REPRINT FROM: PROC. OF IRISCH SIGNAL AND SYSTEM CONFERENCE, DERRY, NORTHERN IRELAND, PP.165-172.

Low Power VLSI CMOS Design

An Image Processing Chip for RGB to HSI Conversion

A.Th. Schwarzbacher and J.B. Foley
Department of Electronic and Electrical Engineering, Trinity College
Trinity College, Dublin2, Ireland

Abstract: Up until recently performance parameters such as area and speed were the important key factors in the design of integrated circuits. With the introduction of smaller technologies these objectives were more easily achieved. However, increasing integration and higher clock speeds has moved power dissipation to the forefront of design challenges facing today's chip designers.

By implementing low power structures from the first stages of the design cycle expensive cooling techniques and packaging can be avoided, even when implementing fast real time image processing algorithms.

Most research towards low power design has been done at the lower levels of the disign hierarchy. However, the field of high level or architectural level low power design is likely to be of increasing significance. The aim of this project is to demonstrate that the use of low power strategies at the initial stages of the design development cycle will lead to a significantly reduced power consumption in the final chip.

Keywords: High Level Low Power CMOS Design, RGB to HSI Conversion.

1 Introduction

Most research towards low power design has been done at the lower levels of integrated circuit (IC) design. However, the aim of this paper is to show that the use of low power strategies at the initial stages of the design development cycle will lead to a significantly reduced power consumption in the final chip. For this purpose Kenders HSI (hue, saturation, intensity) algorithm for the transformation of an eight bit RBG (red, blue, green) camera signal into the HSI code was chosen. The algorithm is shown below without further explanation since it is used in this paper as a vehicle to present an approach taken towards a low power implementation. For further references about the HSI algorithm and Kenders implementation consult paper [1].

1.1 Kenders Algorithm for Faster Computation of HUE

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

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

else if (G > R)

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

else if (B > G)

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

else if (R > B)
hue = 0

else
'achromatic'

1.2 Computation of the Saturation

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

1.3 Computation of the Intensity

$$\text{int}ensity = \frac{R + G + B}{3}$$

2 Power Consumption

Before the implementation of the HSI Algorithm is explained some essential background on power dissipation is presented in this section.

Power dissipation is caused by three major sources. The overall power consumption is calculated by:

$$P_{total} = P_{switching} + P_{short\text{-}circuit} + P_{leakage} \quad (2.1)$$

$P_{switching}$ represents the switching component of the power dissipation, also called dynamic power dissipation, which occurs each time a power consuming transition is performed. The $P_{short\text{-}circuit}$ term represents the short circuit path which arises if both NMOS and PMOS transistors are open and a path is connected directly between supply and ground. The $P_{leakage}$ losses are due to substrate injection and subthreshold effects [2].

This paper will concentrate on the power dissipation caused by transitions. This is also reasonable since the dynamic power dissipation is typically 90% and above of the total power dissipation[2]. Equation 2.2 presents the general equation for the switching power.

$$P_{switching} = \sum (n_{(0,1)} C_{node}) \times V_{dd}^{\ 2} \times f_{clk} \quad (2.2)$$

In this equation f_{clk} represents the clock frequency, $V_{dd}^{\ 2}$ is the supply voltage squared. It is assumed that the supply voltage is equal to the swing voltage, otherwise this term has to be replaced by $V_{dd} * V_{swing}$. The active capacitance represents the sum of all node capacitance multiplied by the number of power consuming transactions at each node per clock cycle.

From the point of view of the supply a power consuming event occurs only when a low to high transition occurs. During such an event the node capacitance C_{node} is charged with the supply voltage V_{dd}. If a high to low transition takes place the energy stored in the node is discharged, but no power is consumed since no power is taken out of the power supply during a high to low switching event.

2.1 Reduction of the Supply Voltage

The most obvious and most common way to reduce power consumption is to reduce the supply voltage. The supply voltage is present in all terms of the total power consumption equation (eq. 2.1). Therefore the most important factor in reducing the power consumption is a reduction in the supply voltage.

2.2 Delay and Voltage

Figure 2.1, (adapted from [2]) shows that the delay is highly supply voltage dependent which limits the reduction in the supply voltage.

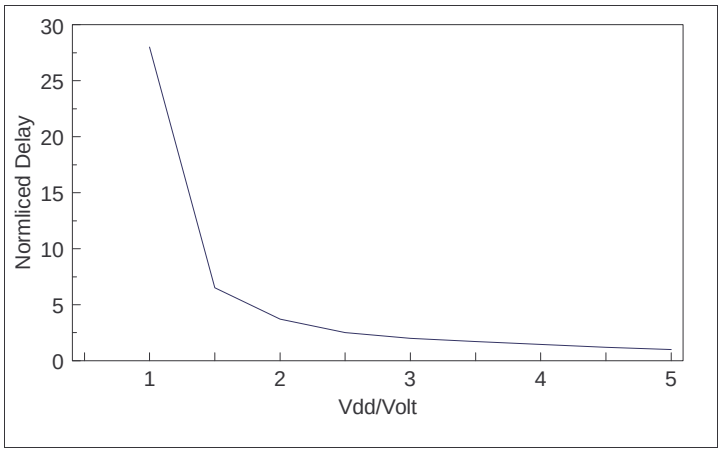

Figure 2.1

As seen in Figure 2.1 the delay will double at a voltage of approximately 2.9V compared to 5V. At this point a reduction in the dynamic power dissipation of approximately 66% can be achieved when compared to the 5V standard. Therefore it is essential to run the circuit at the lowest possible supply voltage at which the minimum throughput is guaranteed in order to achieve the lowest possible power consumption.

To run a circuit at the lowest possible voltage it is important to minimise the critical path in the design so that not one path limits the required throughput significantly more than any other path. Typically this has been done by using techniques such as pipelining and parallelism.

2.3 Reduction of the Number of Power Consuming Events

2.3.1 Number Representation

Because the total power consumption is highly dependent on the switching activity, this section analyses the effects caused by the representation of numbers in the design.

Most signals are represented by two's-complement, because this makes arithmetic processes such as addition and subtraction very easy to execute. Positive values are expressed as a bit integer. Negative values are obtained from the positive values were the number one is added and the result is completely inverted. This means that the most significant bit (MSB) and all non-information carrying bits represent the sign-extension. This is also the reason why two's-complement is not recommended for low power implementations as the following example illustrates. A signal consisting of eight bits is be considered. If the present state of the signal is +1 the bus is set to "0000 0001". If the signal is changes to -1 the bus now changes to "1111 1111". Each time the sign changes all higher bits also perform a change, because the sign is represented in all of the bits which are not used to represent a number. Therefore two's-complement consumes a lot of power while transmitting minimum information. In the example above only two bits are necessary to transmit the information, but seven power consuming

transitions occur. These transitions consume 60% of the total dynamic power and are redundant.

The best solution for this problem is to split the signal into sign and magnitude. Now only the most significant bit carries the sign information and all other bits are used to represent the unsigned number. If we now take another look at the example above, a change from +1 to -1 would only cause the highest and lowest bit to change from 0 to 1 and consume only 12.5% of the switching power compared to the previous example. This will also cause a reduction in the activity of computation. If during a computation the sign changes the same rules apply as shown above. Therefore it is also useful to compute the sign and magnitude in separate units of a functional block instead of using two's-complement circuits. This requires special circuits which are more complex and larger than those used to compute in two's-complement format. Therefore the trade-off between larger capacitance and logic on one hand and the lower switching activity of the sign-magnitude representation has to be analysed before deciding on an implementation.

2.3.2 Minimising Glitching Activity.

Due to finite propagation delay through logic blocks the output of a device can have different values during one clock cycle before settling to the correct value. This is called glitching and amounts to unnecessary transitions. Typically these glitches produce around 20% of the total power consumption, which might rise up to 70% of the total power in some cases such as combinatorial adders[3].

To alleviate this problem balanced design structures such as tree adder should be used. Highly pipelined structures also help in avoiding critical races with the tradeoff of higher area. Additional logic can also reduce glitching significantly, e.g. the precomputation of the carry in adders.

2.3.3 Optimisation of Constant Operation

The extensive use of Hardware Description Languages (HDLs) leads to the use of multipurpose functions or design units from previous projects. This guarantees the shortest possible design cycles and a very cost effective project management. However, in DSP applications multiplying with fixed coefficients is often required. These coefficients are normally known before the actual design process begins. By using optimised multiplier structures rather than a general-purpose multiplier, the number of operations, the delay through the block or area and therefore capacitance can effectively be reduced [4].

3 Implementation of the HSI Algorithm

The following section describes the implementation of the RGB-HSI algorithm and explains what high-level design decisions were taken in order to reduce the power consumption of the overall chip.

3.1 A Low Power Implementation of Kender's Algorithm for Fast Computation

3.1.1 Basic Implementation Considerations

As shown in the previous section it is most desirable to implement designs unsigned. The key to the implementation of an unsigned version of Kenders algorithm is understanding the arguments of the arctan function. The arctan function appears in the first three parts of the hue algorithm and can generally be written as follows:

$$\arctan\left(\frac{\sqrt{3}\times(X-Y)}{X-Z+Y-Z}\right) \tag{3.1}$$

In this equation Z is the smallest of the three variables. 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 has point symmetry through the origin, the behaviour of the function can also be described as:

$$\arctan(x) = -\arctan(-x) \tag{3.2}$$

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. This also halves the range of the arctan function which has to be calculated.

The normal way to calculate the arctan function is by using the Taylor Series presented in equation (3.3).

$$\arctan\begin{cases} x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots & ;|x| < 1 \\ \pm\pi + x + \frac{x^3}{3} - \frac{x^5}{5} + \ldots & ;|x| \geq 1 \end{cases} \tag{3.3}$$

That Taylor Series has proven to be a poor way of realising a low power arctan function. The increasing exponents will increase the number of bits which have to be computed. This results not only in large buses with correspondingly high activity but also in huge functional blocks. Other "intelligent" implementations would need complex control logic which itself would consume a significant part of the power. Another problem of the arctan function is the presentation of the factor $\sqrt{3}$. Since it is not possible to present this value with an accuracy of 100%, an implementation would either contain a significant error or be unreasonably large. Large busses not only have the drawback of higher switching but also lead to longer interconnection lengths and larger functional units due to the fact that the subsequent units have to compute more bits.

For these reasons a different way of implementing the arctan function was needed. The most reasonable way to do this was by implementing a Look Up Table (LUT). Look Up Tables are simple Read Only Memory (ROM) storing devices. They contain only one set of data (one number) for each input address. These numbers represent the result of the process which is performed by the LUT. Because the LUT is only a storing device the result has to be calculated for each possible input value before implementing the LUT. Because all possible output values are already calculated LUT's are very fast compared to an algorithmic implementation.

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 8 by 8 bit multiplier would result in a memory of more than 1.5 Mbits. This excludes most computations. Normal LUT's consume a lot of power due to precharging. Therefore special circuits as presented in [5] should be used in order to reduce the power up to 75%, depending on the minimal allowable swing voltage at the bitlines.

At this stage a look at the possible output values of the hue algorithm should be undertaken. Because today's digital RGB standard uses a quantisation of the input signal into 8 unsigned bits for each colour, the hue output can also only 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 two bits have to be reserved for singularities where hue is zero or achromatic. This leaves a possible range of 253 for describing the hue space. The algorithm splits the output range in three separate units each of the same range. Therefore the output range is split into three parts, each containing 84 values. This gives a total of 252 plus two values for singularities. Because it is not possible to use one value the dynamic range of hue is reduced by 0.39%.

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 algorithm ensure that the hue space is unique for all three cases and that each of the ranges spread from -

90° to +90° 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 of (3.2) the number of values can be reduced by a factor of two since only the positive values have to be stored. 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 (3.1) the maximum value of the argument of the function is $\sqrt{3}$. Therefore the maximum value of the remaining terms must be one. 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.

3.2 Implementation of the Hue Algorithm

As shown in 2.3.1 it is desirable to represent numbers using the sign-magnitude system. As described in (3.1) only the subtraction in the divisor determines the sign of the arctan function. Normally determination of the sign would be done by using a sign-magnitude subtractor. Since all comparisons are done in order to determine which of the five cases of Kender's algorithm is true, the two values in the dividend can now be sorted and the smaller value is subtracted from the larger one. This has the advantage that an ordinary unsigned subtractor can be used instead of a larger and more complex sign magnitude subtractor. This also has the advantage that the sign is excluded at the earliest possible stage and therefore it was possible to reduce the bus size by one bit which results in smaller functional blocks in the following stages.

If one of the last two cases in Kender's algorithm is true then the result is determined and need not to be computed. Therefore the previous result is kept at the output of this module, all remaining stages are disabled and only the control bus changes value. This ensures that the minimum number of transactions is performed.

In the next stage of the process the argument of the arctan function is calculated. The maximum value of ((X-Y)/(Y-Z+X-Z)) is one. Because it is most desirable to work with integer numbers the value of the dividend is multiplied with an integer number before it is divided. This integer number not only defines the maximum output value, but also the error of the module.

In order to reduce the switching activity it is necessary to reduce the width of buses. Simulations have shown that it is possible to reduce the range of numbers down to 60. This results in a rounding at 0,7 instead of 0,5 depending of the address accessed in the LUT or in other words the output of the hue algorithm might have a maximum absolute error of +0.2 when compared with a floating point computation without the need of a rounding logic.

A different implementation using a multiplication with the smallest possible value was designed. This time the value of the dividend was multiplied by the value 63. This can be done by simply shifting the output bus 5 times. Since this operation doesn't require any logic it does not consume any power. But the drawback of this approach is that the output bus is now one bit larger and therefore the area and delay of the following functional block will increase.

In order to keep the clock frequency as low as possible a pipelined parallel array divider structure suitable for implementation with hardware description languages (HDLs) was chosen in order to compute the argument of the arctan function.

3.3 Implementation of the Saturation Algorithm

The algorithm for the computation of the saturation is given in 1.2. The first step in order to compute the saturation is to determine which input signal (R, G or B) has the smallest value. As described in 3.3 this was already done in order to avoid two's-complement notation. Therefore this value already exists on a bus and can be used for this computation.

The next value to compute is the sum of all three input signals. The divisor of (3.1) can also be written as follows.

$$\text{Divisor} = X + Y - 2Z \qquad (3.4)$$

As already explained value Z is the smallest of the three input signals and this value also exists on a bus in the hue module. Now instead of calculating the sum of the input signals, 3Z is added to this value in order to compute the sum of the input signals. This reduces the amount of functional logic as well as the amount of pipeline latches.

3.4 Implementation of the Intensity Algorithm

As presented in section 1 the intensity is the sum of the input signals divided by three. The sum of all input signals was calculated for the saturation and so this module need only divide this value by three. Instead of using a universal dividing structure and applying the constant three to the input, this fixed value was declared as a constant in the HDL description. This allowed optimisation during compilation. Table 3.1 compares performance measures before and after optimisation.

	Standard	Optimized	Reduction / [%]
Max. delay / [ns]	34.24	15.4	55
Number of nets	318	50	84
Area / [μm^2]	127018	22496	82
C_{active} / [pF]	9.3	2.6	72

Table 3.1

Even if the first three values do not reflect the power consumption directly the active capacitance does have a direct effect on the power consumption. Additional power savings of more then 60% can be achieved if the supply voltage for the optimised divider is halved.

The overall structure of the proposed design is presented in Figure 1.

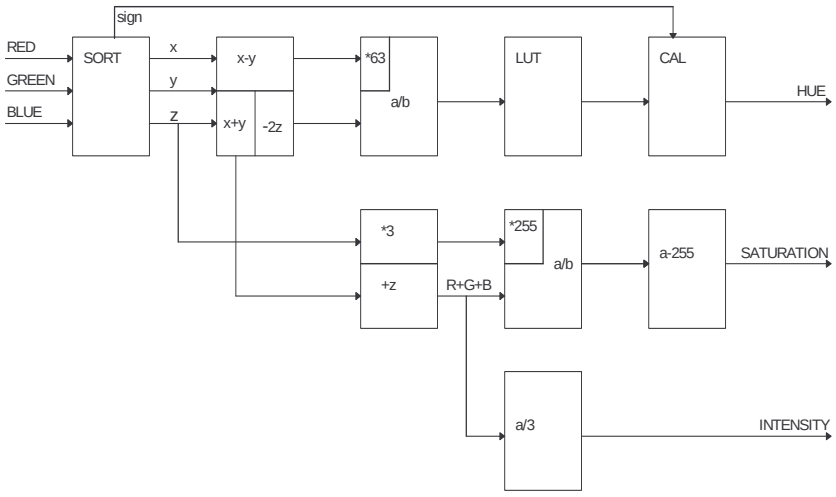

Figure1

4 Conclusions

A theoretical approach towards the low power implementation of Kenders Algorithm for the fast computation of hue was presented. Emphasis was placed upon the need to use unsigned logic as well as the early extraction of the sign information. This resulted in a smaller and less complex logic with lower switching activity and lower area (and consequently lower capacitance). The computation of the arctan function was performed by the use of an LUT in

order to avoid the otherwise complex and therefore power consuming computation. The hue space also allowed a reduction in the number of bits down to 6 through uniformly distributing the output range with the drawback of a decrease in range of 0.39%.

The power consumption in the saturation and intensity algorithm is reduced by the use of common subexpressions as well as the use of optimised logic blocks. Furthermore, investigation of control logic has indicated more power savings as switching is disabled in unused stages.

5 Acknowledgements

The authors wish to thank the Dublin Institute of Technology, in particular Mr. Chris Cowley and Mr. Paul Comiskey for access to their research facilities.

6 References

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

[2] A. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 84, no. 4, pp.498-523, Apr. 1995.

[3] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*. vol. 2, no. 4, pp. 446-455, Dec. 1994.

[4] R. W. Brodersen, J. Rabaey, *et al*, "Optimising power using transformations," *EECS Department, University of California*, http://infopad.EECS.Berkeley.edu/infopad-ftp/papers.

[5] A. Chandrakasan, A. Burstein, *et al*, "A low-power chipset for a portable multimedia I/O terminal". *IEEE J. of Solid-State Circuits*, vol. 29, no. 12, pp.1415-1428, Dec. 1994.