Design of
8 x 8 Dadda & Wallace Multiplier
(Self Project)

Done by:

Souhardya Mondal ,213070092,Department of EE, IIT Bombay

Dadda Multiplier

Dadda multipliers use 3 stages:

1. Generate all bits of the partial products in parallel.
2. 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.
3. 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.

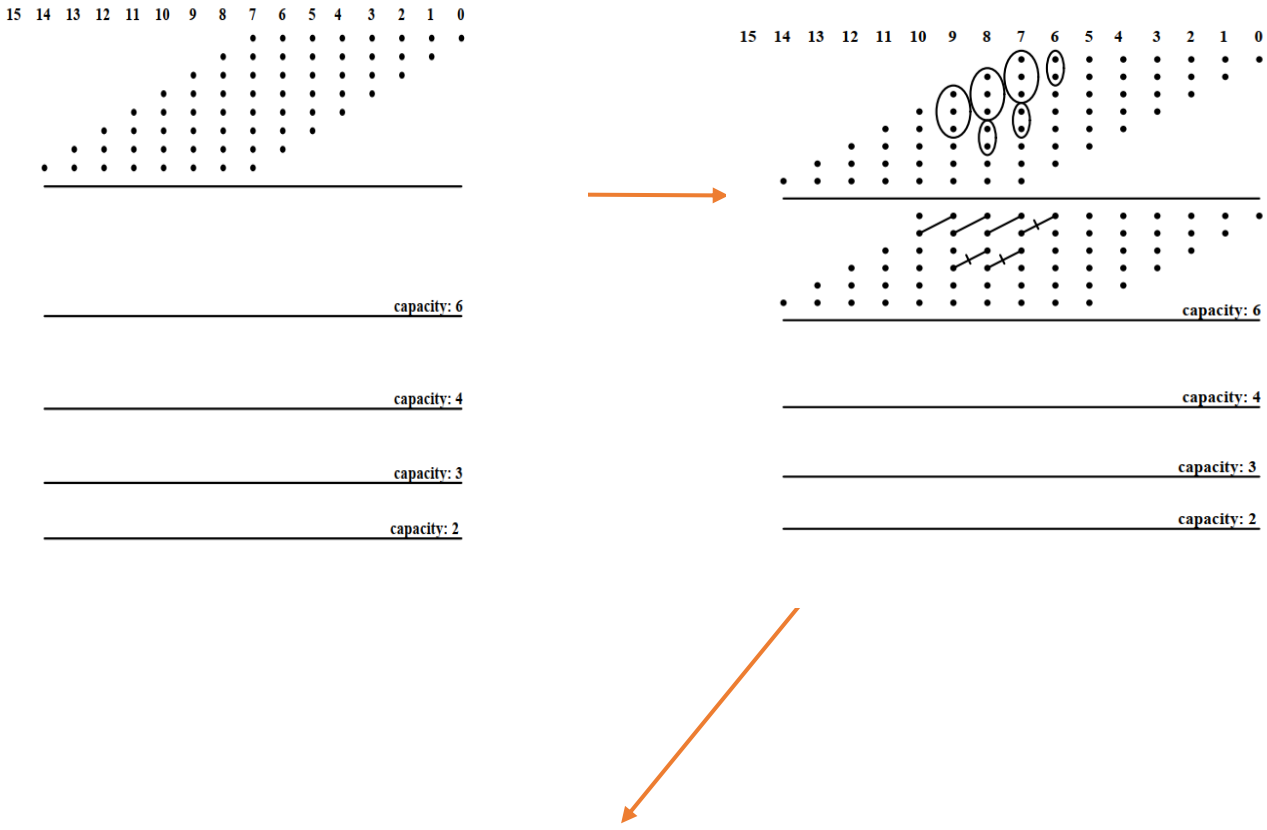
Dadda Multiplier tries to use as few adders and as small as possible (i.e prefers using HA instead of FA wherever possible). Also, it tries to reduce the number of columns as late as possible . These approaches are opposite to that of a Wallace multiplier. Dadda multipliers plan on reducing the final number of wires for any weight to 2 with as few and as small adders as possible.

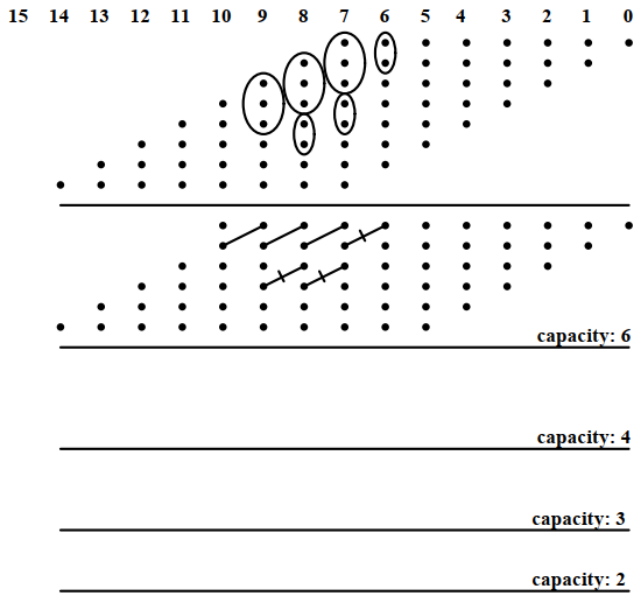
We work back from the final adder to earlier layers till we find that we can manage all wires generated by the partial product generator. The final addition will have at most 2 bits in each column. So, the capacity for the last stage should be 2.

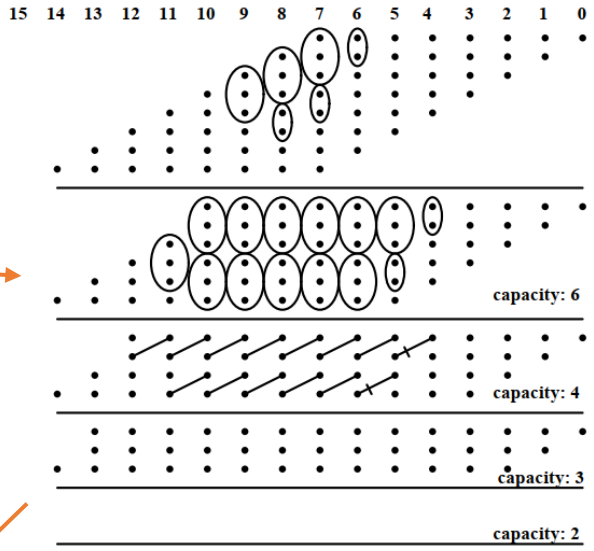
The capacity for layer i is integer of the 3/2 of the capacity of layer i-1.This implies that the capacity is 9,6,4,3,2 etc.

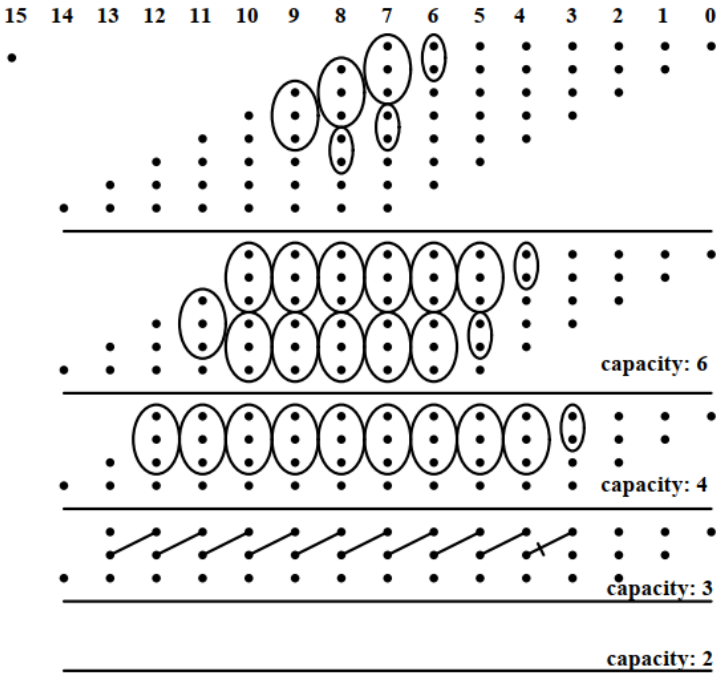
Say,We have a maximum of 8 bits in a column, then the capacity of this top layer=9. Thus, number of stages = 5 and reduction is done in 4 stages ,except the final stage where addition will be required .

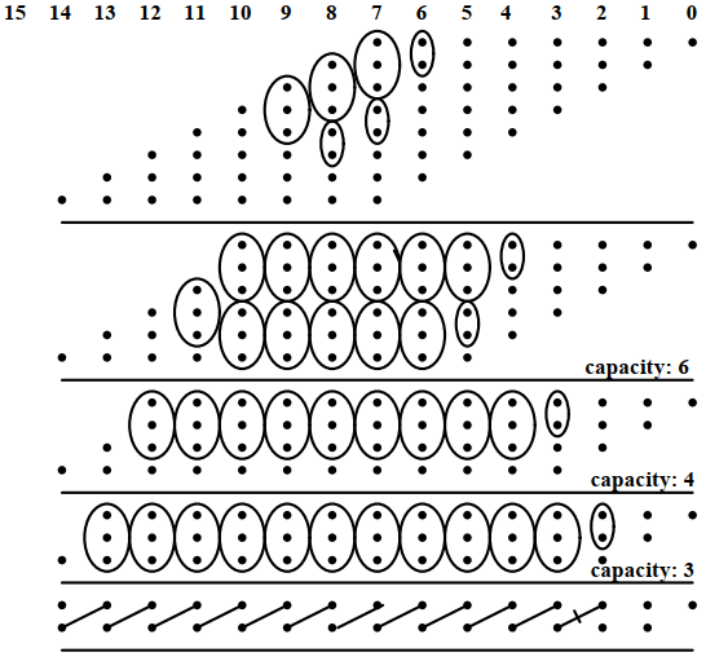
Demonstration of 8x8 multiplication:




15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
capacity: 6
capacity: 4
capacity: 3
capacity: 2


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
capacity: 6
capacity: 4
capacity: 3
capacity: 2

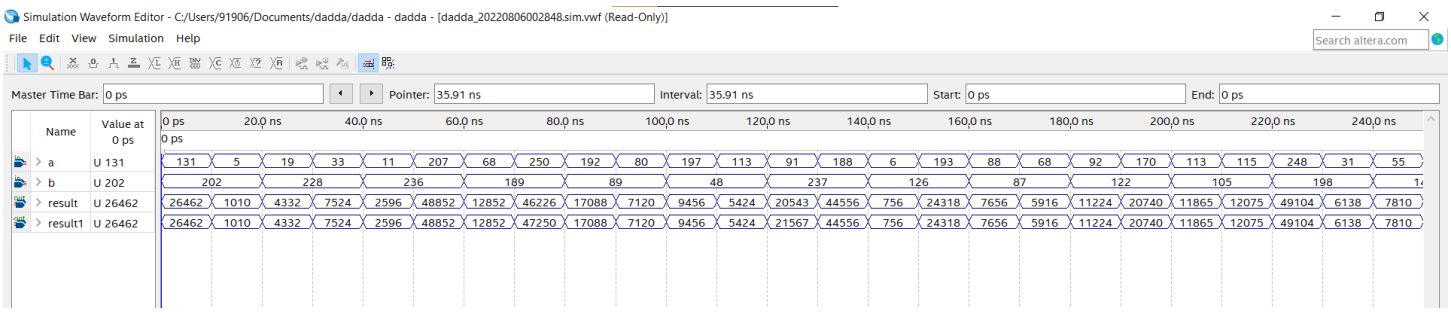

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
capacity: 6
capacity: 4
capacity: 3
capacity: 2


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
capacity: 6
capacity: 4
capacity: 3

Simulation Results of Dadda

Synthesis and simulation done in Quartus Prime

Not considering any delay here.



a → 1st 8 bit operand (shown above in decimal format)

b→ 2nd 8 bit operand (shown above in decimal format)

result→ Obtained using Dadda's Algorithm.

Result1→ Obtained by Verilog Multiplication operation.

Can be observed that result= result1 , thus correct.

Source code shown at the end of report.

Wallace Multiplier

Wallace multipliers act in three stages:

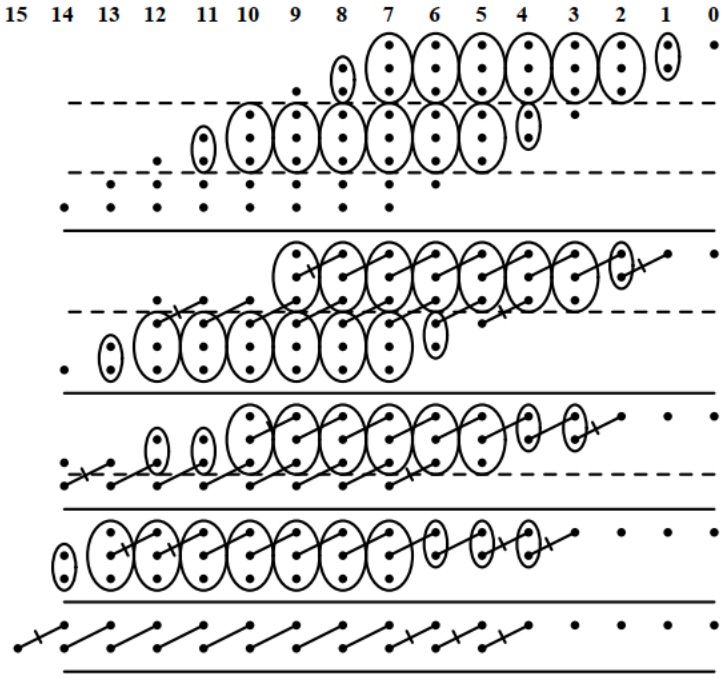
1. Generate all bits of the partial products in parallel.
2. Collect all partial products bits with the same place value in bunches of wires and reduce
these in several layers of adders till each weight has no more than two wires.
3. For all bit positions which have two wires, take one wire at corresponding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

Wire Reduction Scheme for Wallace Multipliers

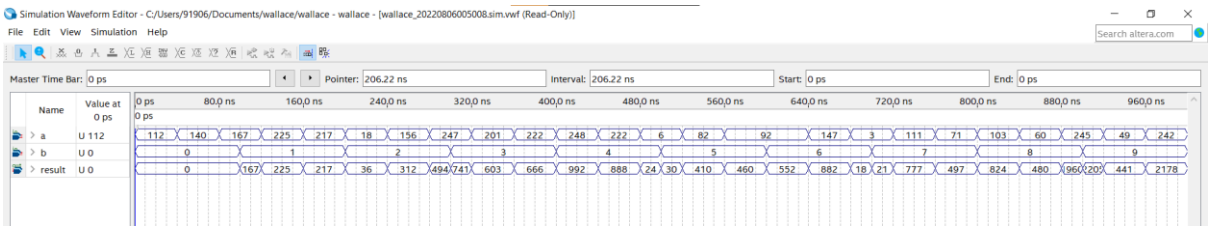
If any weight contains more than two wires, we use a reduction algorithm to reduce the number of wires. We divide the total number of rows to be added in groups of 3 rows. Any rows which are additional to these groups are passed on to the next stage as they are. Next we take groups of 3 rows one at a time. If there are 3 wires in any column of this
group, we place a full adder. The sum output of this adder goes to the same column in the next stage. The carry output of the adder goes to the column with next higher weight. If a column has two wires, it is reduced with a half adder. Again the sum output goes to the same column in the next stage and the carry wire joins the column with higher weight. If a column has a single wire, it is passed through to the next stage.
This is repeated for all groups of 3 rows.
When all groups of three rows have been reduced this way, we count wires at all weights and continue this procedure till no weight has more than two wires.
At the end, many contiguous bits at the least significant end will have a single wire. These are carried through to the result. For bits which have two wires, one is allocated to one word and the other to another word, and these two words are added using a fast adder.

Demonstration of 8x8 multiplication:



Simulation Results of Wallace

Synthesis and simulation done in Quartus Prime.
Not considering any delay here.



a → 1st 8 bit operand (shown above in decimal format)

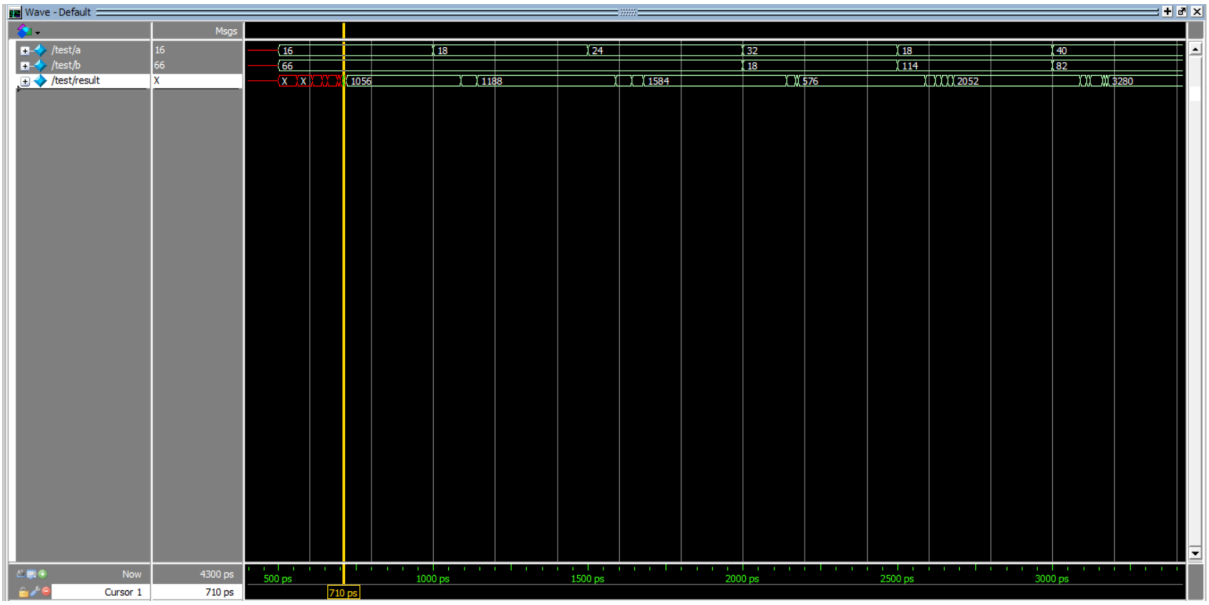
b→ 2nd 8 bit operand (shown above in decimal format)

result→ Obtained using Wallace's Algorithm.

Source code shown at the end of report.

Comparison between the Delays of Dadda & Wallace:

Wallace Multiplier testing with delays



Dadda Multiplier testing with delays

Analysis

Input A	Input B	Time of inputs	Result of Wallace	Result of Dadda	Result Obtained for Wallace	Result Obtained for Dadda
16	66	500ps	1056	1056	720ps	690ps
18	66	1000ps	1188	1188	1140ps	1140ps
32	18	2000	576	576	2180ps	2160ps
18	114	2500	2052	2052	2680ps	2600ps
40	82	3000	3280	3280	3180ps	3100ps

Conclusion: Dadda is faster than Wallace