Design Activity 2
Pipeline Analysis

Name: Sreepathy Jayanand
Roll No: CED17I038

Objective :

To perform stage by stage analysis of different functional units like

1. Integer adder using recursive doubling based carry look-ahead adder.

2. Integer multiplier using Wallace Tree multiplier.

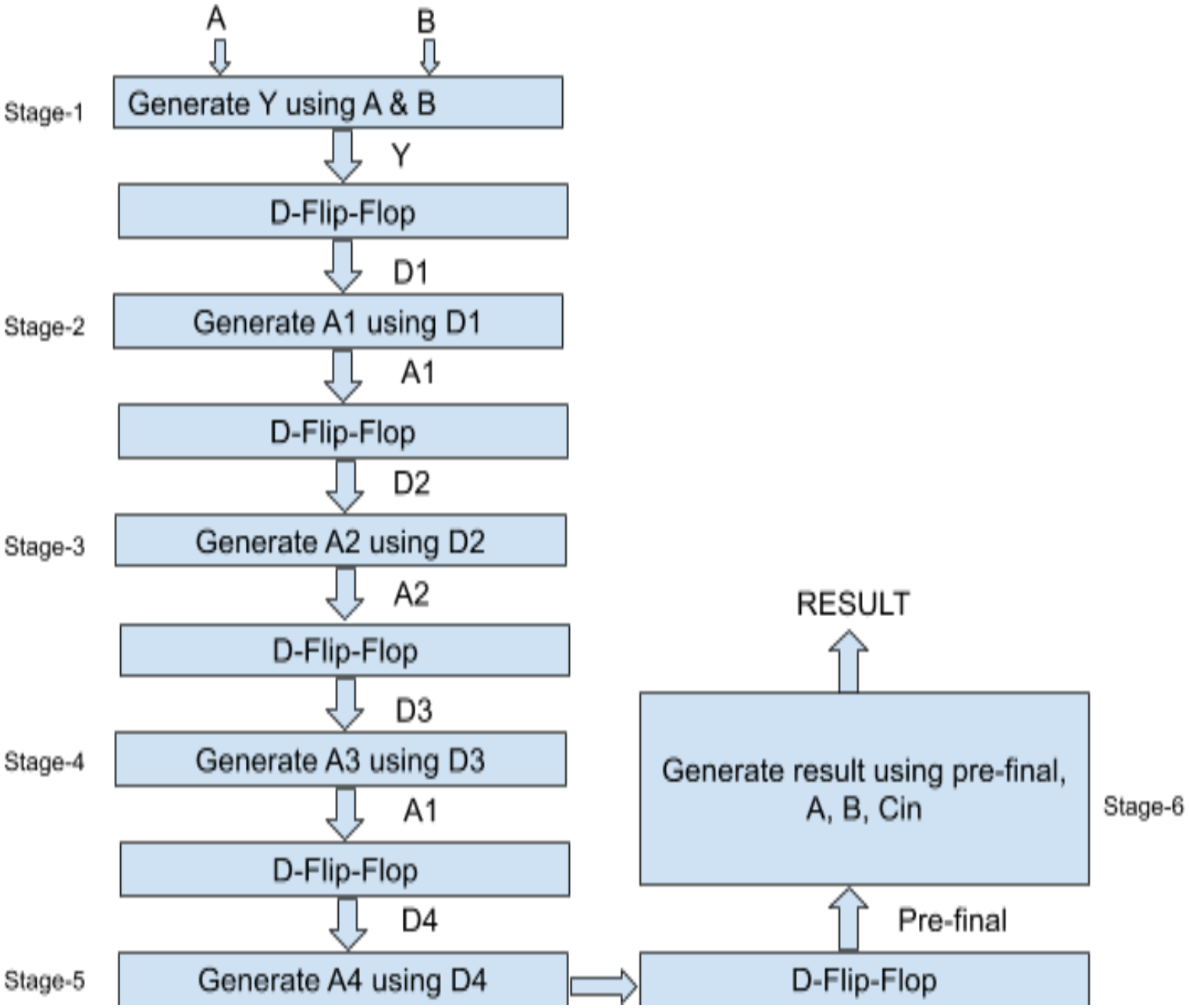
3. Half precision floating point adder.

4. Half precision multiplier

in both pipelined and non pipelined versions.

Analysis of Integer adder using recursive doubling based carry look ahead adder

Pipeline diagram of Integer adder using recursive doubling technique based carry look-ahead adder:



Analysis:

The inputs are A, B, and Cin(16-bits). This is a 6 stage pipelined functional unit.

Stage 1:

Based on the table below we decide whether it is a delete(kill), propagate or generate state.

A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

Stage 2:
Output D1 from A & B is the input based on the table. The last bit of D1 is taken the same for A1 and the rest of the bits are computed using the consecutive bits of D1.

Stage 3:
D2 is the input and A2 is the output. Similar to stage 2 the first two bits are directly copied the rest are computed using 2 bit separated values.

Stage 4:
D3 is the input and A3 is the output. Similar to stage 3 first four bits are directly copied and rest are computed using 4 bit separated values.

Stage 5:
D4 is the input and pre-final is the output. Similar to stage 4 first eight bits are directly copied and rest are computed using 8 bit separated values.

Stage 6:
The final result is computed using pre-final, A, B, and Cin with a carry in the first position.

Overview:

Stage No.	#AND	#OR	#XOR	Net Delay
1	21	19	0	3D
2	30	30	0	2D
3	28	28	0	2D
4	24	24	0	2D

5	16	16	9	2D
6	13	0	32	4D

Pipeline vs Non Pipeline Analysis:

Consider 1000 additions.

Non pipeline delay = 3D + 2D + 2D + 2D + 2D + 4D = 15D
Exec time = 15000D
Throughput = #instructions/unit time
= 1000/15000 = 0.066

For pipeline: Exec time =
Time taken for 1 task + time taken for remaining 999 tasks.
= 6 x 4 + 4 x (1000-1) = 4020D
Throughput = #instructions/unit time
= 1000/4020 = 0.24

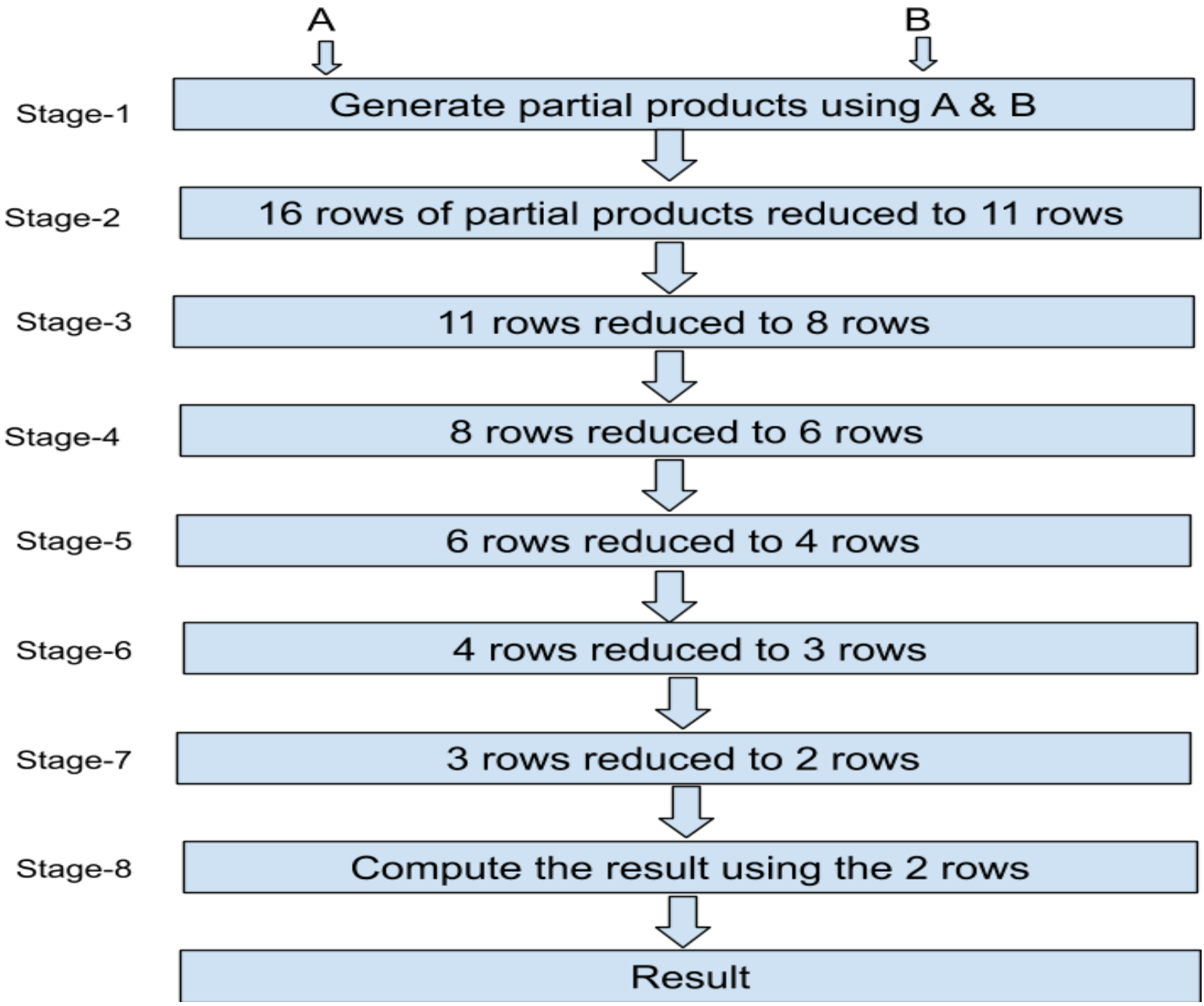
=>Pipeline exec time = 3.73 times faster than non pipelined adder.

%improvement on using pipelined version = speedup/maxspeedup = 3.73/6 = 62%

Speedup = 3.73

Analysis of Integer Wallace Tree multiplier

Pipeline diagram of Integer(16 bit) multiplier using Wallace tree multiplier:



Analysis:

The inputs are A, B(16 bits). The pipeline has 8 stages. We use carry save adders in stages 1-7 and recursive doubling carry look ahead adder in stage 8.

Stage 1:

Each bit of input A is anded with each input bit of B and partial products are produced.

Stage 2:

The 16 rows of partial products are grouped into 5 groups of 3 each and 1 separate row.
The 3 rows in each group are added to convert it into 2 rows without changing the 1 separate row.
So a total of 11 rows.

Stage 3:
The 11 rows from stage 2 are grouped into 3 groups of 3 rows and 1 group with 2. The 3 groups of 3 are converted to 3 groups of 2 without changing the 1 group of 2. So a total of 8 rows.

Stage 4:
The 8 rows from stage 3 are grouped into 2 groups of 3 rows and 1 group with 2. The 2 groups of 3 are converted to 2 groups of 2 without changing the 1 group of 2. So a total of 6 rows.

Stage 5:
The 6 rows from stage 4 are grouped into 2 groups of 3 rows and converted into 2 groups of 2 rows after the addition. So a total of 4 rows.

Stage 6:
The 4 rows from stage 5 are grouped into 1 group of 3 and one group of 1. The one group with 3 is added to make 2 rows. So a total of 3 rows.

Stage 7:
Now the 3 rows are added to make 2 rows.

Stage 8:
The 2 rows are added to get the product of A and B.

Overview:

Stage No.	#AND gates	#Full-Adder	#Half-adder	Delay
1	64	0	0	1D
2	0	71	10	4D
3	0	43	7	4D
4	0	27	10	4D
5	0	28	9	4D
6	0	15	9	4D
7	0	15	10	4D
---	---	---	---	---

In stage 8 we use 16 bit recursive doubling based CLA - delay of 15D

Pipeline and Non pipeline analysis:

Assumption: All the basic gates have a delay of 1D.

Consider 1000 multiplications.

Non pipelined delay = 1D + 4D + 4D + 4D + 4D + 4D + 4D + 15D = 40D

Exec time = 40000D

Throughput = 1000/40000 = 0.025

Pipeline Delay =

Time taken for 1 task + time taken for remaining 999 tasks.

8 x 15 + 15 x (1000-1) = 15105D

Throughput = 1000/15105 = 0.066

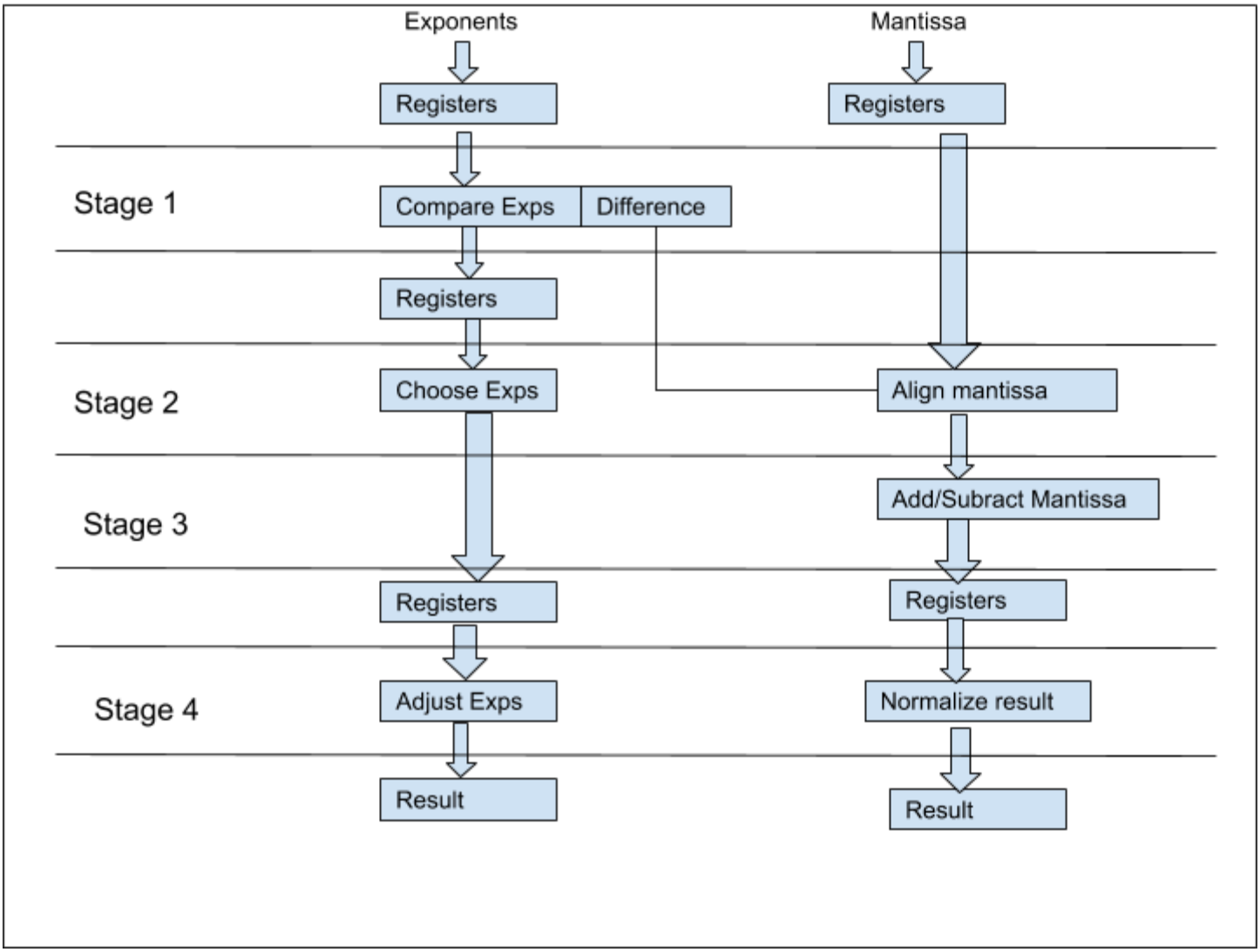
%improvement on using pipelined version = speedup/maxspeedup = 2.64/8 = 33%

=> Pipeline exec time = 2.64 times faster than non pipelined multiplier.

Speedup = 2.64

Analysis of half-precision floating point adder

Pipeline diagram of Floating point adder:



Analysis:

The number of pipeline stages = 4

Stage 1:

We compare the exponents to get the difference between them and are given to the next stage so that we can align mantissa to add the mantissas.

Stage 2:

We choose the larger exponent to which we align and the mantissas are aligned.

Stage 3:

We add/subtract the mantissa based on sign values.

Stage 4:

We adjust the exponents and shift the mantissa to get leading bit 1 to get the result.

Overview:

Stage 1:
5 bit ripple carry adder to subtract the exponents, delay = 12D

Stage 2:
Shifter for mantissa,Eff. Delay = 15D

Stage 3:
11 bit ripple carry adder for the mantissa,Eff. Delay = 24D

Stage 4:
5 bit ripple carry adder and a shifter,Eff. Delay = 18D

Pipeline and Non pipeline analysis:

Assumption: All the basic gates have a delay of 1D.
Consider 1000 floating point additions.

Non pipelined delay = 12D + 15D + 24D + 18D = 69D
Exec time = 69000D
Throughput = 1000/69000 = 0.014

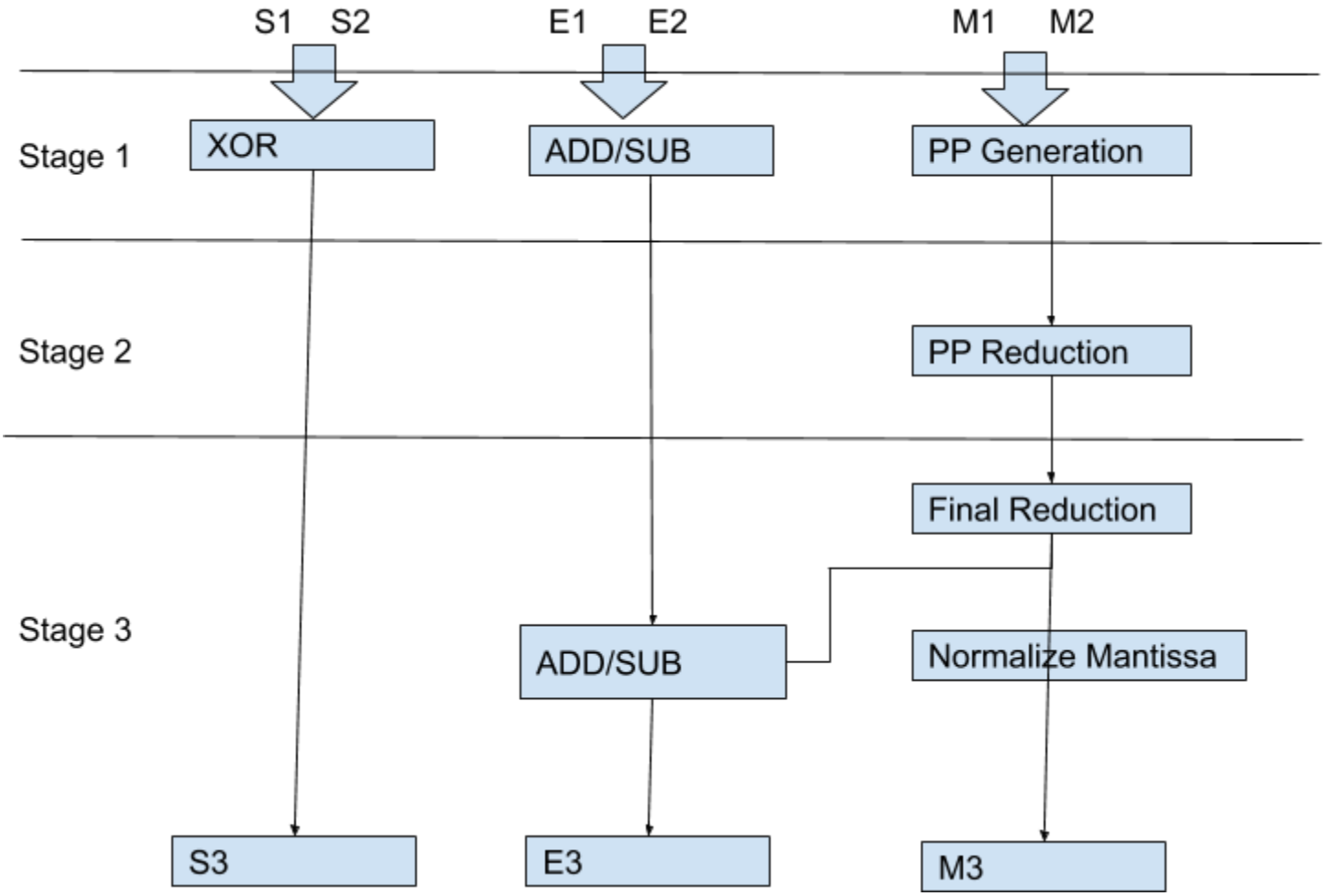
Pipeline Delay =
Time taken for 1 task + time taken for remaining 999 tasks.
4 x 24 + 24 x (999-1) = 24072D
Throughput = 1000/24072 = 0.04

%improvement on using pipelined version = speedup/maxspeedup = 2.86/4 = 71%

=> Pipeline exec time = 2.86 times faster than non pipelined multiplier.
Speedup = 2.86

Analysis of half-precision floating point multiplier

Pipeline diagram of half precision multiplier:



Analysis:

Stage 1:
The sign of the final result is computed using xor operation, the exponents are added or subtracted based on their sign values and the partial products of the mantissas are computed using the wallace multiplier
Stage 2:
The partial product in the first stage is reduced to a 22 bit product.
Stage 3:
The final reduction is done by choosing the first 11 bits out of the 22 bits. The result is normalized by finding the position of the most significant bit and the exponent is incremented/decremented accordingly.

Overview:

Stage 1:

We use 22 and gates, 1 xor and a 5 bit ripple carry adder.Eff. Delay = 12D

Stage 2:

We use a carry save adder and a 22 bit ripple carry adder.Eff. Delay = 46D

Stage 3:

We use a 5 bit ripple carry adder.Eff. Delay = 18D

Pipeline and Non pipeline analysis:

Assumption: All the basic gates have a delay of 1D.

Consider 1000 floating point additions.

Non-pipelined delay = 12D + 46D + 18D = 76D

Total delay = 76000D

Throughput = 1000/76000 = 0.013

Pipelined Delay = Time for 1 task + time for remaining 999 tasks.

= 46 x 3 + 46 x (1000-1) = 46092D

Throughput = 1000/46092 = 0.021

%improvement on using pipelined version = 1.64/3 = 54%

=> Pipeline execution time = 1.64 times faster than non pipelined execution time

Speedup = 1.64