CHAPTER 3

METHODOLOGY

3.1 Introduction

The process flow in order to make this project succeed begins with research activities to get as much as detail data and information about the selected multipliers from journals, books and internet. The finding from these activities is important to get clear view and concept about the multipliers, and to make proposal report. After proposal's approval received from supervisor, the schematic of the selected multiplier is designed using Altera's Quartus II Software. The speed performance is tested and verified using Timing Analyzer that provided in Altera's Quartus II Software.

The detail descriptions and information collected from journals, books and websites, make possible to design the schematic of Modified Baugh-Wooley Two' Complement Signed Multiplier. The schematic is designed by referring to the tabular form of bit-level Modified Baugh-Wooley Two's Complement Signed multiplication. The schematic design of the multiplier consist operation of AND gate, NAND gate, Half Adder, Carry Save Adder and D Flip-flop. The schematic is designed, compiled and simulate using Altera's Quartus II Software.

3.2 Flow Chart of the Project

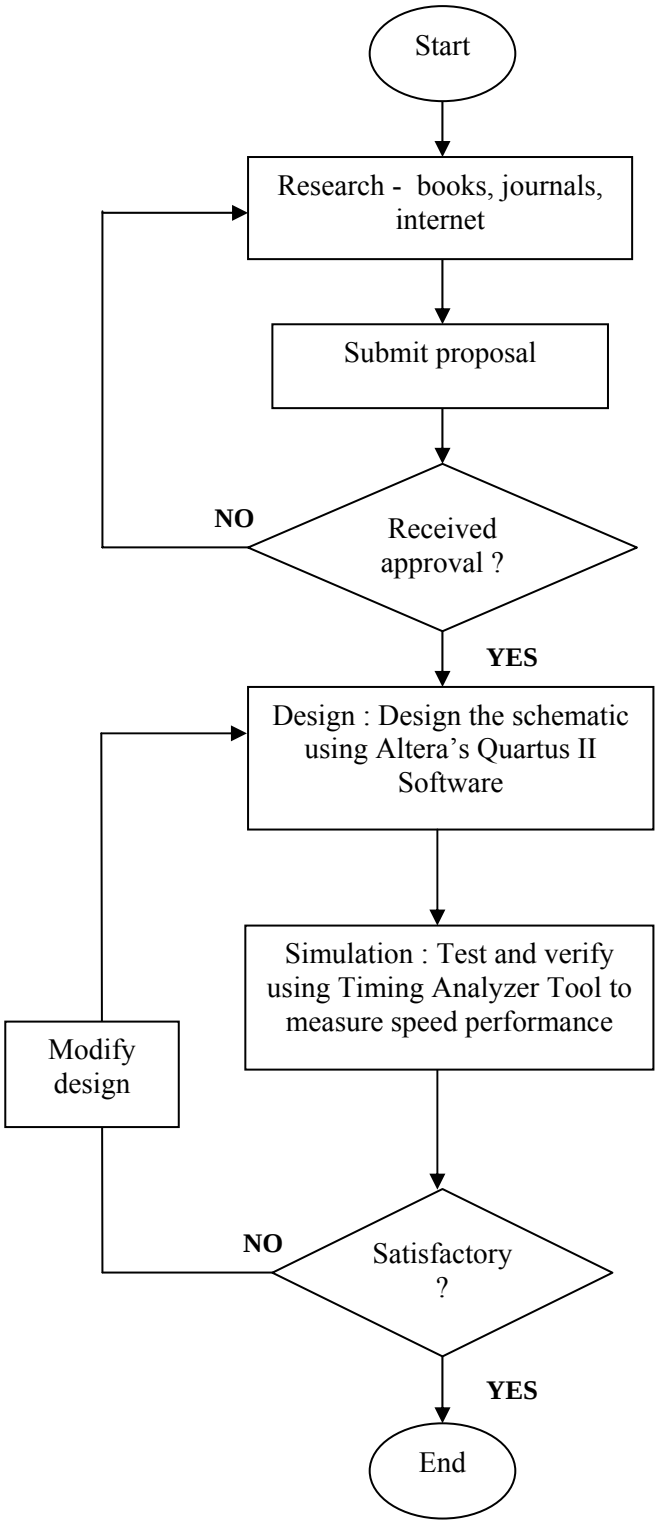

Figure 3.1: Flow Chart of the Project

As shown in Figure 3.1, this project is begins with research activities to get as much as detail data and information about the selected multipliers from journals, books and internet. The finding from these activities is important to get clear view and concept about the multipliers. From the finding of the researched, the best high speed multiplier is selected which is Modified Baugh-Wooley Two's Complement Signed Multipler and a proposal report is made. After proposal's approval received from supervisor, the schematic of the selected multiplier is designed using Altera's Quartus II Software.

The schematic is designed by referring to the tabular form of bit-level Modified Baugh-Wooley Two's Complement Signed multiplication. The schematic design of the multiplier consist operation of AND gate, NAND gate, Half Adder, Carry Save Adder and D Flip-flop. The design has to be complied to check for errors and also to measure speed performance using Timing Analyzer which provided in Altera's Quartus II Software. The output of the design is check and compare with theoretical. If the multiplication does not produce correct output, a modification on design will be made until the multiplier produce correct output.

3.3 Subcircuits Design

The schematic design of the multiplier consist operation of AND gate, NAND gate, Half Adder, Carry Save Adder and D Flip-flop. The schematic is designed, compiled and simulate using Altera's Quartus II Software and obtained on EPF10K70. The speed performance is tested and verified using Timing Analyzer to define the critical path in the design.

3.3.1 AND Gate

The operation of the AND gate as shown in Figure 3.2 is simple and is defined as follows; the output, X, is 1 if input A and input B are both 1 [10]. In other words, if A=1 and B=1, then X=1. If either A or B or both are 0, the output will be 0 as in Table 3.1.

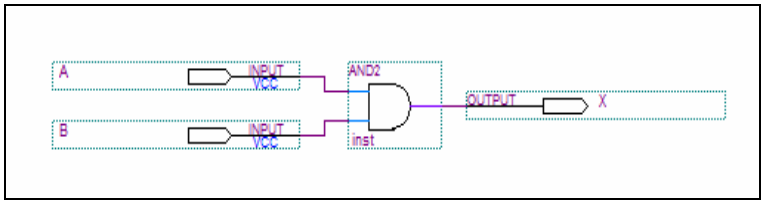

Figure 3.2 : Logic Diagram of Two Input AND Gate

Table 3.1 : Truth Table for A Two Input AND Gate

Inputs		Outputs
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

3.3.2 NAND Gate

The operation of the NAND gate as shown in Figure 3.3 is the same as the AND gate except that its output is inverted. The symbol for a NAND gate is made from and AND gate with a bubble at its output, denoting the inversion operation.

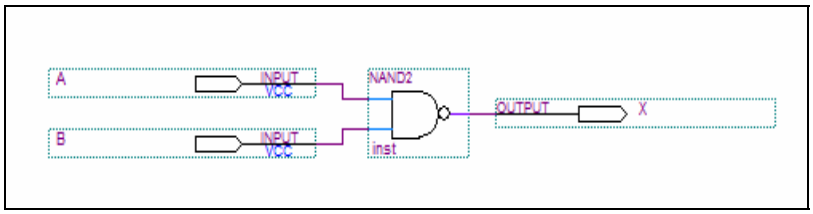

Figure 3.3 : Logic Diagram of Two input NAND Gate

The NAND gate produces a 0 output only when all the inputs are 1. Then any of the inputs is 0, the output will be 1 as shown in Table 3.2.

Table 3.2 : Truth Table for A Two Input AND Gate

Inputs		Outputs
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

3.3.3 Half Adder

A Half Adder is an arithmetic circuit that generates the sum of two binary digits. The HA can be implemented with one exclusive OR gate and one AND gate as shown in Figure 3.4 below ;

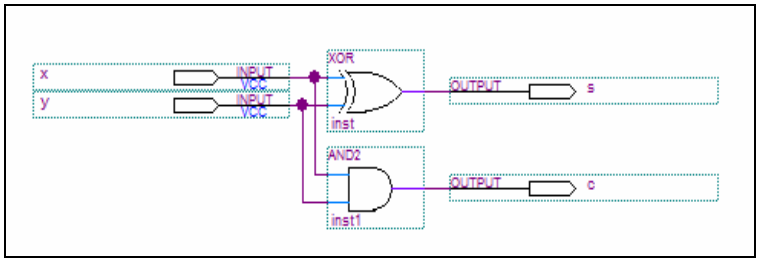

Figure 3.4 : Logic Diagram of Half Adder

The circuit has two inputs and outputs. The input variables are the augend and addend bits to be added, and the output variables produce the sum and carry. The C output is 1, only when both inputs are 1. The S output represents the least significant bit of the sum. The Boolean functions for the two outputs are given below;

$$\begin{aligned} S &= \overline{X}Y + X\overline{Y} \\ &= X \oplus Y \\ C &= XY \end{aligned} \tag{3.1}$$

Below in Table 3.3 represent the truth table for Half Adder.

Table 3.3 : Truth Table for Half Adder

Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3.3.4 Carry Save Adder

Carry Save Adder accepts three n-bit operands and generates two n-bit results, an n bit partial sum, and n-bit carry. Carry Save Adder also known as (3, 2) counter. The n-bit Carry Save Adder consists of n (3,2) counters in parallel with no carry shown below in Figure 3.5;

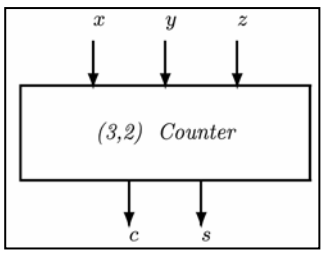

Figure 3.5 : The (3,2) Counter Block Diagram

The basic implement of the Carry Save Adder is exactly same as Full Adder with three inputs, X, Y and Z, and two outputs, S and Y. Input X and Y represent the two significant bits to be added and Z represents the carry from the previous lower significant position as shown in Figure 3.6.

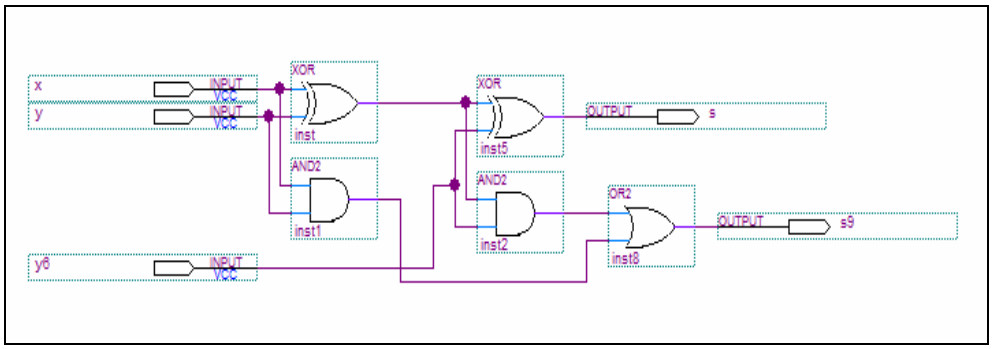

Figure 3.6 : Logic Diagram of Full Adder

The values for the outputs are determined from the arithmetic sum of the three input bits. When all the input bits are 0, the outputs are 0. The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to1. The C output has a carry of 1 if two or three inputs are equal to 1 as shown in Table 3.4.

$$\begin{aligned} S &= \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ \\ C &= XY + XZ + YZ \end{aligned} \tag{3.2}$$

Table 3.4: Truth Table for Full Adder

Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3.3.5 D Flip-Flop

D flip-flop can be formed from the gated S-R flip-flop by the addition of an inverter as shown in Figure 3.7. This enables just a single input D to both Set and Reset the flip-flop.

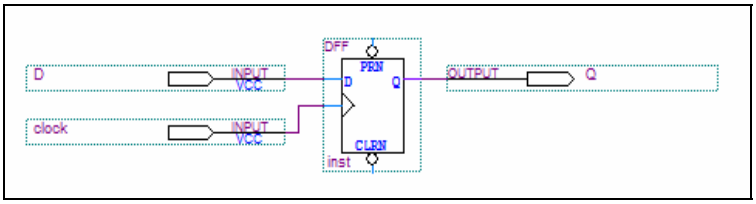

Figure 3.7 : Block Diagram of D Flip-Flop

The operation is such that Q will be the same as D while clock is 1, and Q will remain latched when clock goes 0 as shown in Table 3.5.

Table 3.5 : Truth Table for D Flip-Flop

Inputs				Outputs
Set	Reset	Clock	D	Q
0	1	X	X	1
1	0	X	X	0
0	0	X	X	undefined
1	1	↑	0	0
1	1	↑	1	1

3.4 The Design of Modified Baugh-Wooley Two's Complement Signed Multiplier

The design of Modified Baugh-Wooley Two's Complement Signed Multiplier is done by referring to the tabular form of bit-level Modified Baugh-Wooley Two's Complement Signed multiplication as shown in Figure 3.8. Since, there is no tabular form available for 8-bits x 8-bits Modified Baugh-Wooley Two's Complement Multiplier, the tabular form for it is design referring to tabular form 5-bits x 5-bits as shown in Figure 2.7 in Section 2.3.2 that has been discussed before. The implementation of this circuit needed AND gates, NOT gates, Half Adders and Carry Save Adders to form the partial product bits. When the operands a, b are presented as inputs, the two's complement multiplier will produce the correct answer, except for the sign bit of the result [11]. However, the sign bit can be computed separately and appended at the sign position of the final result, disregarding the resulting sign bit of the two's complement multiplication.

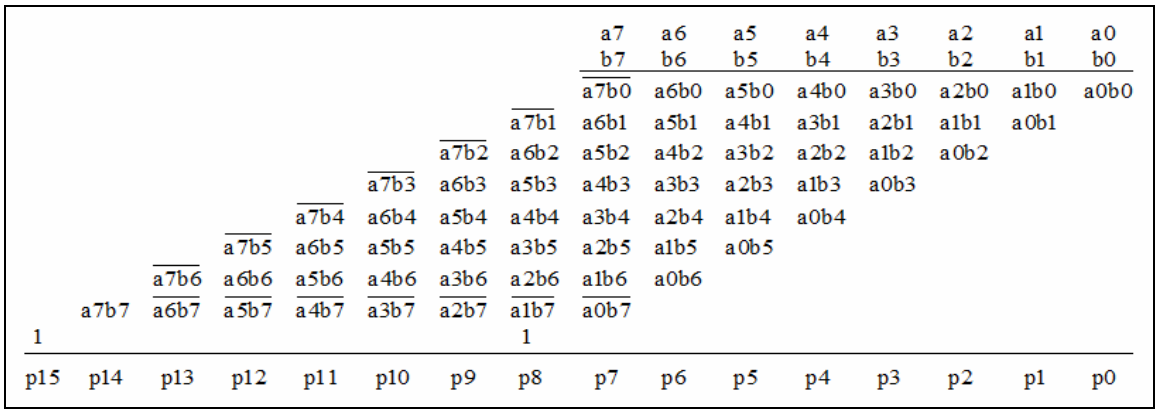

Figure 3.8 : Tabular Form of Bit-Level Modified Baugh-Wooley Two's Complement Signed Multiplication

Altera's Quartus II Software is used to draw, compile and analyze the design. Block Design File.(bdf) is created by drawing the schematic of the multiplier. Altera's Quartus II perform an analysis of the bdf file to make sure that there are no errors in the schematic. To simulate the schematic, Vector Waveform File.(vwf) is created. It provides a way to draw the waveforms that step through all the possible combinations of the inputs and produce the resulting outputs. The resulting output is compared to the expected result to ensure that the schematic is functioning properly.

A detailed analysis of all timing delays for the schematic performed using Timing Analyzer Tool. It provided analysis such as maximum clock frequency (f_{MAX}), setup (t_{SU}), hold (t_H), clock-to-output (t_{CO}), and pin-to-pin (t_{PD}) timing. f_{MAX} is a function of the longest propagation delay between two registers in the schematic. t_{CO} is the time elapsed from the active edge of the clock signal at the clock source until a corresponding output signal is produced (from a flip-flip) at an output pin. t_{SU} is the length of time before the active edge of a trigger pulse that the inputs of a digital device must be in a stable digital state. For example, if the setup time of a device is 20 ns, the inputs must be held stable and will be read, 20 ns before the trigger edge. t_H is the length of time after the active clock edge that the input data to be recognized must be held stable to ensure recognition.

3.5 Approaches to Increase the Speed Performance of the Modified Baugh-Wooley Two's Complement Signed Multiplier.

There are three ways that can speed up multiplication: reduce the number of the partial products, accelerate the accumulation and pipelining. However, in this project only two ways have been used to speed up multiplication which is reduce the number of partial product and accelerate the accumulation.

3.5.1 Reduce the Partial Products

Conventional Baugh-Wooley Multiplier method increases the height of the longest column by two, which may lead to a greater delay through the Carry Save Adder tree. In the given example of Figure 3.9 column height is 7, requiring an extra Carry Save Adder level.

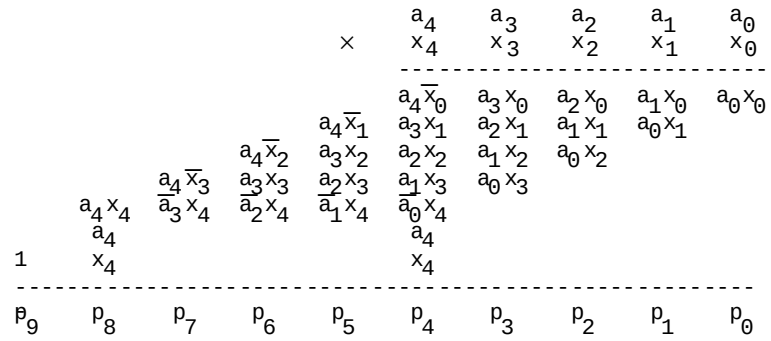

Figure 3.9 : Baugh Wooley Two's Complement Multiplication

Removing x_4 from fourth column and writing two x_4 entries in the third column, which has only four entries, can reduce the extra delay caused by the additional Carry Save Adder level. Thus, the maximum number of entries in one column becomes six, which can be implemented with three level Carry Save Adder tree.

All negatively weighted a_4x_4 terms can be transferred to the bottom row, which leads to two negative numbers in the last two rows, where a subtraction operation from the sum of all the positive elements is necessary. Instead of subtracting a_4x_4 two's complement of a can be added x_4 times. This method is known as the Modified Baugh-Wooley algorithm as shown in Figure 3.10.

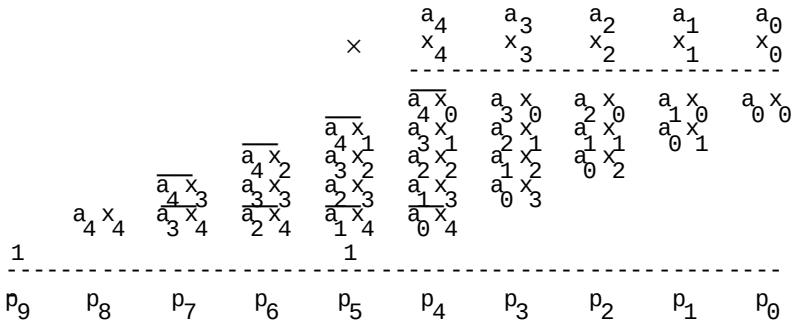

Figure 3.10 : Modified Baugh-Wooley Two's Complement Signed Multiplication

3.5.2 Accelerate the Accumulation

There are two methods used in this Modified Baugh-Wooley Two's Complement Signed Multiplier to reduce the time needed to accumulate the partial products. First is using Half Adder instead of Full Adder [12] and use faster adder which is Carry Save Adder [13].

In this project, Modified Baugh-Wooley Two's Complement Signed Multiplier use carry save addition concept. The partial product is replaced by a partial sum and partial carry, which are saved and passed on to the next row. The advantage of carry save addition is that the additions at different bit positions in the same row are independent of each other and can be carried out in parallel, which essentially speed up the addition phase of each cycle, and hence speeds up the multiplication.

The addition of partial sum and partial carry is performed by a vector merging adder. The dependence graph of bit level carry save multiplication is shown in Figure 3.11. The Modified Baugh-Wooley Two's Complement Signed Multiplier is designed from this dependence graph.

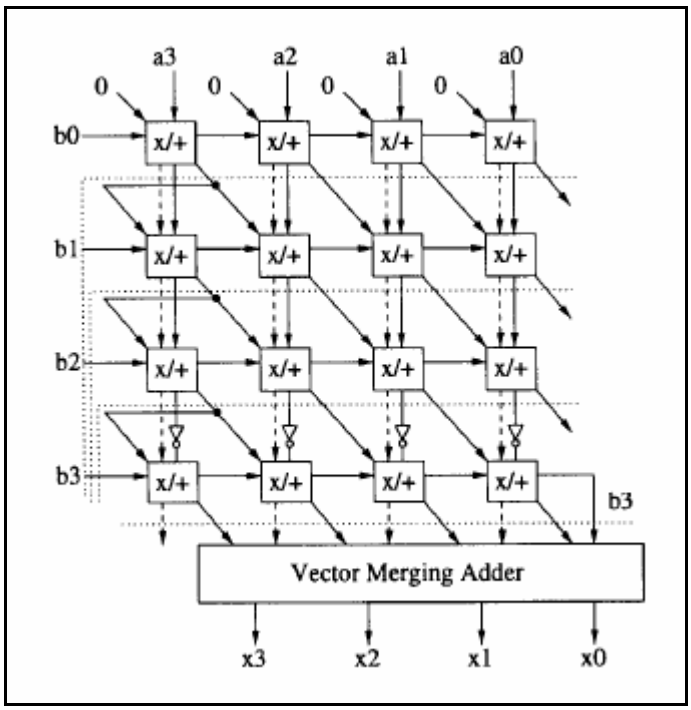

Figure 3.11 : Dependence Graph for the 4-bits x 4-bits Carry Save Array Multiplication

The vector merging adder is implemented as 4-bits x 4-bits carry save array which contains only Half Adder and has a more regular structure is shown in Figure 3.12. The advantage of the Half Adder array is that the delay of each stage is half of the delay of the Full Adder stage.

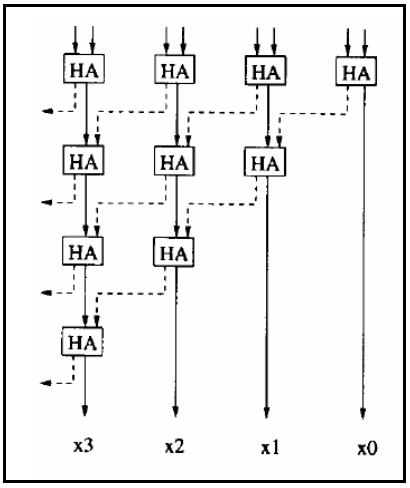

Figure 3.12 : A Carry Save Vector Merging

3.5.3 Pipelining

The need for fast, yet small registers (REG) which is D flip-flop has been used as pipelined data path structures [15]. Pipelining is a popular design technique often used to accelerate the operation of the data paths of micro- and signal processors. This statement can be easily explained with the Figure 3.13 below. A REG has been put at input and output to provide a clock signal to measure the speed.

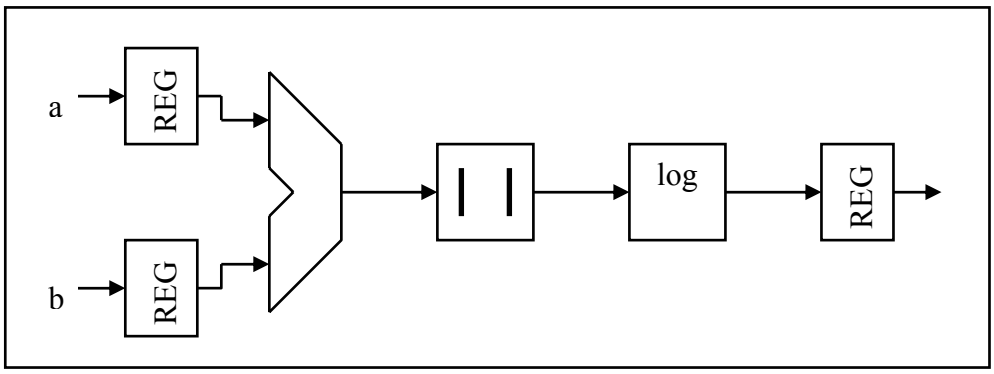

Figure 3.13 : Non-pipelined Version

The goal of the presented circuit is to compute log ($|a-b|$), where both a and b represent streams of numbers, that is, the computation must be performed on large set of input values.

Now, by adding register between the logic blocks, as shown in Figure 3.14, the result for the data set (a_1, b_1) only appears at the output after three clock-periods. At that time, the circuit has already performed parts of the computations for the next data sets, (a2, b2) and (a3, b3). The computation is performed in an assembly-line fashion, hence the name pipeline. The advantage of the pipelined operation becomes apparent when examining the maximum clock speed (or the minimum clock period) of the modified circuit. However, adding extra pipeline stages only makes sense up to a certain point. When the delays of the registers become comparable to the logic delays, no extra performance is gained. Adding extra stages only increases the hardware overhead and its area.

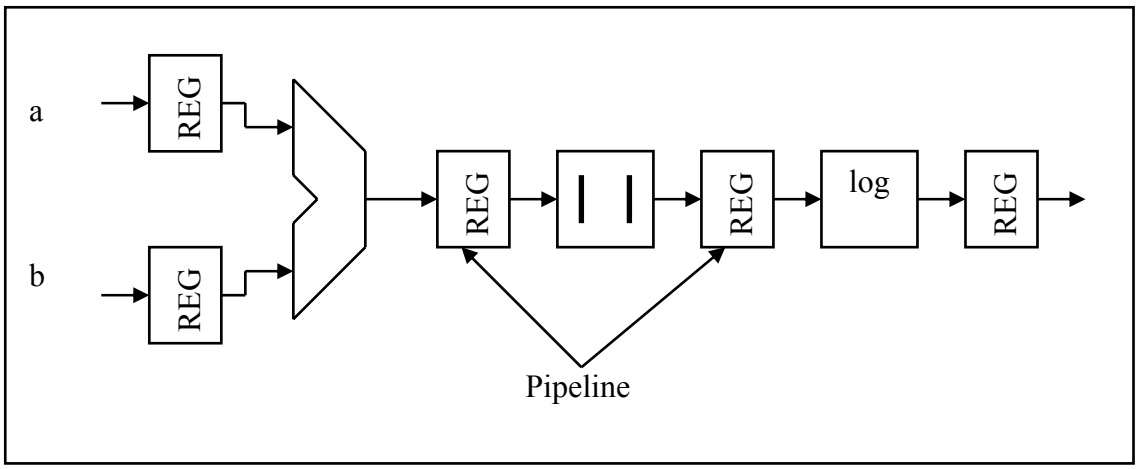

Figure 3.14 : Pipelined Version