Amar Kumar-CED17I029

16 bit Wallace Multiplier

A **Wallace tree** is an efficient hardware implementation of a digital circuit that multiplies two integers.

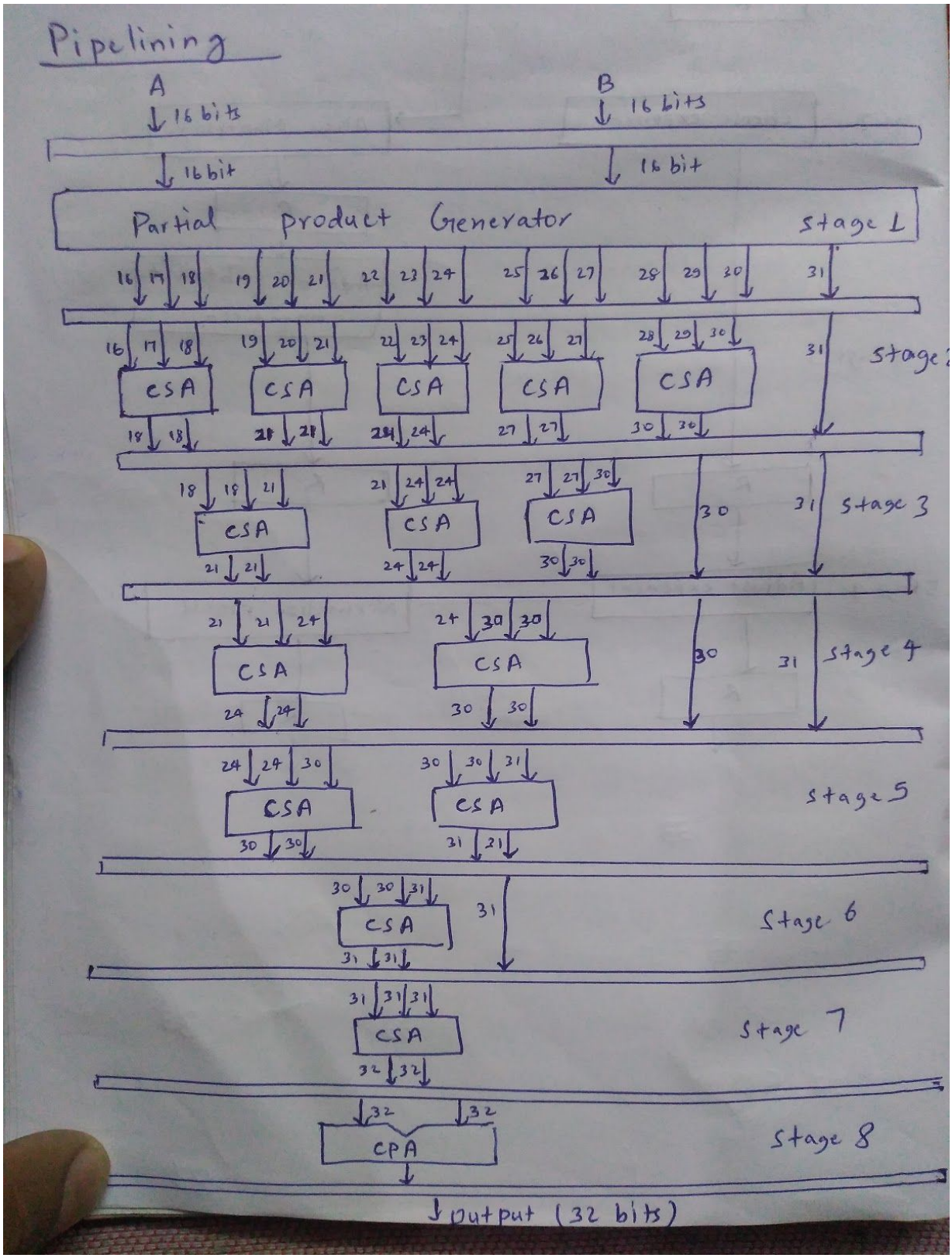
The Wallace tree has three steps:

1. Multiply (that is – AND) each bit of one of the arguments, by each bit of the other, yielding n^2 results.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Group the wires in two numbers, and add them with a conventional adder.

The second step works as follows. As long as there are three or more wires with the same weight add a following layer:-

- Take any three wires with the same weights and input them into a full adder. The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.
- If there are two wires of the same weight left, input them into a half adder.
- If there is just one wire left, connect it to the next layer.

Pipelining


Pipelining
A
16 bits
B
16 bits
16 bit
16 bit
Partial product Generator
stage 1
CSA
stage 2
stage 3
stage 4
stage 5
Stage 6
Stage 7
CPA
Stage 8
Output (32 bits)

Logic Components

(assume delay of the gates as following:- NOT-1D, AND-1D, OR-1D, XOR-3D)

Stage 1 - 16 x 16 AND gate in parallel. Delay is 1D.

Stage 2 - 70 full adders(2 XOR gates, 2 AND gates, 1 OR gate). Full adders have critical path of 2 XOR gates. So delay is 3D*2 = 6D

10 half adder(1 XOR gate, 1 AND gate) where critical path is of XOR hence delay is 3D. So, delay in this stage is 6D.

Stage 3 - 43 full adders and 7 half adders. So, delay in this stage is same as previous stage as all full adders and half adders are running in parallel.

Stage 4 - 27 full adders and 10 half adders. Delay in this stage is 6D.

Stage 5 - 30 full adders and 6 half adders. Delay in this stage is 6D.

Stage 6 - 15 full adders and 9 half adders. Delay in this stage is 6D.

Stage 7 - 15 full adders and 10 half adders. Delay in this stage is 6D.

Stage 8 - 1 delay for carry generate and propagate, then 8 groups each 4 bits.

$C_1 = G_0 + C_0P$

$C_2 = G_1 + G_0P_1 + C_0P_0P_1$

$C_3 = G_2 + G_1P_2 + G_0P_1P_2 + C_0P_0P_1P_2$

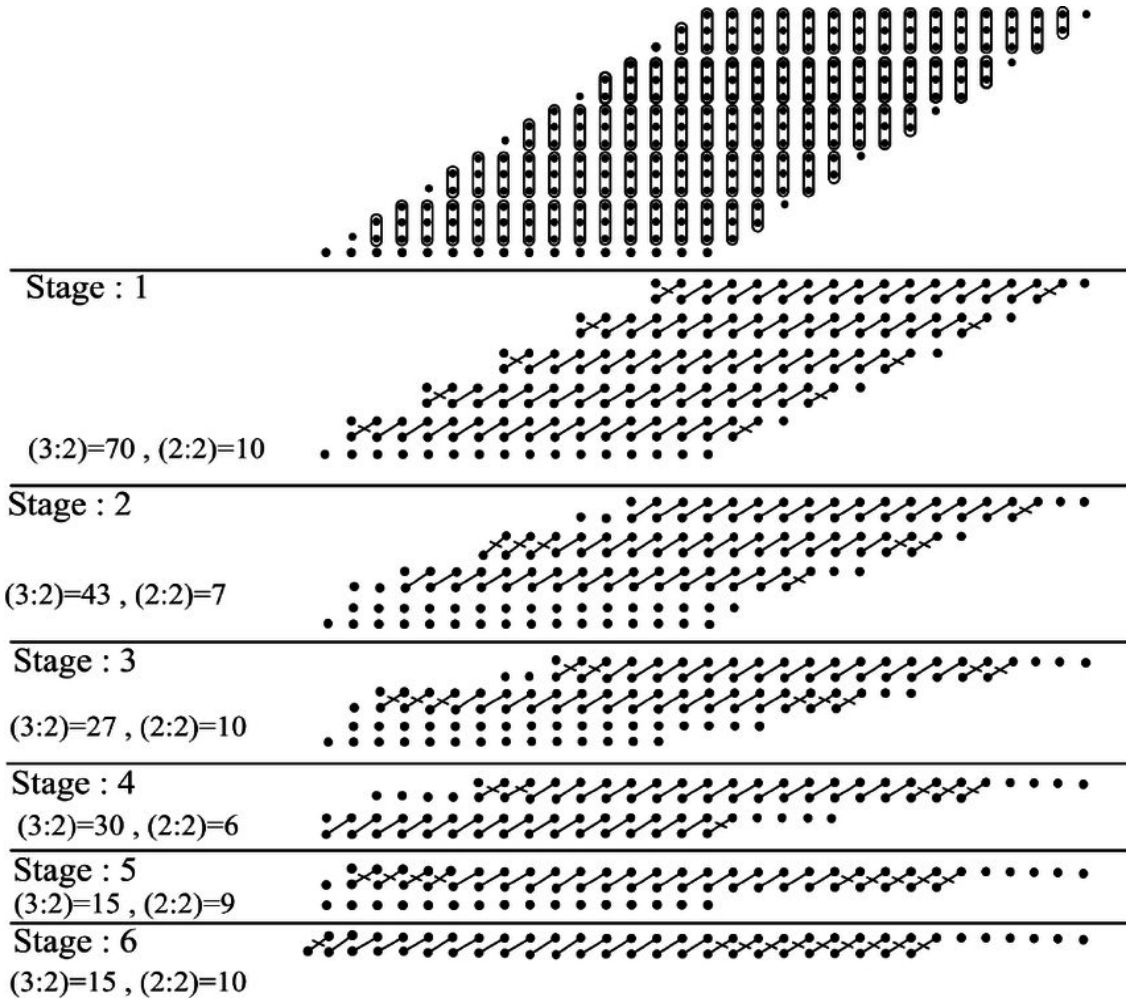
$C_4 = G_3 + G_2P_3 + G_1P_2P_3 + G_0P_1P_2P_3 + C_0P_0P_1P_2P_3$

$C_n \; as \; C_0$ for group 2 and so on.

Each group delay is 2D, for 8 group delay is 16D

AND : 10*8 = 80 OR : 4*8 = 32

Finally the sum generator uses 32 XOR gates in parallel which has a delay of 3D.Hence total delay in this stage is 21D



This figure is just for reference of number of full and half adder used

Comparison of pipelined and non-pipelined architecture

Let us assume that we have 1000 instructions

In pipelined version,
The first instruction will take delay of 58D
All other instructions will take delay of 21D.
Hence total delay for 1000 instructions will be 21D*999+58D*1 = 21037D
On average, delay is 21D*1000 = 21000D

In non-pipelined version,
All instructions will take delay 58D
So total delay for 1000 instructions will be 58000D.

% of Improvement can be achieved with respect to non-pipelined architecture.

%Improvement using pipeline = $\frac{non-pipelined}{pipelined} x\ 100$

Hence % improvement using pipelined architecture is $\frac{58000}{21000}$ x 100 ≃ 276%