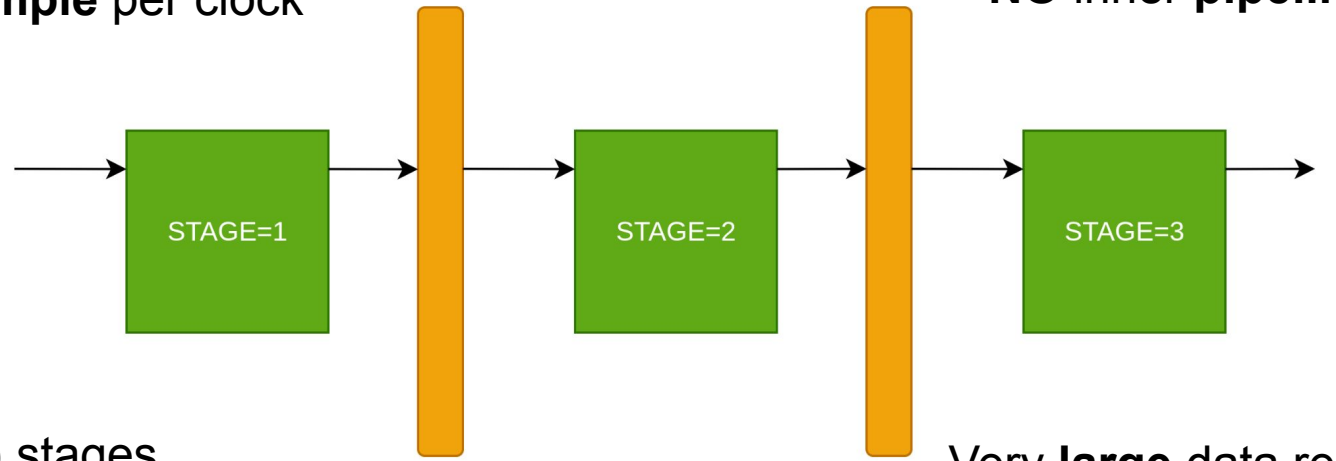
Scalable FFT processor implementation on FPGA

Christodoulos Zerdalis
03531

Previous Radix-2 SDF implementation

ONE sample per clock

NO inner **pipelining**



log2(N) stages

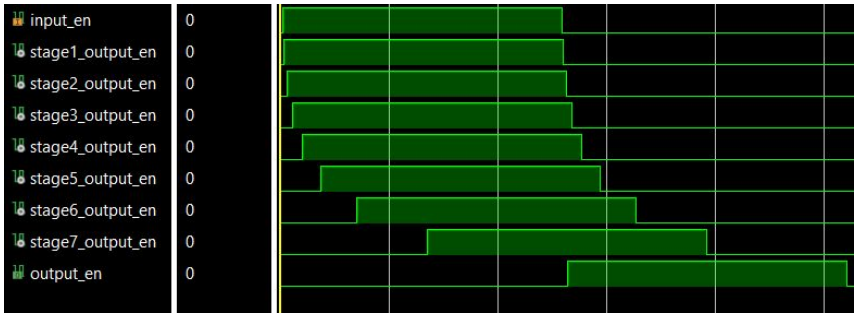
Very **large** data resolution **32bit**

Previous Radix-2 SDF implementation problems

1. For a **N** point FFT, **2N - 2 + logN** cycles needed to complete **computation**

$$N + 1 + \sum_{i=1}^{log_2 N - 1} (2^i + 1)$$

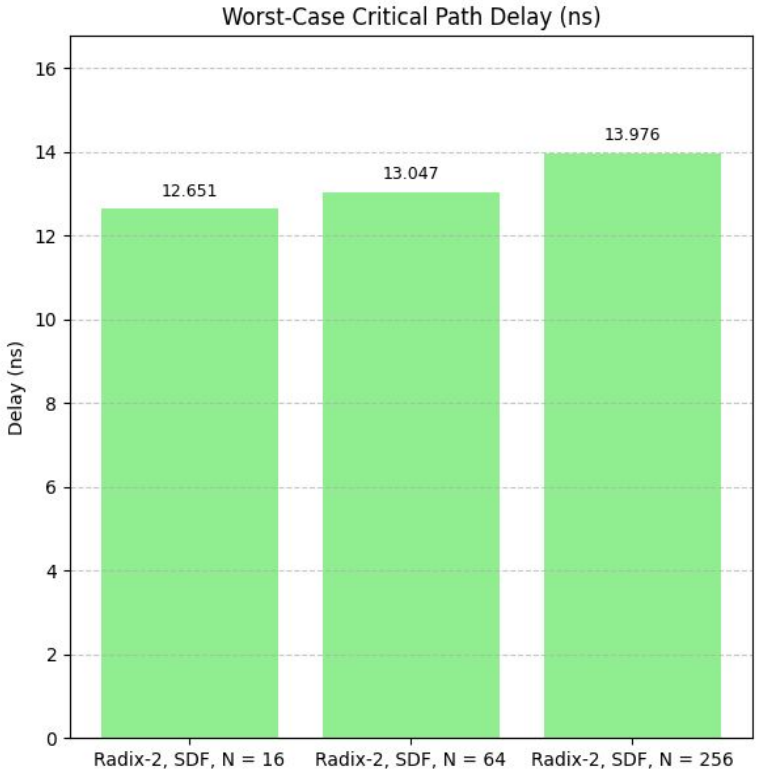
SLOW



Previous Radix-2 SDF implementation problems

2. **Big** logic delays, **inability** to run on **high frequencies**

SLOW

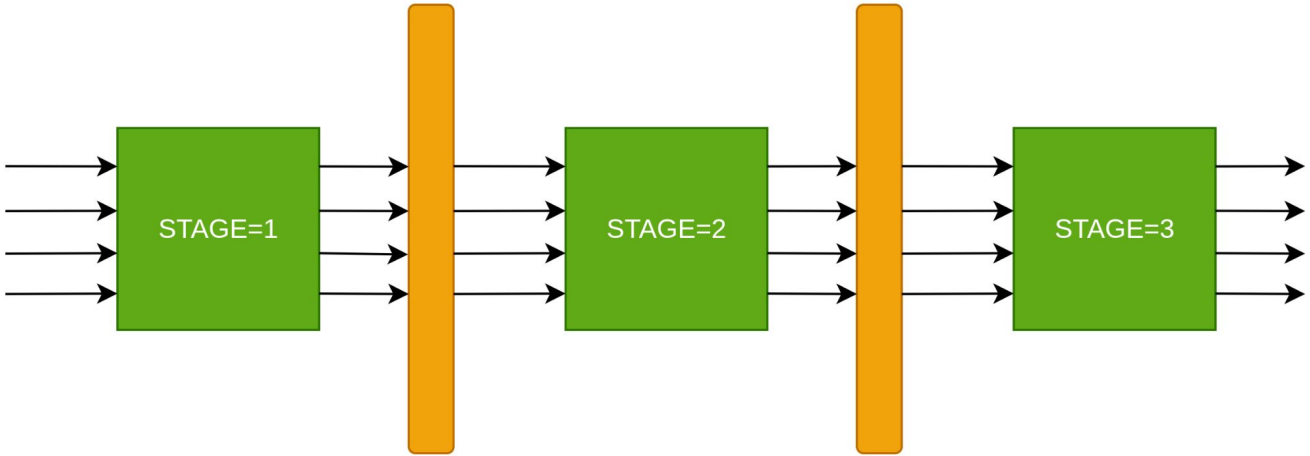


Improvements

1. Use a **radix-4** unit instead of a radix-2, **fewer stages**
2. Process **4 samples** rep clock **more bandwidth**
3. **Pipeline** the **butterfly** unit responsible for the main computations
4. **Data Width** optimization

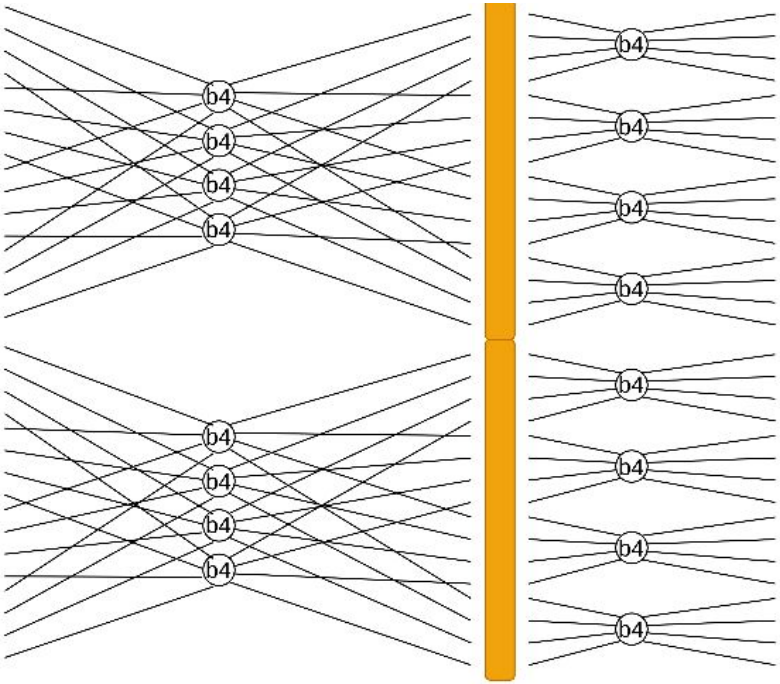
Radix-4 implementation

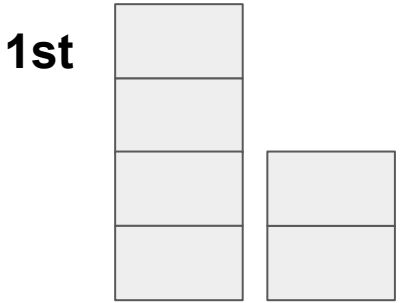
Instead of processing **1 sample** pre clock we processes **4 samples**

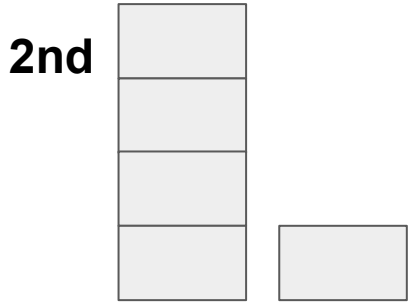


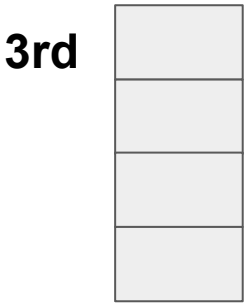
This optimization increases the **complexity** of how we **store** the incoming data and time their **delay** in every stage

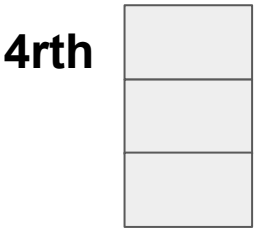
New buffers / Delays (N=16 example)



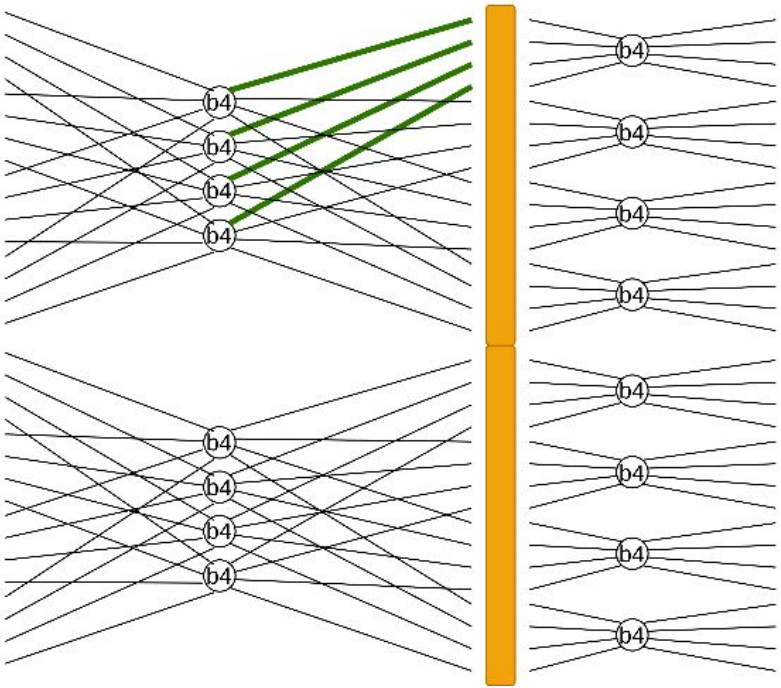


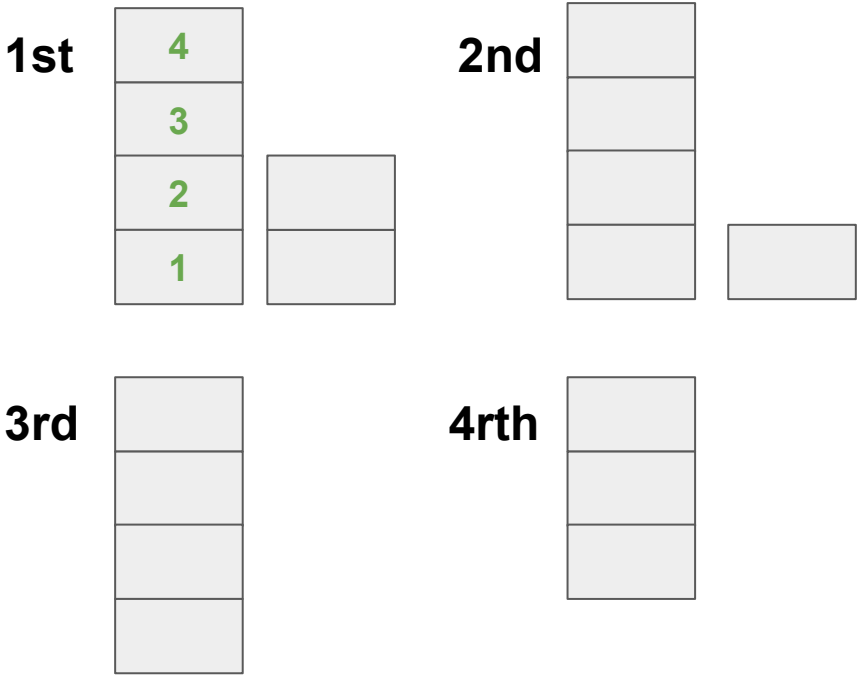




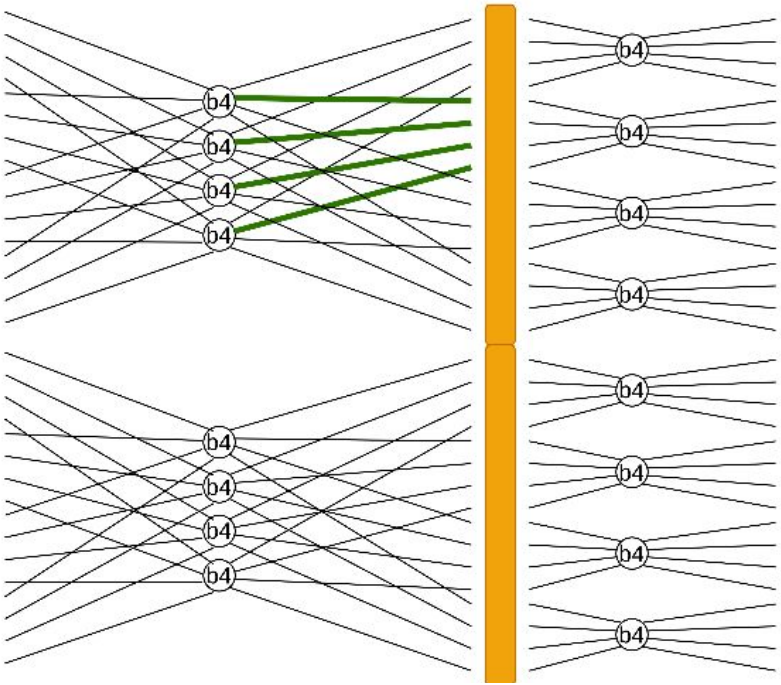


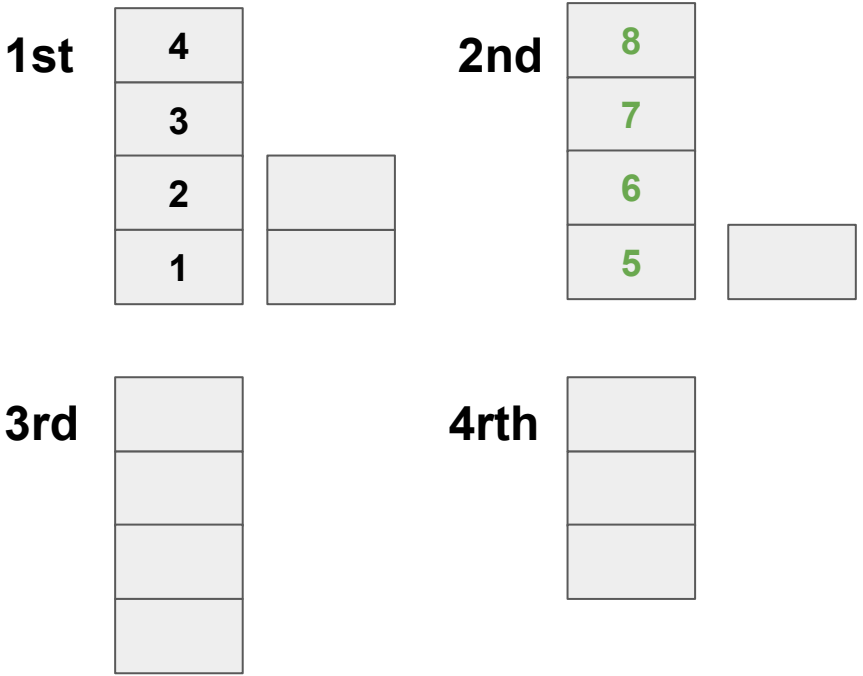
New buffers / Delays (N=16 example)



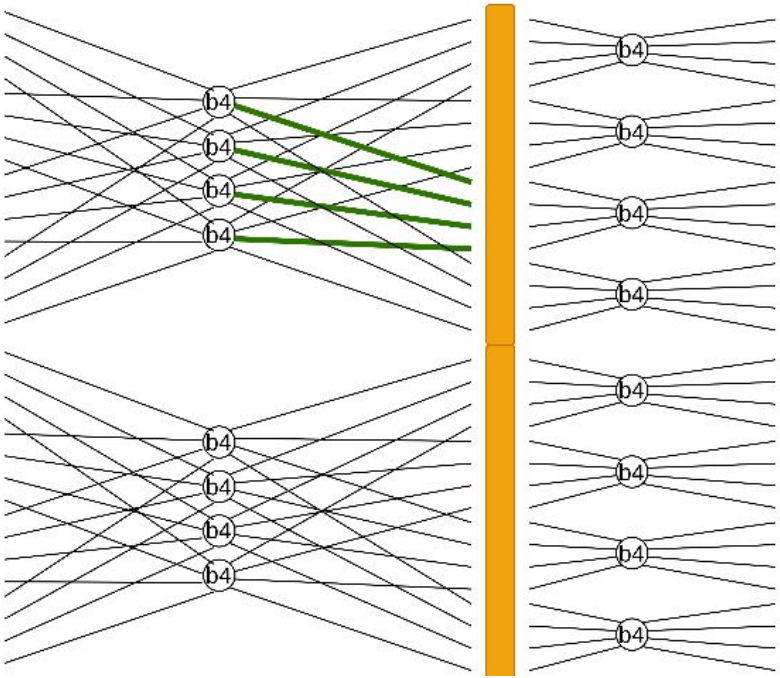


New buffers / Delays (N=16 example)

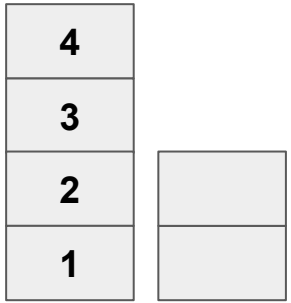




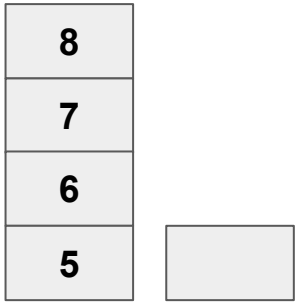
New buffers / Delays (N=16 example)



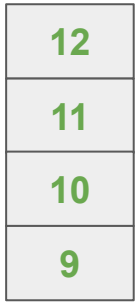
1st



2nd

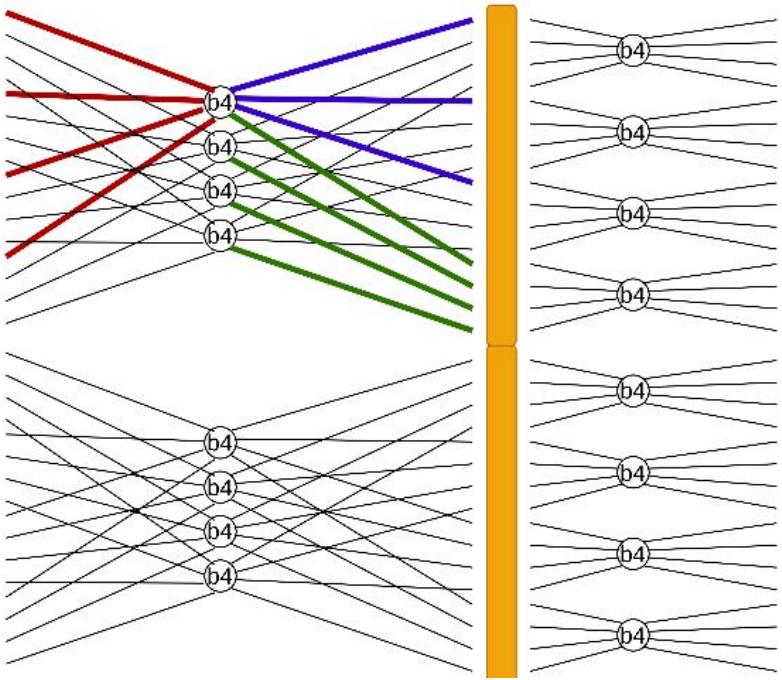


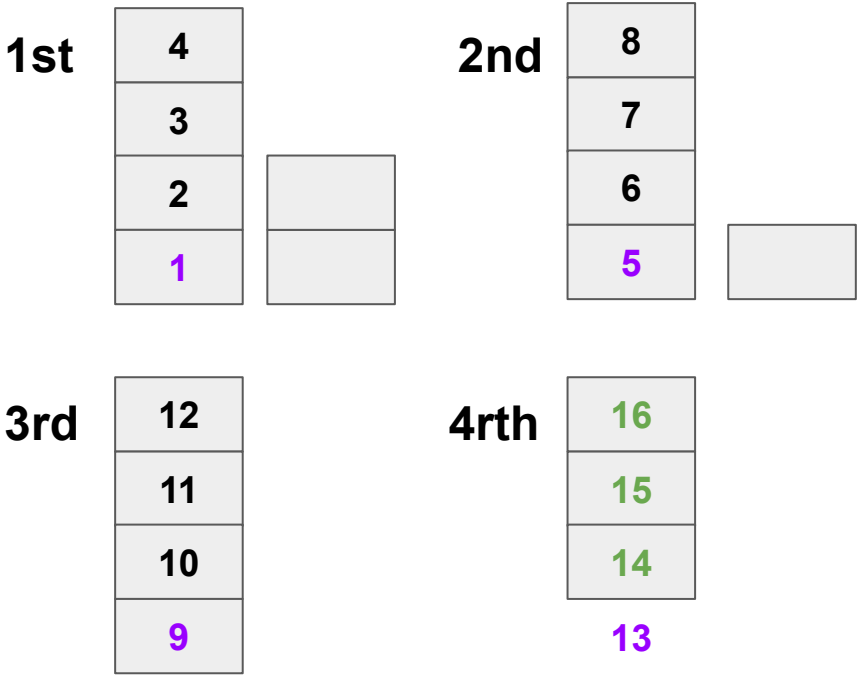
3rd



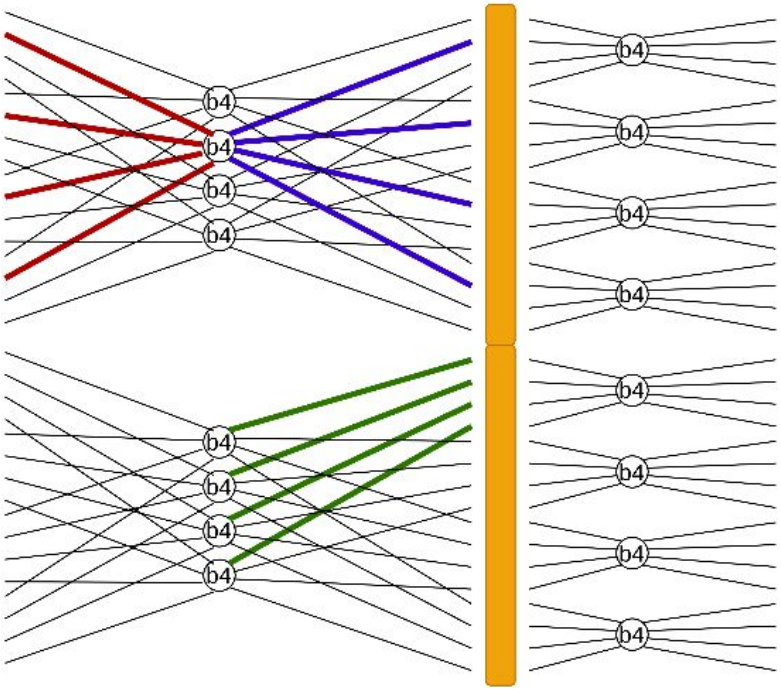
4rth

New buffers / Delays (N=16 example)





New buffers / Delays (N=16 example)



1st

20	
4<-19	
3	18
2	17

2nd

8	
7	
6	

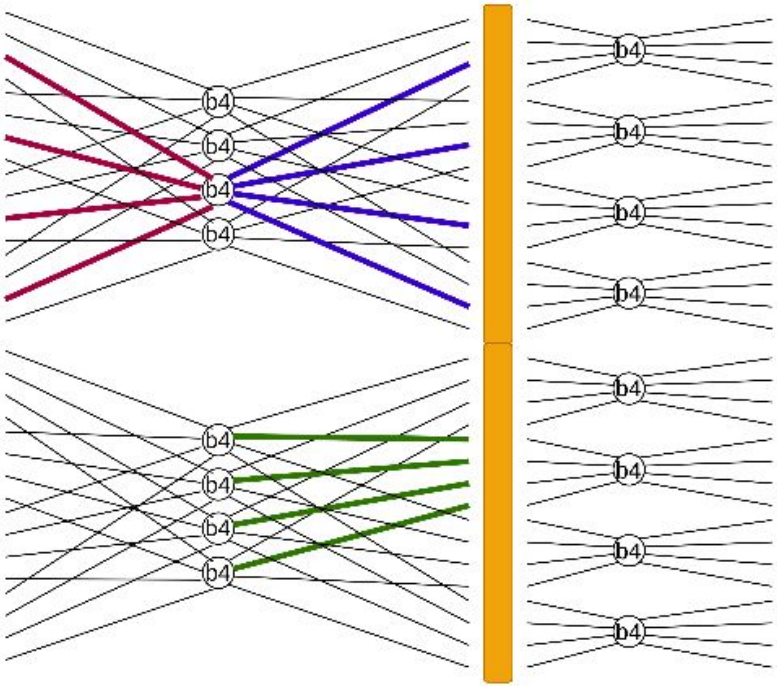
3rd

12
11
10

4rth

16
15
14

New buffers / Delays (N=16 example)



1st

20	
19	
4	18
3	17

2nd

24	
23	
8<-22	
7	21

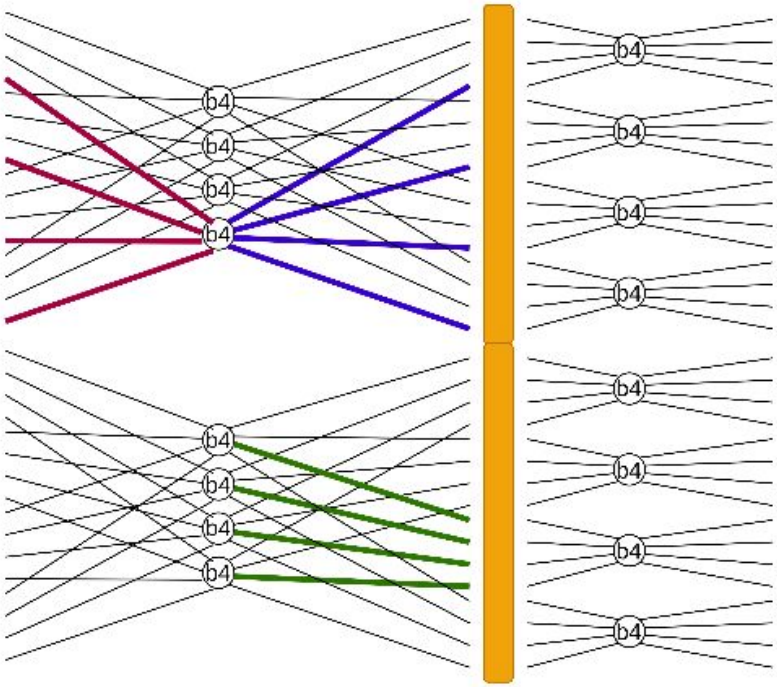
3rd

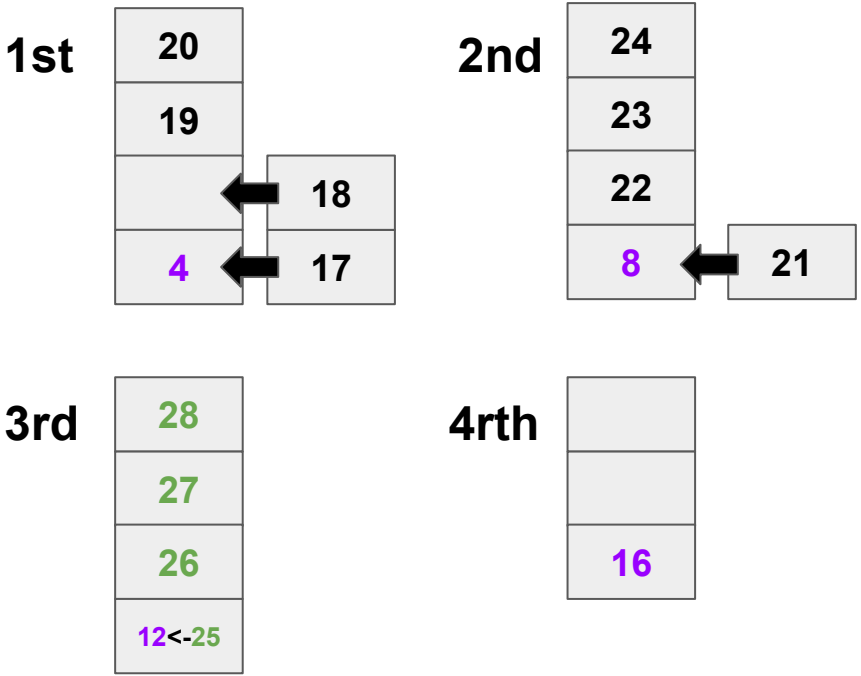
12
11

4rth

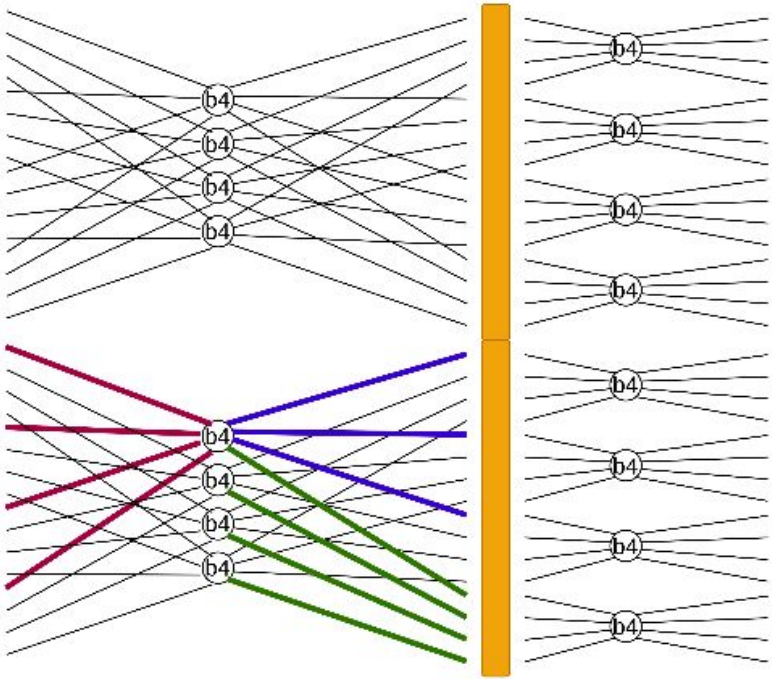
16
15

New buffers / Delays (N=16 example)





New buffers / Delays (N=16 example)



1st

20
19
18
17

2nd

24
23
22
21

3rd

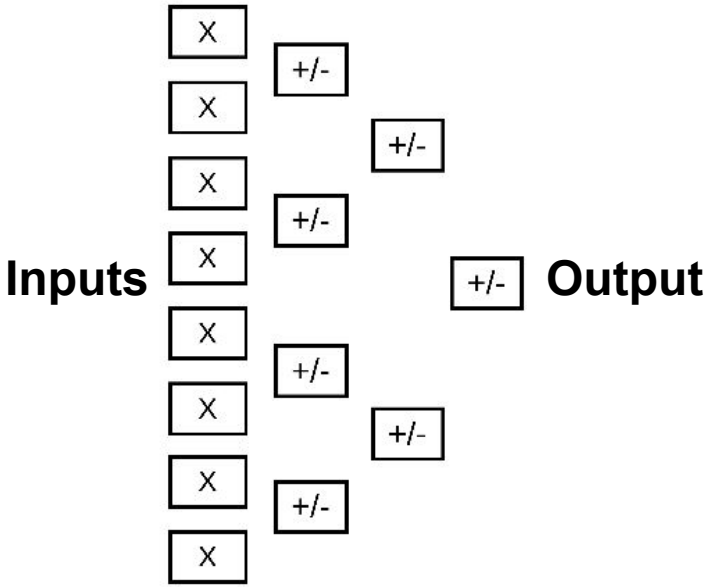
28
27
26
25

4rth

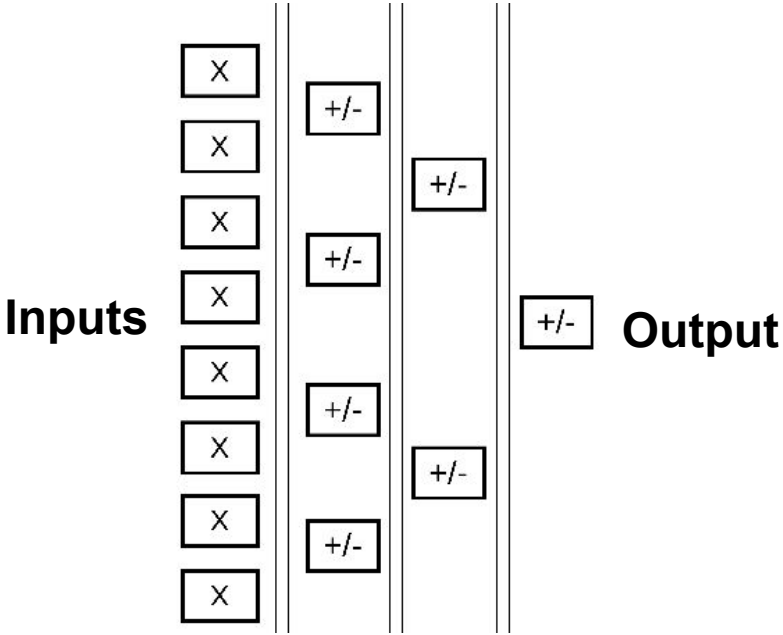
32
31
30
29

Pipelining the butterfly unit

No pipeline dataflow



With pipelining



Results before and after

Name	Slack ^1	Levels	Routes	Hig...	Fr...	To	Total Delay	Logic Delay	Net Delay	Req...
Path 1	-6.780	14	8	66	...	g...	16.602	9.421	7.181	10.0
Path 2	-6.758	17	8	66	...	g...	16.770	9.716	7.054	10.0
Path 3	-6.746	15	8	66	...	g...	16.755	9.444	7.311	10.0
Path 4	-6.741	15	8	66	...	g...D	16.644	9.796	6.848	10.0
Path 5	-6.734	14	8	66	...	g...	16.618	8.852	7.766	10.0
Path 6	-6.724	17	8	66	...	g...	16.546	9.601	6.945	10.0
Path 7	-6.715	17	8	66	...	g...	16.727	9.709	7.018	10.0

Name	Slack	Lev...	Ro...	Hig...	... ^1	...	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	3.580	0	1	328	g...C	...	5.714	0.419	5.295	10.0
Path 2	3.580	0	1	328	g...C	...	5.714	0.419	5.295	10.0
Path 3	3.580	0	1	328	g...C	...	5.714	0.419	5.295	10.0
Path 4	3.580	0	1	328	g...C	...	5.714	0.419	5.295	10.0
Path 9	3.864	0	1	328	g...C	...	5.431	0.419	5.012	10.0
Path 10	3.864	0	1	328	g...C	...	5.431	0.419	5.012	10.0
Path 5	3.834	0	1	327	g...C	...	5.286	0.419	4.867	10.0

DSP optimization

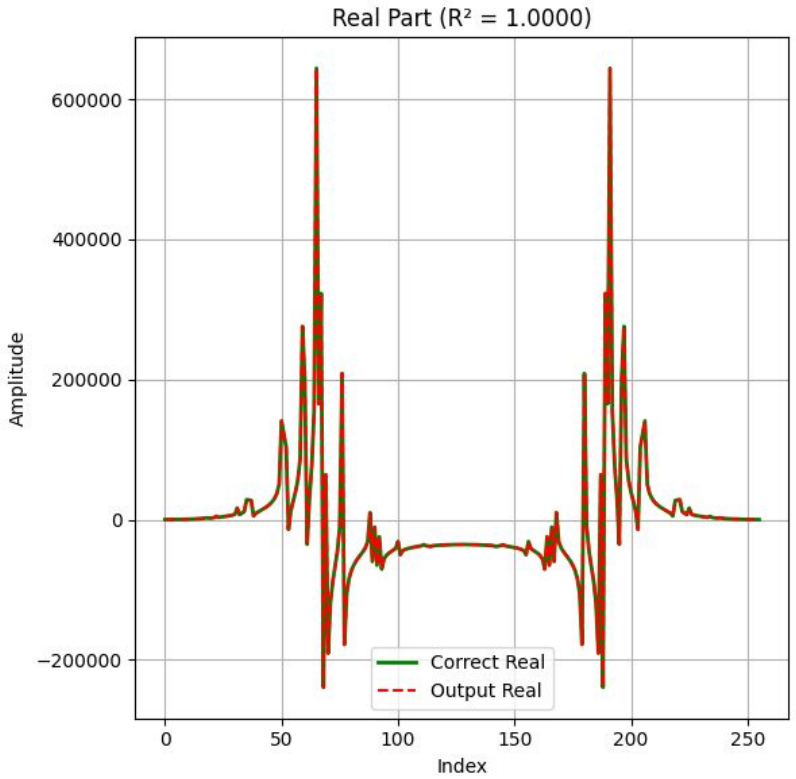
The DSPs in zedboard (DSP48E1) can handle up to 25x18 bit multiplication. This means that for data sizes bigger that this we need two DSPs in series and that increases the delay of the critical paths.

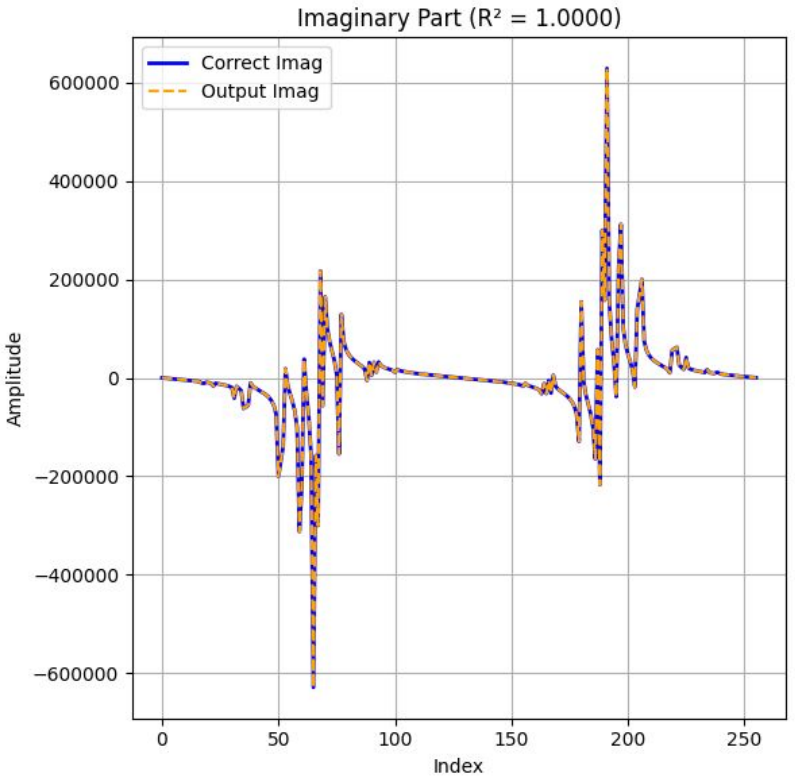
Data size optimization

4 different sizes

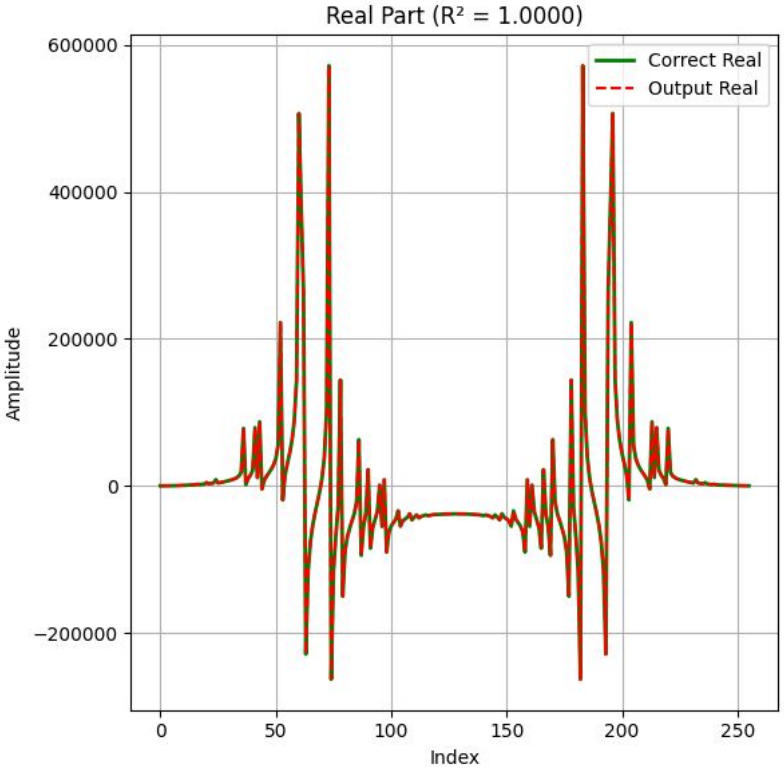
- 32 bit with 16 bit twiddle (16 bit from -1 to 1)
- 24 bit with 12 bit twiddle (optimal for our DSPs)
- 18 bit with 9 bit twiddle
- 16 bit with 8 bit twiddle

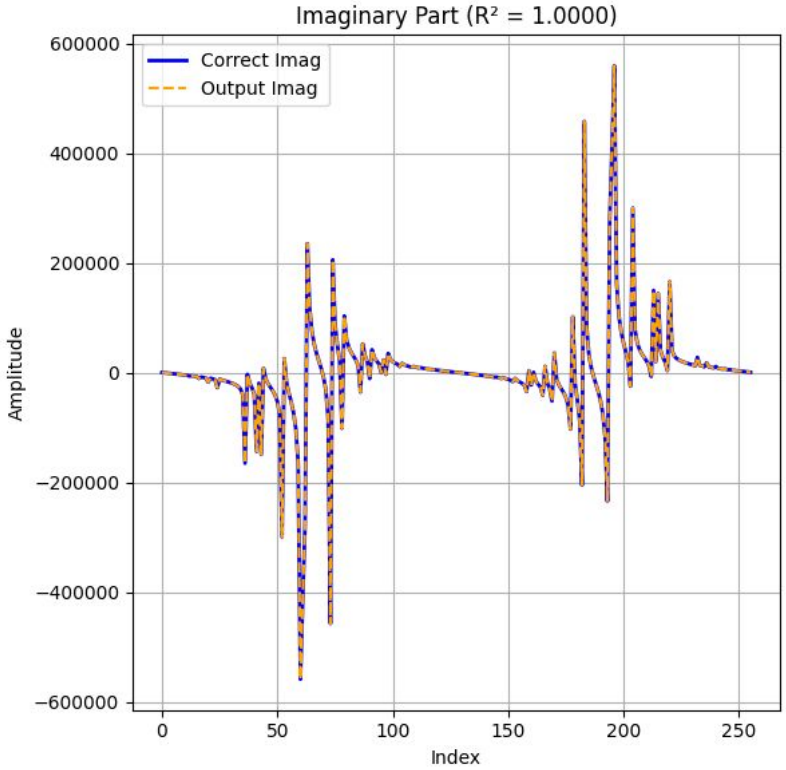
Results 32 bit



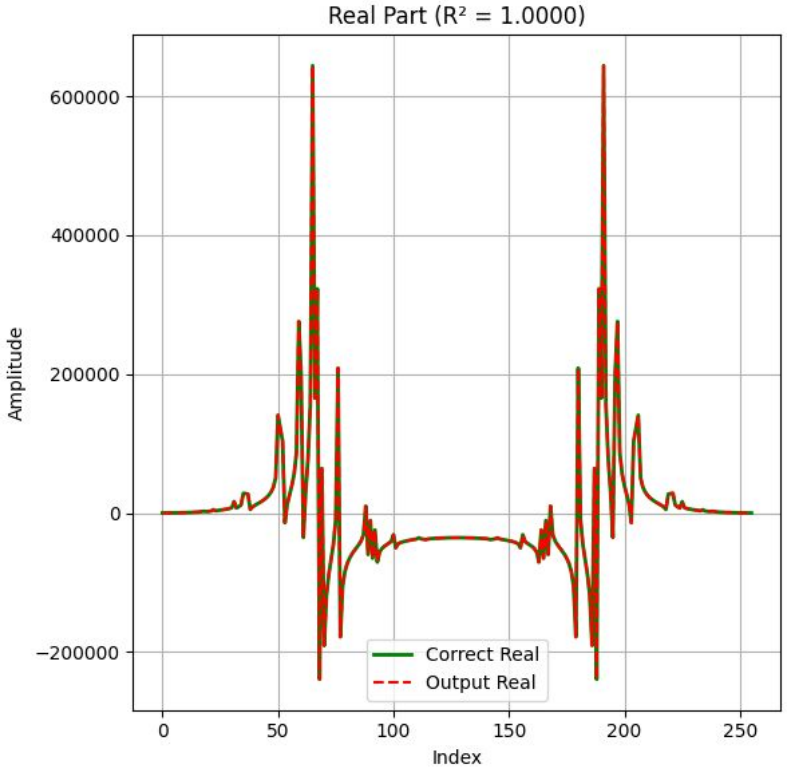


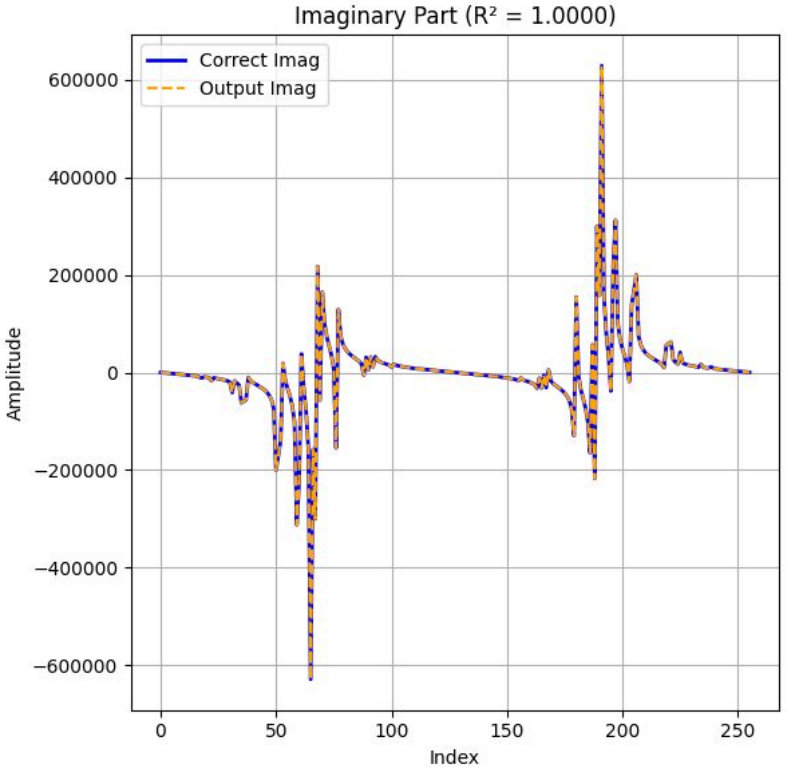
Results 32 bit



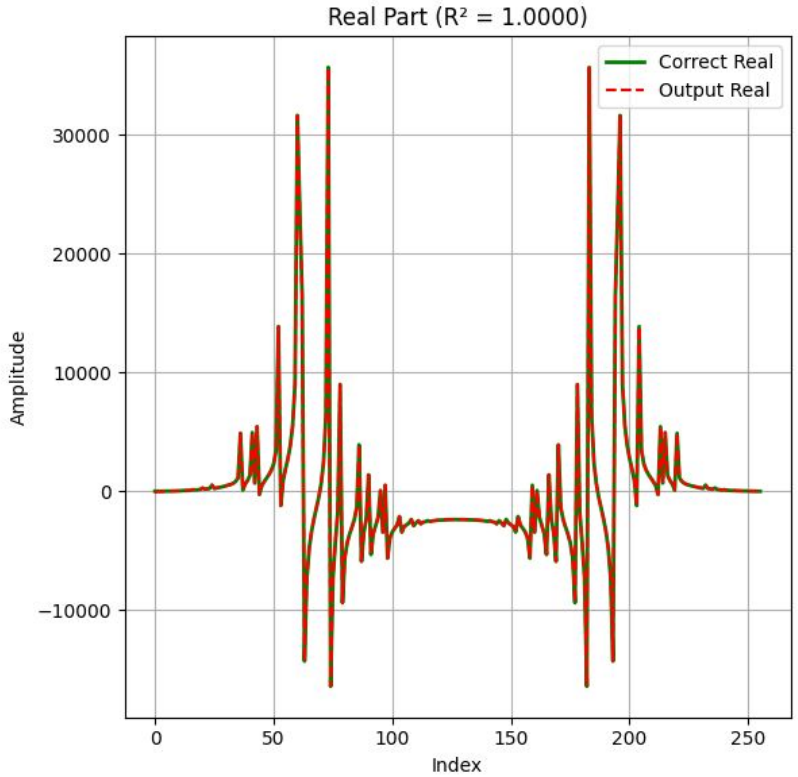


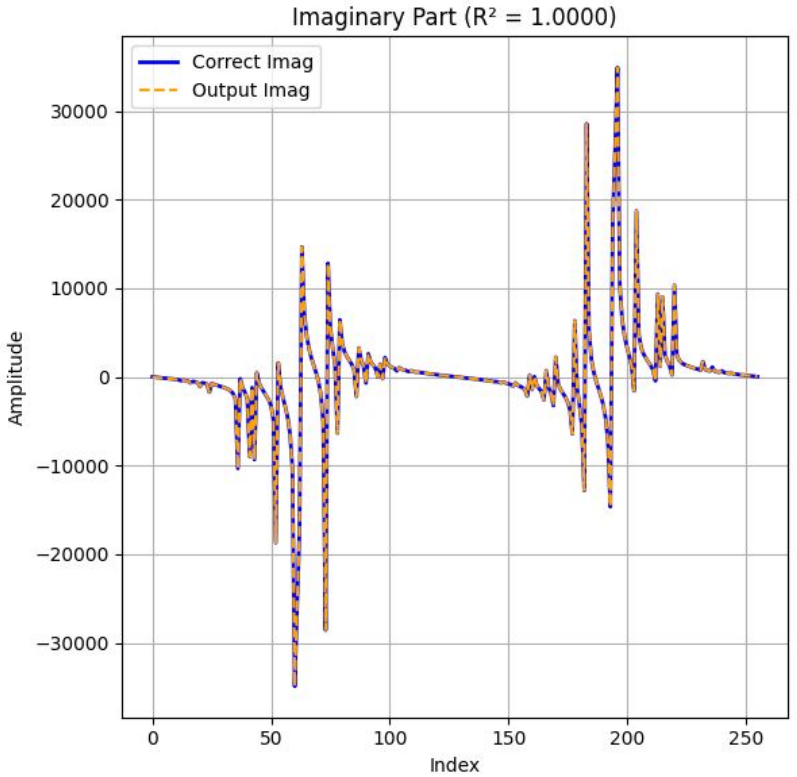
Results 24 bit



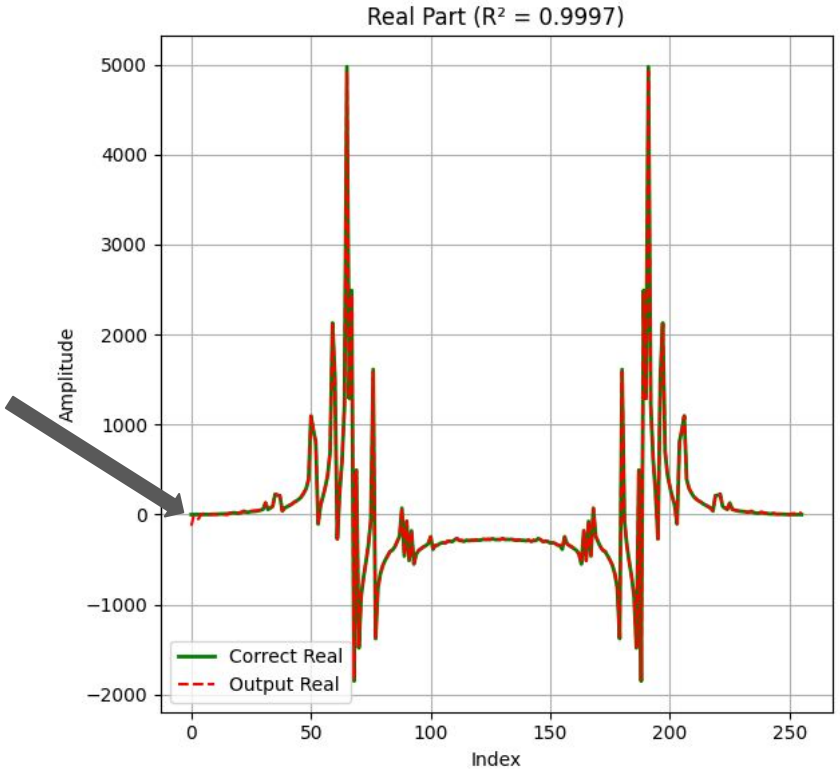


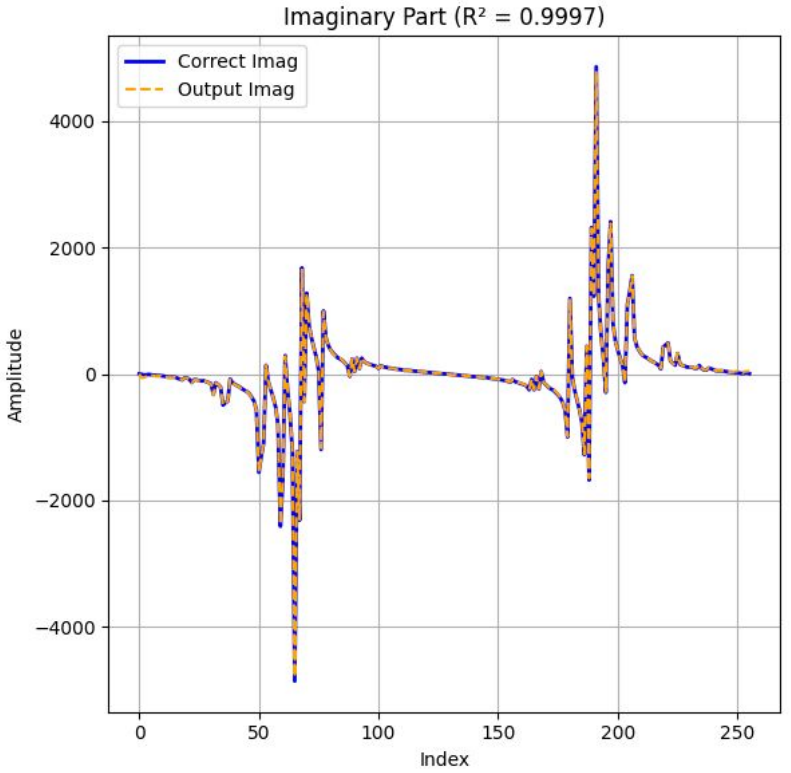
Results 24 bit



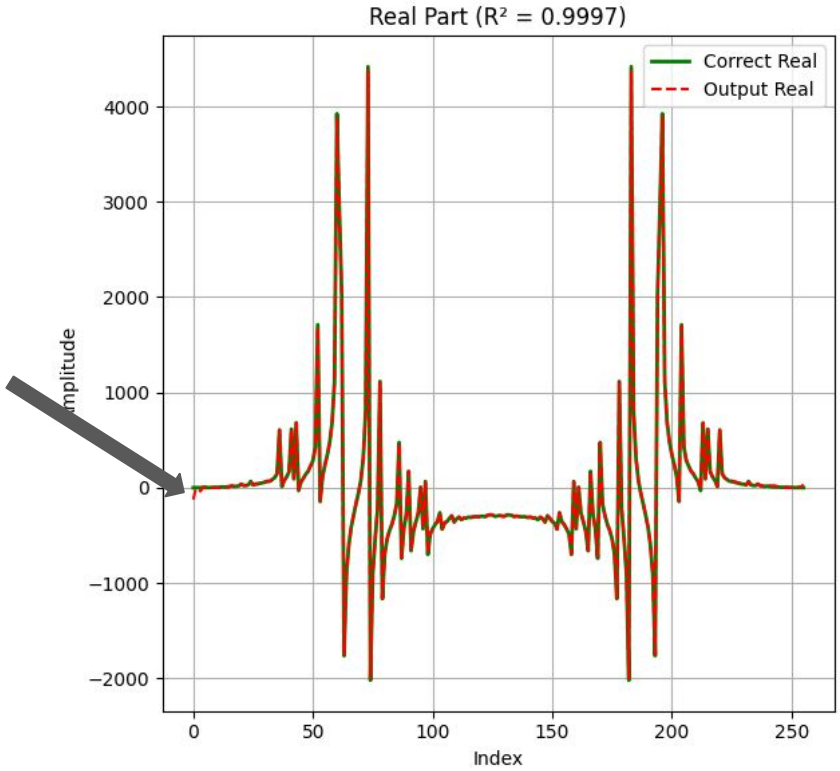


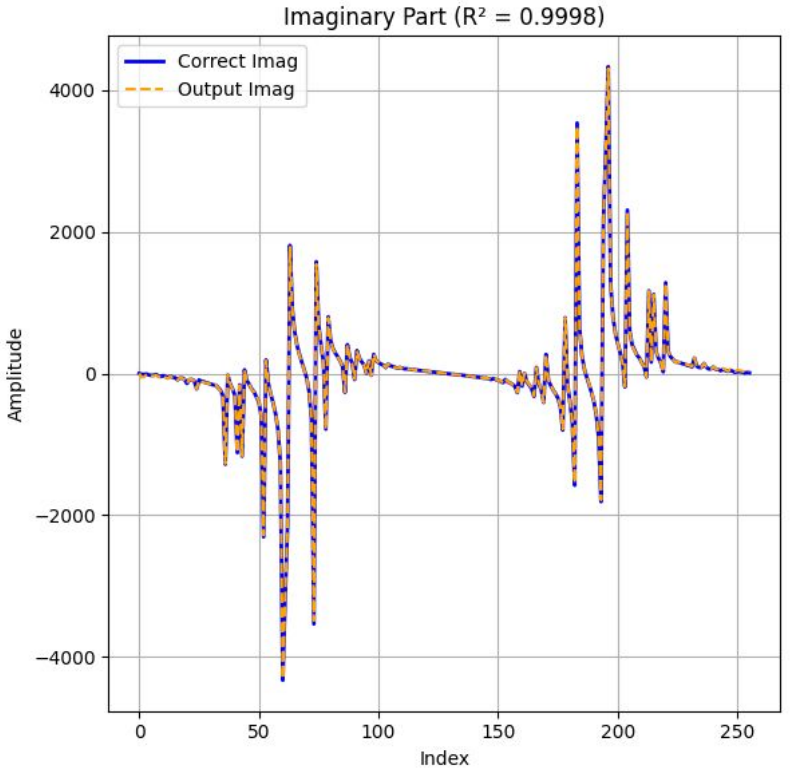
Results 18 bit



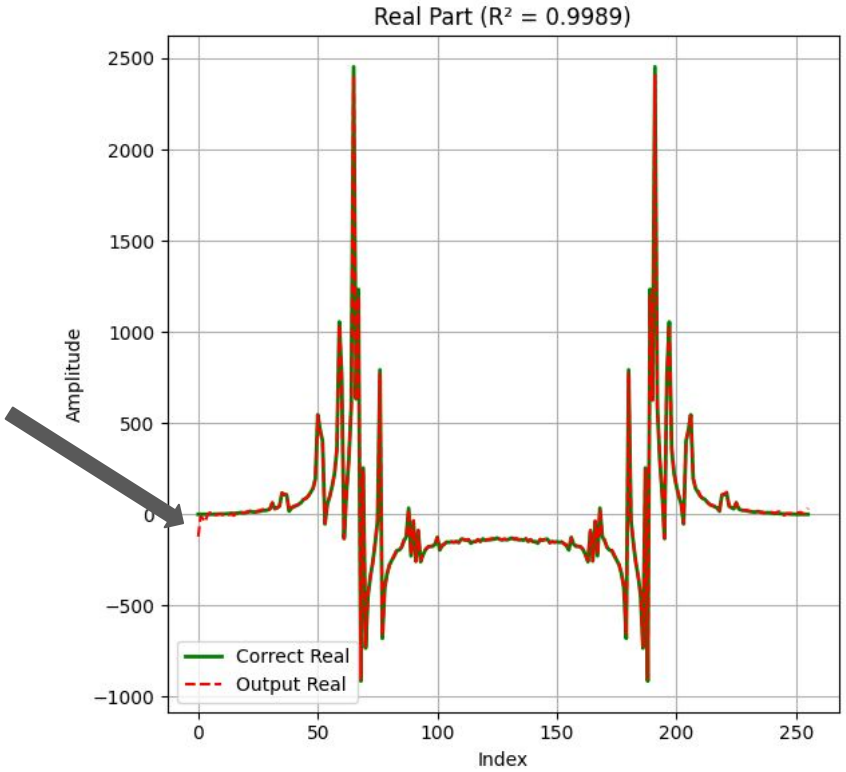


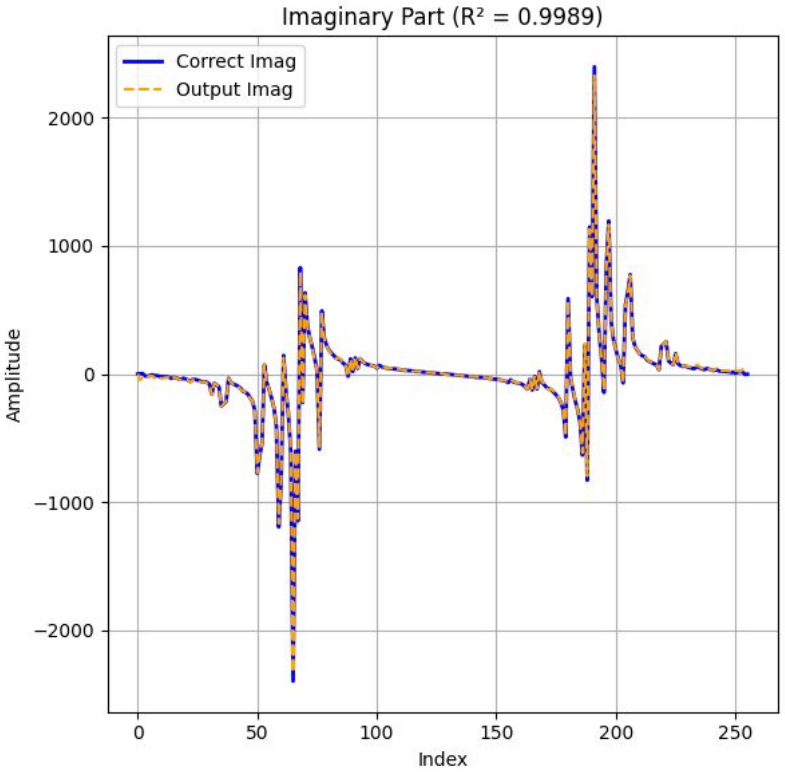
Results 18 bit



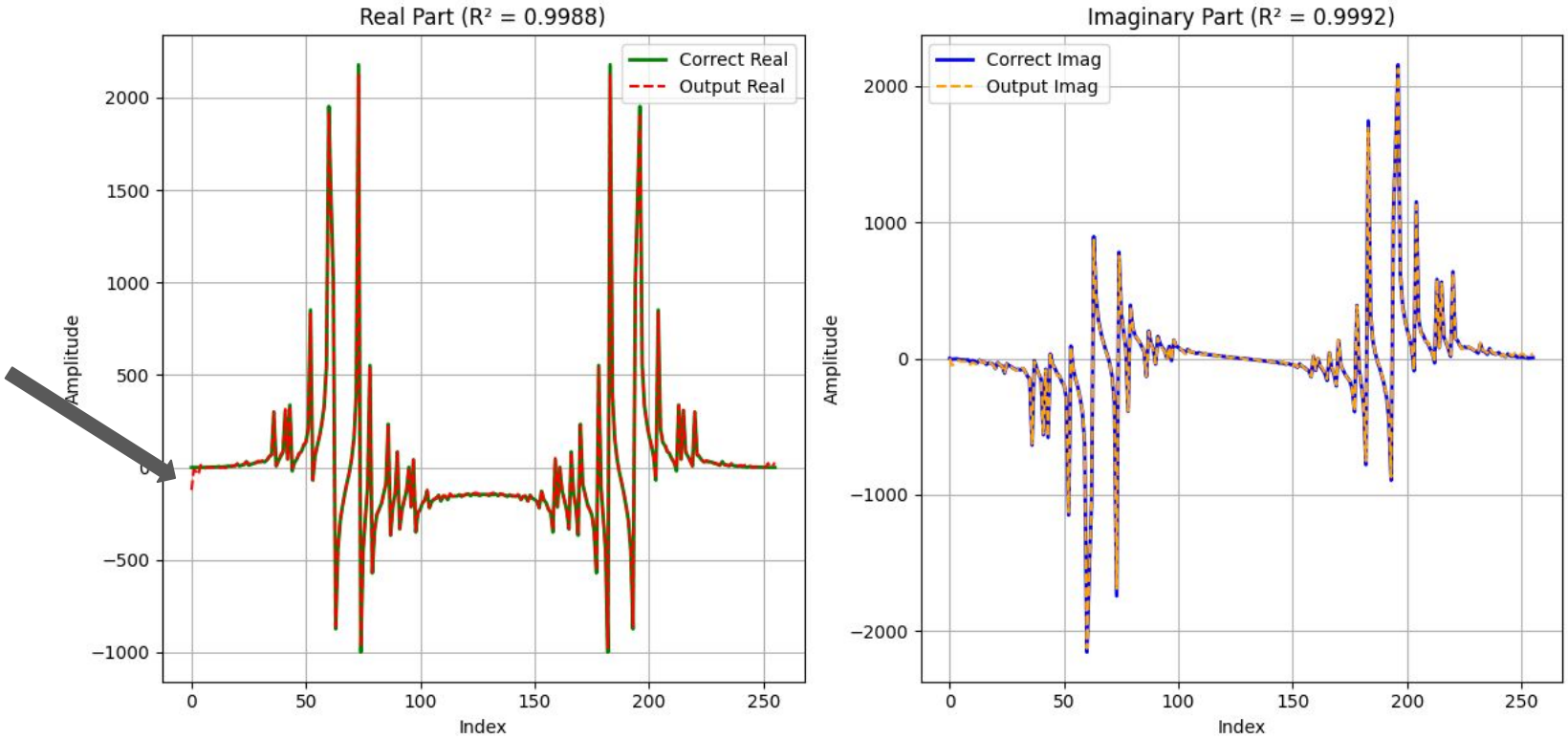


Results 16 bit

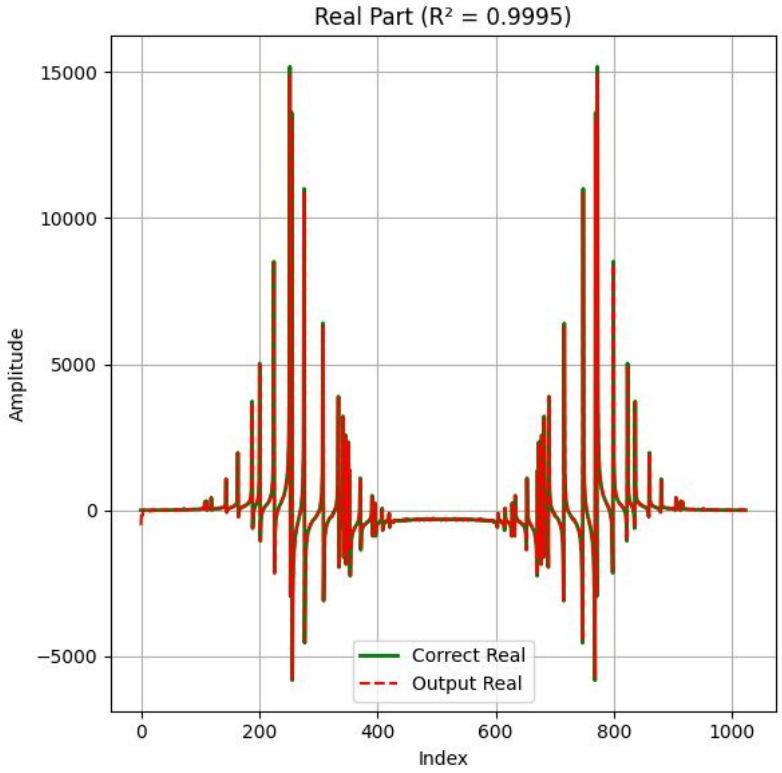


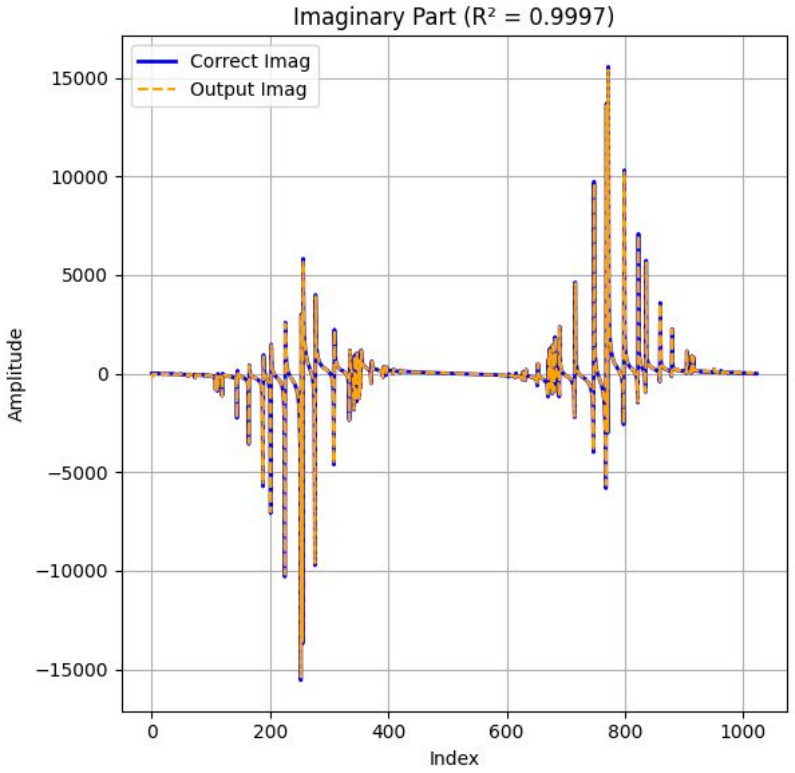


Results 16 bit

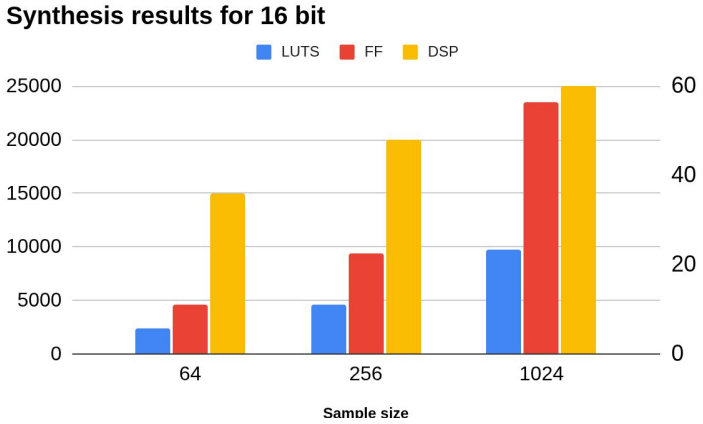


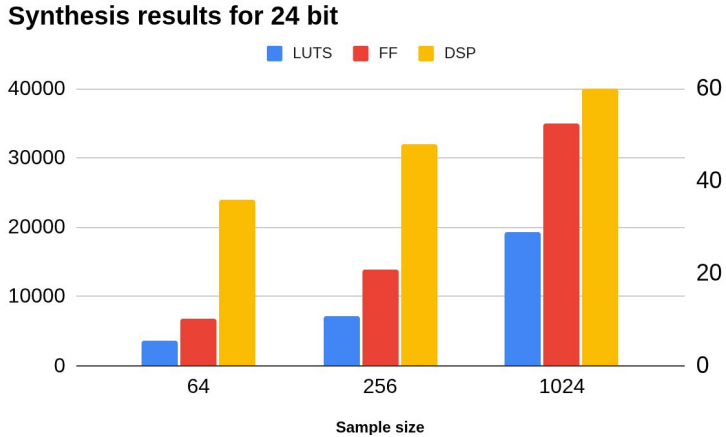
18 bit for N=1024

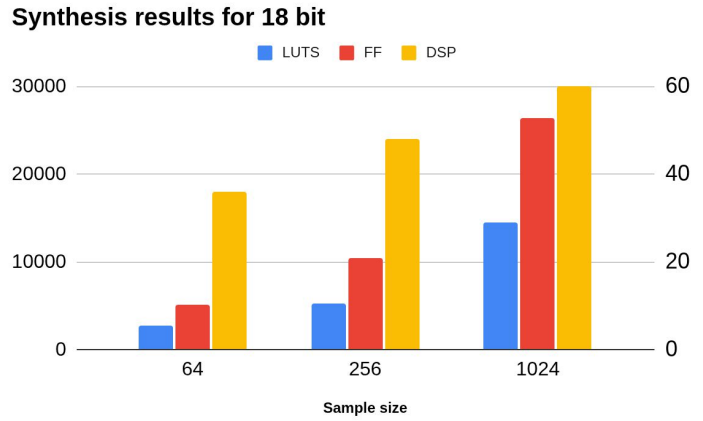


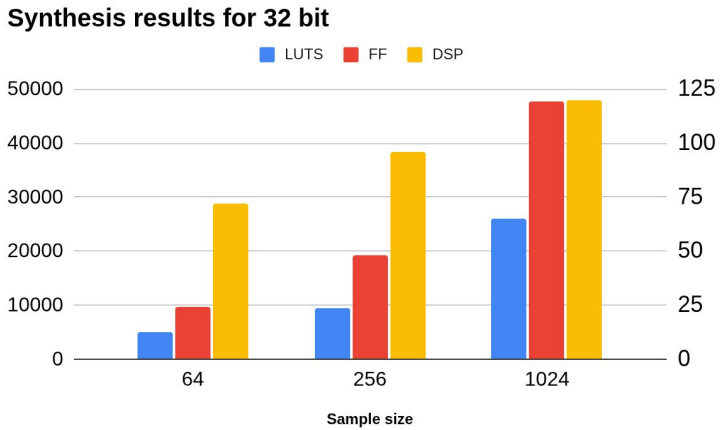


Synthesis results (DSPs on the right axis)

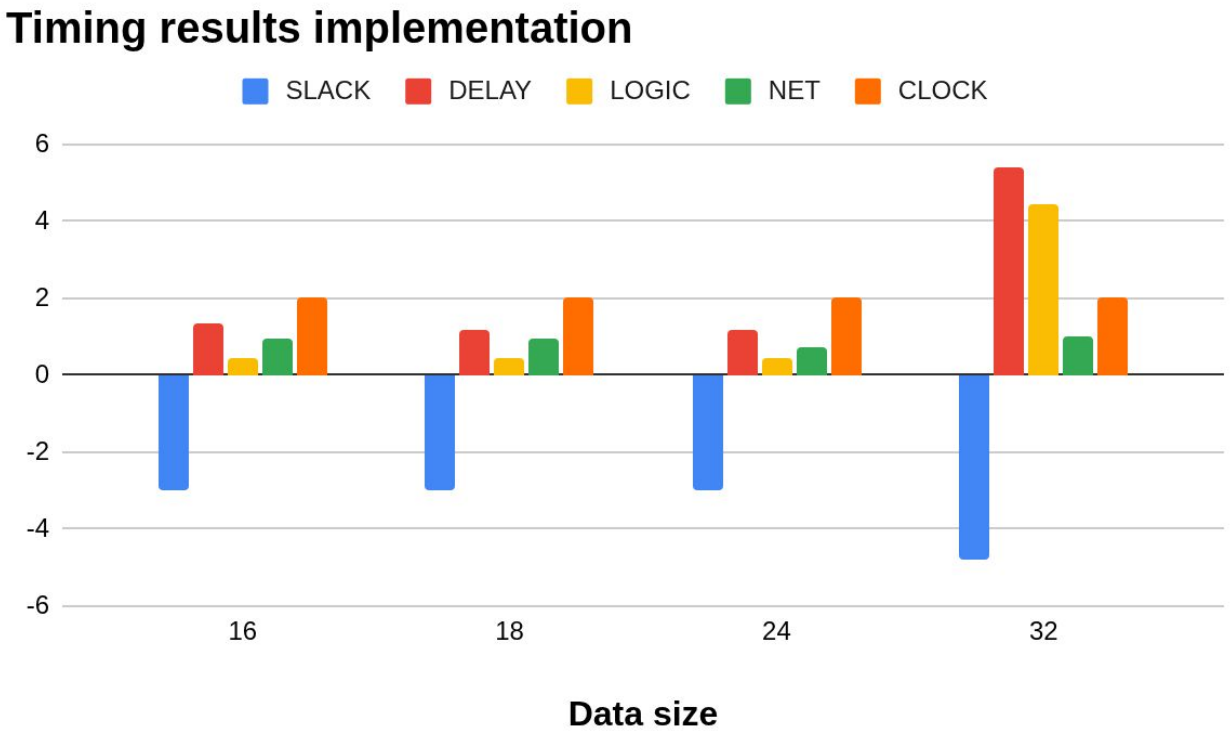








Timing results



Here i tried to implement the design with a clock of 2ns. Even though for 16,18 and 24 bits we can see that the logic+net delay is < 2ns there is still slack. Maybe with an FPGA rated for higher frequencies the design can reach higher speeds.

Comparison

The paper uses this metric to calculate the execution time.

N=256	Last in first out	Last in Last out	First in last out	Execution time 100MHz
clock cycles	23	87	151	0.230μs

If we take into account the pipeline the delay is roughly N/2 clock cycles

Processor	[20]	Altera FFT Core [21]	Xilinx FFT Core [22]	Our Design-Altera	Mine
Number of points	1024	256	256	256	256
Datapath width (bit)	16	16	16	16	16
Combinational ALUTs	------	2144	2027	6702	4635
Logic registers	------	3758	------	6498	9342
18 x 18 multipliers	------	12	30	48	48
Memory	------	19 (9 K)	3 (36 K)	5 (4 K)	0
Execution time(μs)	10.1	0.69	0.59	0.123	0.230
Frequency (MHz)	127	370	432	350	100
FPGA family	Virtex II Pro 30	Stratix III	Virtex 5	Stratix II	Zynq-7000 SoC
Device	XCV2P30	EP3SL70F484C2	5VSX35T	EP2S15F672C3	XC7Z020-CLG484-1

https://scispace.com/pdf/fpga-realization-of-a-split-radix-fft-processor-1fp4drkb34.pdf

Peper results

FPGA realization of a Split Radix FFT processor