Design and Implementation of Dadda & Wallace Multiplier

Submitted By

Harsh Gupta (22M1169)

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Powai, Mumbai - 400076

Contents

List of Figures

Chapter 1

Introduction

The concept of Binary Multiplier

A binary multiplier is an integrated circuit used in digital circuits and so many applications, such as a microprocessor, to multiply two binary numbers. There are so many different kind of Digital devices where multipliers are being, very, frequently.

Now a days, so many applications are using Multipliers like these are generally being utilized in digital signal processing. Increasing technology has maximized the demand for rapid and efficient real-time digital signal processing applications.

A digital multiplier can be implemented using a variety of computer arithmetic techniques. The majority of techniques involve computing a set of partial products, which are then summed using binary adders.

The process of multiplication can be split into 3 steps:

- generating partial product
- reducing partial product
- computing final product

Now a days most of multiplier architectures use **Wallace trees** or **Dadda multipliers** to add the partial products together in a single cycle.

Finally at last stage in Multipliers, shift-and-add multipliers require a fast adder (something faster than ripple-carry) for increasing their speed.

1.1 Wallace Multiplier

Since, Multipliers do not have the same number of bits in every column. Therefore in 1964, Wallace proposed a method for a carry save like reduction scheme which is valid for columns of variable length.

Wallace multiplier uses adders which take multiple inputs of the same weight and produce sum outputs of the same weight and carry outputs with higher weights.

These are combined in stages to reduce the number of terms at each weight to 2 or less. These two terms are then added by a conventional adder to produce the final result.

Wallace multipliers act in three stages:

- Generate all bits of the partial products in parallel.
- Collect all partial products bits with the same place value in bunches of wires and reduce these in several layers of adders till each weight has no more than two wires.
- For all bit positions which have two wires, take one wire at corresponding place values to form one number, and the other wire to form another number.

Add these two numbers using a fast adder of appropriate size.

Reduction Stage of Wallace Multipliers

- Partial products are grouped in multiples of three rows.
- Rows which are additional over multiples of three are just passed on to the next stage.
- Wires are now reduced for each group of 3 rows.

- For any group of 3 rows, if we find 3 wires for any weight, we place a Full Adder, which generates 1 wire of the same weight and 1 wire with the next higher weight.

- If there are two wires left, we place a half adder to reduce these.

- If only one wire is left, it is carried through to the next layer.

- The reduction procedure is carried out for all weights. starting from the least significant weights.

- At the end of of each layer, we count wires for each weight again, and if none has more than 2 wires, we proceed to the final addition stage.

- If any weight has 3 or more wires, we add another layer, and repeat this procedure till the number of wires for all weights is reduced to 2 or less.

- Now we compose one number from one of the left over wires at corresponding weights and another from the remaining wires.

- Finally, we use a conventional fast adder of appropriate size to add the two numbers.

Redundant MSB in large Wallace Multipliers

- The reduction scheme as described above sometimes produces a redundant most significant bit.

- The result is still correct and the redundant bit will always be zero.

- To see this effect & understand, let us apply the above scheme to an 8x8 Wallace multiplier and observe its Dot Diagram

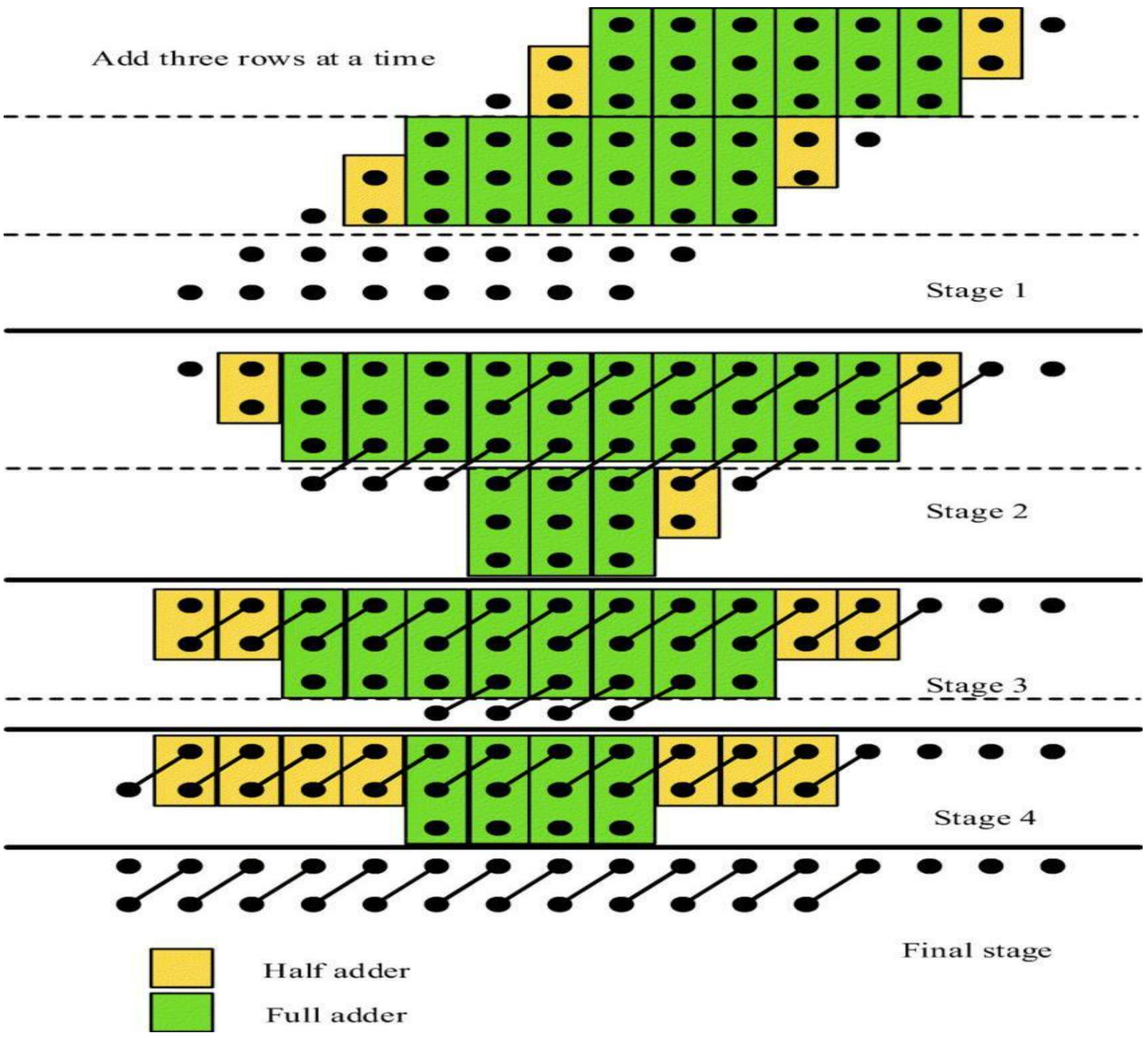


Figure 1.1: 8X8 Wallace Multiplier

1.2 Dadda Multiplier

- Wallace multipliers reduce as soon as possible, while Dadda multipliers reduce as late as possible.

- Dadda multipliers plan on reducing the final number of wires for any weight to 2 with as few and as small adders as possible.

- We determine the number of layers required first, beginning from the last layer, where no more than 2 wires should be left.

- The number of layers in Dadda multipliers is the same as in Wallace multipliers.

Dadda Multipliers: Layer wise division Procedure

- We work back from the final adder to earlier layers till we find that we can manage all wires generated by the partial product generator.

** We know that the final adder can take no more than 2 wires for each weight.

- Let dj represent the maximum number of wires for any weight in layer j. where j= 1 for the final adder (thus $d_1 = 2$).
- The maximum number of wires which can be handled in layer j+1 (from the end) is the integral part of $3/2 d_j$.

- We go up in j. till we reach a number which is just greater than or equal to the largest bunch of wires in any weight.

- The number of reduction layers required is this $J_{final} - 1$.

Wire Reduction in Dadda Multipliers

- At each layer we know the maximum number of wires which should be left for the next layer.

- For each weight, we place the least number of smallest adders, such that the wires going out to the next layer do not exceed the maximum number of wires it can handle.

- At each weight, we must consider all the sum and pass through wires at this weight, as well as the wires which will be transferred through carry of the less significant weights, to the next layer.

.

- That is why we must begin with the lowest weight and go towards higher weights in each layer.

❖ To understand this, let us apply the above scheme to an 8x8 Dadda multiplier and observe its Dot Diagram

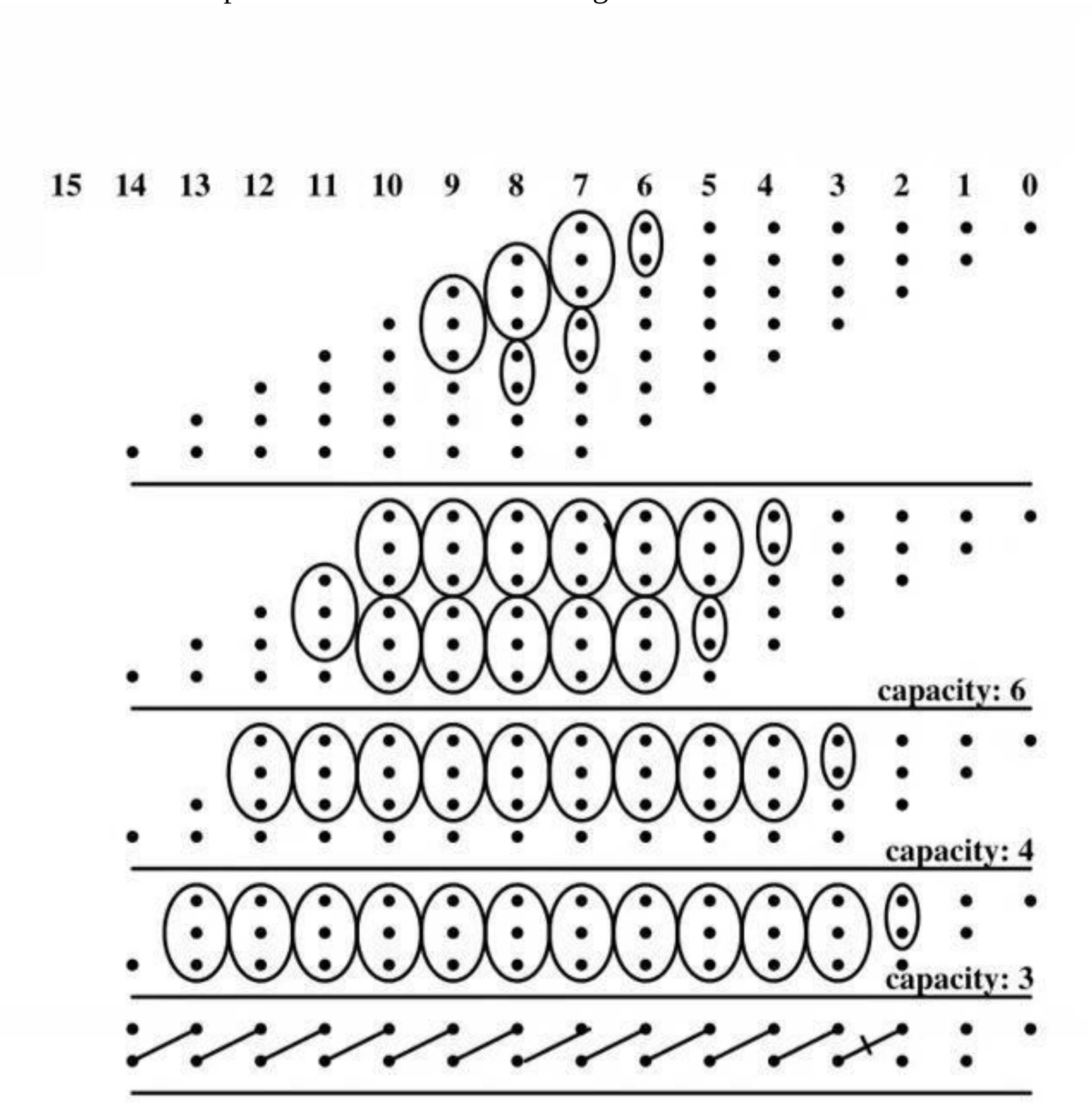
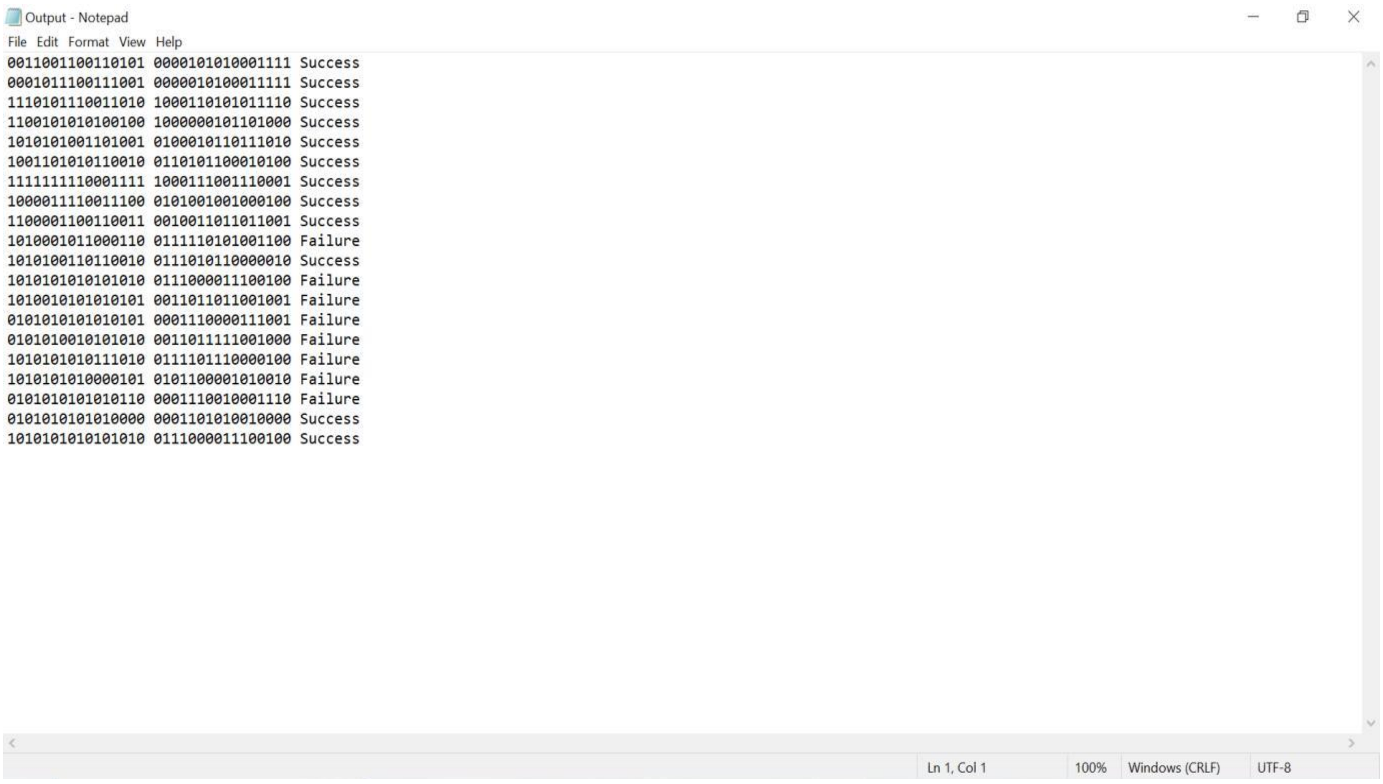


Figure 1.2 : 8x8 Dadda Multiplier

Implementation on Xen-10 board (Altera Max-10 faimily)

- First, we design both Wallace & Dadda multiplier in either VHDL or Verilog.
- For wrapping whole design unit , we design a DUT
- Since here we are having 2 inputs of 8 bits i.e. total of 16 bits so for doing that we used **VIRTUAL J-tag**
- To cross check our Result we used **scan-chain**
- Hence , To instantiate both V-jtag and DUT components into one design we made Top Level design.
- With the help of Quartus prime we generated finally **SVF file** by putting proper device in setting

Following Output file Observed through scan_vjtag:

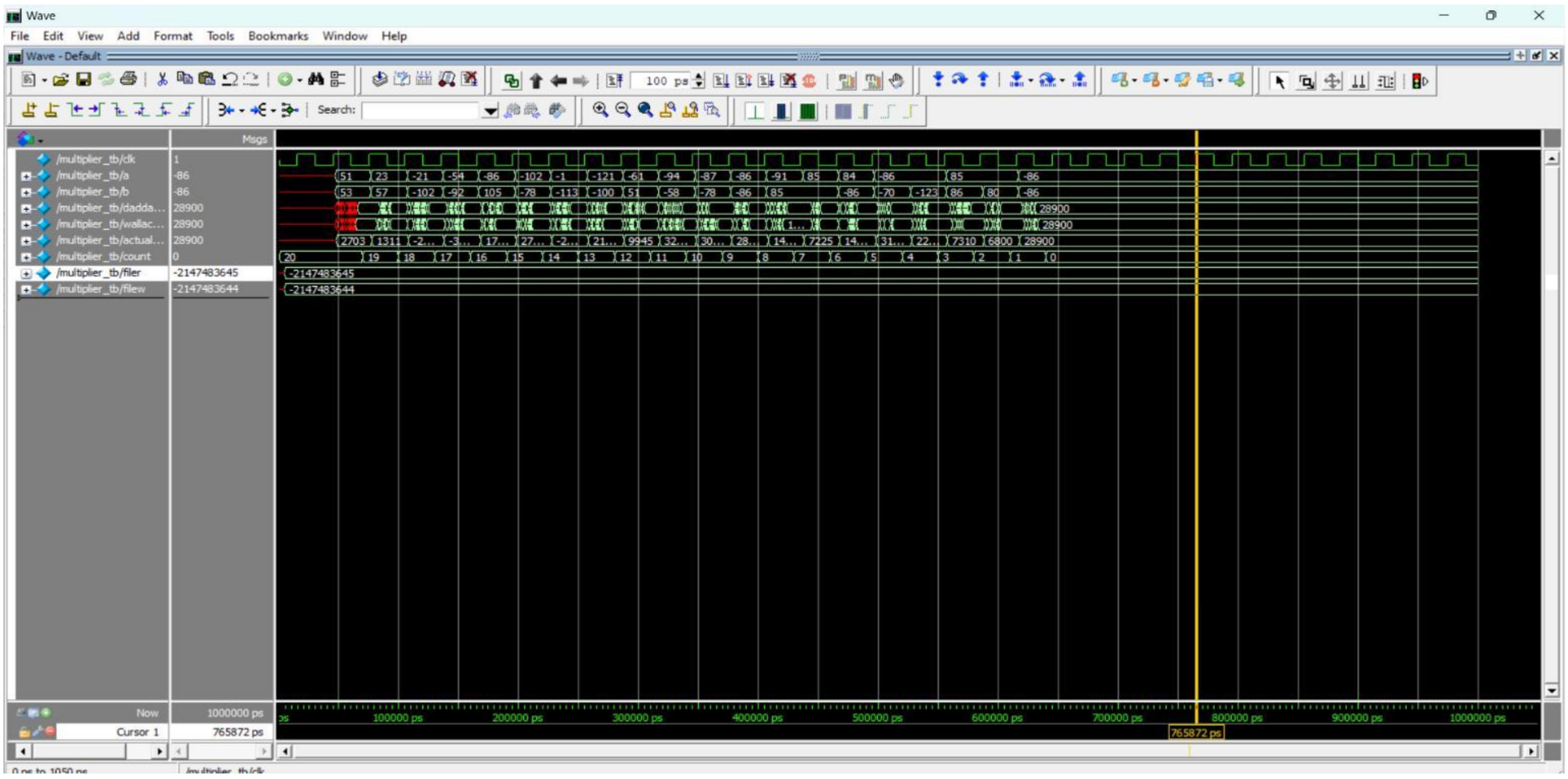


```
0011001100110101 0000101010001111 Success
0001011100111001 0000010100011111 Success
1110101110011010 1000110101011110 Success
1100101010100100 1000000101101000 Success
1010101001101001 0100010110111010 Success
1001101010110010 0110101100010100 Success
1111111110001111 1000111001110001 Success
1000011110011100 0101001001000100 Success
1100001100110011 0010011011011001 Success
1010001011000110 0111110101001100 Failure
1010100110110010 0111010110000010 Success
1010101010101010 0111000011100100 Failure
1010010101010101 0011011011001001 Failure
0101010101010101 0001110000111001 Failure
0101010010101010 0011011111001000 Failure
1010101010111010 0111101110000100 Failure
1010101010000101 0101100001010010 Failure
0101010101010110 0001110010001110 Failure
0101010101010000 0001101010010000 Success
1010101010101010 0111000011100100 Success
```

Figure 1.3 : Output file through scan_chain for cross checking Multiplier

Chapter 2

Simulation Results



	Msgs
/multiplier_tb/clk	1
/multiplier_tb/a	-86
/multiplier_tb/b	-86
/multiplier_tb/dadda...	28900
/multiplier_tb/wallac...	28900
/multiplier_tb/actual...	28900
/multiplier_tb/count	0
/multiplier_tb/filer	-2147483645
/multiplier_tb/filew	-2147483644
Now	1000000 ps
Cursor 1	765872 ps

Figure 2.1 : Simulation Result

Delay Comparison between Dadda & Wallace Multiplier

	Dadda	**Wallace**
Input-1	19 unit	17 unit
Input-2	16 unit	16 unit
Input-3	18 unit	16 unit
Input-4	17 unit	16 unit
Input-5	19 unit	13 unit
Input-6	14 unit	14 unit
Input-7	16 unit	16 unit
Input-8	16 unit	13 unit
Input-9	18 unit	14 unit
Input-10	19 unit	20 unit
Input-11	11 unit	19 unit
Input-12	14 unit	15 unit
Input-13	17 unit	13 unit
Input-14	15 unit	14 unit
Input-15	15 unit	15 unit
Input-16	13 unit	15 unit
Input-17	14 unit	12 unit
Input-18	20 unit	13 unit
Input-19	15 unit	15 unit
Input-20	14 unit	16 unit
Average	16 unit	15.1 unit

Area Comparison between Dadda & Wallace Multiplier

Dadda Multiplier : No. of Half Adders =7
No. of Full Adders = 35

Wallace Multiplier : No. of Half Adders = 16
No. of Full Adders = 38

Chapter 3

Conclusion

3.1 Conclusion and Outlook

** Hence , we can see from our analysis that Dadda & Wallace both the multipliers shows same order of delay but Wallace requires more No. of Hardware to perform the same computation.

Comparison of Wallace and Dadda Multipliers

- Wallace and Dadda multipliers need the same number of layers.
- Dadda multiplier minimizes the number of adders, so it has the potential for lower complexity and power.
- Dadda multiplier uses more pass throughs and smaller adders which have lower delay. Thus it can also minimize the critical delay for reaching ` the final 2 wire stage.
- However, The final addition in Dadda multiplier needs wider carry propagating adders, which can slow it down.
- A careful evaluation inclusive of wiring and parasitic delays has to be made to determine which is the faster adder for a given configuration and process.