMOS circuit operation," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 45, No.3, pp. 415-417, March 1998.

[Beni00] L. Benini, G. DeMicheli, A. Macii, E. Macii, M. Poncino and R. Scarsi, "Glitch power minimization by selective gate freezing," IEEE Transactions on VLSI Systems, Vol. 8, No. 3, pp. 287-298, June 2000.

[Best98] G. W. den Besten and B. Nauta, "Embendded 5V-to-3.3V voltage regulator for supplying digital IC's in 3.3V CMOS technology," *IEEE Journal of Solid-State Circuits*, Vol.33, No. 7, pp. 956-962, July 1998.

[Blair94] G. M. Blair, "Designing low-power digital CMOS," *Electronics and Communication Engineering Journal*, October 1994.

[Brust93] L. Brust and M.S. Tsay, "Mixing signals and voltages on chip," *IEEE Spectrum*, August 1993.

[Burd94] T. Burd and B. Peters, "A power analysis of a microprocessor: a study of the MIPS R3000 architecture," May 1994.
http://infopad.eecs.berkeley.edu/infopad-ftp/

[Burd95] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," *Proc. of the 28th Annual HICSS Conference*, January 1995.

[Burd96] T. Burd, "Low-power CMOS libraries design methology," *MSc Thesis. University of California, Berkeley,* 1996.

[Calv00] G. Calvini and G. Sandini, "Color segmentation for vegetables quality control," http://www.lira.dist.unige.it/Projects/Research/Stuff/vegetables.html

[Cast99] K. Castille, "TMS320C6000 Power Consumption Summary," *Application Report*, Texas Instruments, November 1999,
http://www.ti.com/sc/docs/psheets/abstract/apps/spra486b.htm

[Cava99] C. Cavadore, B. Gaillard, P. Martinole and S. Charbonnel, "Transformation HSI vers RGB et RGB vers HSI,"
http://prism.astroccd.com/aide/Trichro/4.html

[Ccma99] DIT VLSI Research Group, *CapCount Manual V.06/99*, DIT VLSI Research Group, Ireland, 1999.

[Chan92] A. Chandarakasan and S. Sheng, "Low-power CMOS digital design," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 4, April 1992.

[Chan94] A. Chandrakasan, M. Potkonjak, R. Mehra, *et al.*, "Optimizing power using transformations," *IEEE Transaction on CAD of Integrated Circuits and Systems*, Vol. 14, No. 1, January 1994.

[Chan94b] A. Chandrakaan, A. Burstein and R. Brodersen, "A low-power chipset for a portable multimedia I/O terminal," *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 12, December 1994.

[Chan95a] A. Chandrakasan, M. Potkonjak, R. Mehra, *et.al.* "Optimizing power using transformations," *Transactions on CAD,* January 1995.

[Chan95b] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, Vol. 83, No. 4, April 1995.

[Chan95c] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Third Printing (1998), 1995.

[Danc00] A. P. Dancy, R. Amirtharajah and A. P. Chandrakasan, "High-efficient multiple-output DC-DC conversion for low-voltage systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 3, pp. 252-263, June 2000.

[Es2] European Silicon Structures, *ES2 ECPD07 Library Databook,* 1996.

[Fren98] J. Frenkil, "A multi-level approach to low-power IC design," *IEEE Spectrum*, pp. 54-60, February 1998.

[Giak98] I. Giakoumis and I. Pitas, "Digital restoration of painting cracks," *IEEE Int. Symposium on Circuits and Systems (ISCAS'98),* California, USA, May1998.

[Gonz96] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits*, Vol. 31, No.9, pp. 1277-1284, September 1996

[Good98] J. Goodamn, A. P. Dancy and A. P. Chandrakasan, "An energy/security scalable encryption processor using an embedded variable voltage DC/DC

converter," *IEEE Journal of Solid State Circuits*, Vol. 33 No. 11, pp. 1799-1809, November 1998.

[Hama00] G. Hamarneh, A. Chodorowski, and Tomas Gustavsson, "Active contour models: application to oral lesion detection in color images," *IEEE Conference on Systems, Man, and Cybernetics,* 2000.

[Irsim] IRSIM Reference Manual, http://www.research.digital.com/wrl/projects/magic/magic.html

[Jou98] S.-J. Jou and T.-L. Chen, "On-chip voltage down converter for low-power digital systems," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 45, No. 5, pp. 617-625, May 1998.

[Kapa99] H. Kapadia, L. Benini and G. DeMicheli, "Reducing Switching Activity on datapath buses with control-signal gating," *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 3, pp. 405-414, March 1999.

[Kender] J. Kender, "Saturation, hue and normalized color," *Carnegie-Mellon University, Computer Science Dept.,* Pittsburgh PA. 1976.

[Kosc96] A. Koschan, "Using perceptual attributes to obtain dense depth maps," *IEEE Southwest Symposium on Image Analysis and Interpretation,* San Antonio, USA, pp. 155-159, April 1996.

[Laks99] G. Lakshminarayana, A. Raghunathan, N. K. Jha and S. Dey, "Power management in high-level synthesis," *IEEE Transactions on VLSI Systems*, Vol. 7, No. 1, pp.7-14, March 1999.

[Land94] P. Landman and J. Rabaey, "Black-Box capacitance models for architectural power analysis," *Proc. of 1994 International Workshop on Low-Power Design,* April 1994.

[Li85] S.-Y. R. Li, "Fast constant division routines," *IEEE Transactions on Computers*, Vol. C-34, No. 9, September 1985.

[Liev99] M. Liévin and F. Luthon, "Unsupervised lip Segmentation under natural conditions," *IEEE Conf. on Acoustics, Speech and Signal Processing, ICASSP'99,* Phoenix, Arizona, vol. 6, pp. 3065-3068, March 1999.

[Liu93] D. Liu and C. Svensson, "Trading speed for low power by choice of supply and threshold voltages," *IEEE Journal of Solid State Circuits*, Vol. 28, No. 1, January 1993.

[Lyon93] R.F. Lyon, "Cost, power, and parallelism in speech signal processing," *IEEE Custom Integrated Circuits Conference,* San Diego, USA, May 1993.

[Mach00] Mentor Graphics Corporation, Mach TA, http://www.mentorg.com/mach/

[Malh94] S. Malhi and P. Chatterjee, "Scaling on schedule for personal communications," *IEEE Journal*, March 1994.

[Mart99] B. Martin, "Electronic design automation: technology 1999 analysis and forecast," *IEEE Spectrum*, pp. 57-61, January 1999

[Mata94] A. Matsuzawa, "Low-voltage and low-power circuit design for mixed analog/digital systems in portable equipment," *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 4, April 1994.

[Mehr94] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," *Proc. First Inter. Workshop on Low Power Design*, Napa Valley, CA, pp.197-202, April 1994.

[Mehr97] R. M. Mehra, L. M. Guerra and J. M. Rabaey, "A partitioning scheme for optimising interconnect power," *IEEE Journal of Solid-State Circuits*, Vol. 32, No. 3, pp. 433-443, March 1997.

[Micr00] E. Sicard, Microwind2, http://intrage.insa-tlse.fr/~etienne/Microwind/index.html

[Mone95] J. Moneiro, J. Rinderknecht, S. Devadas, *et.al.* "Optimisation of combinational and sequential logic circuits for low power using precomputation," *1995*

Chapel Hill Conference on Advanced Research in VLSI, Chapel Hill, North Carolina, March 1995.

[Moor65] G.E. Moore, "Cramming more components onto integrated circuits" *Electronics,* Vol. 38, No.8, pp. 114-117, April 1965.

[Mult00] Electronics Workbench, MultiSim, http://www.interactiv.com/

[Msim00] Model Technologies, ModelSim/VHDL Simulator, http://www.model.com/products/vhdl.asp

[Naim94] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, December 1994.

[Naka94] Y. Nakagome, K. Itoh, M. Isoda, K. Takeuchi and M. Aoki, "Sub-1-V swing internal bus architecture for future low-power ULSI's," *IEEE Journal of Solid-State Circuits*, Vol. 28, No. 4, April 1994.

[Nema99] M. Namani and F.N. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No.6, pp.697-713, June 1999.

[Nguy00] H. T. Nguyen and A. Chatterjee, "Number-splitting with shift-and-add decomposition for power and hardware optimization in linear DSP synthesis," *IEEE Transactions on VLSI Systems*, Vol. 8, No. 4, pp. 419-424, August 2000.

[Niel94] L. S. Nielsen, C. Niessen and K. van Berkel, "Low-power operation using self-timed circuits and adaptive scaling of the supply voltage," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, December 1994.

[Nose00] K. Nose and T. Sakurai, "Analysis and future trend of short-circuit power," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol.19, No.9, pp. 1023-1030, September 2000.

[Papu90] L. Papular, *Mathematik für Ingenieure 2,* Viewegs Fachbücher der Technik, 5. Auflage, pp. 455-493, 1990.

[Parc00] Synopsys Inc., PowerArc, http://www.synopsys.com/products/etg/powerarc_wp.html

[Pcal00] IBM Corp., ASIC Design Methodology: PowerCalc Tool Supplement, http://www.chips.ibm.com/products/asics/document/appnote/231500_0.pdf

[Pcut00] Legend Design Technology Inc., Power-Cut, http://www.legenddesign.com/

[Petry93] F. E. Petry, P. Srinivasan, "Division techniques for integers of the form 2^n+1 and 2^n-1," *Int. J. Electronics*, Vol. 74, No. 5, pp 659-670, 1993.

[Petry94] P. Srinivasan, F.E. Petry, "Constant-division algorithms," *IEE Proc.-Comput. Digit. Techn.*, Vol. 141, No. 6, November 1994.

[Pmil98] Synopsys, *PowerMill Reference Manual,* Synopsys, Inc., 1998.

[Ppow00] Synopsys Inc., PrimePower, http://www.synopsys.com/products/etg/primepower_wp.html

[Ptoo00] Veritools, Inc. Power Tool, http://www.veritools-web.com/

[Pwat00] Sequence Design Inc., Peak Watcher, http://www.senteinc.com/2_solutions/2b3_pwatcher.html

[Ragh99] A. Raghunathan, S. Dey and N.K. Jha, "Register transfer level power optimisation with emphasis on glitch analysis and reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 18, No.8, pp. 1114-1131, August 1999.

[Ramp99] S. Ramprasad, N. R. Shanbhag and I. N. Hajj, "Decorrelating (DECOR) transformations for low-power digital filters," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 46, No. 6, pp. 776-787, June 1999.

[Ramp99b] S. Ramprasad, N. R. Shanbhag and I. N. Hajj, "Signal Coding for low power: fundamentals limits and practical realizations," *IEEE Transactions on Circuits*

and Systems-II: Analog and Digital Signal Processing, Vol. 46, No. 7, pp. 923-929, July 1999.

[Rior97] M. Riordan and L. Hoddeson, "Birth of an era," *Scientific America, Special Issue – The Solid-Sate Century,* 1997.

[Sanc99] H. Sanchez, J. Siegel, C. Nicoletta, J.P. Nissen and J. Alvarez, "A versatile 3.3/".5/1.8-V CMOS I/O driver built in a 0.2-μm, 3.5-nm tox, 1.8-V CMOS technology," *IEEE Journal of Solid State Circuits*, Vol. 34, No. 11, pp. 1501-1511, November 1999.

[Schw97] A. Schwarzbacher and B. Foley, "Low power CMOS design : a chip for RGB to HSI conversion," *Trans. of ISSC 97,* pp. 165-172, June 1997.

[Schw98] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Powercount: measuring the power at the VHDL netlist level," Electronic Devices and Systems Conference, Bruno, Czech Republic, June 1998.

[Schw99] A. Schwarzbacher and S. Roth, *Graphical and Statistical Investigation of Various Implementations of the HSI Algorithm,* Technical Report, Dublin Institute of Technology, Dublin, Ireland, July 1999.

[Shim93] K. Shimohigashi and K. Seki, "Low-voltage ULSI design," *IEEE J. of Solid-State Circuits,* No. 4, April 1993.

[Shiu00] W.-T. Shiue and C. Chakrabarti, "Low-power scheduling with resources operating at multiple voltages," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 47, No. 6, pp.536-543, June 2000.

[Sieg96] A.F. Siegel, C.J.Morgan, *Statistics and Data Analysis,* John Wiley & Sons, 2nd edition, 1996.

[Smbt00] Avant! Corporation, Star-MBT, http://www.avanticorp.com/product/1,1172,25,00.html

[Sobo98] K. Sobottka and I. Pitas, "A novel method for automatic face segmentation, facial feature extraction and tracking," *Signal Processing: Image Communication,* Vol. 12, No. 3, pp. 263-281, June, 1998.

[Spice] Cadence, *Cadence SPICE Reverence Manual,* Cadence Design Systems, Version 4.2, March 1992.

[Spie88] M.R. Spiegel, *Schaum's Outline of Theory and Problems of Statistics,* McGraw-Hill, 2nd edition, 1988.

[Ssim00] Avant! Corporation, Star-Sim http://www.avanticorp.com/product/1,1172,29,00.html

[Stork95] J. Stork, "Technology leverage for ultra-low power information systems," *Proceedings of the IEEE*, Vol. 83, No. 4, April 1995.

[Synop] Synopsys, *VHDL Compiler Reference Manual Version 3.0*, November 1992.

[Tcd00] School of Mathematics, Trinity College Dublin, Online Image Archive, http://www.maths.tcd.ie/pub/images/images.html

[TI98] Texas Instruments, "How to Begin Development Today With the TMS320C6211 DSP," *Application Report*, Texas Instruments, September 1998, http://www.ti.com/sc/docs/psheets/abstract/apps/spra474.htm

[Tsui95] C. Tsui, J. Monteiro, M. Pedram, *et al.* "Power estimation for sequential logic circuits," *IEEE Transactions on VLSI Systems,* pp. 404-416, September 1995.

[Uscs] USC-SIPI Image Database, University of Southern California, http://sipi.usc.edu/services/database/Database.html

[Veen84] H. Veendrick, "Short-circuit dissipation of static CMOS circuits and its impact on the design of buffer circuits," *IEEE Journal of Solid State Circuits,* Vol. 19, No. 4, August 1984.

[Veri00] Cadence Design Systems, Inc., Verilog-XL, http://www.cadence.com/eda_solutions/flv_ver_vhdl_sim_l3_index.html

[Vold59] J.E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, Vol EC-8, no.3, September 1959.

[Vpow00] Veritools, Inc. VeriPower, http://www.veritools-web.com/

[Wang98] C.C. Wang and J.C. Wu, "A 3.3-V/5V low power TTL-to-CMOS input buffer," *IEEE Journal of Solid State Circuits*, Vol. 33, No. 4, pp. 598-603, April 1998.

[Wei99] G.-Y. Wei and M. Horowitz, "A fully digital, energy-efficient, adaptive power-supply regulator," *IEEE Journal of Solid State Circuits*, Vol. 34, No. 4, pp. 520-528, April 1999.

[Wu99] Y. Wu, Q. Liu, T.S. Huang Beckman, "Robust real-time human hand localization by self-organizing color segmentation," *Proc. of ICCV99 Workshop RATFG-RTS,* Greece, Sep. 1999.

[Wwat00] Sequence Design Inc., Watt Watcher, http://www.senteinc.com/2_solutions/2b1_wwatcher.html

[Xant99] T. Xanthopoulos and A. P. Chandrakasan, "A lop-power IDCT marcocell for MPEG-2 MP@ML exploiting data distribution properties for minimal activity," *IEEE Journal of Solid State Circuits*, Vol. 34, No. 5, pp. 693-702, May 1999.

[Xhdl00] X-Tek Corporation, X-HDL3, http://www.x-tekcorp.com/xhdl3.htm

[Yama96] T. Yamauchi, Y. Morooka and H. Ozaki, "A low power and high speed data transfer scheme with asynchronous compressed pulse width modulation for AS-memory," *IEEE Journal of Solid State Circuits*, Vol. 31, No. 4, pp. 523-530, April 1996.

[Zhan00] H. Zhang, V. George and J. M. Rabaey, "Low-Swing on-chip signaling techniques: effectiveness and robustness," *IEEE Transactions on VLSI Systems*, Vol. 8, No. 3, pp. 264-272, June 2000.

Authors Publications

[1] A.Th. Schwarzbacher and J.B. Foley, "Low-power design : an image processing chip for RGB to HSI conversion," *Irish Systems and Signals Conference,* Derry, Ireland, pp. 165-172, June 1997.

[2] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Powercount : measuring the power at the VHDL netlist level," *Electronic Devices and Systems Conference,* Bruno, Czech Republic, June 1998.

[3] P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "Power estimation in CMOS circuits using genetically optimised input patterns," *Electronic Systems and Devices Conference,* Bruno, Czech Republic, June 1998.

[4] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Reduction of the power consumption at the algorithmic level of CMOS circuits," *Electronic Systems and Devices Conference,* Bruno, Czech Republic, June 1998.

[5] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "High level power estimation powercount," *Irish Systems and Signals Conference*, Dublin, Ireland, pp. 101-108, June 1998.

[6] P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "Random binary vector generation and analysis using genetic optimisation," *Irish Systems and Signals Conference*, Dublin, Ireland, pp.519-525, June 1998.

[7] A.Th. Schwarzbacher, P.A. Comiskey and J.B. Foley, "Improving the power consumption in image processing algorithms," *UK Low Power Forum,* Sheffield, Untied Kingdom, pp. 4.1–4.5, September 1998.

[8] P.A. Comiskey, A.Th. Schwarzbacher and J.B. Foley, "The effect of input lattice structure in image processing algorithm," *UK Low Power Forum,* Sheffield, Untied Kingdom, pp. 11.1–11.6, September 1998.

[9] A.Th. Schwarzbacher, P.A. Comiskey, J. Neves Rodrigoues and J.B. Foley, "Design of integrated circuits for the power domain," *First International Postgraduate Research Conference*, Dublin, Ireland, November 1998.

[10]A.Th. Schwarzbacher and P.A. Comiskey, "Classification of Uniform White Noise Sources using the Spectral Test," *Fling High - Magazine for Supercomputing,* No.15, Spring 1999.

[11]A.Th. Schwarzbacher, A. Brasching, Th.H. Wahl and J.B. Foley, "Optimisation of trigonometric functions for low power CMOS implementations," *Irish Systems and Signals Conference*, Galway, Ireland, pp. 201-206, June 1999.

[12]A.Th. Schwarzbacher and P.A. Comiskey, "Power estimation at the higher levels of integrated circuit design" *Irish Scientist,* p. 112, Year Book 1999.

[13]A.Th. Schwarzbacher, A. Brasching, Th.H. Wahl, and J.B. Foley, "Optimisation and implementation of the arctan function for the power domain," *Electronic Circuits and Systems Conference,* Bratislava, Slovakia, pp. 33-36, September 1999.

[14]A.Th. Schwarzbacher, "Optimisation of algorithms for fast power analysis in CMOS circuits," *Fling High - Magazine for Supercomputing,* No.16-17, pp. 4-7, 1999.

[15]A.Th. Schwarzbacher, P.A. Comiskey, J.B. Foley, J. Rodrigoues and F. Klemenz, "Rapid estimation of the active node capacitance of VLSI circuits," *Programmable Devices and Systems 2000,* Ostrava, Czech Republic, February 2000.

[16]A.Th. Schwarzbacher, "Energy reduction of transformation of image and video data," *to appear in Proc. of CINECA,* Summer 2000.

[17]A.Th. Schwarzbacher, M. Brutscheck, O. Schwingel and J.B. Foley, "Constant divider structures of the form $2^{n\pm1}$ for VLSI implementation," *Irish Signals and Systems Conference,* Dublin, Ireland, June 2000.

[18]A.Th. Schwarzbacher and J.B. Foley, "Optimisation of real-time signal processing algorithms for low-power CMOS implementations," *Digital Signal Processing 2000,* Bournemouth, United Kingdom, July 2000.

Appendix A: Using PowerCount

This appendix shows the simulation of a 1-bit adder to show the performance and features of PowerCount as well as to illustrate the user-friendliness of its interface. It is not intended to be a tutorial for PowerCount. All options of the power estimation environment can be found in the PowerCount Manual [Ccma99]. The design used as an example is the 1-bit adder, which was introduced in section 4.3. The design is simulated with the following specifications in order to estimate the active capacitance:

Design name	FULL_ADDER
Number of input vectors	100
Desired accuracy	1%
Time base	ns
Scaling factor	0.01
Name of the file containing the results	Results_add_1%

Table A.1: Parameter for the Simulation

Power Count uses a simple command line interface to control the simulation. This enables the designer to run the tool in the background or over slow networks. The simulation is started with:

```
pcount -d FULL_ADDER -iv 100 -acc 0.01 -rf Resultsadd_1% -sf 0.01 -tb ns
```

The design FULL_ADDER is simulated with 100 input vectors per iteration. This is done until $\bar{x}$ is within a maximum deviation of 1% from μ or until 30 iterations are performed. The reason for the upper limit is to shorten the simulation if an unrealistic accuracy target is set. The results of the simulation and the simulation parameter are printed into the file *Resultsadd_1%* and onto the screen. The file *Resultsadd_1%* provides a brief overview of the most important features of the simulation. The output of this file is shown below:

```
sum input_vec:100 runs:6 acc: 0.999 act_cap:0.282 totaltime: 55 s
```

The results of the simulation are as follows:

Number of input vectors: 100

Number of iterations: 6

Deviation $\bar{x}$ from μ: 0.01%

Simulation time: 55s

Estimated active capacitance: 0.285pF

The energy which is required by the circuit is calculated by multiplying the active capacitance for a single vector by the supply voltage squared.

$E=0.285pF\ (5V)^2 = 7.125pJ$

To enable a detailed analysis a file *sim.tra* is created.
Figure A.1 shows this file for the *Full_Adder*.

```
N24                              0.14     23.33       3.27
N23                              0.14     48.67       6.81
N22                              0.14     25.17       3.52
N21                              0.09     33.50       3.01
SUM                              0.03     49.33       1.48
CARRY_OUT                        0.03     34.00       1.02
CARRY_IN                         0.13     23.67       3.08
B                                0.12     24.17       2.90
A                                0.14     24.50       3.43

------------------------------------------------------------
total active capacitance                    0.285 pF

Accuracy : 99.85%

Initialisation vector(s): 1
Input vector(s): 100
Desired accuracy: 0.01
...OK!
```

Figure A.1: A Sample Output File

Using this file it is possible to analyse the design in order to perform a bottleneck analysis. The file gives the designer detailed information about the node and its physical capacitances (second column). Furthermore, the average number of transitions occurred at each node during the iterations (located in the third column) are used to calculate the active node capacitance (fourth column). With this information the designer is able to find 'hot' areas of the design and direct the design efforts accordingly. The file is a simple ANSI file having tab-stops as separators. Therefore, it can be easily imported into spreadsheet programs or other simulation environments such as Matlab. This is particularly useful for further analysis or, as done in this thesis, for graphic representation of the results of power estimations.