Probabilistic Computing for Efficient Robotic Vision in Space

Final Report

Authors: Ala' Qadi[1], Rob Hewitt[2], Guido De Croon[3]
Affiliations: [1]Carleton University, [2]Queens University, [3]ESA Advanced Concepts Team

Date: 10-2-2014

Contacts:

Alex Ellery

Tel:	Office: +1 613 261 3765, Cell: 613-520-2600 Ext 1027
Fax:	+1 613 520 5715
e-mail:	aellery@mae.carleton.ca

Leopold Summerer (Technical Officer)

Tel:	+31(0)715654192
Fax:	+31(0)715658018
e-mail:	act@esa.int

Ariadna ID: 12-5101
Ariadna study type: Standard
Contract Number: 4000105987/12/NL/KML

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In space robotics, energy is at a prime: space robots need to function for extended periods of time with a limited amount of energy. For example, the Mars rover *Opportunity* was only planned to operate for 90 sols (Martian days), since the designers expected the dust to reduce the solar panels' efficiency at a rapid rate. Fortunately, Martian dust devils wipe the solar panels clean, allowing Opportunity to function far beyond its planned operation horizon.

In this study, we focus on reducing the energy spent in computation. While for large robots the energy spent on computation may represent a negligible fraction of the energy budget, its importance increases for ever smaller robots. In particular, we investigate how *probabilistic* computing can be employed for space robotics. In typical processor hardware considerable amounts of energy are spent on obtaining correct calculation results, e.g., for adding or multiplying numbers. In probabilistic computing [8, 21, 17], the energy spent by the processing units is lowered, resulting in an increase of the probability that some operations might go wrong. Fortunately, it has been shown that the amount of energy saved is significantly larger than the amount of probability traded in [8]. As a consequence, in theory, large energy gains (in the order of 5 times) can be obtained at a minimal cost in calculation errors.

Different applications of probabilistic computing have been investigated. The study in [8] focused on the application of probabilistic computing to the calculation of the Fourier transform, showing that for human subjects the probabilistic reconstruction is visually identical to the error-free one. Another study involved the application of probabilistic computing to Continuous Restricted Boltzmann Machines (CBRM) [21]. Such CBRMs normally require the generation of pseudo-random numbers. Instead, in [21] probabilistic computing naturally provided random numbers.

This study proposes to perform research on probabilistic computing in a different context, namely that of a vision algorithm for robotics. The main idea behind the application is that a slight loss in accuracy in the computations of the vision algorithms can still lead to robust and successful retrieval of information on the environment. The study will focus on stereo vision, since it is such an essential part of the current rovers' autonomous navigation capabilities.

In the remainder of the final report, we first discuss the probabilistic circuit design (Chapter 2). Since in space environments radiation has a much larger effect on the calculations in a processor, we also introduce a radiation model (Chapter 3). The simulation of the circuit is explained in Chapter 4. Subsequently, we discuss the experiments used to evaluate the effects of probabilistic computing and radiation on the accuracy of stereo vision (Chapter 5). We draw conclusions in Chapter 6.

Chapter 2

Circuit Design

2.1 Disparity Equations Analysis

In stereo vision, objects that are far away from the cameras have (close to) the same image coordinates in both the left and the right image, while objects close by can be significantly shifted in the x-direction. The shift in pixels between the same scene point in the left and right image is called the 'disparity', which is directly related to the distance from the cameras to that scene point. A typical stereo vision algorithm will attempt to match all locations (pixels) in the left image to a range of disparity values in the right image. This matching is most often not done on the basis of single pixels, but on the basis of small image windows in the left and right image. Hence, it is easy to see why this part of the algorithm is computationally the most intensive part of stereo vision: the computation involves matching two image windows per disparity in the disparity range and per pixel in the left image. The disparity calculation is the focus of this section. Below we will look more in detail at the computational complexity and the type of computations performed in the algorithm.

The disparity between pixels in stereo vision algorithm is generally given by the calculation of either sum of square difference between the pixels in the two images (2.1) or the sum of absolute difference between the two images (2.2).

$$C = \sum_{i,j \in W} \left(I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j)\right)^2 \tag{2.1}$$

$$C = \sum_{i,j \in W} |I_1(x+i, y+j) - I_2(x + \Delta x + i, y+j)| \tag{2.2}$$

Figure 2.1 illustrates the parameters of Equation (2.1) in relation to both images and the windows chosen for the calculation. Before expanding Equation (2.1) into pseudo code for calculation, let's introduce the following notation for the various variables of the equation and the algorithm:

Let *x* denote the horizontal axis of the image and let y denote the vertical coordinates of the image. Let *X* be the set of *x* coordinates of the top corners the windows chosen from image 1. Let *Y* be the set of *y* coordinates of the top corners the windows chosen from image 1.

Let *DX* be the set of the sweep distances *x* from the corner of the windows of image 1 to image 2 windows.

Let *i* denote the relative *x* (horizontal) coordinate of a pixel inside the window $W_{x,y}$

Let *j* denote the relative *y* (vertical) coordinate of a pixel inside the window $W_{x,y}$

Let *m* denote the number of pixel columns in any window *W*

Let *n* denote the number of pixel rows in any window W

Let *Dim(X)* indicate the dimension of *X*. i.e. number of elements in *X*

Let *Dim(Y)* indicate the dimension of *Y* i.e. number of elements in *Y*

Let *Dim(DX)* indicate the dimension of *DX* i.e. number of elements in *DX*.

Clearly (2.1) has to be calculated for every pixel in every window from image 1 with every pixel of sweeping window image 2 in the horizontal direction. If we do that for every window in image 1 with every sweeping window then the calculation of (2.1) involves five loops as shown in Figure 2.2 .

The largest proportion of commuting time will be due to the multiplication element in the equation. Let η denote the number of multiplications in the algorithm used in Figure 2.2 to calculate (2.1) is given by (2.5)

$$\eta = n \cdot m \cdot Dim(DX) \cdot Dim(X) \cdot Dim(Y) \tag{2.3}$$

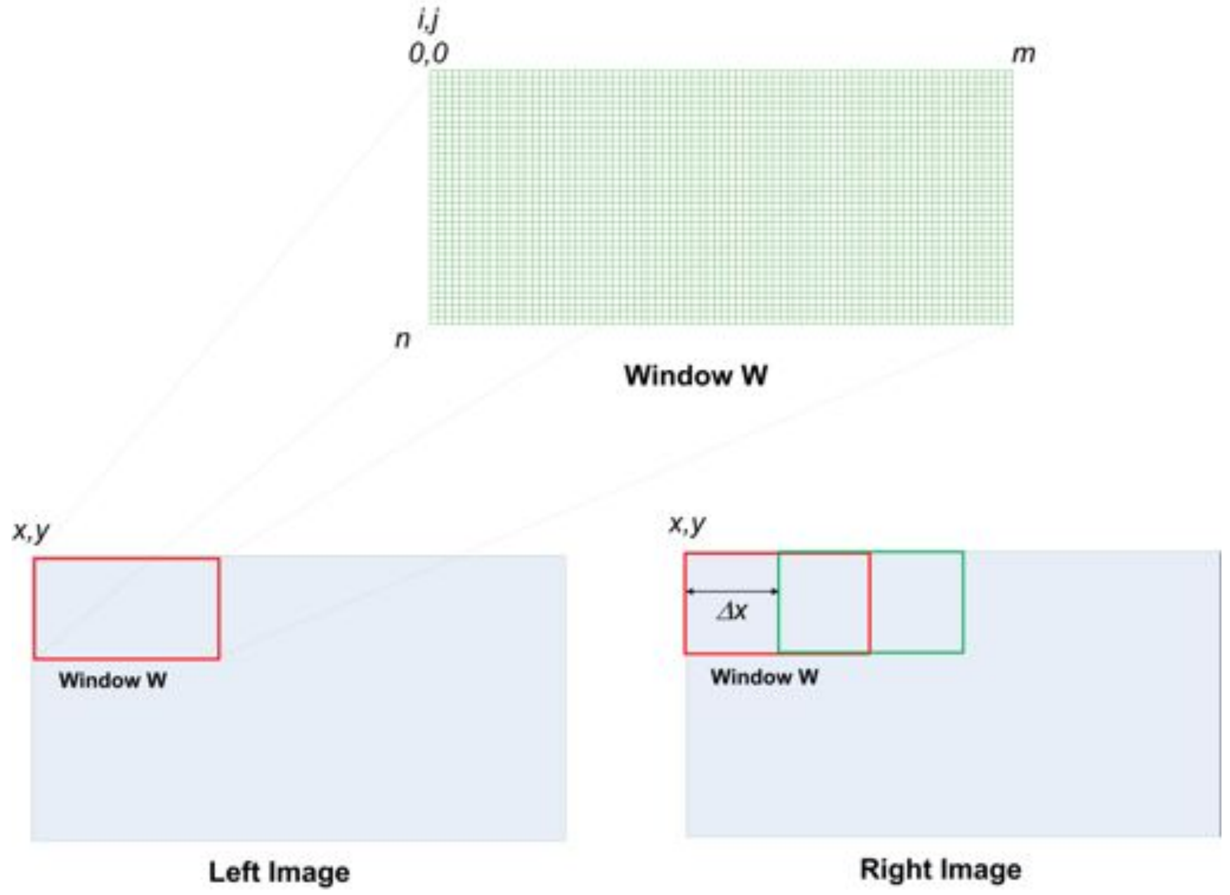


Figure 2.1. Illustration of stereo vision equation variables

```
$For(x \in X)$
        $For(y \in Y)$
                $For(\Delta x \in DX)$
                        $C_{x,y,\Delta x,i,j} = \sum_{i=0}^{n} \sum_{j=0}^{m} (I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j))^2$
                $end$
        $end$
$end$
```

Figure 2.2. Pseudo code for stereo vision equation (square differences)

There are six additions inside the equation

$$C_{x,y,\Delta x,i,j} = (I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j))^2 \tag{2.4}$$

Repetitive additions are needed to calculate the summation. The total number of additions in the algorithm discarding loop counters donated by μ is

```
$For(x \in X)$
    $For(y \in Y)$
        $For(\Delta x \in DX)$
        $sumi_{x,y,\Delta x} = 0$
            $For(i = 0, i \leq n, i++)$
            $sumj_i = 0$
                $For(j = 0, j \leq m, j++)$
                $S = (I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j))$
                $if(S \geq 0)$
                  $C_{x,y,\Delta x,i,j} = (I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j))$
                $else$
                  $C_{x,y,\Delta x,i,j} = -(I_1(x+i, y+j) - I_2(x+\Delta x+i, y+j))$
                $endif$
                    $sumj_i = sumj_i + C_{x,y,\Delta x,i,j}$
                $end$
            $end$
        $end$
    $end$
$end$
```

Figure 2.3. Loop iterative calculation of vision equation (absolute differences differences)

$$\mu = 7 \cdot n \cdot m \cdot Dim(DX) \cdot Dim(X) \cdot Dim(Y) + n \cdot Dim(DX) \cdot Dim(X) \cdot Dim(Y) \tag{2.5}$$

We can see that $Dim(X)$, $Dim(Y)$ and $Dim(DX)$ determine the complexity of the algorithm and the growth of calculation time. Given that multiplications have the greatest impact on delays, it will be the main part targeted for speed up through hardware calculations. Since the calculations are done on the pixels in the array independently inside each window the calculations can be parallellized.

For the sum of absolute differences, the most intensive part would calculation of the absolute value and then the cumulative sum similar to the sum of square differences. The loop iterative calculation of the vision equation for the sum of absolute difference can be expanded as shown in Figure 2.3

The disparity equation is calculated per pixel, so we can parallellize the calculations. To ease the parallelization one can divide every window W into a number of sub windows as shown in Figure 2.4. Choose the number of sub windows to be of size $q = 2^k$, where k is an integer. By choosing q as power of 2 we can assign each block of the to a multi-level hardware tree.

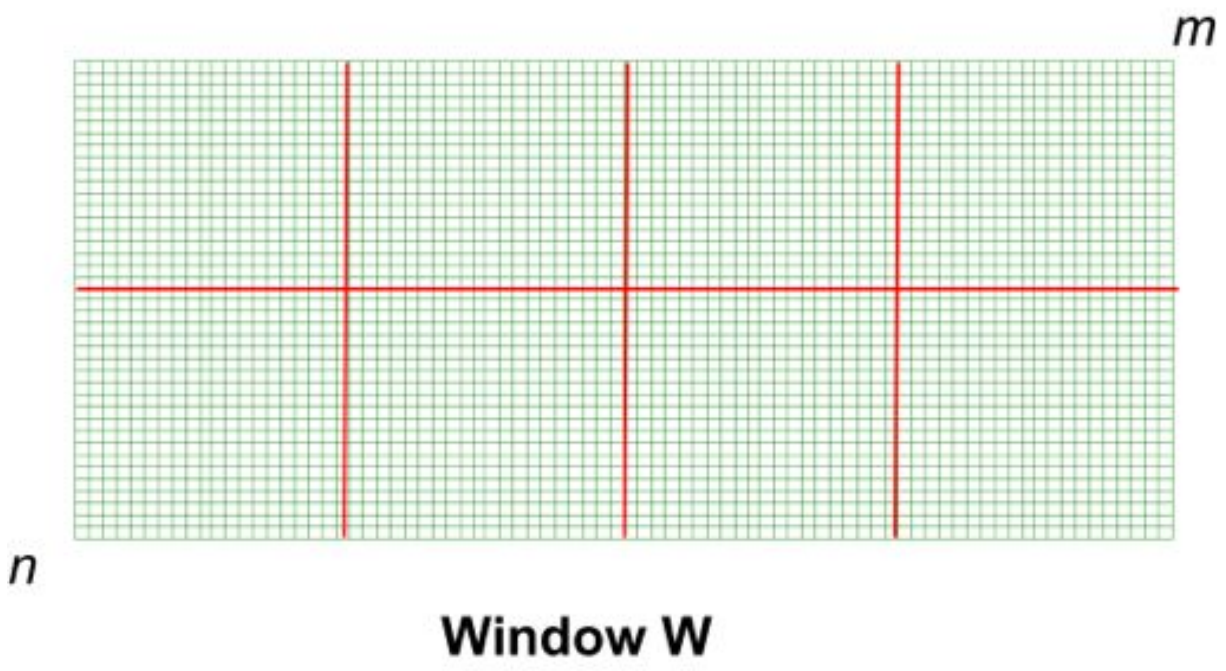


Figure 2.4. Window Parallelization Example $q = 8, k = 3$

2.2 RTL Circuit Level Design

The design of hardware to increase the speed of the vision algorithm will be made on the RTL level and possibly to gate level (if needed), as the device or transistor level circuit design is beyond the time frame and scope of this study. The speed up of the calculation of Equation (2.1) can be done by parallellizing the multiplication and the calculation of partial sums. If we leave the choice of window size aside we can consider the calculation per window by using the assumption above that each window W is of a size of n x m pixels.

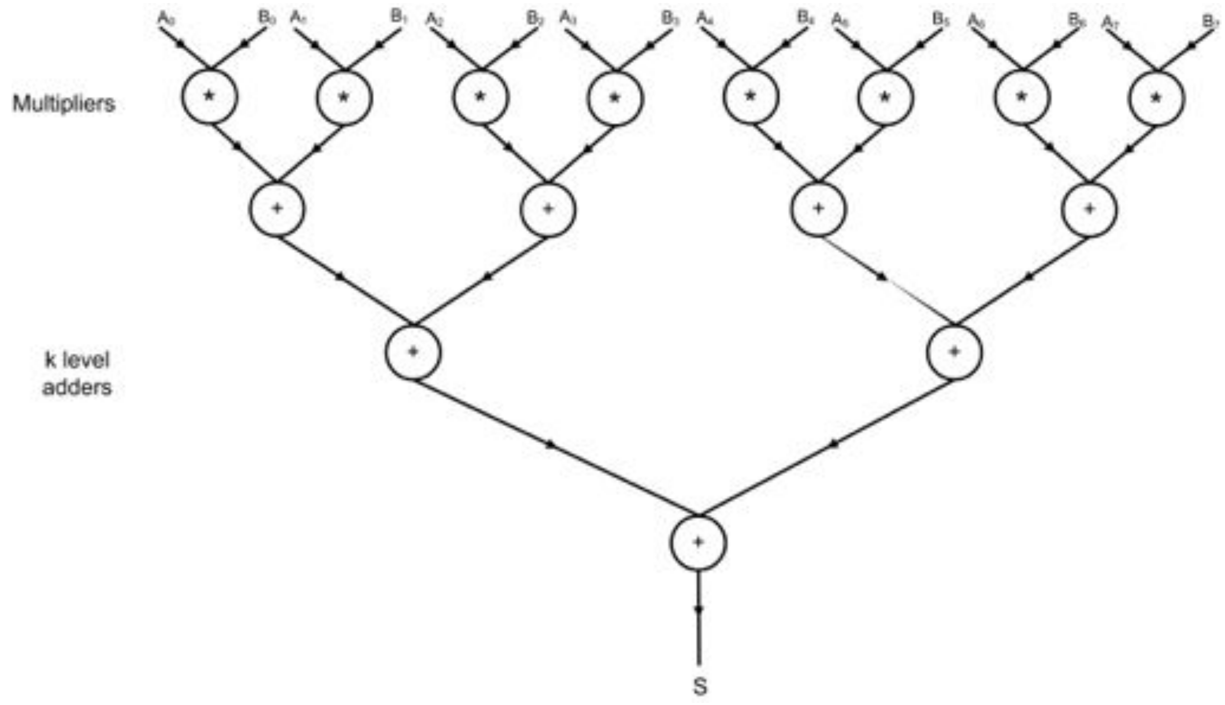
Assume we want to parallellize the multiplication with q multipliers, and q can be written as in (2.6), where k is an integer as

$$q = 2^k \tag{2.6}$$

To calculate the summation after each multiplication we need k levels of adders, where each adder will add the result of two multipliers plus one adder at the end of the last level to add the result to the previous result obtained from the summation. The numbers of adders l needed is given by Equation (2.7)

$$l = 1 + \sum_{i=0} 2^{i^{k-1}} = 1 + (2 \cdot 2^{k-1} - 1) = 2^k \tag{2.7}$$

We can see from Equation (2.7) that the number of adders needed is the same as the number of multipliers. Figure 2.5 shows the combinatorial logic design of an 8 multiplier logic circuit that can be used to calculate a parallellized iteration of Equation (2.1).



A_0 B_0 A_1 B_1 A_2 B_2 A_3 B_3 A_4 B_4 A_5 B_5 A_6 B_6 A_7 B_7
Multipliers
k level adders
S

Figure 2.5. Combinational design of 8 multiplier logic circuit to calculate the vision equation

The next step is to calculate Equation (2.1) for the whole set of points in W. This involved proceeding with an RTL (Register Transfer Logic) level design of the circuit, which adds registers after each stage and one more adder stage to feed back the previous sum in order to accumulate the total sum S (Shown as the red output line in Figures 2.5 and 2.6). Figure 2.6 shows a 4 multiplier RTL circuit that can be used to calculate Equation (2.1). Figure 2.5 shows a general q multiplier k level RTL circuit that can be used to Equation (2.1).

Next we show how parallellizing will speed up the calculation. Let ΔM represent the delay from the multiplier and ΔA represent the time delay for the adder. And let t denote the time for calculating Equation (2.1) per window W ignoring the inner additions before the multiplication (which can be included later in the logic circuit if we need more speed up of calculation).

If the algorithm is done with one ALU, then an estimate of t is given by Equation (2.8),

$$t = n \cdot m \cdot \Delta M + (n \cdot m + 1) \cdot \Delta A \tag{2.8}$$

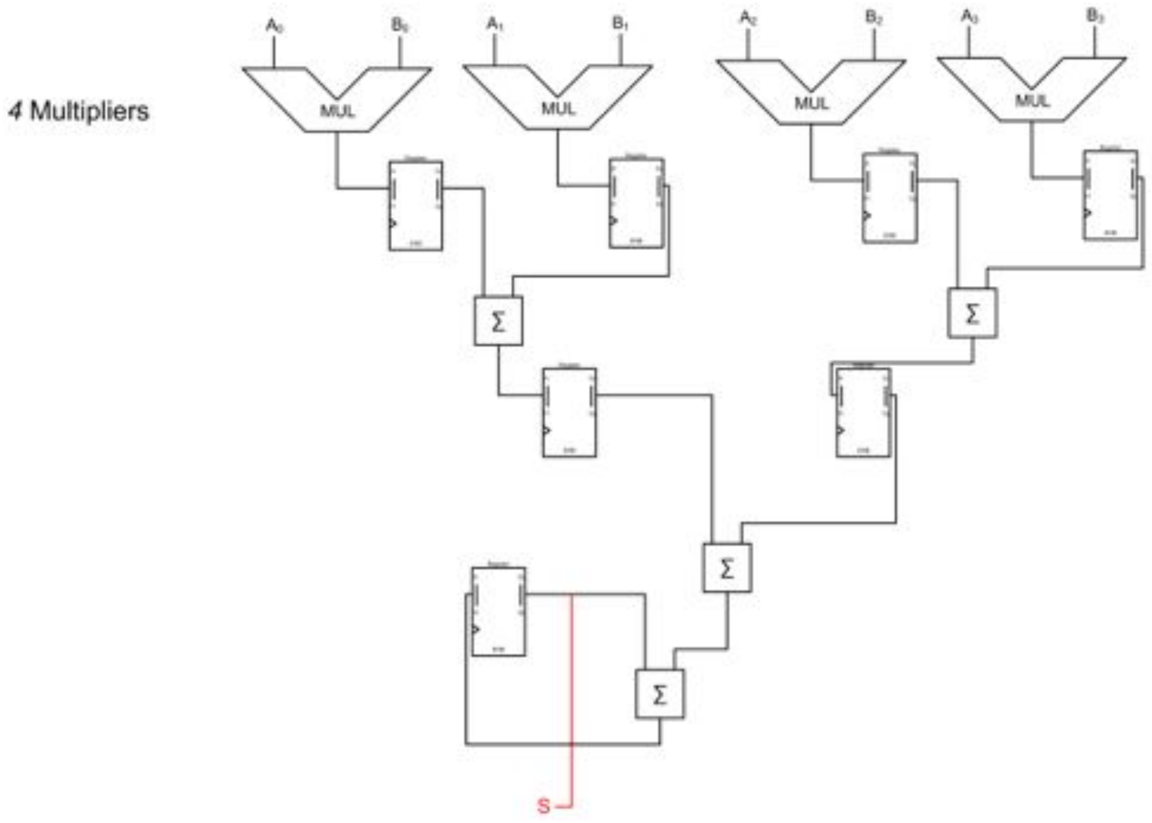


Figure 2.6. An example of 4 multiplier RTL level logic circuit to calculate the vision equation

If the parallellized array multiplier in Figure 6 is used the estimate of t is given by Equation (2.9)

$$t = \frac{n \cdot m \cdot}{2^k}(\Delta M + (k+1) \cdot \Delta A) \tag{2.9}$$

The effect of using the RTL level circuit is that while we get an improvement in the performance we will have more energy consumption from the dedicated logic which might have to be implemented on a separate ASIC chip or combined with the regular onboard computer in an FPGA implementation. The more multipliers we use (i.e. a larger q) the larger the number of gates and RTL units will be needed which will increase the amount of transistors, area on the chip and energy consumption [2]. The compromise between energy consumption and speed-up through the use of dynamic voltage scaling and probabilistic computing explained in the next section.

In the case of sum of absolute differences. If the multiplication operation is changed to an absolute value calculation, it can be calculate by a designated unit. Then we can substitute every unit with an absolute value calculation unit which we will call ABS. An example of an RTL Level circuit to calculate the disparity equation, Sum of Absolute Difference, $q = 4$ is shown in Figure 2.7. The general RTL circuit to calculate the sum of absolute differences with n ABS unit is shown in Figure 2.8.

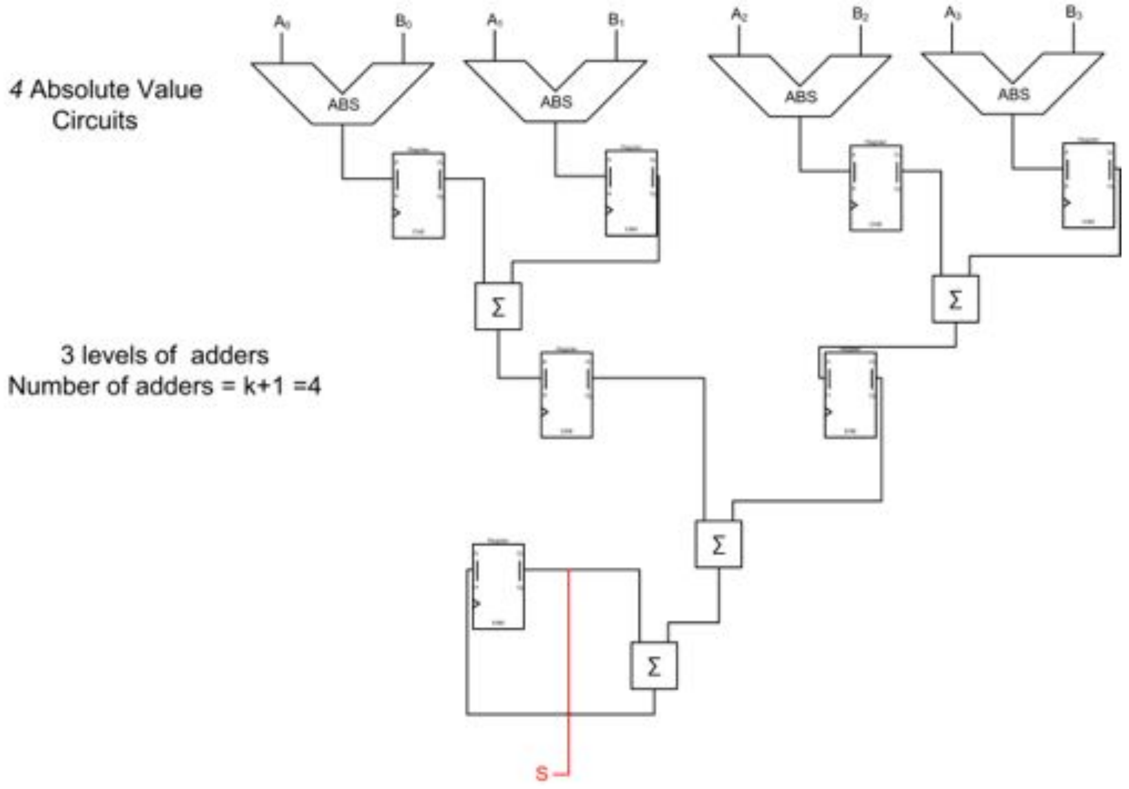


Figure 2.7. An example of 4 multiplier RTL level logic circuit to calculate the vision equation for sum of absolute differences.

It is easy to calculate the absolute value with two's complement calculation. Two's complement representation is used in most digital arithmetic systems. In a twos complement representation, negative numbers are represented as the binary inversion of the number plus one. The most significant bit is the sign bit (1 for negative, 0 for positive). Since we only need the sign of the comparison, we can use the properties of twos complement arithmetic and use an adder instead. Calculating the negative of a number in two's complement can be accomplished by inverting all the bits of the number and then adding one.

Figure 2.12 shows a possible design for the absolute value calculation unit. In this design we use two adders and two multiplexes, the first adder at the input serves as a magnitude comparator while multiplexor 1 passes either A or $\bar{A}$, multiplexor 2 passes either B or $\bar{B}$.

2.3 Control Logic

In order to control the flow of the data from each stage of the adder from the combinational parts (absolute value circuits and adders) to the registers. The control logic is also necessary to control when the circuit starts again after calculating and enables the register for the last stage adder to reload after the loading of each block of pixels. This is achieved by enabling the load the input register for the last adder in the tree and disabling the loading of the other registers in the earlier stages using an inverter. The timing of the loading of each adder after the tree finished the calculation of a complete block is done by comparing the count of a pre-loaded counter with number of cycles required for the tree to calculate the disparity value for a block of pixels as given by Equation (2.10).

$$MaxCount = \left\lceil \frac{NumberofSelectedPixels}{q} \right\rceil + k + 2 \tag{2.10}$$

2.4 Adder Gate Level Design

The base unit of our modular design is the adder. To include both probabilistic computing and radiation effects we have to design the adder down to the gate level. While there are many standard designs for n-bits adders available in industry and literature with various methods of look ahead logic and other more complex methods to reduce delay in the adder, we will use the standard adder design using only full adders. The reason for choosing the design with full adders is to retain the modularity of the design and be able to substitute the adder with the probabilistic BIVOS adder presented in [8].

The design of the full adder is done using a NAND gate. The NAND gate is chosen because NAND is a universal gate that can be used to implement any combinatorial logic circuit.

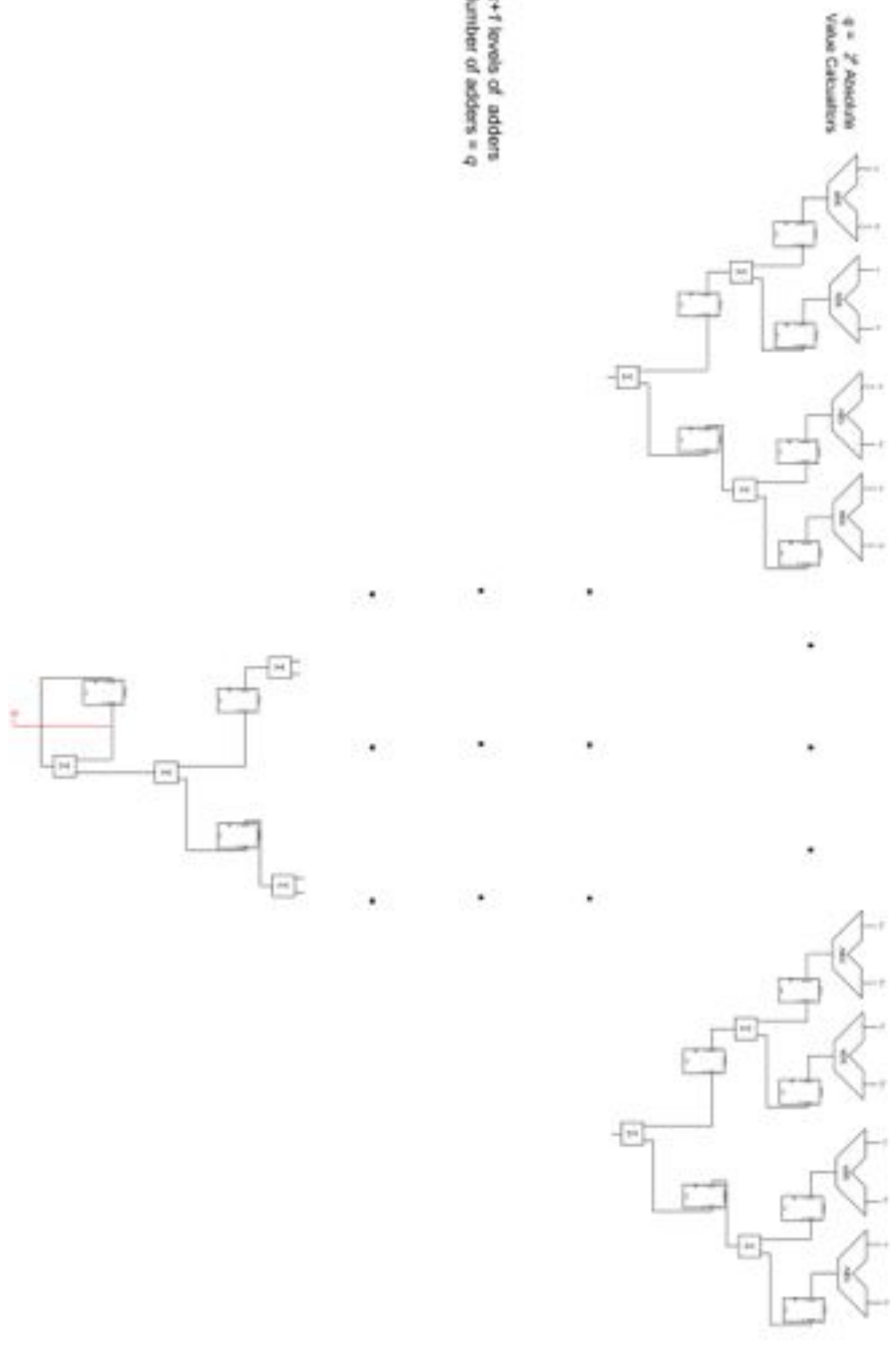


Figure 2.8. A general RTL level circuit with q ABS circuits to calculate the disparity equation for sum of absolute differences.

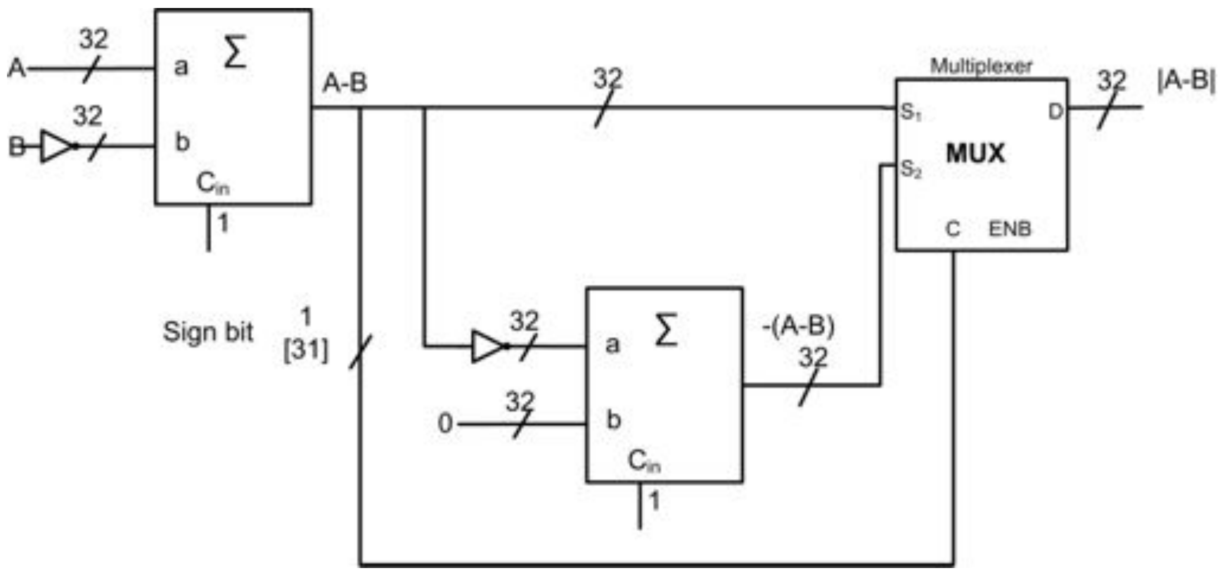


Figure 2.9. Design 1, Absolute Value Calculation Unit

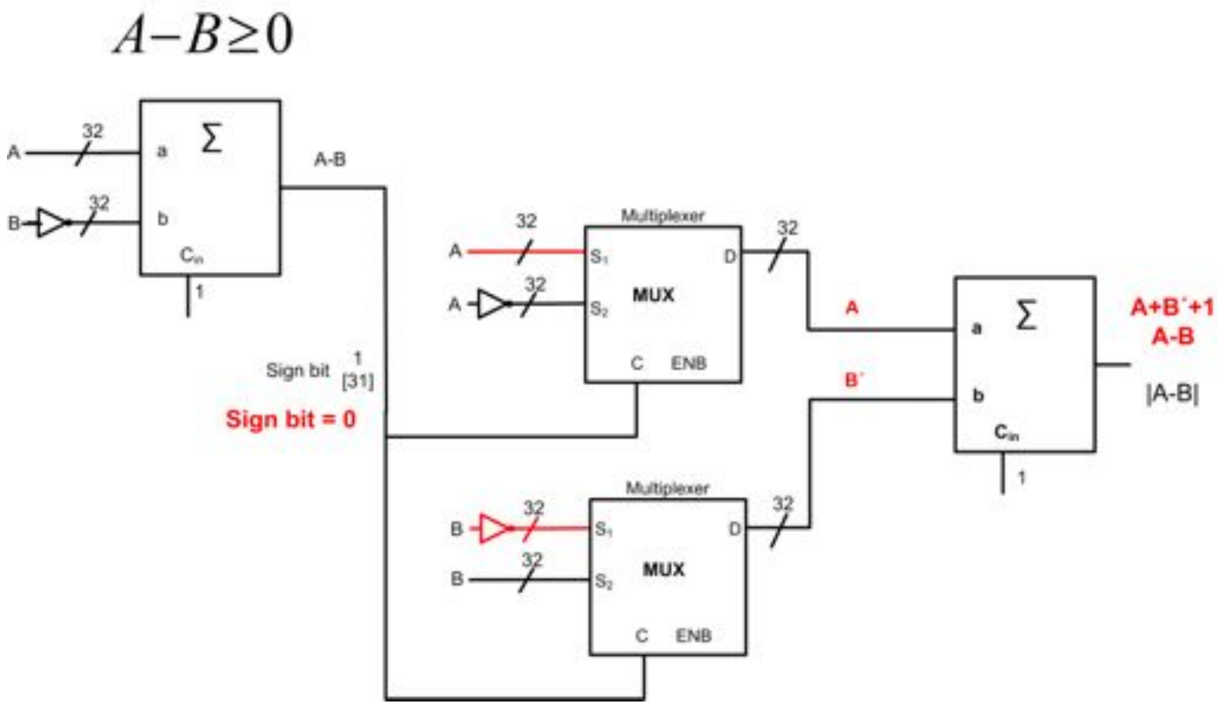


Figure 2.10. Design 1, Absolute Value Calculation Unit, Circuit Operation $A \geq B$

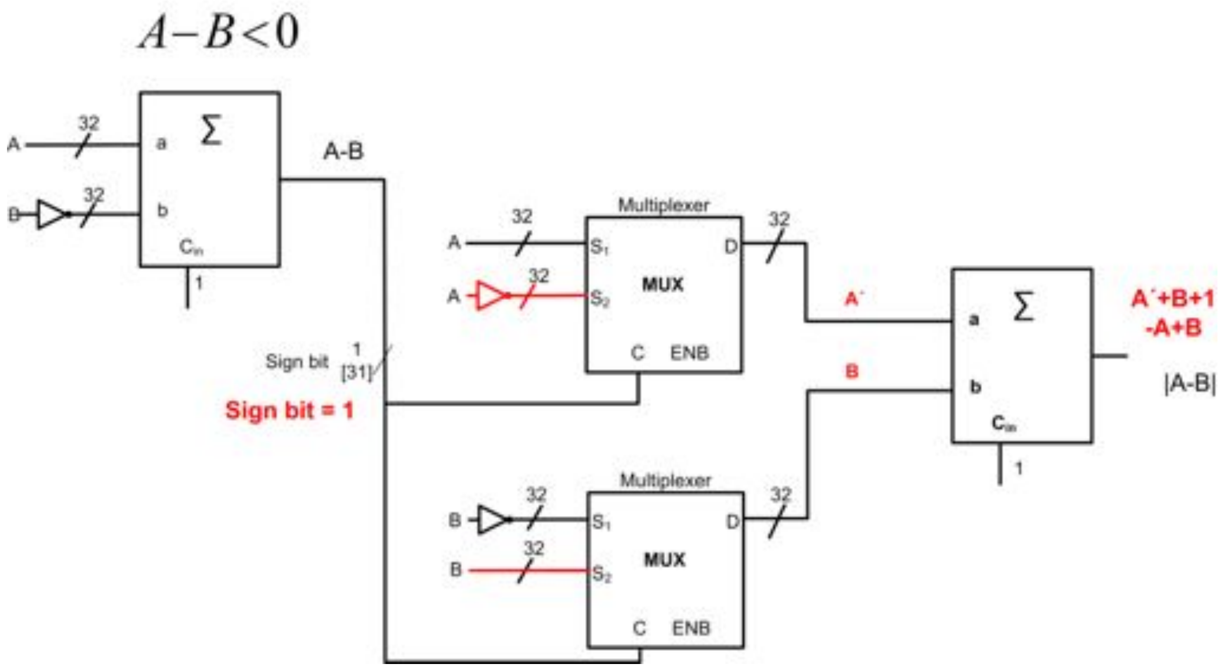


Figure 2.11. Design 1, Absolute Value Calculation Unit, Circuit Operation $A < B$

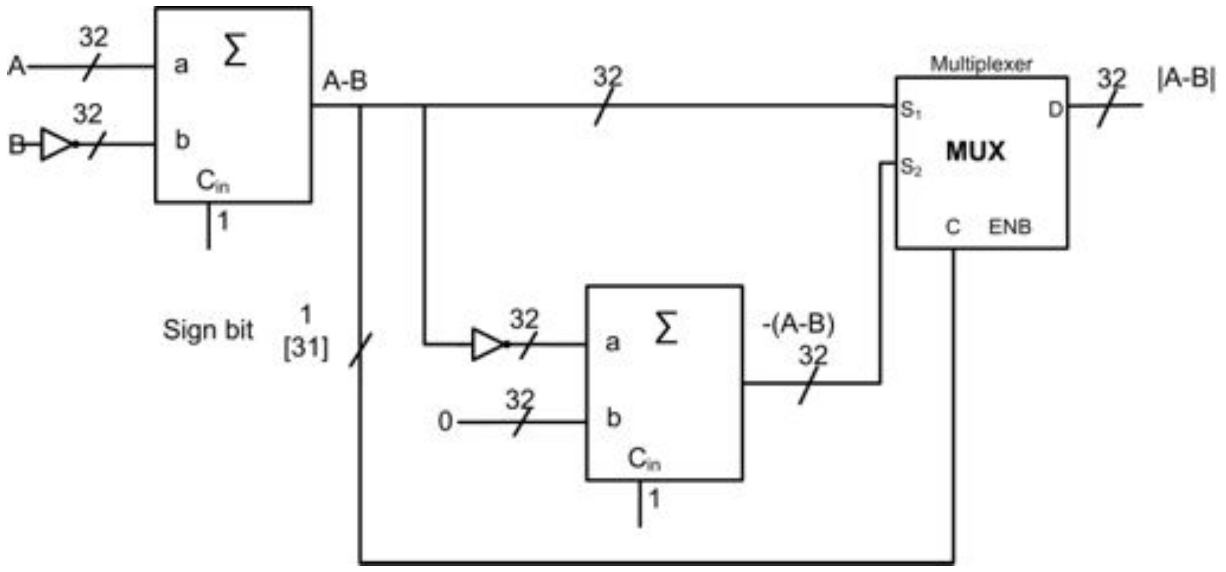


Figure 2.12. Design 2, Absolute Value Calculation Unit

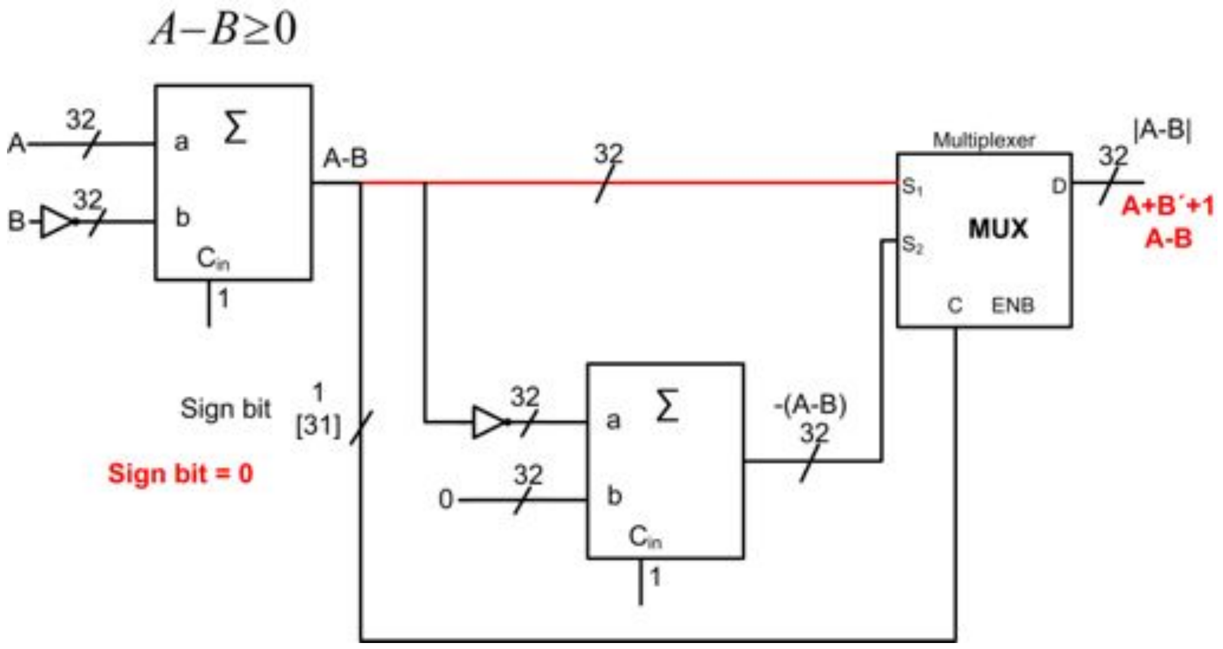


Figure 2.13. Design 2, Absolute Value Calculation Unit, Circuit Operation $A \geq B$

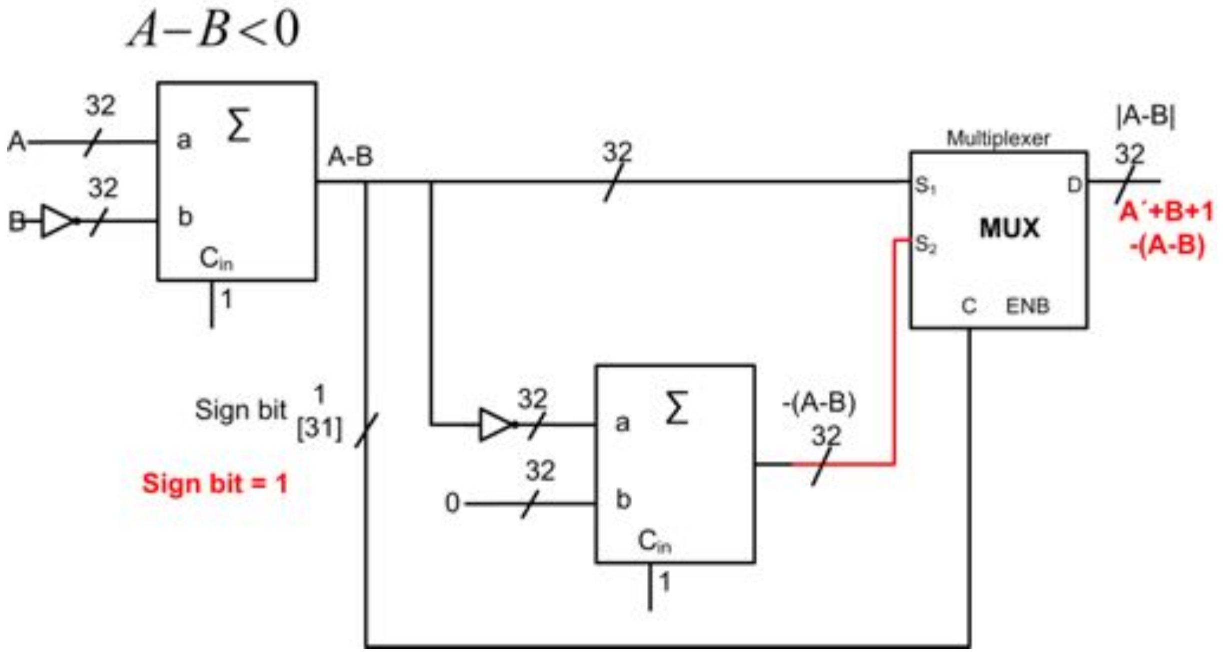


Figure 2.14. Design 2, Absolute Value Calculation Unit,t, Circuit Operation $A < B$

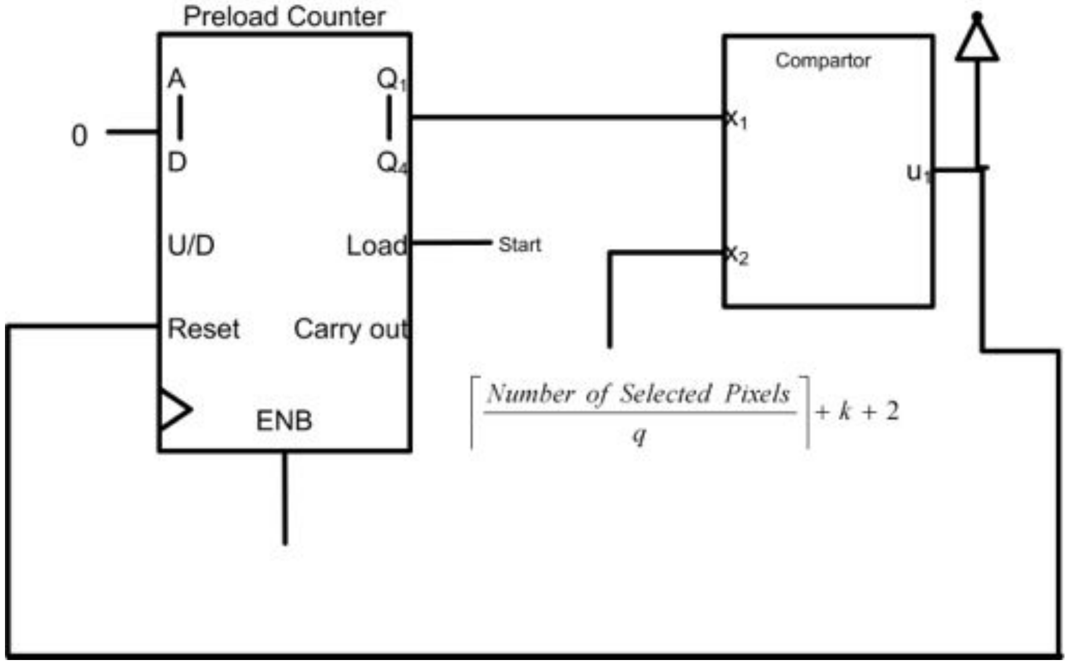

Preload Counter
A
D
0
Q1
Q4
U/D
Load
Start
Reset
Carry out
ENB
Compartor
X1
X2
u1
$\left\lceil \frac{Number\ of\ Selected\ Pixels}{q} \right\rceil + k + 2$

Figure 2.15. Control Logic

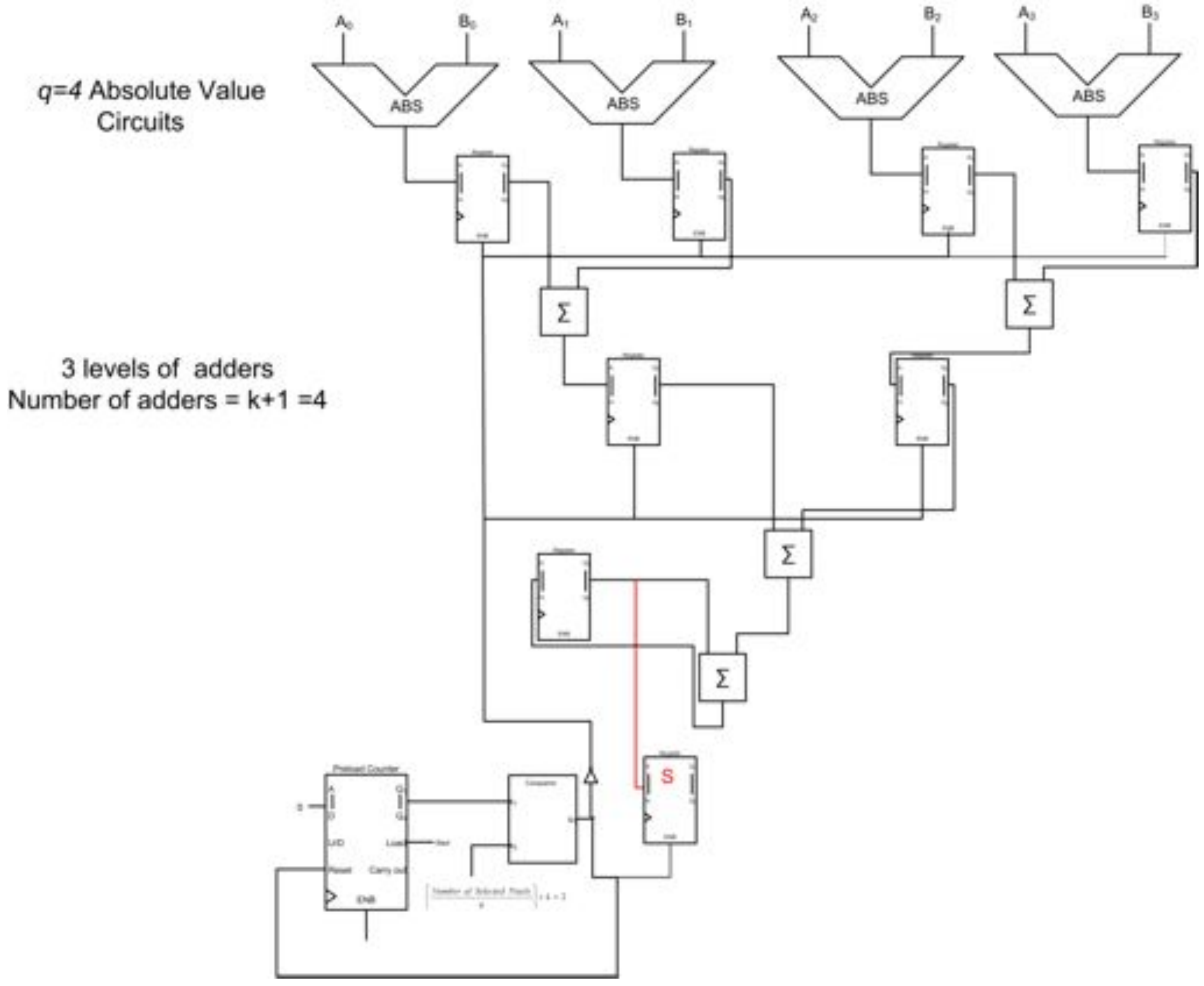


Figure 2.16. RTL Circuit with 4 ABS units and control logic

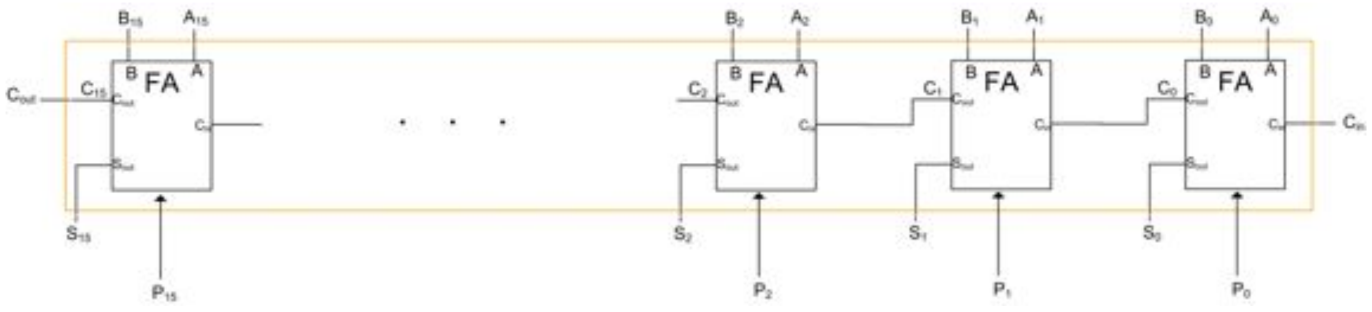


Figure 2.17. n-bit adder design using full adders

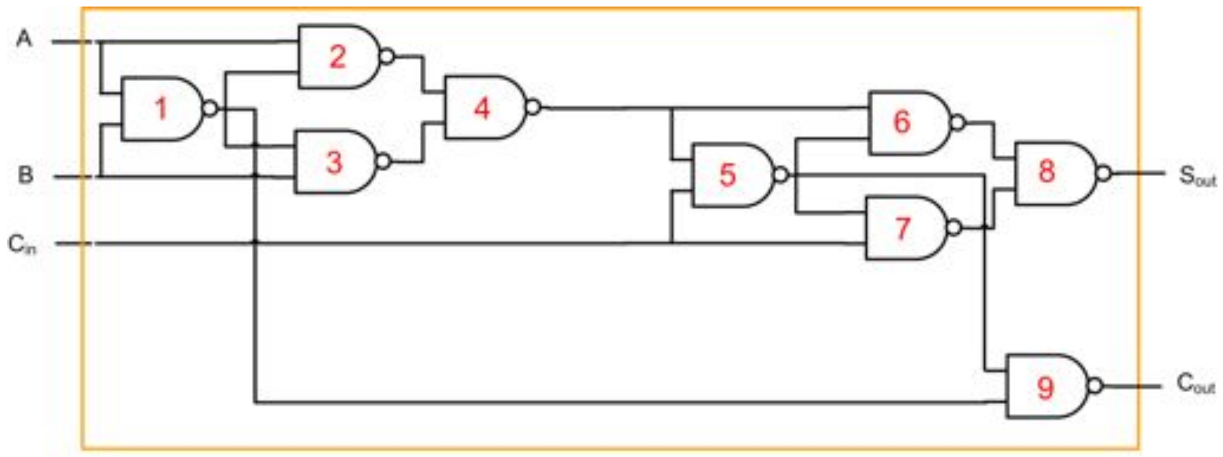


Figure 2.18. Full adder design using NAND gates

Chapter 3

Radiation Effects

For this Ariadna project, using probablistic computing to increase the efficiency of stereo-vision related calculations is being investigated. With a BIVOS design that will be implemented in environments that are high in radiation, it is worth investigating the effects of radiation on any new hardware design. Single Effect Events (SEE) are soft or hard errors that are introduced when radiation imparts some of it's energy into semiconducter components that make up the computer hardware.

For this study, a preliminary investigation into the influence of radiation particles on the calculations performed with the proposed variable voltage IC design has been done. As voltage levels decrease, a circuit becomes more susceptible to high energy particle strikes. When the particle strikes, an electron-hole pair is generated that causes a transient pulse that may alter the logical state of the circuit node [31]. In digital circuits, this temporary disruption of a circuit node is known as a single event transient (SET). In environments such as the Moon, Mars or the moons of the gas giants, robots utilizing stereo vision will be exposed to much higher levels of radiation than on Earth [24].

Radiation can have other effects such as long-term induced parasitic transistor states of low voltage and high current and permanent ”hard” failures [4]. The former depends upon the device and circuit layout while the latter is due to very high and prolonged current levels. While both are important problems that need to be addressed in general radiation hardening of circuits, the most crucial for this

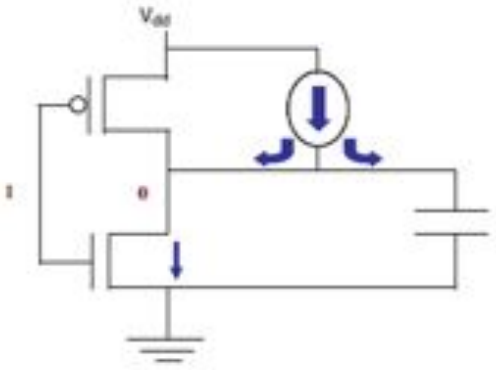


Figure 3.1. Spice level simulation of charge deposition modeled by a transient current source [31].

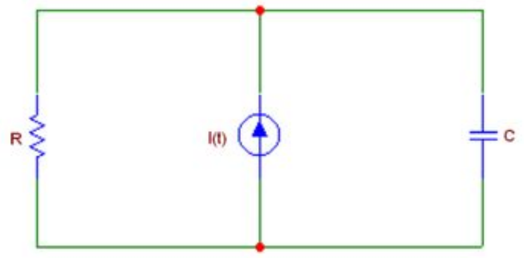


Figure 3.2. Equivalent circuit for circuit response to an energetic particle [31].

study is the SET where the effect of temporary altering of a logical state of a circuit node can propagate throughout the calculations after the event has occurred and are more likely to happen at lower voltage levels like the ones proposed in [8].

As shown in [2], the choice of gates when designing adder and multiplier integrated circuits will affect the susceptibility of the circuit to incorrect inputs. By testing various designs with input error models that represent the various causes of circuit node alteration, the optimal design can be chosen. At a circuit level, SETs can be generated using a Spice-level simulation that uses a transient current source as seen in Figure 3.1.

Due to the short timescale of the project, a full transient simulation at the device or circuit level is too time consuming. As shown in [31] it is possible to develop a simple analytical model that can be used to evaluate a circuit's sensitivity to SETs. An equivalent circuit can be created that represents a circuit response to an energetic particle hit as shown in Figure 3.2.

By modifying the properties of this current source a variety of energetic particle strikes can be simulated and made to follow a probabilistic model representing the planetary surface in question. In [31], these analytical results are compared to Spice level simulations showing good agreement between the two approaches while the analytical method had a speedup of more than 1000 times that of the Spice simulation in runtime. The analytical method is also easier to integrate into the proposed Matlab simulation. The following sections describe the analytical model that represents the energetic particle strikes and their propagation through logic stages.

3.1 NAND gate

Using a NAND gate any other logic gate can be built. Because of this property, the NAND gate was chosen to simulate energetic particle strikes which can then be used to build an adder used in the stereo-vision calculations. The basic logic of a 2 input NAND gate is shown in Table 3.1.

Table 3.1: NAND Gate Behaviour

Input 1	**Input 2**	$Output$
0	1	1
1	0	1
0	0	1
1	1	0

This is accomplished by connecting transistors together to achieve the desired effect. For CMOS, the design is composed of two NMOS transistors connected to the separate inputs at the gate node and to each other in series. Two PMOS transistors are then connected in parallel. The overall design is shown in Figure 3.3.

The top two transistors are PMOS transistors, the bottom two are NMOS transistors. **G** stands for the gate node, **S** stands for the source node, and **D** stands for the drain node. The idea is, if both inputs are high then both NMOS are conductive and the PMOS are not which will connect the output to ground. If either input is low, then the output is connected to V_{dd}. It is useful to think of the top half as the *PMOS block* and the bottom half as the *NMOS* block. If both inputs are high, the lower block is ON, and the upper block is OFF. If either input is low, the lower block is OFF and the upper block is ON.

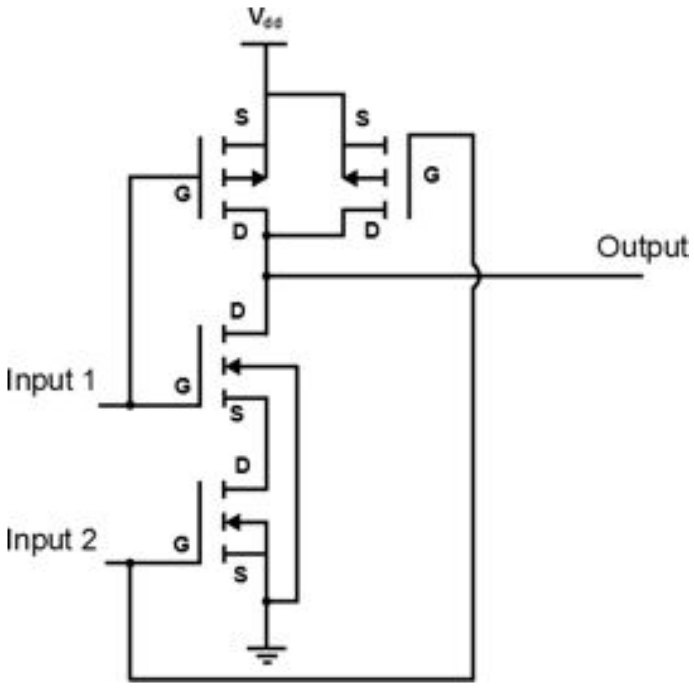


Figure 3.3. CMOS NAND circuit

3.2 SETs effect on transistors

For the Ariadna project, the goal is to determine the effect of having a particle that is energetic penetrate one of the transistors as depicted in Figure 3.4. If it imparts enough energy to cause a change in voltage the output could change from what is expected from the NAND gate given the inputs. This input to the circuit is modeled as a current source so it can be solved analytically. The biggest effect comes when a particle hits a drain of a transistor connected to the output node, so this was the focus for this project. The same techniques can be extended to cases where the particle impacts other nodes as well. In the case of a NAND gate, this means the drains of both PMOS transistors and the drain of the NMOS that are connected to the output will be considered potential impact points.

Linear Energy Transfer (LET) is a common measure of energy transferred by a radiation particle when it strikes a material. This is used to calculate the charge, Q, and collection time constant, τ_α. The relation between the track length, d, the charge, Q, and LET

$$LET = \frac{Energy(MeV)}{density(mg/cm^3) * d(cm)}. \tag{3.1}$$

For instance the amount of energy needed to liberate one electron-hole pair in silicon is 3.6 eV and

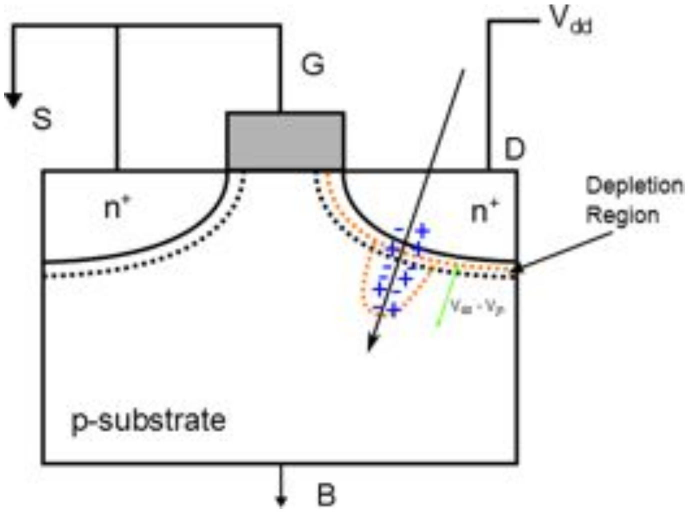


Figure 3.4. Ion tracks through a transistor drain node.

the material density is 2.42 g/cm^3 we can calculate the charge deposited by a unit LET (for a track length of 1 μm) by first calculating the linear energy transfer:

$$\begin{aligned} LET &= \frac{3.6 \times 10^{-6}}{2.42 \times 10^3 \times 10^-4 \times 10^{-4}} \\ &= 1.49 \times 10^{-5} MeV - cm^2/mg, \end{aligned} \tag{3.2}$$

and taking into account the charge of 1 electron the charge deposited per unit LET is calculated as:

$$\begin{aligned} \text{Charge per unit LET} &= \frac{\text{Charge of 1 electron}}{LET} \\ &= \frac{1.6 \times 10^{-7}}{1.49 \times 10^{-5}} \approx 0.01 pC/LET. \end{aligned} \tag{3.3}$$

So in terms of the track length and LET for any given particle, we have:

$$Q(pC) = 0.01 \times LET(MeV - cm^2/mg) \times d(\mu m). \tag{3.4}$$

The current, $I(t)$ is then a function of τ_α, Q, and τ_β, the ion track establishment constant (often $\tau_{beta} \ll \tau_{alpha}$), and time, t. The function is described as:

$$I(t) = \frac{Q}{\tau_\alpha - \tau_\beta} \left(e^{\frac{-t}{\tau_\alpha}} - e^{\frac{-t}{\tau_\beta}} \right) \tag{3.5}$$

This current can be modelled as a network model, where C is the effective capacitance loading lumped on the output node and R is the effective resistance as shown in Figure 3.2.

In the case of the *PMOS block* being *ON*, *R* is the pull up resistance, whereas if the *NMOS block* is *ON* it is the pull-down resistance. The equation describing $V(t)$ at the struck node is:

$$I_p(t) = C\frac{dV(t)}{dt} + \frac{V(t)}{R}, \tag{3.6}$$

the solution for $\tau_\beta \ll \tau_\alpha$ is:

$$V(t) = \frac{I_o \tau_\alpha R}{\tau_\alpha - RC}\left(e^{\frac{-t}{\tau_\alpha}} - e^{\frac{-t}{\tau_\beta}}\right), \tag{3.7}$$

where I_o is the max charge collection current.

The t_{peak} when V is at maximum is then evaluated as:

$$t_{peak} = \frac{\ln\left(\frac{\tau_\alpha}{RC}\right)\tau_\alpha RC}{\tau_\alpha - RC}, \tag{3.8}$$

inserting this back into (3.7) we obtain V_{peak} as:

$$V_{peak} = \frac{I_o \tau_\alpha R}{\tau_\alpha - RC}\left(\left(\frac{\tau_\alpha}{RC}\right)^{\frac{RC}{RC-\tau_\alpha}} - \left(\frac{\tau_\alpha}{RC}\right)^{\frac{RC}{RC-\tau_\alpha}}\right). \tag{3.9}$$

Now, the minimum charge to cause V_{peak} is Q_c, which is simply $I_o\tau_\alpha$, so rearranging we have:

$$Q_c = \frac{V_{peak}\left(\tau_\alpha - RC\right)}{R\left(\left(\frac{\tau_\alpha}{RC}\right)^{\frac{RC}{RC-\tau_\alpha}} - \left(\frac{\tau_\alpha}{RC}\right)^{\frac{RC}{RC-\tau_\alpha}}\right)}. \tag{3.10}$$

If τ_α is much smaller than RC, $Q_c \rightarrow V_{peak}C$. If τ_α is much greater than RC, $Q_c \rightarrow \frac{V_{peak}}{R}\tau_\alpha$.

The transient pulse duration, T_D, is:

$$T_D = t_{peak} - RC\ln\left(\frac{\frac{V_{dd}}{2}}{V_{peak}}\right) - \tau_\alpha \ln\left(\frac{\frac{V_{dd}}{2}}{V_{peak}}\right) \tag{3.11}$$

which is reduced to summing t_{peak} to the second or third term on the right hand side depending on whether $RC \gg \tau_\alpha$ or vice versa.

The effective capacitance, C, is the sum of all parasitic capacitances lumped at the node including interconnect, drain junction and capacitive loading due to the logic gates driven by the struct node. The effective resistance, R, is the mean value of the channel dc resistance for gate voltage = V_{dd} and drain source exponentially decaying to zero with the source connected to ground. For p-type it is the mean channel dc resistance for the gate terminal connected to ground, source drain voltage exponetially decaying from V_{dd} to zero, and the source terminal connected to V_{dd}. Once the effective resistance is calculated, it is then multiplied by the W/L ratio of the actual transistors being used. It is worth noting that this is related to the sheet resistances, ρ (commonly quoted for transistor technology), by:

$$R_{eff} \cdot \frac{W}{L} = \frac{\rho}{d}, \tag{3.12}$$

where d is the thickness of the transistor. Using this relation the value for R_{eff} can be determined given the quoted specifications of the transistor being used.

3.3 SET propagation

Now that we can calculate the amplitude of the transient pulse and it's duration, there is the question of how this pulse will propagate from gate to gate. Within an adder there are several stages of logic gates, but if the pulse does not run it's course through the entire adder design, the effects will be temporary and the output will not change. Only if the transient pulse is able to reach the output of the last stage of the adder is an error registered in the output that is stored in memory.

τ_η is the transient pulse duration at the η^{th} stage. For the first stage, τ_1, it is simply equal to T_D. The response of the gate to an input transient is divided into four regions:

The first region is when the pulse is filtered out. The model is:

$$\text{if}\left(\tau_\eta < k \cdot t_p\right), \quad \tau_{\eta+1} = 0, \tag{3.13}$$

where k is tunable for the technology being used and is equal to the minimum ratio $\frac{\tau_\eta}{t_p}$ for a SET to be propagated. The same k is used for all logic gates of that technology node regardless of inputs or fan out.

The second and third regions model the situation when a pulse is propagated but degraded in amplitude and duration:

$$\begin{aligned} \text{if}\left(k \cdot t_p < \tau_\eta < (k+1) \cdot t_p\right), \quad & \tau_{\eta+1} = (k+1)\, t_p \left(1 - e^{(2.4-(\tau_\eta/t_p))}\right), \\ \text{if}\left((k+1) \cdot t_p < \tau_\eta < (k+3) \cdot t_p\right), \quad & \tau_{\eta+1} = \frac{\left(\tau_\eta^2 - t_p^2\right)}{\tau_\eta}, \end{aligned} \tag{3.14}$$

Finally there is the fourth region, where the pulse does not degrade:

$$\text{if}\left((k+3) \cdot t_p < \tau_\eta\right), \quad \tau_{\eta+1} = \tau_\eta, \tag{3.15}$$

3.4 Application to Probabilistic Computing

Scaling up, we want a system in place for the BIVOS adders being used in this project. Each adder is made up of stages using NAND gates. Given the input value to the adder and the value for V_{dd} that is being supplied, it should be possible to determine what the effect of radiation hitting a drain node of one of the adders will be. The goal is to determine what the output of the adder will be if various types of particles strike the drain junctions of the many transistors within the adder. In the end this will determine what noise will be introduced into the calculations done for the stereo-vision algorithm.

At each stage the change in the output is determined, and then moved on to the next stage as long as $\tau_\eta > 0$. If $\tau_\eta = 0$, then the calculation stops with no change to the adder's output.

To test a radiation spectrum (ie. different τ_α for different particles) we can try different energy levels and based on the Heinrich curves for different environments determine the likely flux for various energetic particles.

To carry out a full simulation, a flux of probabilistically generated particles and incidence angles must be generated and their transient effects are calculated with the software created for this project. The effect on the results of the calculations can then be seen if this software is running in conjunction with a full timing analysis of the circuit being used. By using the pulse durations that are already calculated in the radiation model, this can be run with a timing analysis of the circuit.

3.5 Summary

To summarize, the main concern for temporary calculation errors comes from the SEE, specifically SETs. These transient pulses act as a current source for a short amount of time, and propagate through integrated circuits such as adders used in the stereo-vision calculations created for this project. Through the preliminary investigation done for this project, the characteristics of this transient pulse are calculated at the gate level, and then are scaled up to devices such as adders by determining if the pulse is degraded by the time it reaches the output of the adder. The introduced model will allow us to determine the effect of such SEE on stereo-vision calculations.

Future work outside the scope of this project will involve a full timing analysis that integrates the current pulses generated by radiation together with the adders used in this project in a real-time simulation that executes the stereo-vision algorithm. In addition, by using modeled or measured spectrum from environments such as Mars or the Moon, accurate generation of random particles that will strike the transistors can be done in simulation.

Chapter 4

Simulation Model

The simulation model for the circuit is developed using MATLAB. The reasons for using MATLAB as the simulation method is that the stereo vision code was already written in MATLAB, the shorter development time considering the limited timeline of the project and the need to create a model for hardware that is not manufactured yet and no model for it is available from off-the-shelf circuit simulation software.

Two MATLAB models for the probabilistic computing circuit have been created.

- Model 1: The circuit has two data flow paths, one for always correct results and one for probabilistic results. The circuit model has the following probabilistic components:
  - Probabilistic Adder
  - Probabilistic Inverter
  - Probabilistic Absolute Value Component
- Model 2: The circuit model has the following probabilistic components:
  - Probabilistic NAND
  - Probabilistic Full Adder composed of NANDs.
  - Probabilistic 16 bit Adder
  - Probabilistic Inverter
  - Probabilistic Absolute Value Component

  The model also has the following radiation components:

- Radiation Full Adder composed of NANDs
- Radiation Model for 16 bit Adder

When transforming the radiation model into the logical model that can implemented in MATLAB, we have to take a look at the CMOS design of the CMOS gate shown in Figure 4.1. If the radiation hits any of the transistors then it will flip pull the transistor from high to low or low to high. The transistors in the CMOS NAND gate act similar to an on-off switch. The P-MOS transistor acts as a normally closed switch and the N-MOS acts as a normally open switch. Table 4.1 shows a truth table for the effect of radiation on the output of the gate. This depends on which transistor is hit by radiation and the combination of inputs to the NAND gate.

Extending the concept to the full adder, we substitute the NAND gates in the full adder design shown in Figure 2.18 with the probabilistic NAND gates that include the radiation model.

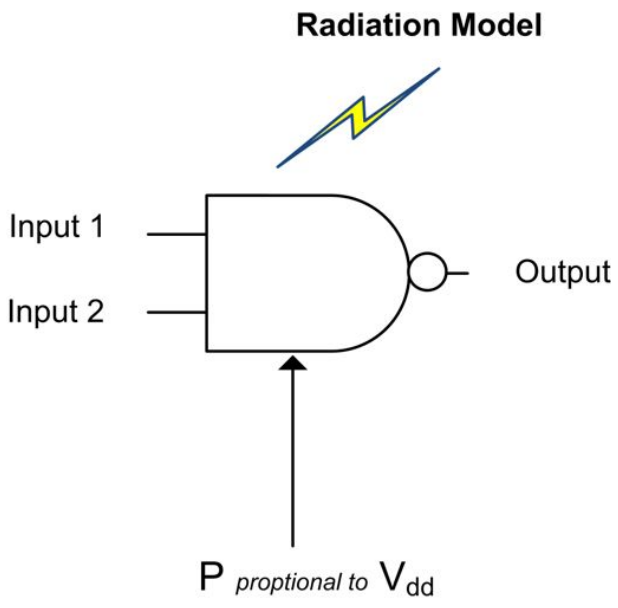


Figure 4.1. Probabilistic NAND, with Radiation Effect

We illustrate the effects of radiation with a few examples. Figure 4.3 shows the effect of the radiation hitting transistor T_1 of gate 6 in the adder and with a propagation lasting four stages. If the inputs to gates 6 and 7 are as shown, then radiation flips the output of gate 6 from 1 to 0. Because of that the inputs of gate 8 are switched from 1,1 to 0,1 the output is flipped from 0 to 1. Since the transit pulse last for four propagation stages, it lasts long enough to keep have the output S_{out} flipped until it reached the register. Figure 4.4 shows an example of the radiation hitting transistor T_0 of gate 2 in the adder

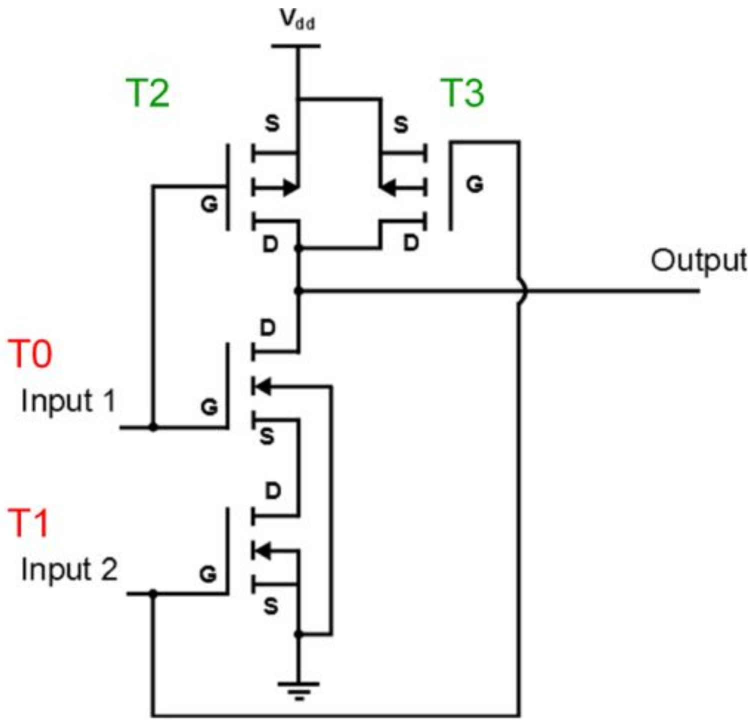
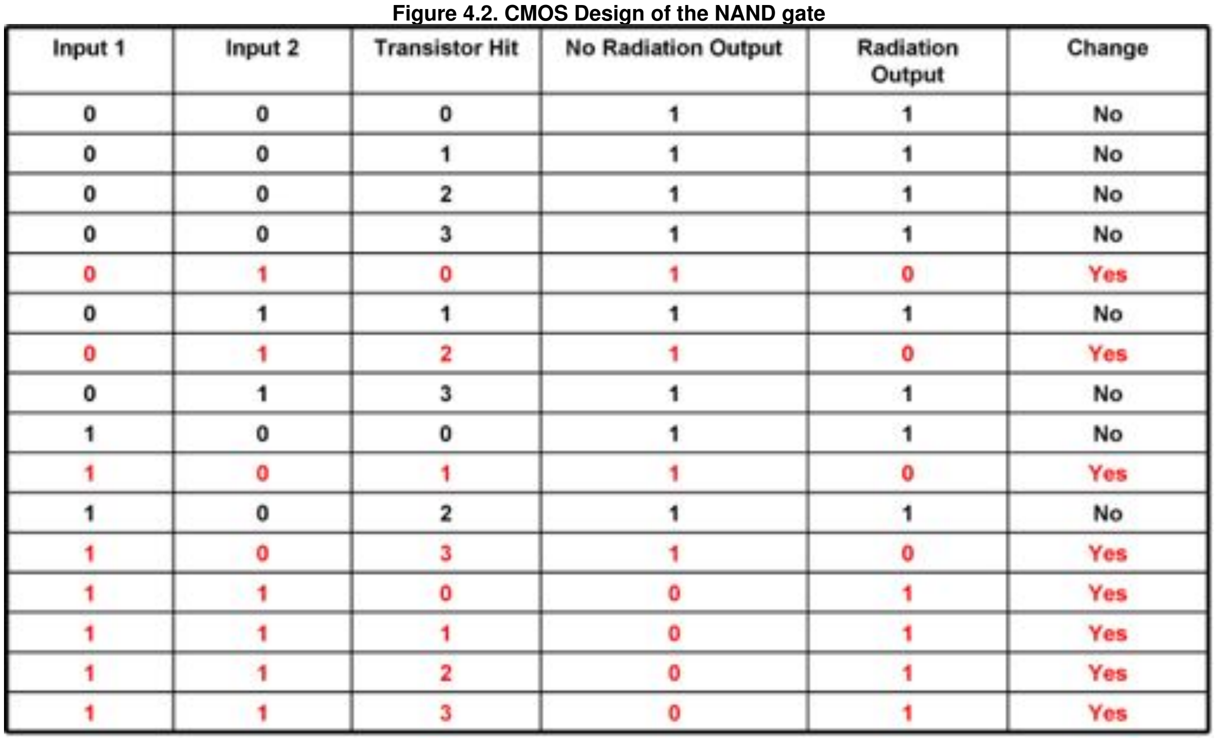


Figure 4.2. CMOS Design of the NAND gate



Input 1	Input 2	Transistor Hit	No Radiation Output	Radiation Output	Change
0	0	0	1	1	No
0	0	1	1	1	No
0	0	2	1	1	No
0	0	3	1	1	No
0	1	0	1	0	Yes
0	1	1	1	1	No
0	1	2	1	0	Yes
0	1	3	1	1	No
1	0	0	1	1	No
1	0	1	1	0	Yes
1	0	2	1	1	No
1	0	3	1	0	Yes
1	1	0	0	1	Yes
1	1	1	0	1	Yes
1	1	2	0	1	Yes
1	1	3	0	1	Yes

Table 4.1. Truth Table for the NAND gate hit by radiation

and with a propagation lasting two stages. There are four stages from gate 2 to both S_{out} and C_{out}. Therefore the radiation will not cause any change in the final result of the full adder inputs.

By using the modular probabilistic and radiation model full adder, we can construct the probabilistic radiation 16-bit adder shown in Figure 4.5 using the same design we had for the normal adder. We can

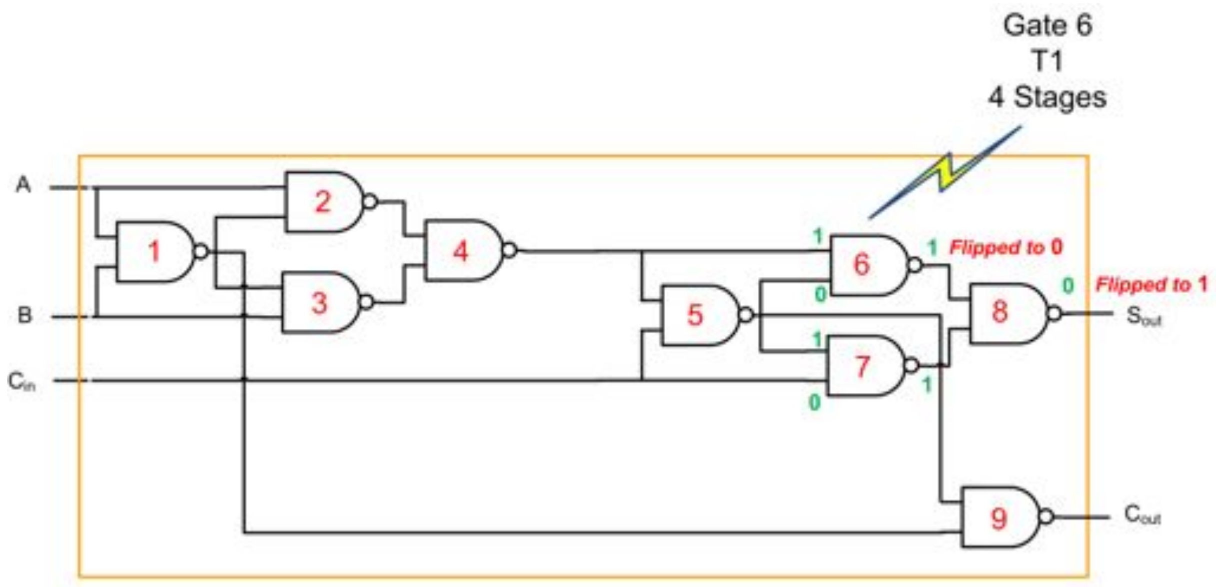


Figure 4.3. Full adder hit by radiation, example 1

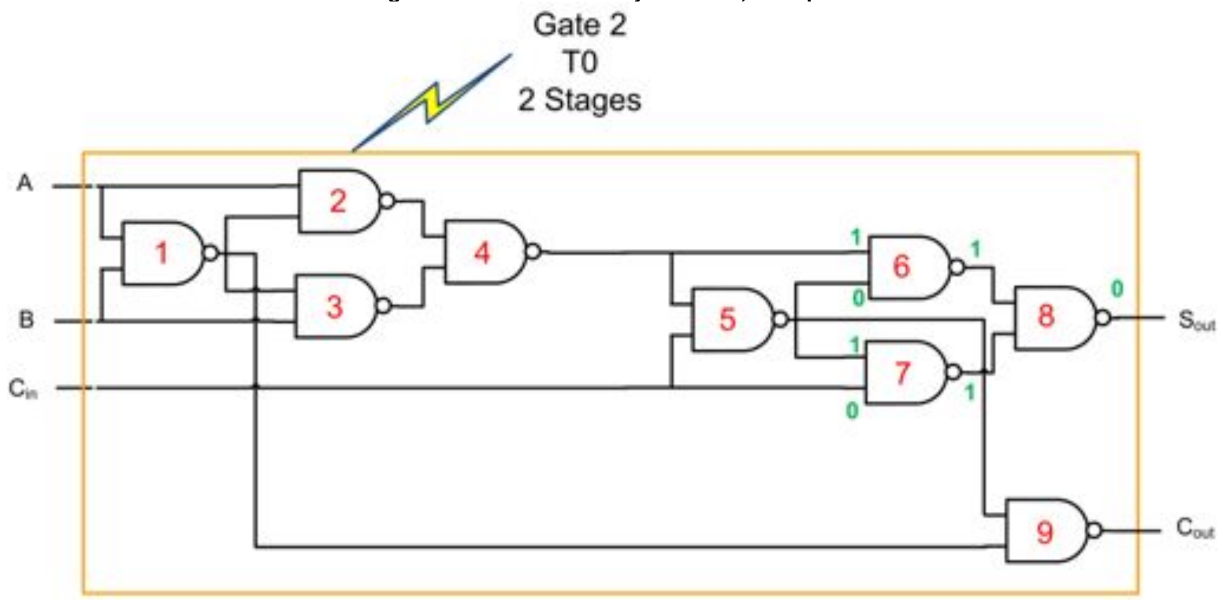


Figure 4.4. Full adder hit by radiation, example 2

extend the modules to use the 16-bit adder to construct the absolute value circuit shown in Figure 4.6. Then we can use the adders and the absolute value circuit components to construct the full probability radiation circuit for the disparity calculation using the same design we used in Figure 2.16.

Deterministic operation in the control logic is required for correct output of the circuit as any errors in the control logic can be very costly and might stop the whole operation of the circuit and propagation from combinational components to registers. Therefore the probabilistic and radiation effect models are not added to the components of the control logic.

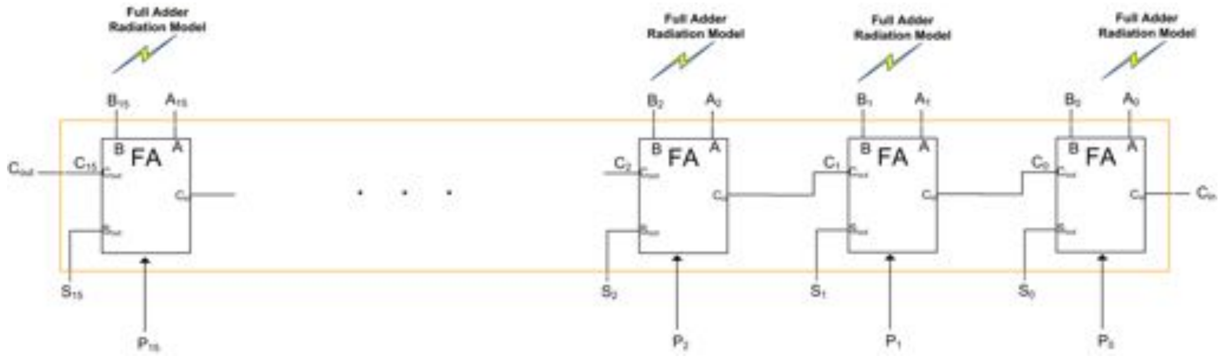


Figure 4.5. 16Bit Adder with radiation model

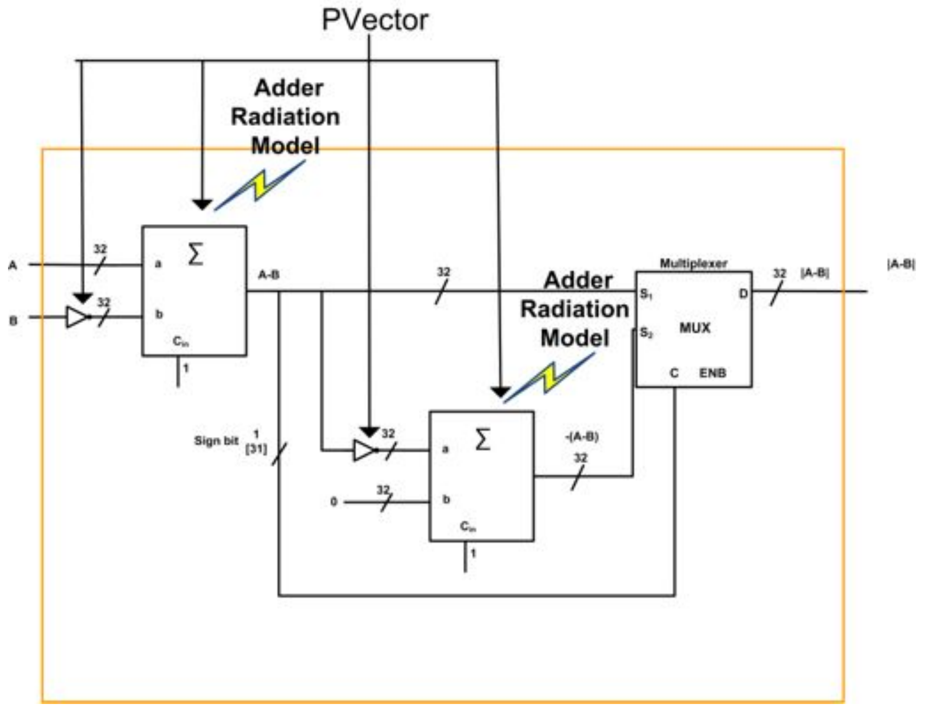


Figure 4.6. Absolute value circuit with radiation model

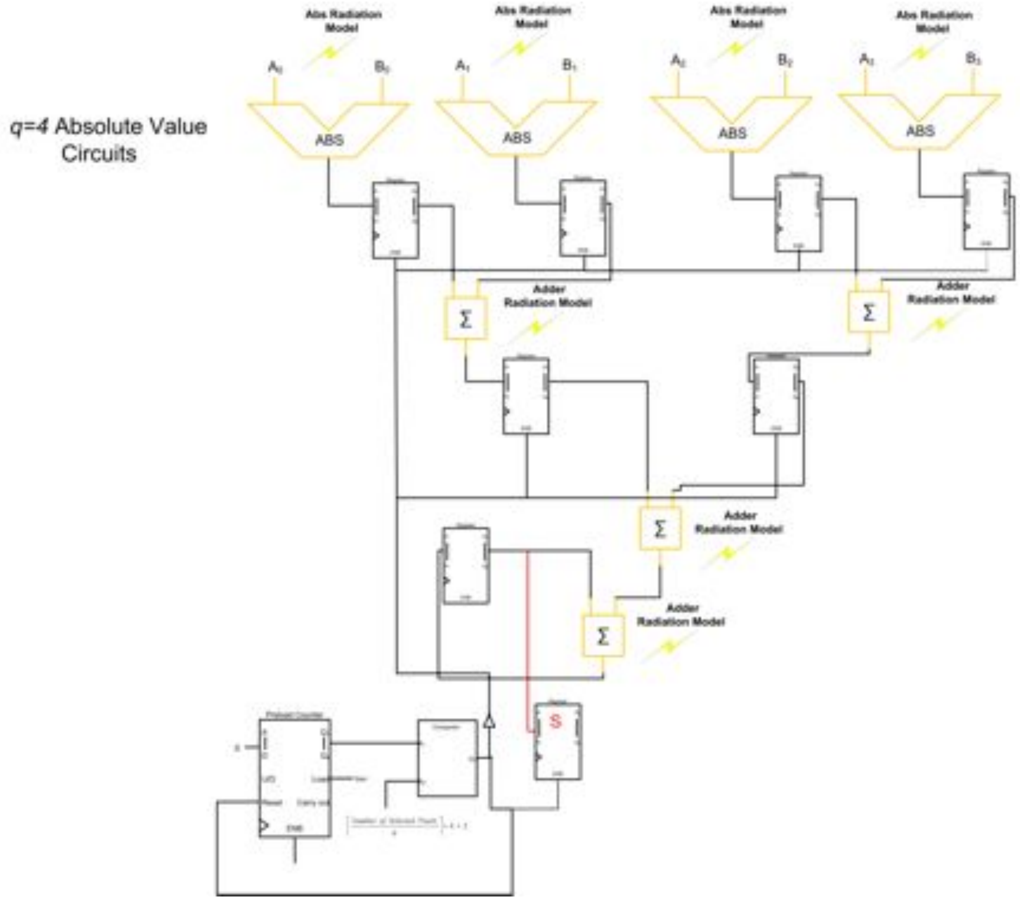


Figure 4.7. Full Circuit with disparity calculation

4.1 Supply Voltage Probability Function

The supply voltage distribution function as introduced by [14] is the function that relates the probability of correctness of each bit to the supply voltage. It is assumed that the supply voltage is directly proportional to the probability of correctness of the bit. [14] introduces two functions, a linear supply probabilistic voltage distribution function and a geometric probabilistic supply voltage distribution function. The linear supply voltage is shown in Equation (4.1) and the geometric equation is shown in Equation (4.3).

$$p_i = \begin{cases} p_0 & i = 0 \\ p_{i-1} + \delta & i \neq 0 \end{cases} \tag{4.1}$$

Where $delta$ is given by Equation (4.2)

$$\delta = \frac{V_{dd(L)} - V_{dd(0)}}{L} \tag{4.2}$$

where $L = 1, 2, ..., L$

$$p_i = \begin{cases} p_0 & i = 0 \\ p_{i-1} + ar^{i-1} & i \neq 0 \end{cases} \tag{4.3}$$

where p_0 : probability of correctness of output bit 0

p_i : probability of correctness of output bit i

i : bit position

a : scale constant

r : ratio constant

It is not clear from George et al. [14] what are optimal range of values for for the parameters p_0, a, r or how to obtain that range. After investigation of linear, quadratic, and geometric functions, we decided to use a bounded geometric function. The bounded geometric function is given by Equation (4.4):

$$p_i = \begin{cases} p_0 & i = 0 \\ p_{i-1} + ar^{i-1} & i \neq 0 \end{cases} \quad if\ p_i > 1\ then \quad p_i = 1 \tag{4.4}$$

, with the bounded geometric function ensuring that the probability value of any bit is no greater than one. We can also further modify the bounded geometric function so that we apply it only to a limited number of bits, starting with the least significant bit. If the number of desired lower significant bits is n then the bounded geometric function will be given by Equation (4.5)

$$p_i = \begin{cases} p_0 & i = 0 \\ p_{i-1} + ar^{i-1} & i \neq 0 \\ 1 & i = m \end{cases} \quad if\ p_i > 1\ then \quad p_i = 1 \tag{4.5}$$

In order to find the best fit values for p_0, a and r we need to maximize the probability of correctness and hence maximize the product in Equation (4.6). On the other hand we have to minimize the energy and therefore we have to minimize the summation in Equation (4.7).

$$\prod_{i=0}^{n} \left(p_0 + a \cdot \sum_{j=0}^{i-1} r^j \right) \tag{4.6}$$

$$\sum_{i=0}^{n} \left(p_0 + a \cdot \sum_{j=0}^{i-1} r^j \right) \tag{4.7}$$

As we can see from Figure 4.8, increasing a leads to making the energy-probability curve more convex, but also as we change a we start having a smaller range of the 6 bits with voltage level less than one. The color scale shows increasing values of a from red to blue. Figure 4.9 shows that increasing r leads to making the curve more convex, similar to a but to a greater level, but changing r higher than 3 causes almost all the bits to be equal to 1, therefore limiting energy savings and voltage scaling. The color scale shows increasing values of r from red to blue.

To see the effect of changing p_δ on the performance of the probabilistic voltage scaling algorithm we

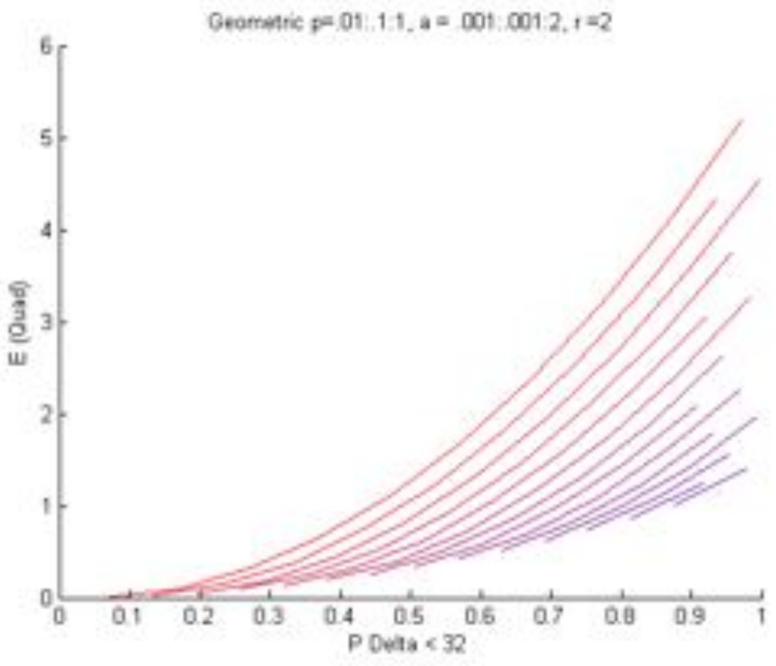


Figure 4.8. Effect of changing a

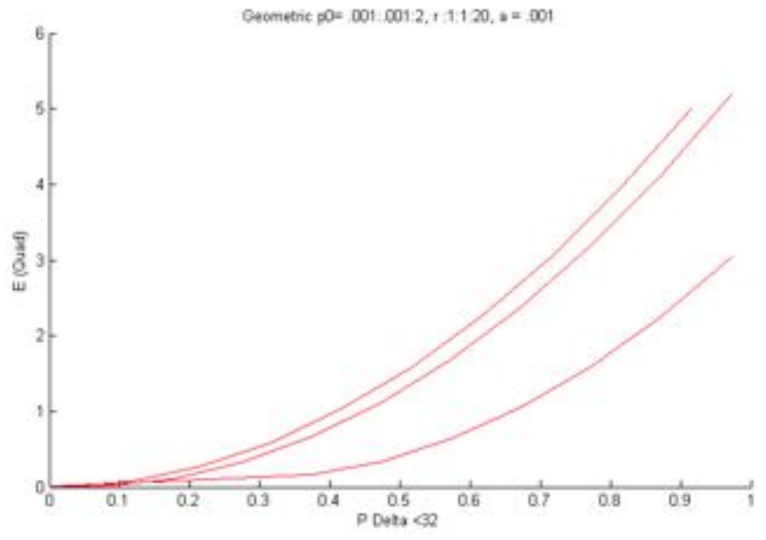


Figure 4.9. Effect of changing r

varied the variables p_0, a and r according to Table 4.2

Parameter	Range	Step
p_0	.01-1	.01
a	.001-.2	.001
r	1-20	1

Table 4.2. Parameter range for testing p_δ change effect

Figure 4.10 shows that increasing p_δ leads to achieving a better performance for the bounded geometric function and the shape of the function changes from concave to convex. We can see the the

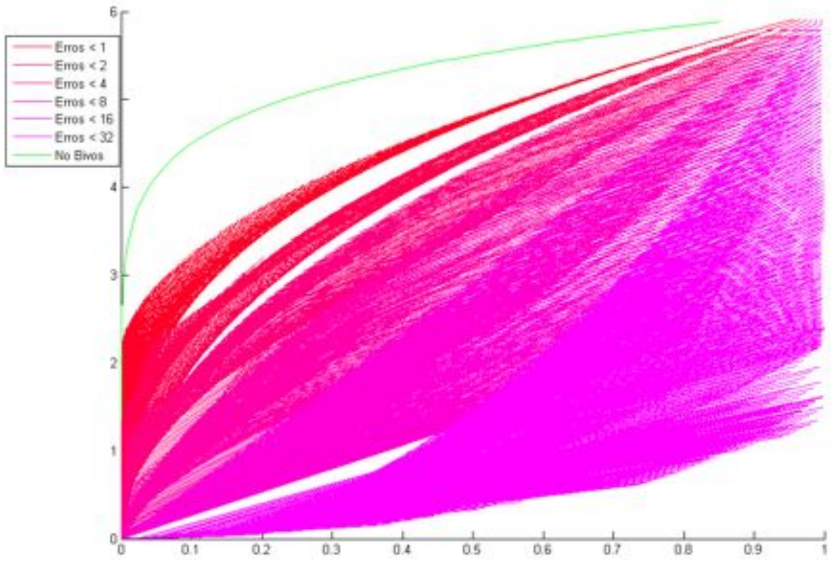


Figure 4.10. Effect of changing p_δ

magnitude of acceptable error p_δ or ”large error” as defined by George [14]. Therefore, judging the performance of the probabilistic voltage scaling algorithm can be deceiving as what is the magnitude of what it is acceptable as a ”large error” is dependent on the application for which probabilistic computing is used.

After running the tests for the range of values given in Table 4.2 we came with two sets of best set parameters that we can use for the bounded geometric function. The best fit parameter sets are given in Table 4.3 and Table 4.4. These are the parameter sets that we used for generating our simulation sets for the probabilistic computing circuit.

p_0	a	r
0.0100	0.0150	2.0000
0.1100	0.0140	2.0000
0.2100	0.0120	2.0000
0.3100	0.0110	2.0000
0.4100	0.0090	2.0000
0.5100	0.0070	2.0000
0.6100	0.0060	2.0000
0.7100	0.0040	2.0000
0.8100	0.0030	2.0000
0.9100	0.0010	2.0000

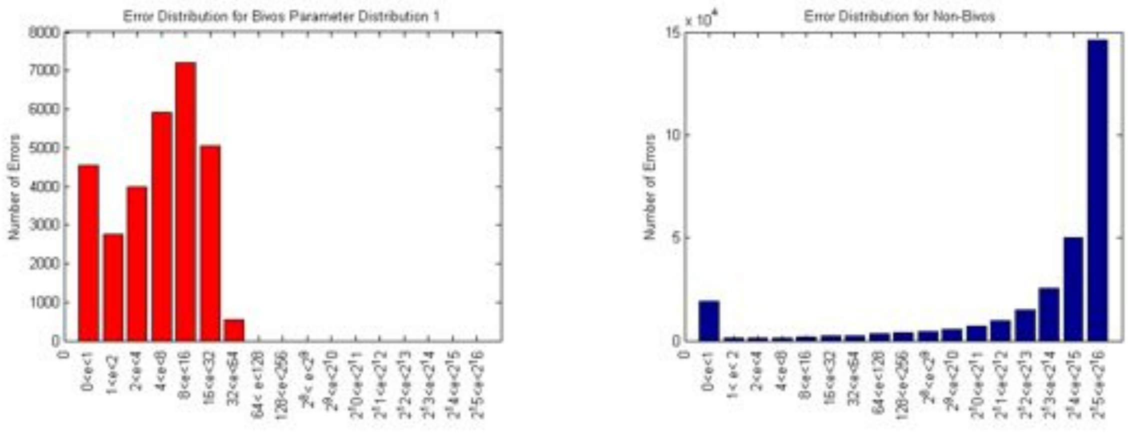
Table 4.3. Best fit parameter set 1

Running the error simulation for the best 10 parameter error distribution for the various components

p_0	a	r
0.0100	0.0020	3.0000
0.1100	0.0020	3.0000
0.2100	0.0020	3.0000
0.3100	0.0010	3.0000
0.4100	0.0010	3.0000
0.5100	0.0010	3.0000
0.6100	0.0010	3.0000

Table 4.4. Best fit parameter set 2

Figure 4.11. Error Distribution for Adder, Best 10 Parameter Combinations, 3000 of 8 additions pixels each



(a) Error Distribution for Bivos Adder

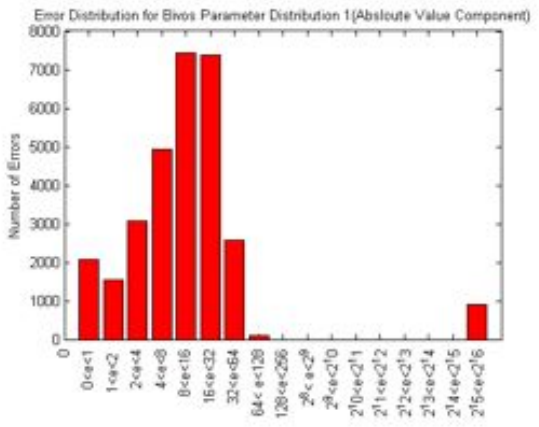
(b) Error Distribution for non Bivos Adder

of the circuit from a component level up to the full circuit. This implies that we first test the full adder, then the absolute value circuit, and then the full circuit. The error was determined on 5000 random disparity calculations of 8 pixels each. Figure 4.12(a) and Figure 4.12(b) show the error distribution for the adder, Figure 4.13(b) and Figure 4.12(b) show the error distribution for the absolute value circuit, Figure 4.14(a) and Figure 4.14(b) show the error distribution for the full circuit. The x-axis shows the range in which the errors occur. The range is divided into 16 sub ranges where each one of them is in the powers of 2 from $2^0 to 2^1 6$.

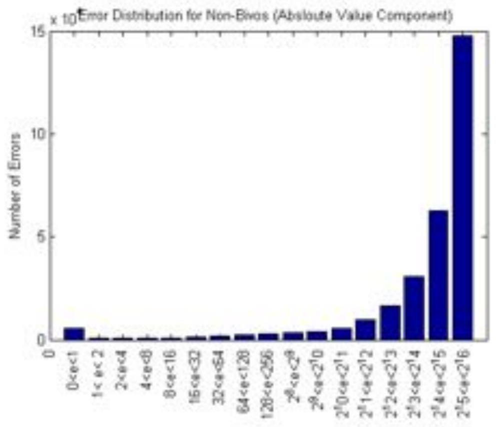
Figure 4.15(a) and Figure 4.15(b) show the error for a bitwise circuit at the bit level, assuming that the components do not have uniform error and the error varies by the bit. Figure 4.15(c) shows the error distribution for the same circuit with the radiation effects added at the bit wise level in the circuit combined with probabilistic computing.

Figure 4.16(a) and Figure 4.16(b) show the effect of radiation on the error distribution of the full circuit with a portability of error due to radiation set to $p = 1/384$. Figure 4.16(a) shows the effect only

Figure 4.12. Error Distribution for Absolute Value Component

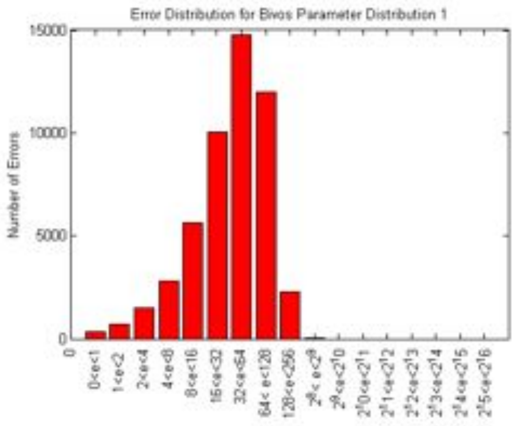


(a) Error Distribution for Bivos Absolute Value Component

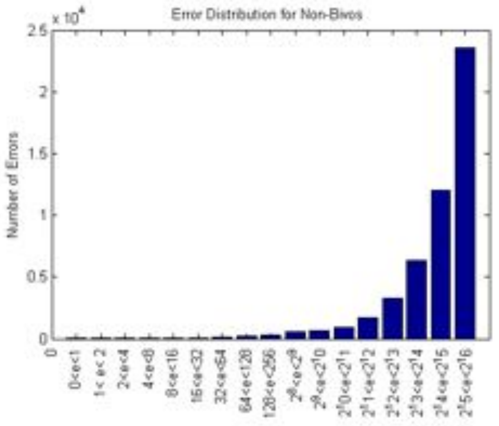


(b) Error Distribution for non Bivos Absolute Value Component

Figure 4.13. Error Distribution for Full Circuit



(a) Error Distribution for Bivos Full Circuit



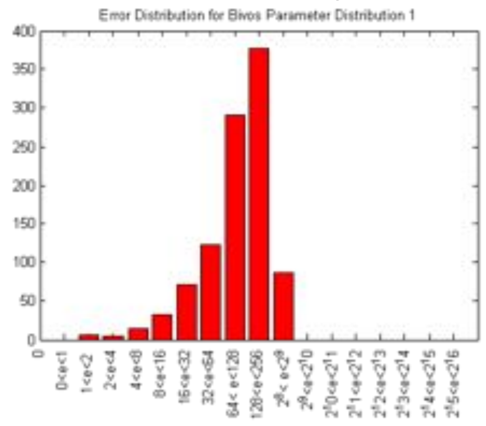
(b) Error Distribution for Bivos Full Circuit Component

of radiation with probabilistic error included in the model while Figure 4.16(b) shows the combined effect of the probabilistic and radiation error on the circuit.

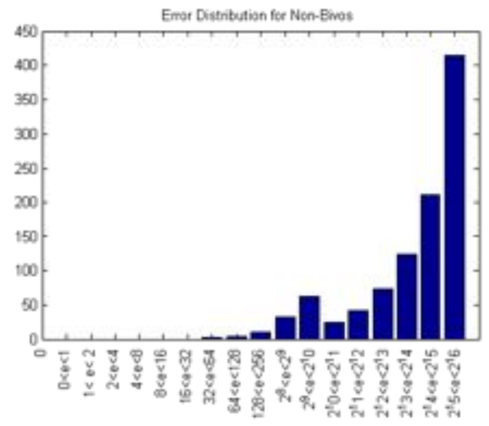
Figure 4.16(a) and Figure 4.16(b) shows the effect only of radiation with probabilistic error and combined effect of probabiliistic and radiation error on the circuit respectively for $p = 1/384$.

The energy savings for both probabilities of $p = 1/192$ and $p = 1/384$ versus p_{delta} are shown in figures Figure 4.17(a) and Figure 4.17(b)

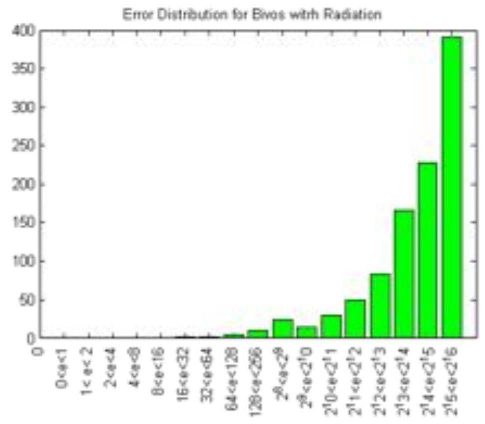
Figure 4.14. Error Distribution for Full Bitwise Circuit



(a) Error Distribution for Bivos Bitwise Full Circuit

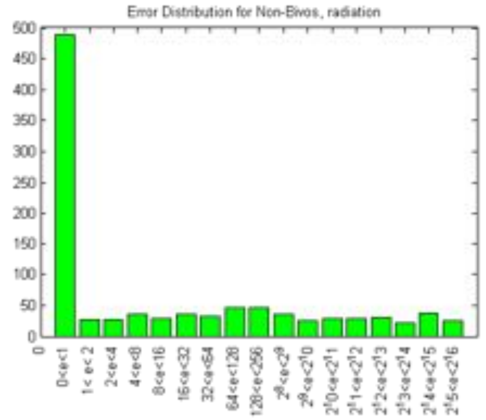


(b) Error Distribution for Bivos Bitwise Full Circuit Component

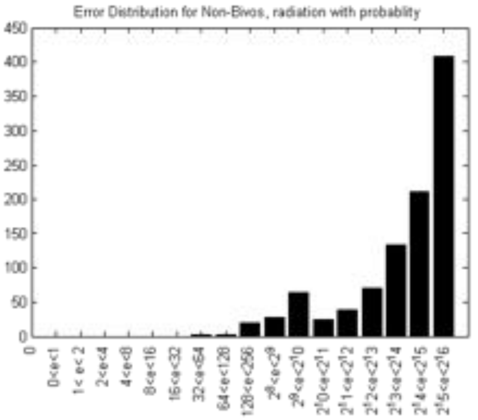


(c) Bivos Full Circuit, Only Radiation

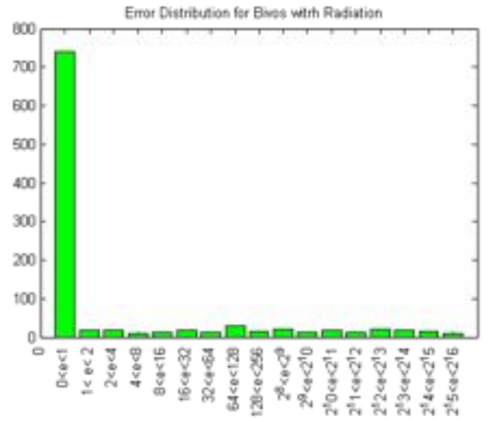
Figure 4.15. Error Distribution for Full Circuit Probabilistic and Radiation Error included



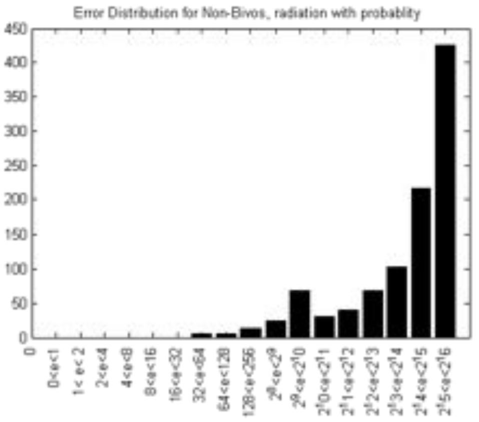
(a) Error Distribution for Bivos Full Circuit, Radiation Effect, p rad = 1/384



(b) Error Distribution for non Bivos Full Circuit, Radiation Effect, p rad = 1/384

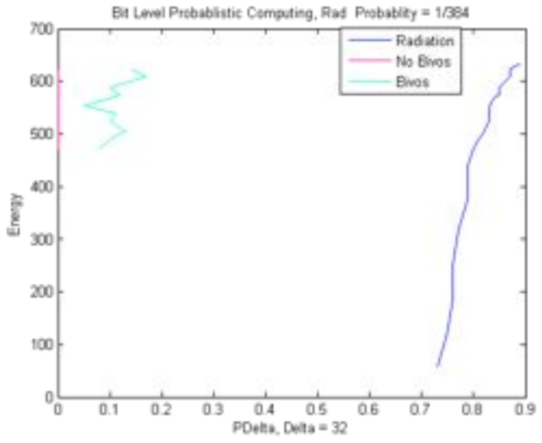


(c) Error Distribution for Bivos Full Circuit, Radiation Effect, p rad = 1/192

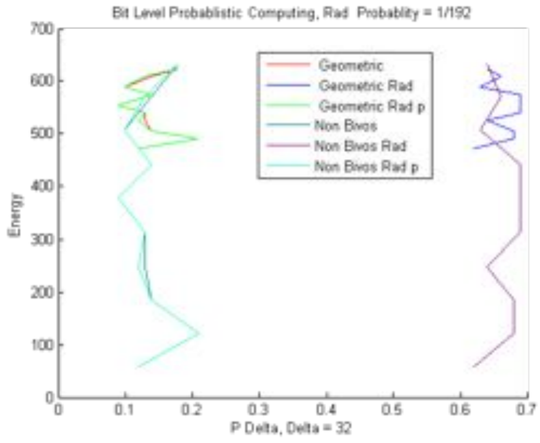


(d) Error Distribution for non Bivos Full Circuit, Radiation Effect, p rad = 1/192

Figure 4.16. Energy Savings



(a) Bit Level Probabilistic Computing, Radiation Probability = 1/384



(b) Bit Level Probabilistic Computing, Radiation Probability = 1/192

Chapter 5

Stereo Vision

We investigate the effects of probabilistic computing on the outcome of a stereo vision process. In particular, we focus on the algorithm explained in [10], since it is the basis for the stereo vision on the Mars rovers Spirit and Opportunity. For this reason we refer to it as the Mars Exploration Rover (MER) stereo vision algorithm. What sets the algorithm apart from the many stereo vision algorithms introduced in the field of computer vision, e.g., [15, 33, 29], is that it does not only focus on accuracy, but also on computational efficiency. The main reasons for this are the limited processing power onboard the Mars rovers and the need to react to the environment in real-time.

In this chapter, we first explain the steps of the MER stereo algorithm (Section 5.1). Then we explain the experimental setup for evaluating the algorithm's accuracy utilizing different settings for the probabilistic computing architecture (Section 5.2). Finally, we show and analyze the results of the stereo vision experiments (Section 5.3).

5.1 Stereo Vision Algorithm

The algorithm explained in [10] consists of the following steps:

1. Downscale images from 1024×1024 to 256×256 pixels.
2. Rectify images: after rectification, corresponding points lie on the same image line.
3. Calculate Laplacian: convolve the image with a difference of Gaussians in order to make the images independent of absolute illuminance.

4. 1-D correlator matching: along the image lines, matches are searched by minimizing the Sum of Absolute Differences between of 7×7 pixel windows.
5. Peak-filter: the selected disparity should have a significantly lower error than adjacent disparities.
6. Left-right consistency check: the correlator is also applied from the right to the left image. The so-determined disparity values should be sufficiently close to the disparitie values determined in the left image for the values to be accepted.
7. Filtering of noise: isolated disparity values are removed (blob filter).

Please remark that the stereo algorithm checks the outcomes of the correlator by means of the peak filter, left-right consistency check, and filtering of noise. Therefore, we expect that possibly large errors caused by the probabilistic computing are caught by these checks and hence do not significantly deteriorate the final results.

5.2 Experimental setup

We determine the performance of the stereo vision algorithm under two conditions: (1) normal computing, and (2) probabilistic computing. Below, we explain how we determine the performance of the algorithm under these conditions, and how we apply probabilistic computing.

5.2.1 Performance Determination

For determining the performance of the algorithm, we use the 'Middlebury' stereo vision data set [23], available at `vision.middlebury.edu/stereo/`. The data set contains ground truth disparities with which the estimated disparities can be compared. It has become an important benchmarking data set for the field of computer vision, with currently (February 10, 2014) 166 algorithms of which the performance is publicly known.

In our performance evaluation, we only evaluate the disparity of pixels that pass the checks of the MER stereo algorithm. In addition, we keep track of the percentage of pixels for which the algorithm outputs a disparity (percentage of 'certain' pixels). As a data set, we use the image pairs: Teddy, Cones, Art, Books, Dolls, Laundry, Moebius, and Reindeer.

5.2.2 Application of Probabilistic Computing

As explained in Chapter 2, the probabilistic computing is applied to the 1-D correlator matching. This algorithmic step is most expensive, since per pixel a 1-D search is performed for d disparity values, involving SAD calculations for a pixel window in the left and right image.

Specifically, the probabilistic computing structure is applied to the SAD calculation on two pixel windows. Please note that in order to fit with the probabilistic computing architecture, we use 8×8 pixel windows instead of 7×7 pixel windows. The left and right window are inserted into the probabilistic computing architecture, which returns a (probabilistic) sum of the pixels' absolute differences. All initial and subsequent algorithmic steps are identical to the normal operation of the algorithm.

5.3 Results

Before we show the stereo vision performance for different energy levels, we first want to illustrate the effect of probabilistic computing with BIVOS on the SAD calculations. Figure 5.1 shows the effects of probabilistic computing on the calculations of the SAD values. There are two plots, both showing the SAD-values (y-axis) over the disparity range (x-axis) for normal computing (blue line) and probabilistic computing (green line). The plots show that probabilistic computing adds noise to the SAD-values. The left plot shows a case where this noise does not influence the determination of the minimal SAD-value (the best matching disparity), while the right plot shows a case where probabilistic computing does influence the determination of the minimal SAD-value, shifting it significantly with 9 pixels. Please remark the difference in range of the y-axis: the determining factor for whether probabilistic computing causes bad results depends on the relation between the magnitude of the introduced noise and the differences in SAD values over the disparity range. Consequently, texture-poor image regions are likely to suffer more from probabilistic computing than texture-rich regions. Of course, these regions are inherently less reliable for stereo matching.

If a false match is introduced by probabilistic computing, it is likely to be filtered out by for example the left-right check in the stereo vision algorithm. The effect of probabilistic computing on the determination of a disparity map can be seen in Figure 5.2. The left image shows the ground-truth disparity

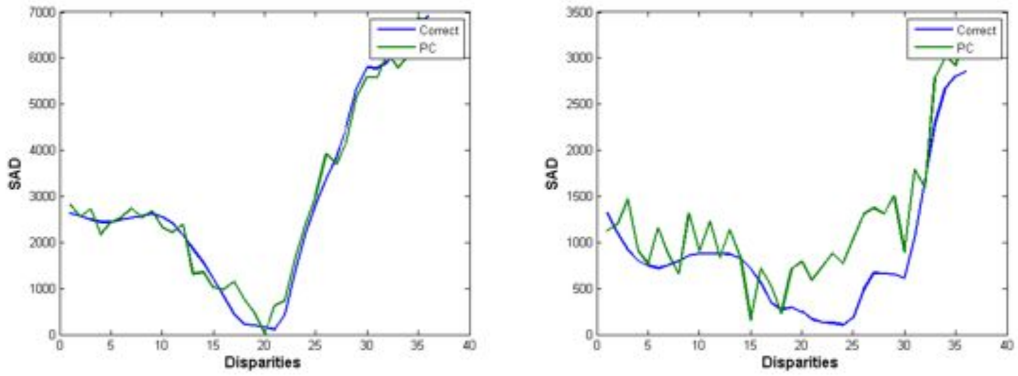


Figure 5.1. Effects of probabilistic computing on the calculations of the SAD values. Both plots show the SAD-values (y-axis) over the disparity range (x-axis) for normal computing (blue line) and probabilistic computing (green line). Left: probabilistic computing adds noise but does not change the selected disparity value. Right: probabilistic computing changes the selected disparity value.

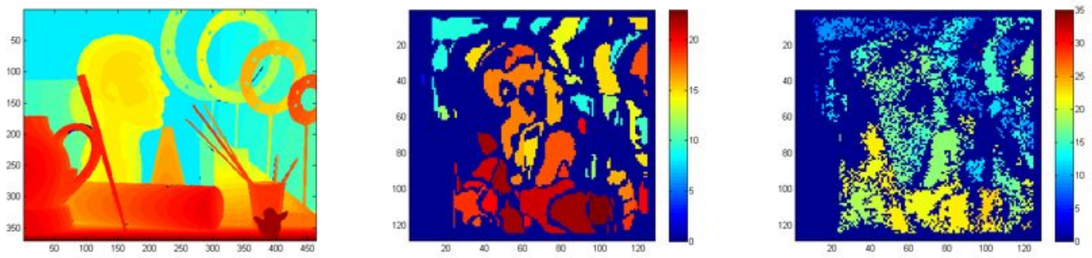


Figure 5.2. The left image shows the ground-truth disparity values, the center image the disparity map with normal computing, and the right image the disparity map with probabilistic computing. Please remark that each image has a different scale bar.

values, the center image the disparity map with normal computing, and the right image the disparity map with probabilistic computing. On the basis of these disparity maps, we make two main observations. First, compared with some state-of-the-art stereo vision algorithms, the MER stereo algorithm does not provide very good coverage of the image - a lot of disparity values are discarded. This is due to the fact that the algorithm has to deal with many more restrictions than just performance, e.g., real-time determination of the disparity values. Second, the disparity map determined with probabilistic computing still represents rather well the general structure of the scene. However, it results in fewer certain pixels and there are a few outliers, which stretch the range of determined disparity values (red is 23 in the normal computing disparity map and 35 in the probabilistic disparity map).

We now turn to the main result of this Ariadna study: the relation between the energy spent on computing and the performance of the stereo vision algorithm. The left part of Figure 5.3 shows the relation between energy and the percentage of good, certain pixels. Certain pixels are pixels for which

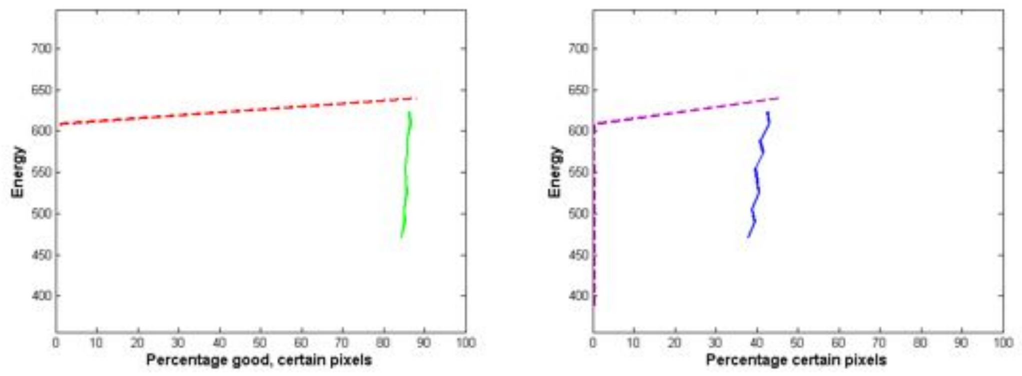


Figure 5.3. Left: relation between energy and the percentage of good, certain pixels. Right: relation between energy and the percentage of certain pixels. In both plots, the dashed lines represent the effects of linear probabilistic computing, while the solid lines represent the results with BIVOS probabilistic computing.

the disparity determination has passed all verifications of the stereo algorithms. Good pixels are pixels with a disparity close to the ground truth (with an absolute error ≤ 2). The green solid line shows the results for probabilistic computing with BIVOS, while the red dashed line shows the results for linearly scaled probabilistic computing. Two main observations can be made from this plot. First, the linearly scaled probabilistic computing directly deteriorates to a performance of 0 percent for a decreasing energy level. Second, the performance of BIVOS probabilistic computing only deteriorates with 2% while the energy is reduced from 625 to 475 (an energy reduction of 24%).

The right part of Figure 5.3 shows the relation between the percentage of certain pixels and the energy level. This relation is shown for linear probabilistic computing (dashed purple line) and BIVOS probabilistic computing (solid blue line). The plot shows that the price paid for the lower energy level also results in a lower percentage of certain pixels. Specifically, the percentage goes down from 42% to 38%. Please note that a normal computing setting leads to a percentage of 45% of certain pixels.

Finally, we would like to remark that it would be interesting to study also lower energy levels. However, with the chosen functions for implementing BIVOS it was difficult to test out such energy levels. Testing the results of larger energy gains will have to be addressed in future work.

Chapter 6

Conclusions

We have designed a probabilistic computing architecture for a robotic stereo vision computing task. The main idea is to save on energy spent on the stereo vision processing, while only slightly degrading the stereo vision performance. We conclude that within the context of the MER stereo algorithm, reducing the energy level by 24% results only in a small decrease in performance (2% fewer good certain pixels and 4% fewer certain pixels).

For future research, we would first recommend studying the effect of even lower energy levels on the stero vision performance. Furthermore, the effect of additional radiation on the stereo vision performance also deserves some attention.

Bibliography

[1] S. G. A. Qadi and S. Farritor. A dynamic voltage scaling algorithm for sporadic tasks. In *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pages 52–62, Cancun, Mexico, 2003.

[2] B. D. D. V. A. B. Marr, J. George and P. Hasler. Analysis of preeptive periodic real-time systems using the (max,plus) algebra with applications in robotics. *VLSI Design*, 2010:1–9, 2010.

[3] M. Choudhury, Q. Zhou, and K. Mohanram. Design optimization for single-event upset robustness using simultaneous dual-VDD and sizing techniques. *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 204–209, Nov. 2006.

[4] W. Dawes. *Hardening semiconductor components against radiation and temperature*. Noyes Data Corporation, Park Ridge, New Jersey, U.S.A., 1989.

[5] W. R. Dawes. *Hardening Semiconductor Components Against Radiation and Temperature.* Noyes Data Corporation, Park Ridge, NJ, United States of America, 1989.

[6] G. De Angelis, F. Badavi, S. Blattnig, M. Clowdsley, J. Nealy, G. Qualls, R. Singleterry, R. Tripathi, and J. Wilson. Modeling of the Martian Environment for Radiation Analysis. *Nuclear Physics B - Proceedings Supplements*, 166:184–202, Apr. 2007.

[7] G. De Angelis, F. Badavi, J. Clem, S. Blattnig, M. Clowdsley, J. Nealy, R. Tripathi, and J. Wilson. Modeling of the Lunar Radiation Environment. *Nuclear Physics B - Proceedings Supplements*, 166:169–183, Apr. 2007.

[8] J. George, B. Marr, E. Bilge, S. Akgul, and K. Palem. Probabilistic arithmetic and energy efficient embedded signal processing. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*.

[9] E. H. N. Gilson I. Wirth, Michele G. Vieira and F. L. Kastensmidt. Modeling the sensitivity of cmos circuits to radiation induced single event transients. *Microelectronics Reliability*, 48:29–36, January 2008.

[10] S. Goldberg, M. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. *IEEE Aerospace Conference Proceedings*, March 2002.

[11] M. Gurtner, L. Desorgher, E. Flückiger, and M. Moser. Simulation of the interaction of space radiation with the Martian atmosphere and surface. *Advances in Space Research*, 36(11):2176–2181, Jan. 2005.

[12] N. Hamid, a.F. Murray, D. Laurenson, S. Roy, and B. Cheng. Probabilistic Computing with Future Deep Sub-Micrometer Devices: A Modelling Approach. *2005 IEEE International Symposium on Circuits and Systems*, pages 2510–2513, 2005.

[13] G. Q. M. P. I. Hong, D. Kirovski and M. Srivastava. Power optimization of variable-voltage core-based systems. *IEEE Transactions on Computer-Aided Design*, 18(12):1702–1714, 1999.

[14] A. B. E. S. J. George, B. Marr and K. Palem. Probabilistic arithmetic and energy efficient embedded signal processing (received best paper award.). In *Proceedings of the Intl. Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 158–168, Seoul, Korea, 2006.

[15] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR 2006*, 2006.

[16] M. Lu. *Arithmetic and Logic in Computer Systems*. John Wiley & Sons, Hoboken, NJ, 2004.

[17] B. Marr, J. George, B. Degnan, D. Anderson, and P. Hasler. Error immune logic for low-power probabilistic computing. In *VLSI Design*, pages 1–10, 2010.

[18] G. C. Messenger. Collection of Charge on Junction Nodes from Ion Tracks. *IEEE Transactions on Nuclear Science*, 29(6):2024–2031, 1982.

[19] K. Mohanram. Closed-Form Simulation and Robustness Models for SEU-Tolerant Design. *23rd IEEE VLSI Test Symposium (VTS'05)*, pages 327–333.

[20] K. Mohanram. Simulation of transients caused by single-event upsets in combinational logic. *IEEE International Conference on Test, 2005.*, pages 973–981, 2005.

[21] D. L. Nor H. Hamid, Alan F. Murray. Probabilistic computing with future deep sub-micrometer devices: A modelling aproach. *Symposium A Quarterly Journal In Modern Foreign Literatures*, pages 2510–2513, 2005.

[22] K. V. Palem, L. N. Chakrapani, and B. E. S. Akgul. Review on Probabilistic 25 A CMOS (PCMOS) Technology : From Device Characteristics to Ultra-Low-Energy SOC Architectures. (1977), 2008.

[23] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, 2002.

[24] I. Schneider and J. Kasting. Radiation environments on Mars and their implications for terrestrial planetary habitability. In K. Meech, J. Keane, M. Mumma, J. Siefert, and D. Werthimer, editors, *Bioastronomy 2007: Molecules, Microbes and Extraterrestrial Life*, volume 420, page 149, 2009.

[25] I. Schneider and J. F. Kasting. Radiation environments on mars and their implications for terrestrial planetary habitability. *Bioastronomy 2007: Molecules, Microbes and Extraterrestrial Life ASP Conference Series*, 420:149, 2009.

[26] V. Sheel and S. Haider. Calculated production and loss rates of ions due to impact of galactic cosmic rays in the lower atmosphere of Mars. *Planetary and Space Science*, 63-64:94–104, Apr. 2012.

[27] S. Skutnik and J. Lajoie. A GEANT-Based Model for Single Event Upsets in SRAM FPGAs for Use in On-Detector Electronics. *IEEE Transactions on Nuclear Science*, 53(4):2353–2360, Aug. 2006.

[28] F. G. W. van der Mark and J. van den Heuvel. Stereo based navigation in unstructered environments. In *IEEE Instrumentation and Measurements Conference*, pages 2038–2042, Budapest, Hungary, 2001.

[29] L. Wang and R. Yang. Global stereo matching leveraged by sparse ground control points. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pages 3033–3040, 2011.

[30] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley, 2011.

[31] G. I. Wirth, M. G. Vieira, E. H. Neto, and F. L. Kastensmidt. Modeling the sensitivity of CMOS circuits to radiation induced single event transients. *Microelectronics Reliability*, 48(1):29–36, Jan. 2008.

[32] K. C. Y. Shin and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the International Conference on Computer-Aided Design*, pages 365–368, 2000.

[33] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. *IEEE PAMI*, 2008.

[34] A. D. YF. Yao and S. Shenker. A scheduling model for reduced cpu energy. In *IEEE Symposium on Foundations Computer Science*, pages 374–382, 1995.