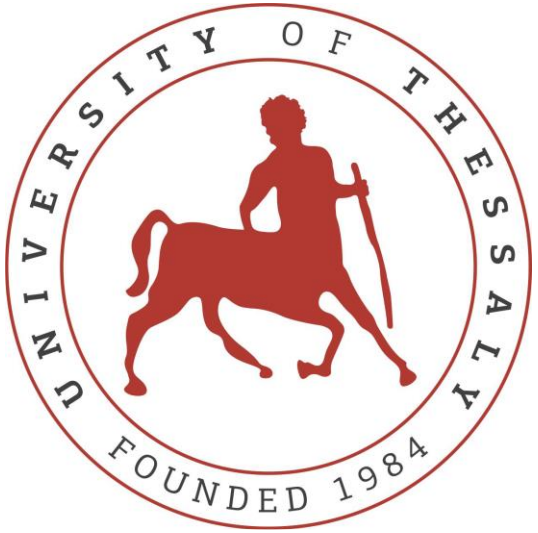
ECE494 MICROPROCESSOR DESIGN

PIPELINED FFT IMPLEMENTATION



Christodoulos Zerdalis 03531

Athanasios Gerampinis 03466

26/6

UNDERSTANDING THE FAST FOURIER TRANSFORM

- ***What is FFT*?**
  - An optimized algorithm to compute the **Discrete Fourier Transform** (DFT) and its inverse, reducing time complexity from **$O(N^2)$** to **O(N log N)**.
- ***Why is FFT Important*?**
  - Essential in **digital signal processing** (DSP)
  - Used in **image processing**, **audio analysis**, **wireless communications.**
- ***Basic Principle:*** FFT breaks down a **large DFT** into **smaller DFTs** using divide-and-conquer strategies (e.g., Radix-2, Radix-4).

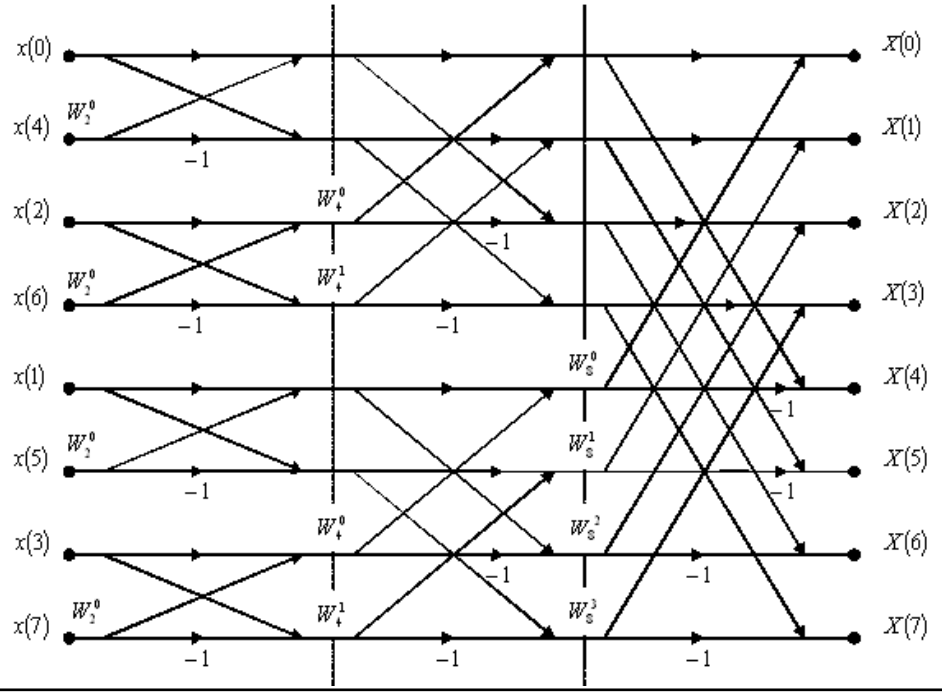
MOTIVATION FOR FFT ACCELERATION

- **High Performance Required in Real-Time Systems**:
  - Voice recognition, and real-time video processing need fast FFT computation.
- **Energy & Area Constraints in Embedded Systems**:
  - Optimized FFT cores reduce power and hardware resources.
- **Large-Scale Data Processing**:
  - FFT is a core computational kernel in large systems like wireless communication stations

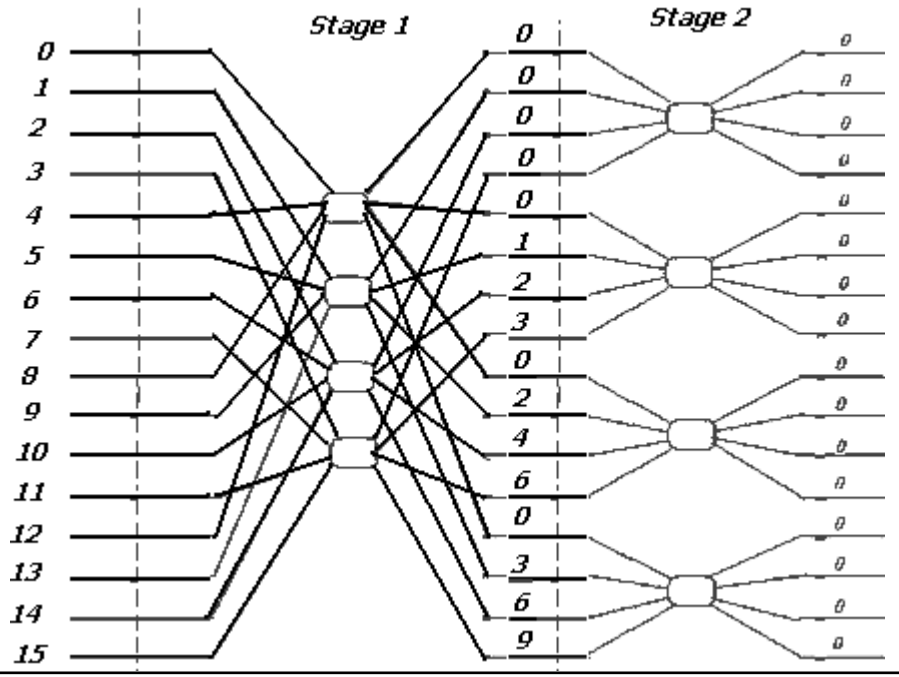
PROJECT FOCUS

Our goal is to develop hardware implementations of two different FFT algorithms (Radix-2 and Radix-4) and compare them in terms of speed, power consumption, and area for various input sizes.

Radix-2



Radix-4



TWIDDLE COMPUTATION

- Since the twiddle factors W involve complex number calculations, their values are **not computed on-the-fly in hardware**. Instead, these values are **precomputed beforehand** and stored in a **lookup table (LUT)**.

This approach reduces computational complexity and saves hardware resources

We save the real and imaginary values in 16-bit buffers using Q1.15 fixed point arithmetic

Index	W_Real	W_Imag	Complex number
0	32768	0	1 + 0j
1	23170	-23170	0.7071 - 0.7071j
2	0	-32768	0 - 1j
3	-23170	-23170	-0.7071 - 0.7071j
4	-32768	0	-1 + 0j
5	-23170	23170	-0.7071 + 0.7071j
6	0	32768	0 + 1j
7	23170	23170	0.7071 + 0.7071j

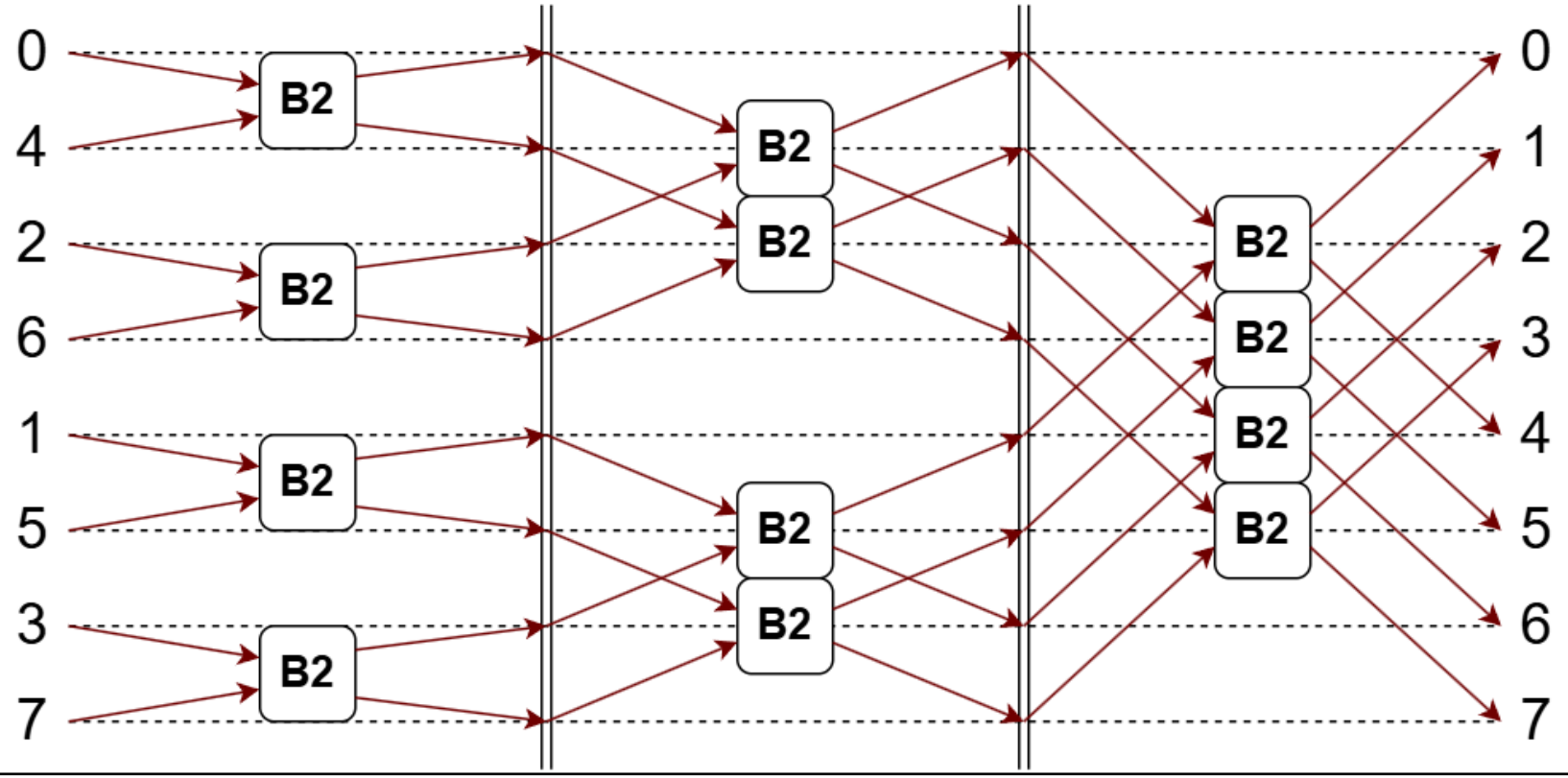
REORDER INPUT DATA

- Before starting butterfly calculations in an FFT we need to **reorder** the data, for this we use **bit-reversal**

Index	Binary	Bit-Reversed Binary	Bit-Reversed Index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

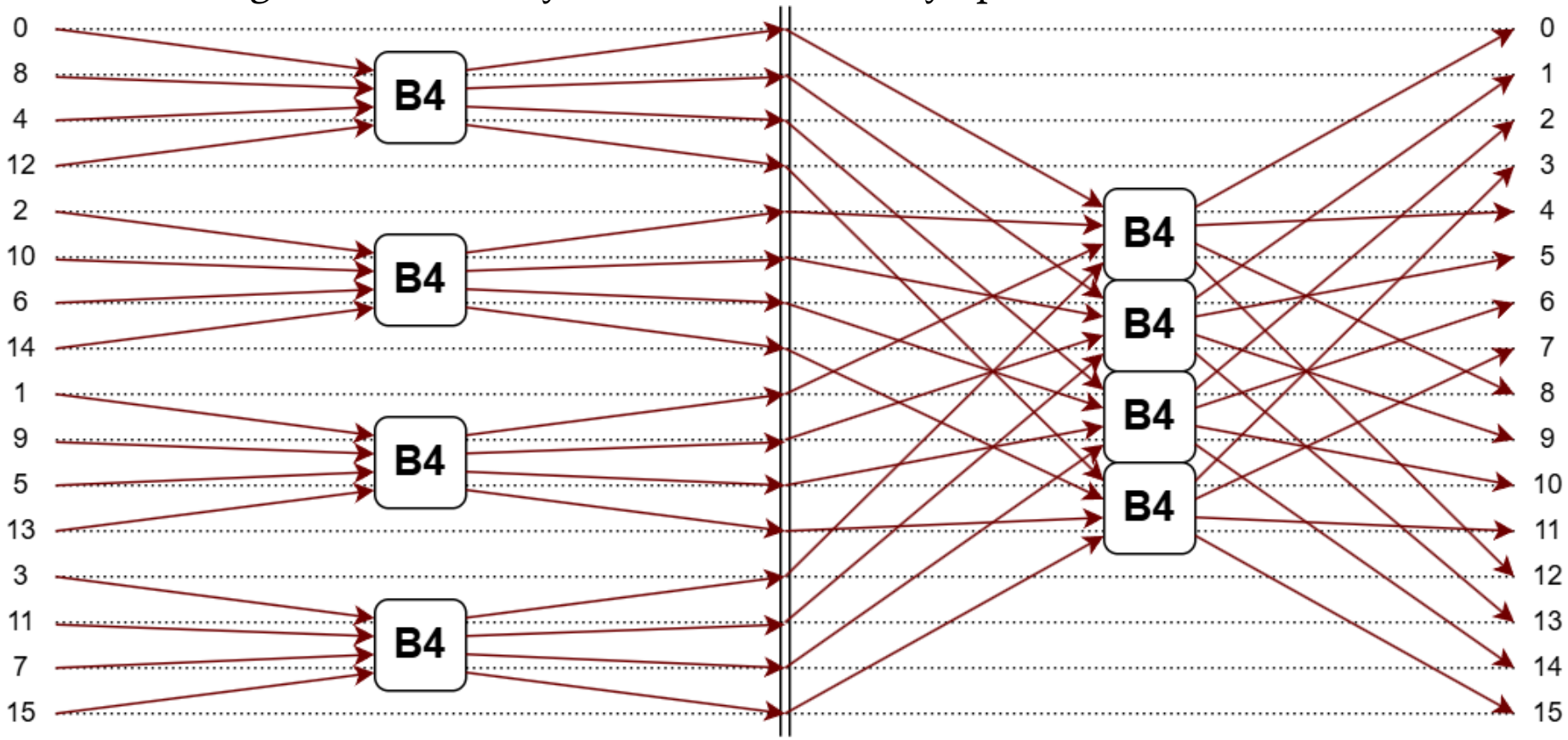
RADIX-2 (unrolled)

This implementation of the Radix-2 algorithm fully utilizes its parallelism, resulting in a fully unrolled design that uses a dedicated butterfly unit for each butterfly operation.



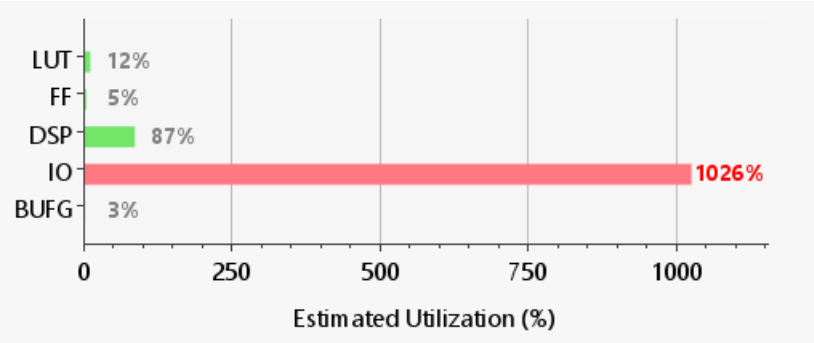
RADIX-4 (unrolled)

Again, the unrolled design uses a butterfly unit for each butterfly operation.

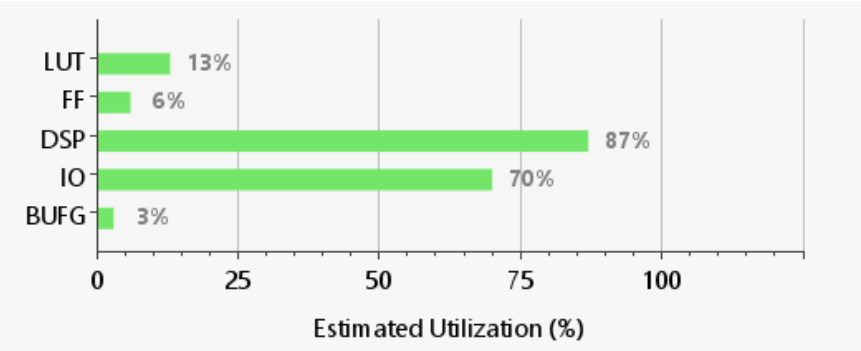


DATA BANDWIDTH LIMITATION

- These designs require all input **data to be available simultaneously** to begin computation. For example, 32 bits × 16 inputs × 2 types x 2 input/output = 2048 bits, **which far exceeds the I/O capabilities** of our FPGA. To overcome this, we designed a **memory controller** that inputs data one at a time, stores them until needed, and then outputs the results sequentially.
- This necessary change makes the design slower. Initially, the computation time was ***T = Number_of_stages***, where ***Number_of_stages = log(Input_size)***. For example, $T_{16} = 4$ for a 16-point FFT. Now this time is ***Tnew = = log(Input_size) + 2xInput_size.***

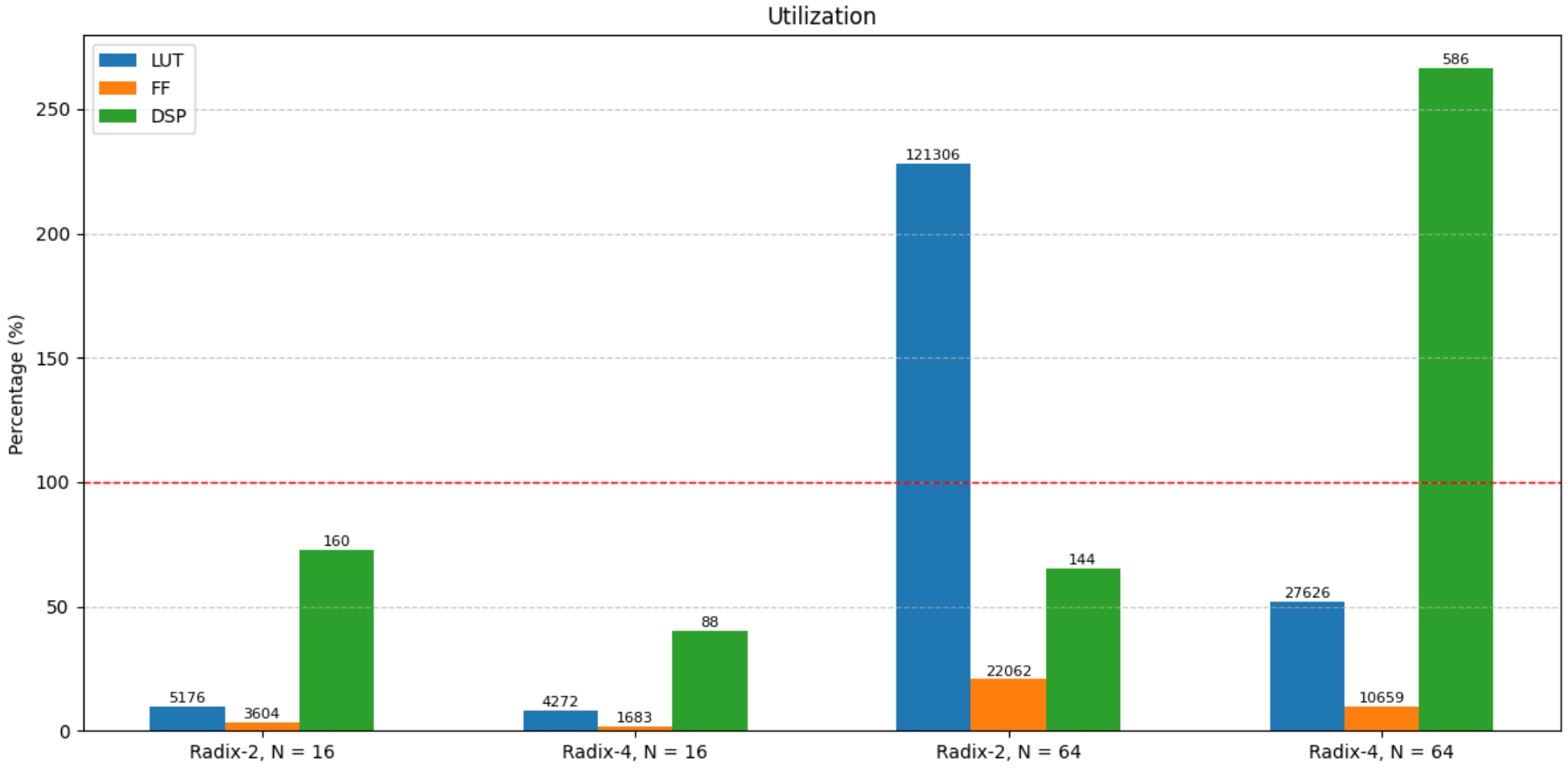


No memory control

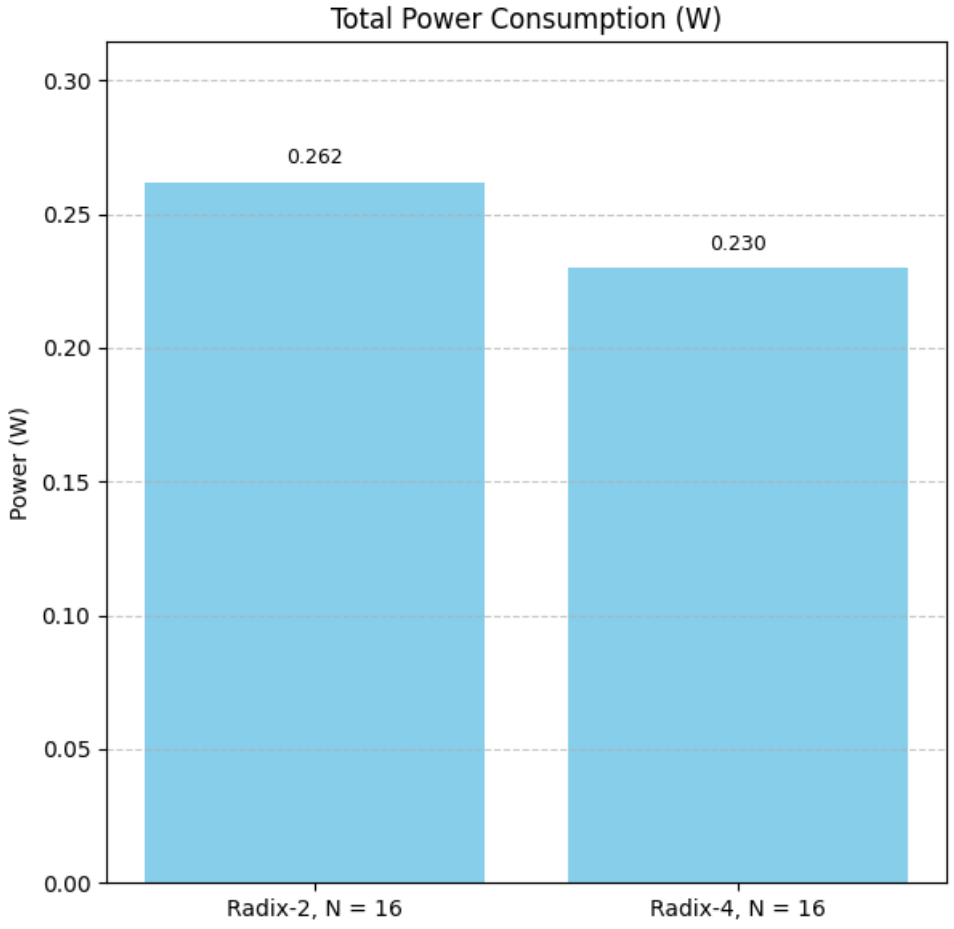


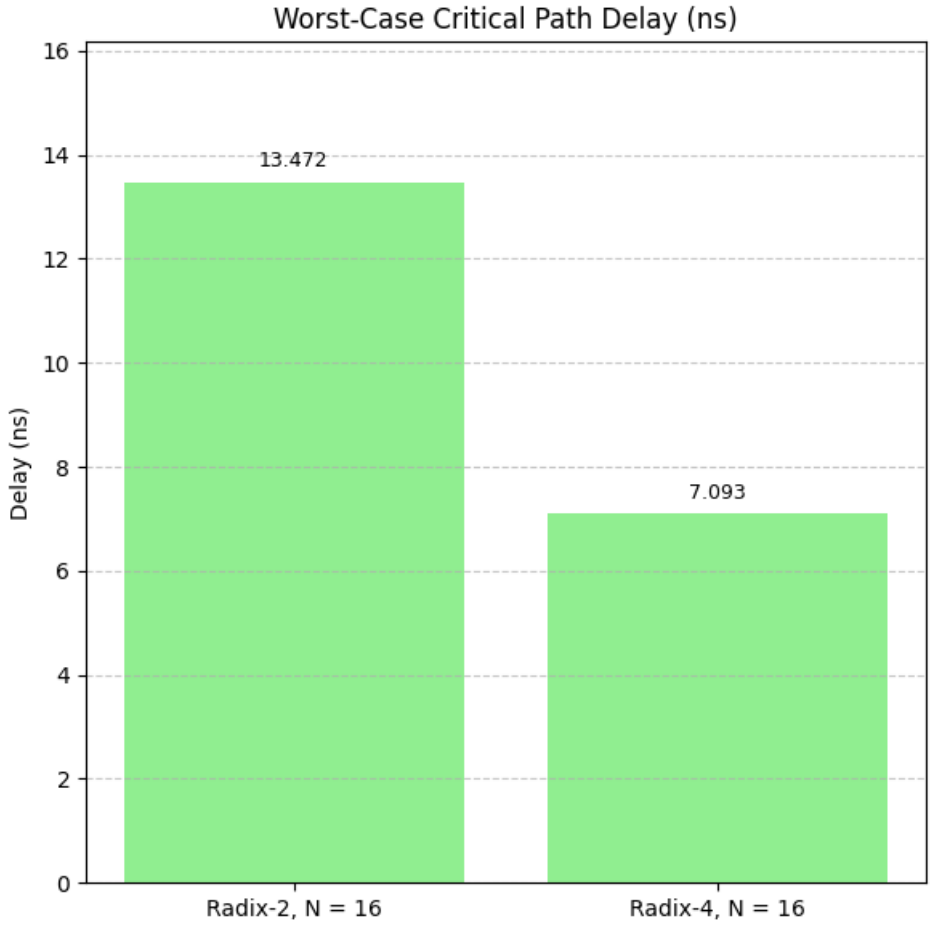
Memory control

RESULTS

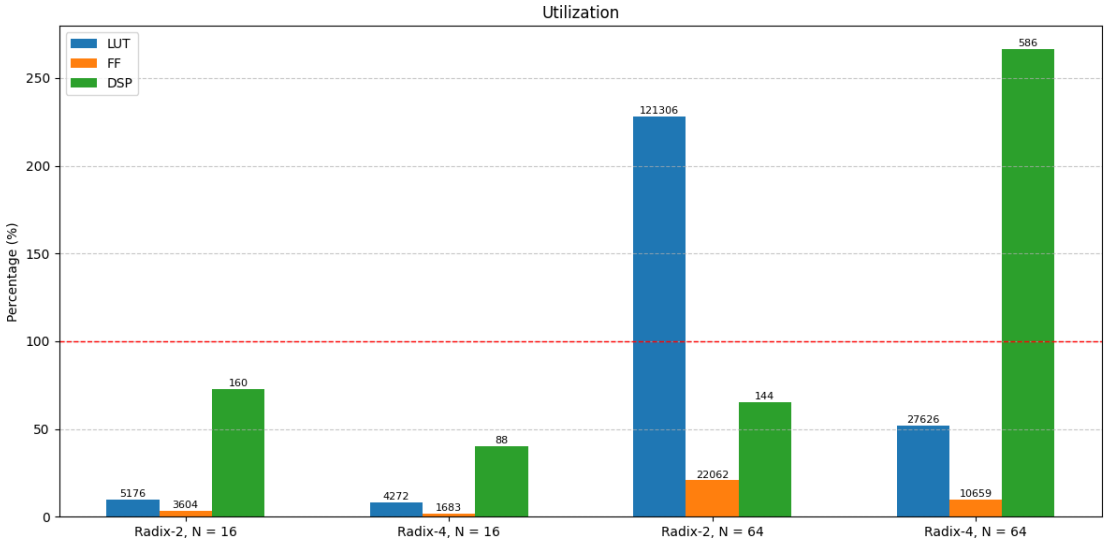

Utilization
LUT
FF
DSP
Percentage (%)
0
50
100
150
200
250
5176
3604
160
4272
1683
88
121306
22062
144
27626
10659
586
Radix-2, N = 16
Radix-4, N = 16
Radix-2, N = 64
Radix-4, N = 64

RESULTS


Total Power Consumption (W)
Power (W)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.262
0.230
Radix-2, N = 16
Radix-4, N = 16


Worst-Case Critical Path Delay (ns)
Delay (ns)
16
14
12
10
8
6
4
2
0
13.472
7.093
Radix-2, N = 16
Radix-4, N = 16

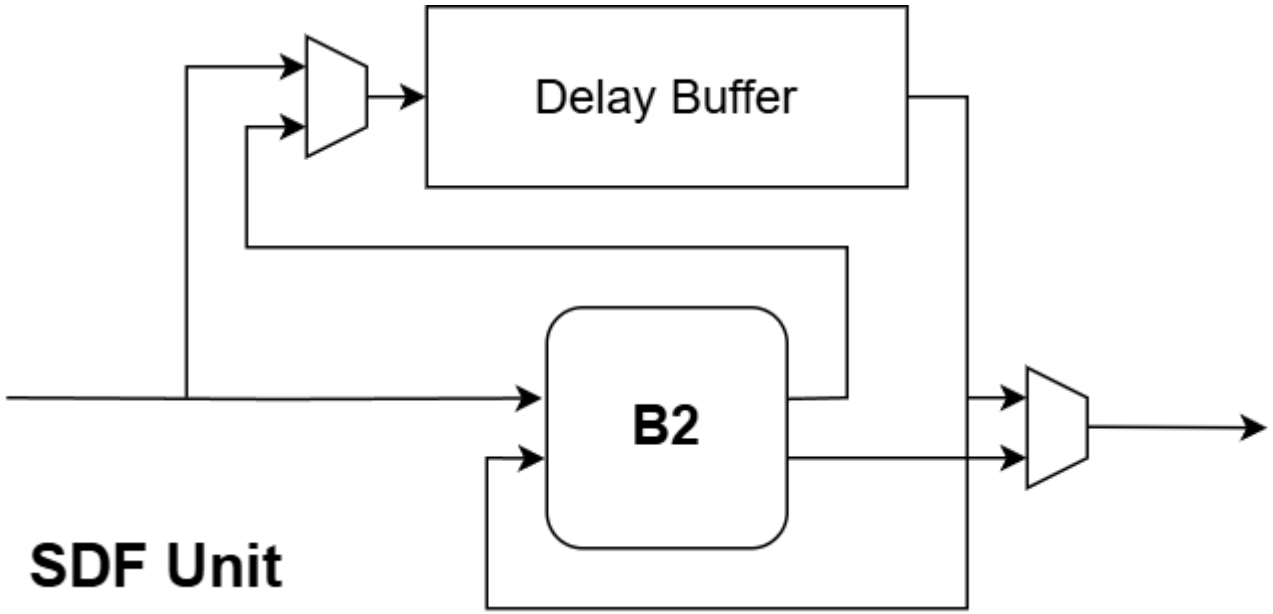
LIMITATIONS (scalability)



- We can see that as the **input size increases**, the number of **resources and the area required grow significantly**. This becomes a problem—even for a small input size of 64, the design **cannot be implemented** on our FPGA.

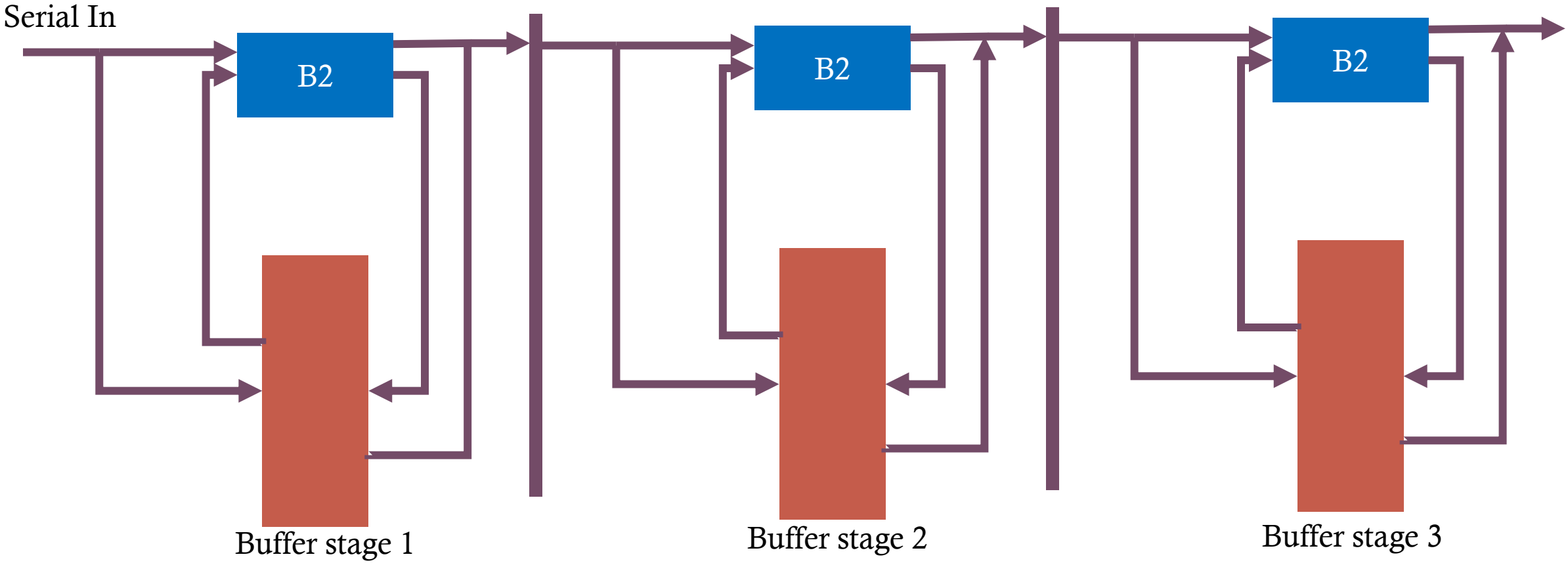
NEW APPROACH

- Radix-2 SDF performs one stage of radix-2 FFT using a **Single-Path Delay Feedback** architecture by **streaming** complex input samples, applying butterfly and twiddle operations with proper data alignment via a **delay buffer**, and outputting the transformed results **sequentially**.



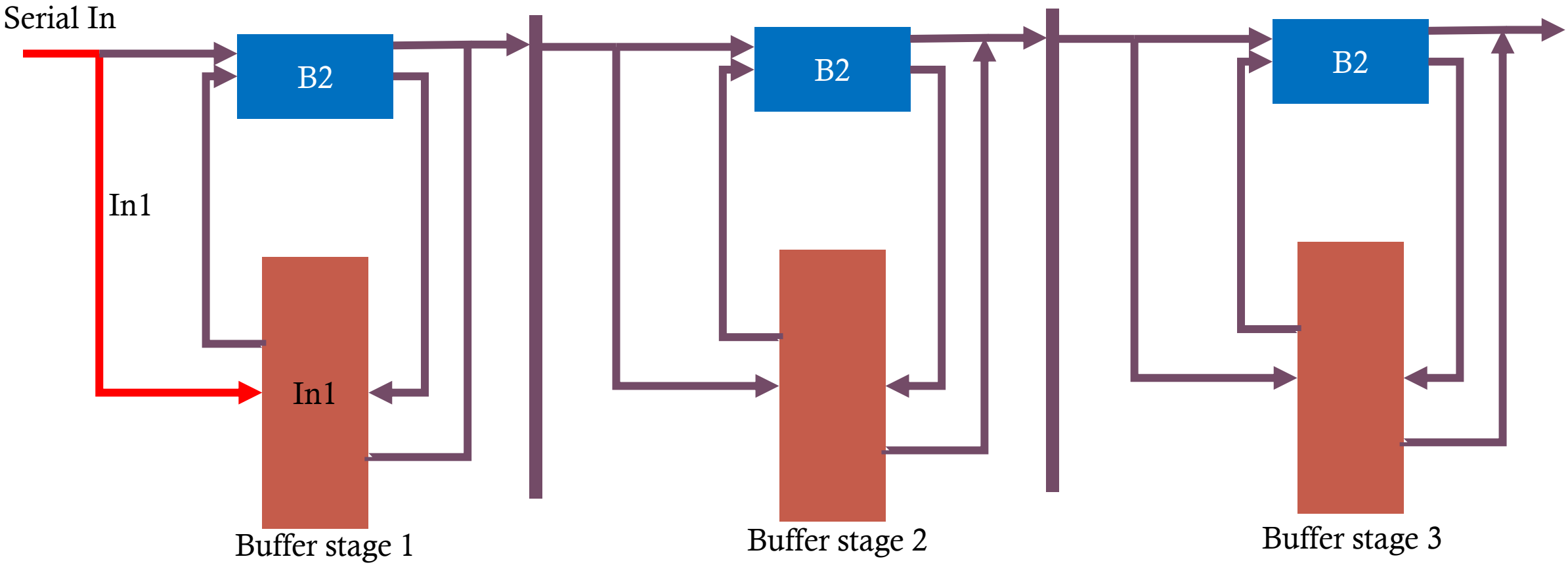
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)



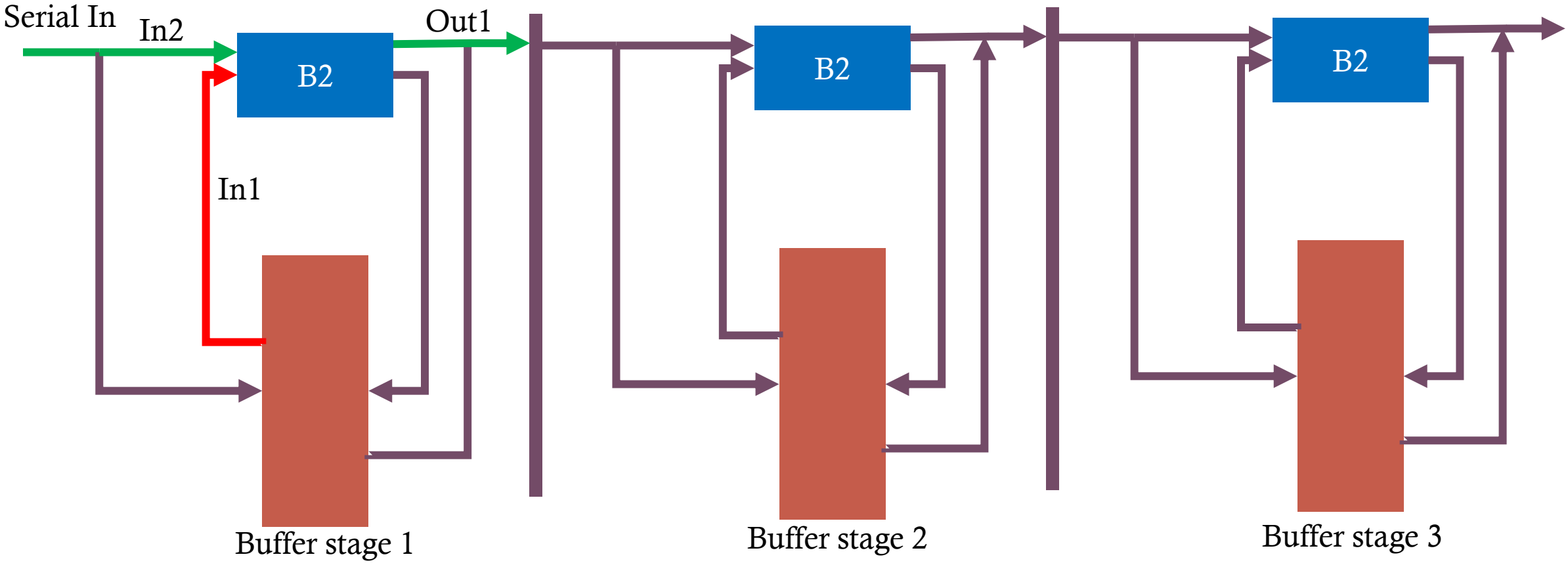
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)



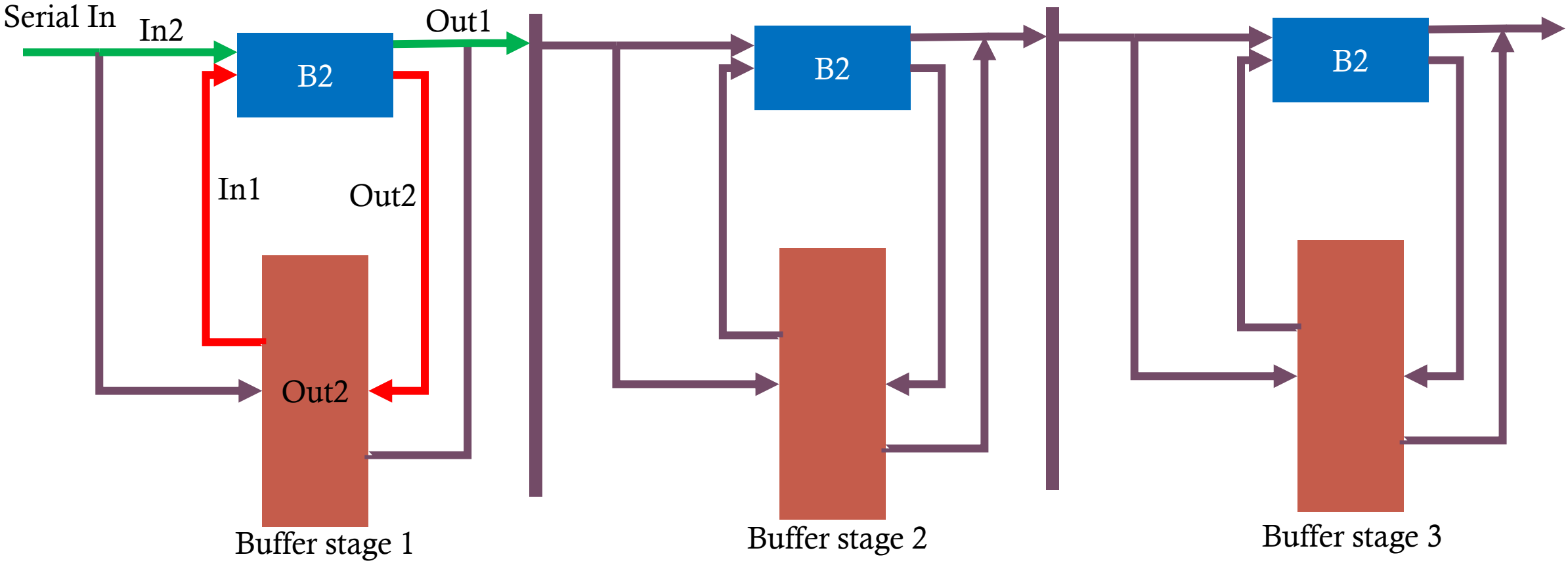
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)



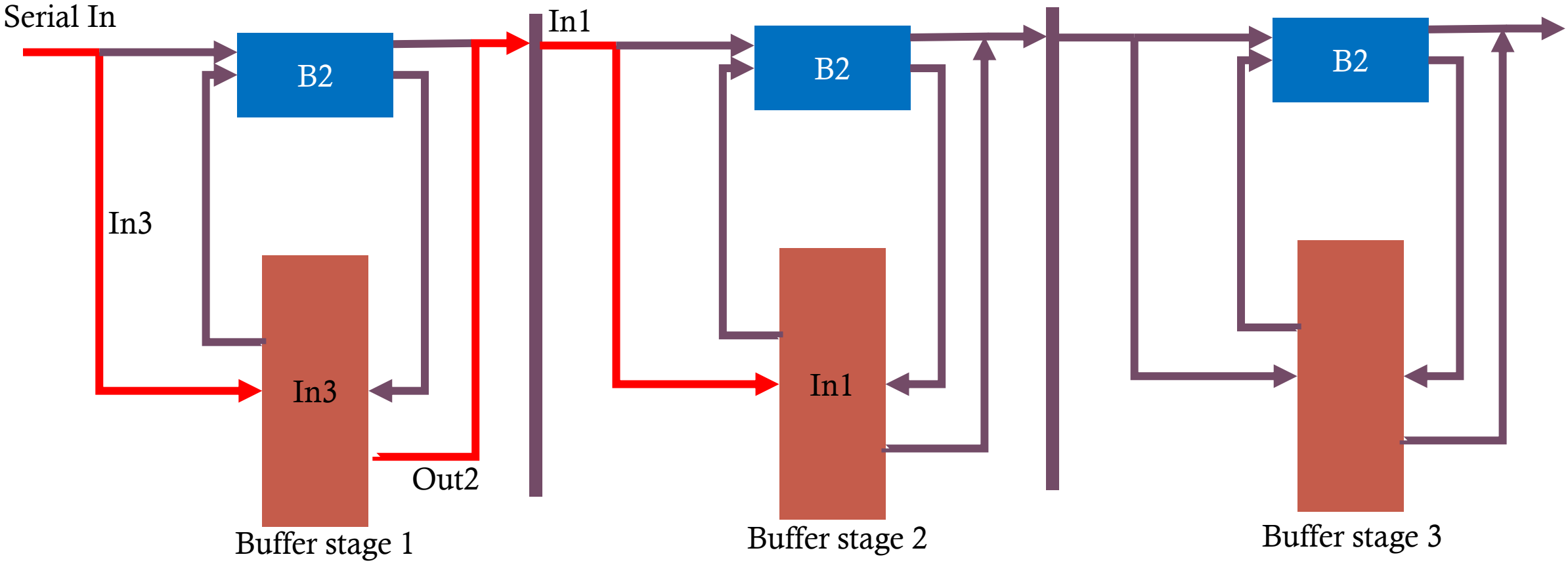
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)



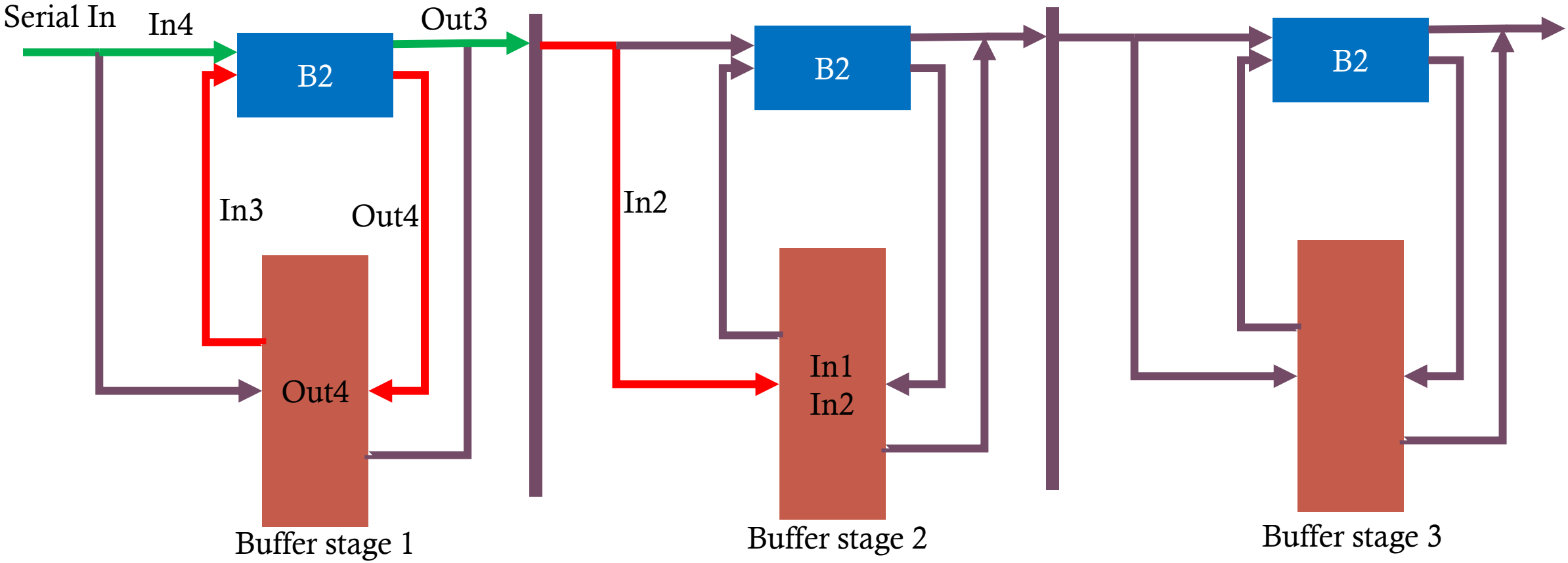
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)



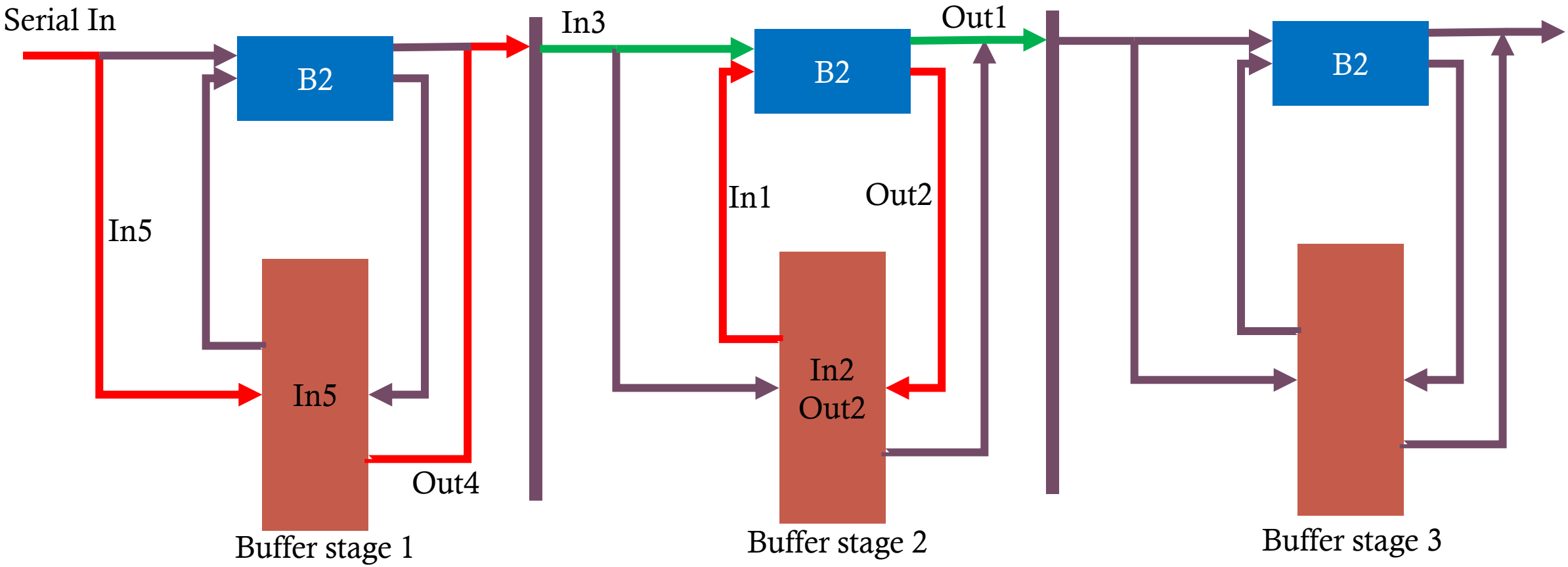
NEW APPROACH

- Radix-2 SDF (Single path delay feedback)

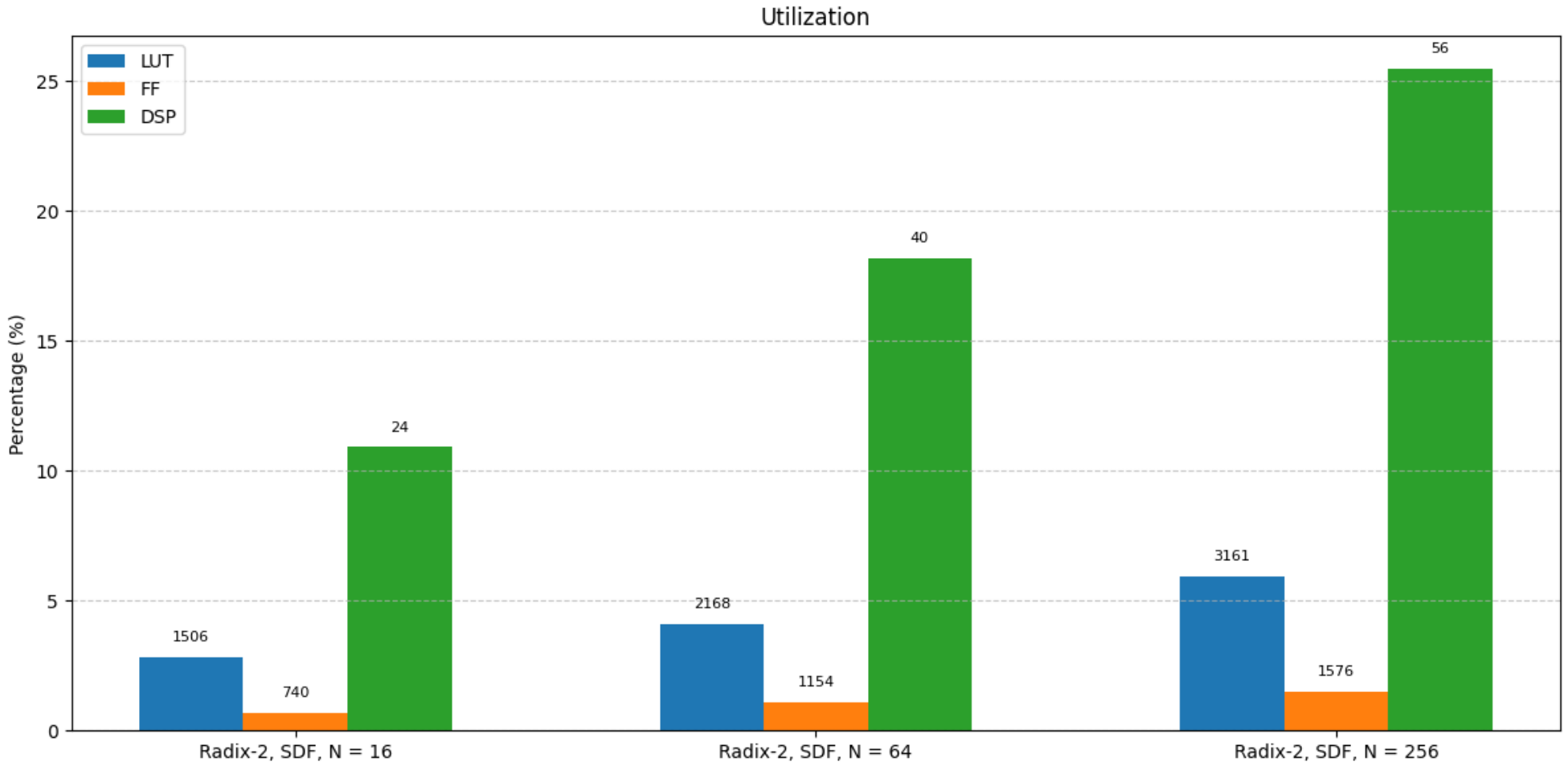


NEW APPROACH

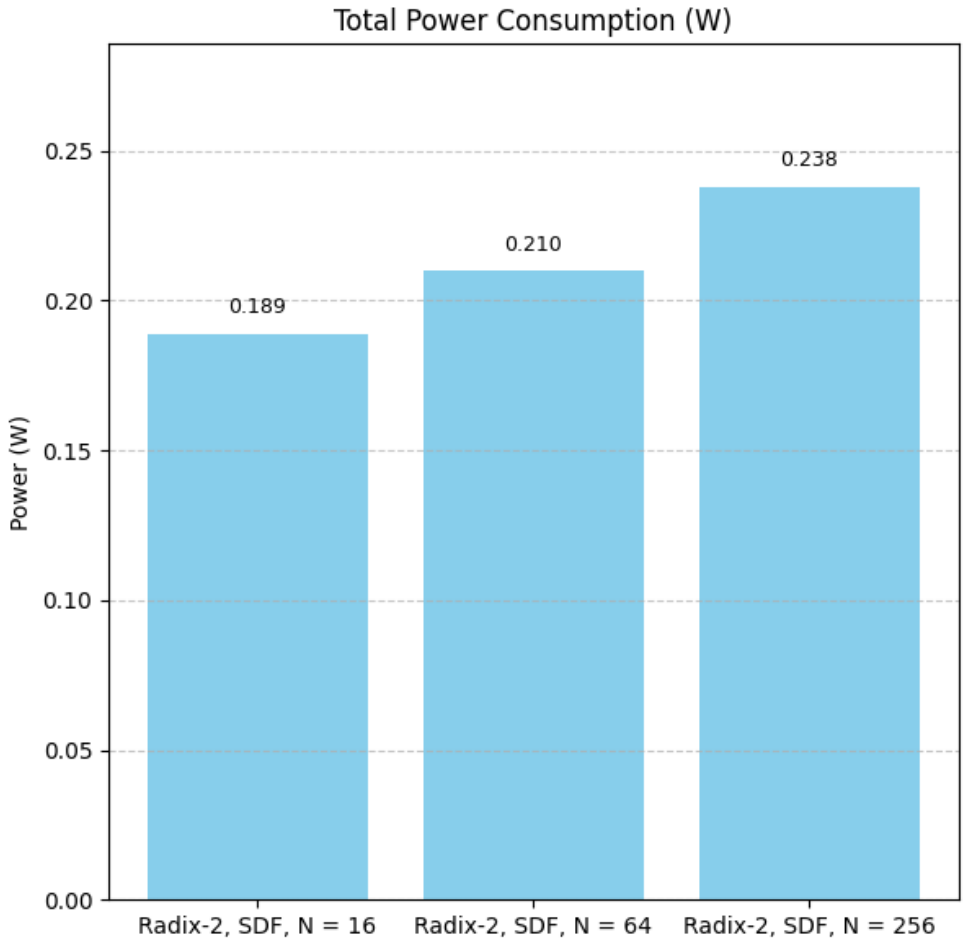
- Radix-2 SDF (Single path delay feedback)

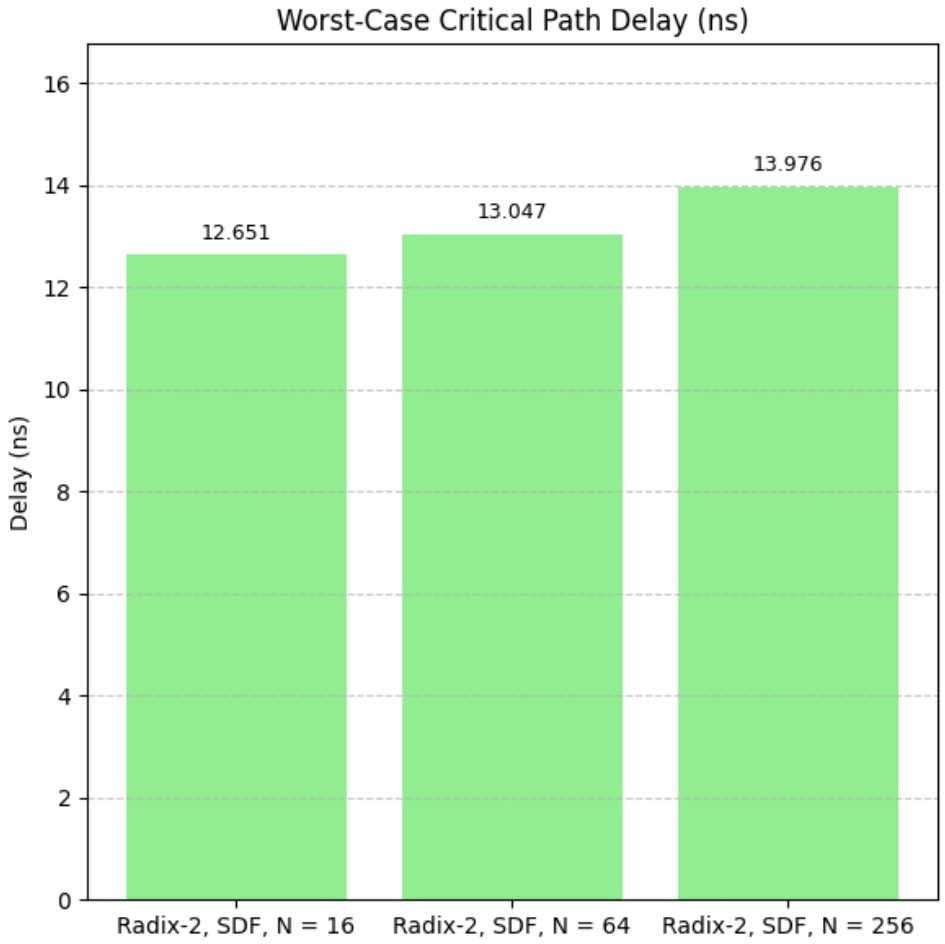


RESULTS

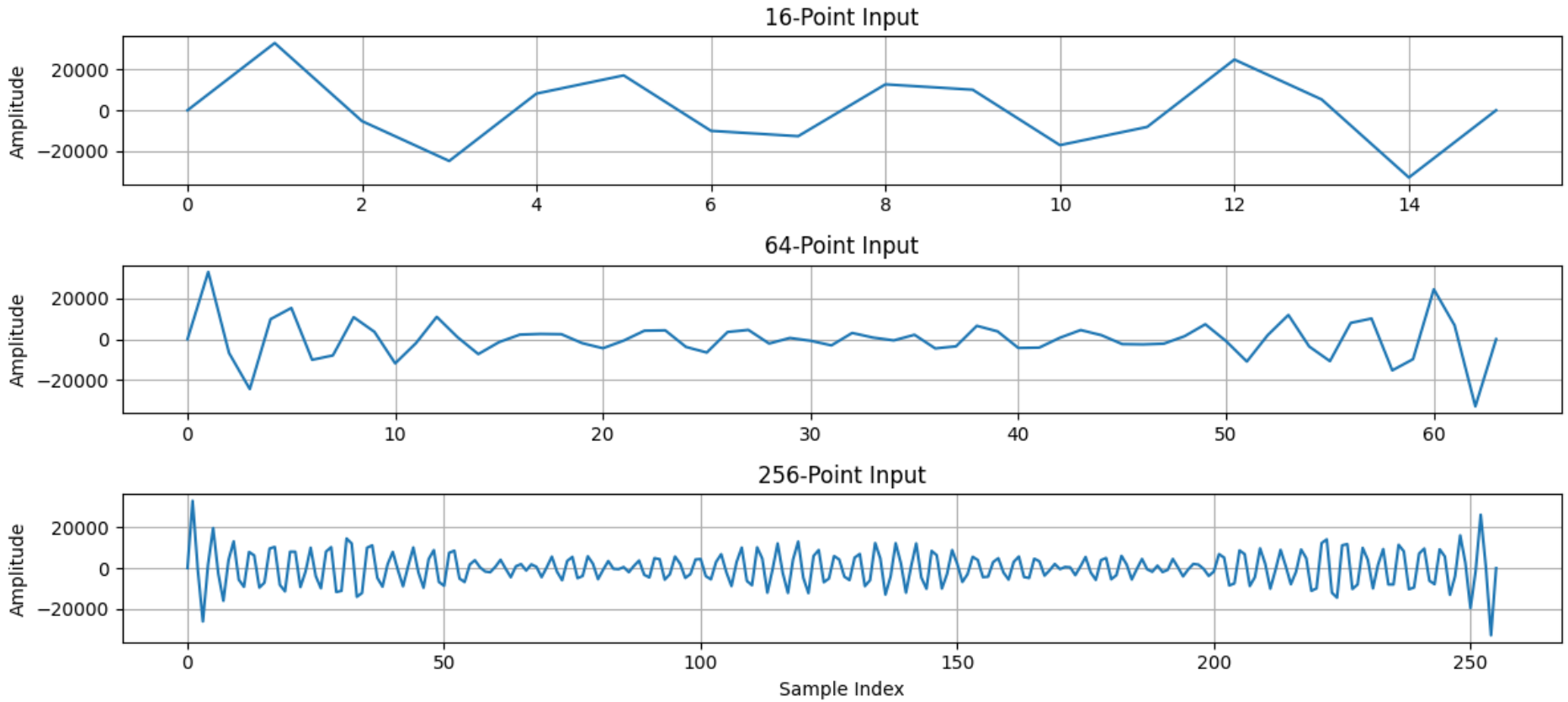

Utilization
LUT
FF
DSP
Percentage (%)
0
5
10
15
20
25
1506
740
24
2168
1154
40
3161
1576
56
Radix-2, SDF, N = 16
Radix-2, SDF, N = 64
Radix-2, SDF, N = 256

RESULTS

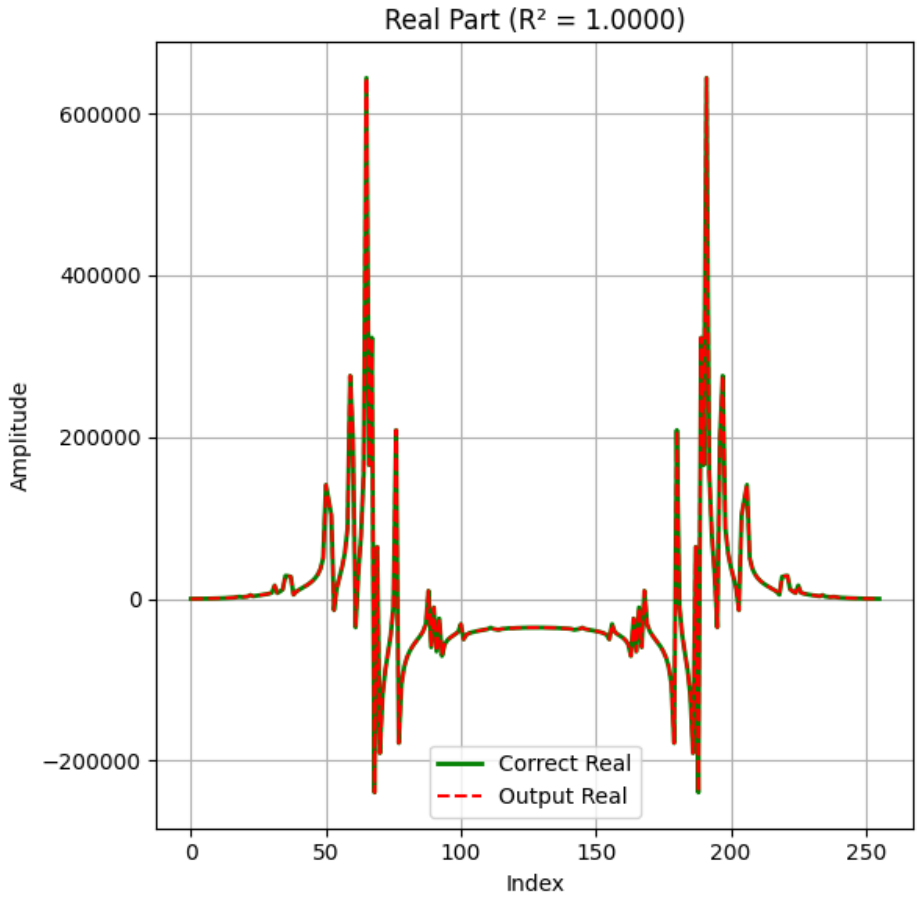

Total Power Consumption (W)
Power (W)
0.00
0.05
0.10
0.15
0.20
0.25
0.189
0.210
0.238
Radix-2, SDF, N = 16
Radix-2, SDF, N = 64
Radix-2, SDF, N = 256

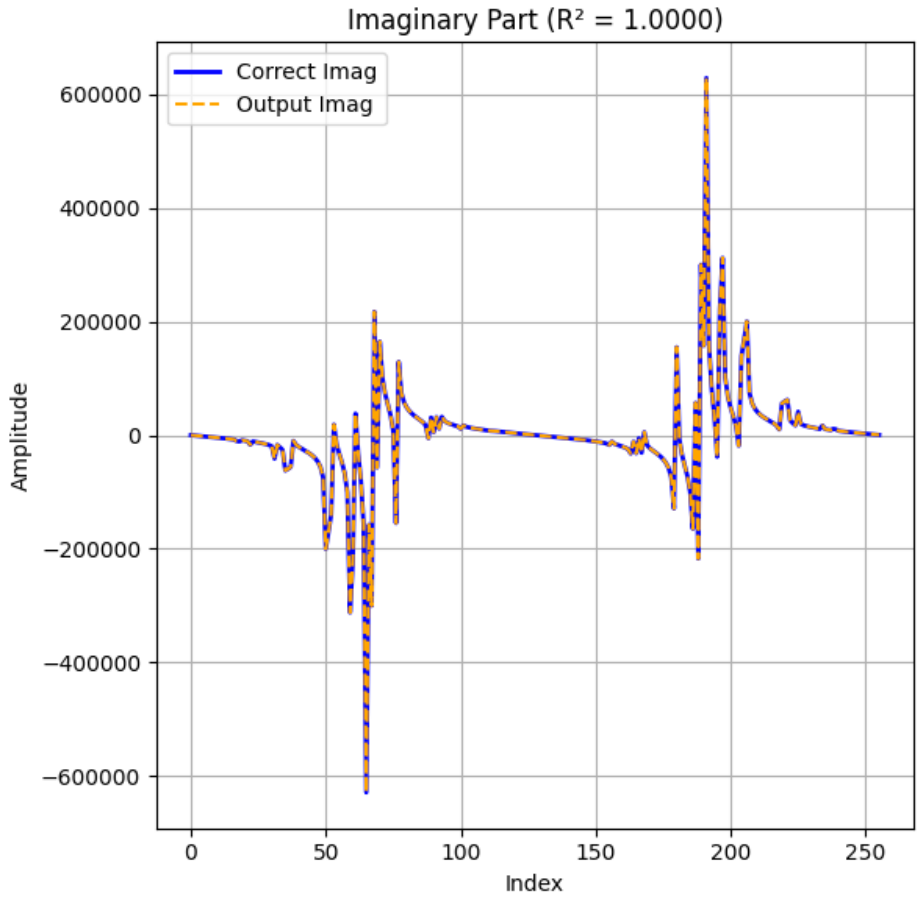

Worst-Case Critical Path Delay (ns)
Delay (ns)
0
2
4
6
8
10
12
14
16
12.651
13.047
13.976
Radix-2, SDF, N = 16
Radix-2, SDF, N = 64
Radix-2, SDF, N = 256

RESULTS


16-Point Input
Amplitude
20000
0
−20000
0
2
4
6
8
10
12
14
64-Point Input
Amplitude
20000
0
−20000
0
10
20
30
40
50
60
256-Point Input
Amplitude
20000
0
−20000
0
50
100
150
200
250
Sample Index

RESULTS


Real Part (R² = 1.0000)
Amplitude
Index
Correct Real
Output Real
600000
400000
200000
0
−200000
0
50
100
150
200
250


Imaginary Part (R² = 1.0000)
Amplitude
Index
Correct Imag
Output Imag
600000
400000
200000
0
−200000
−400000
−600000
0
50
100
150
200
250

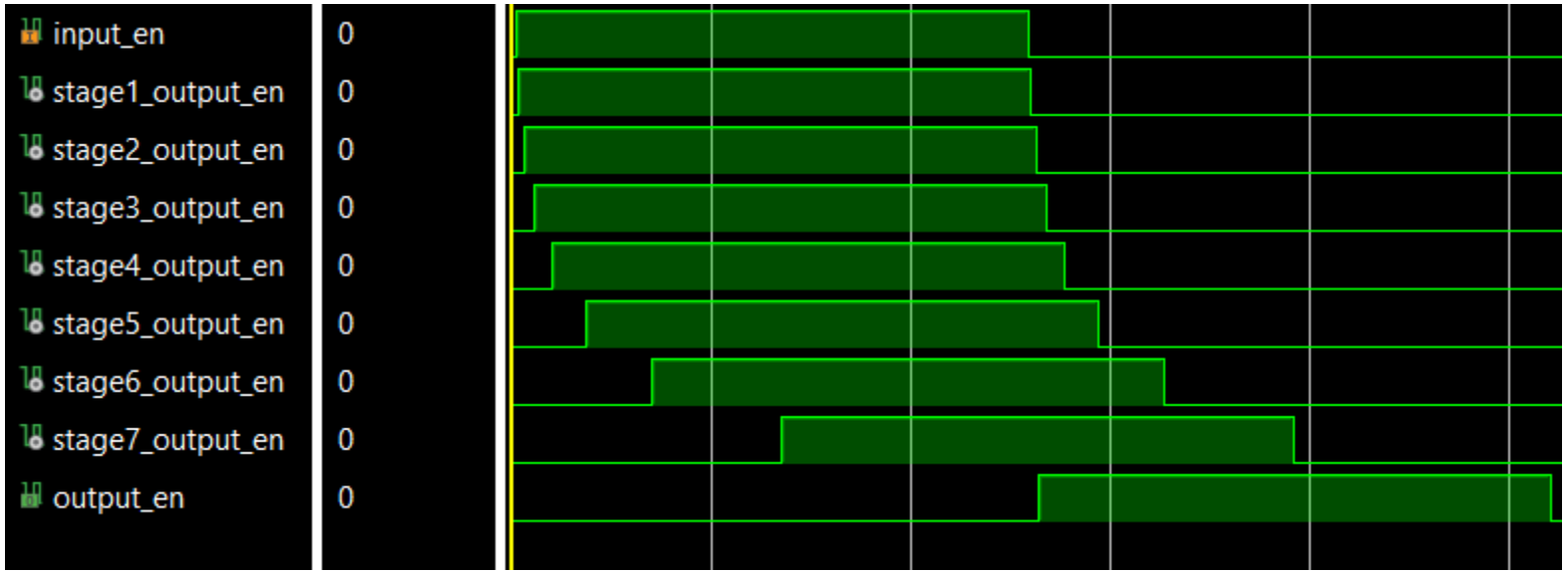
EXECUTION TIME

- As you can see, larger delay buffers with each stage cause the latency from one to another to increase. For example, when measuring the offsets between output enables for consecutive stages we get 1 + 3 + 5 + 9 + 17 + 31 + … + 256 clock cycles (N = 256). This can be generallized by the following formula:

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

which gets simplified to

2N - 2 + log(N)

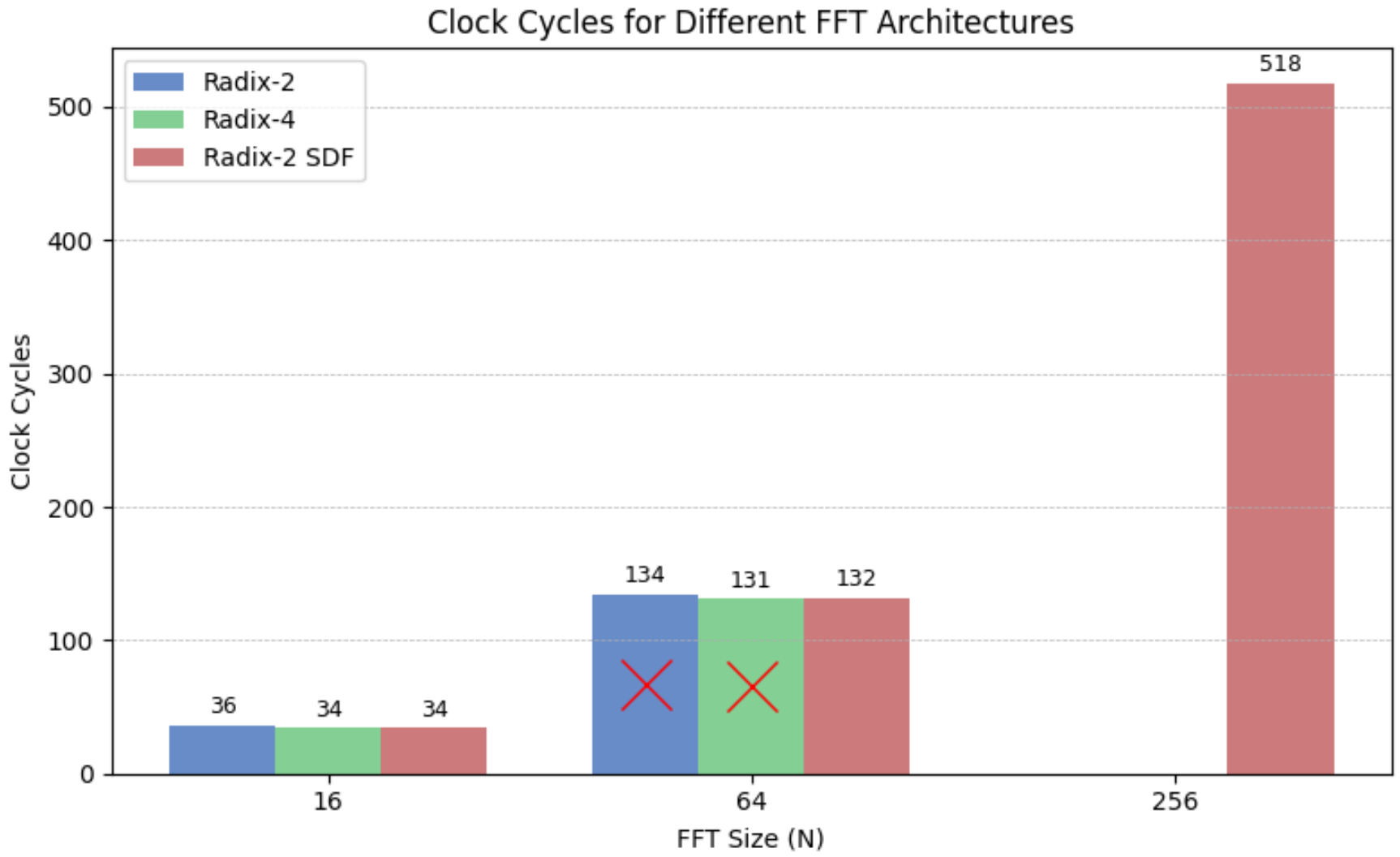


THEORETICAL EXECUTION TIMES

- **Unrolled:** $T = log(N)$

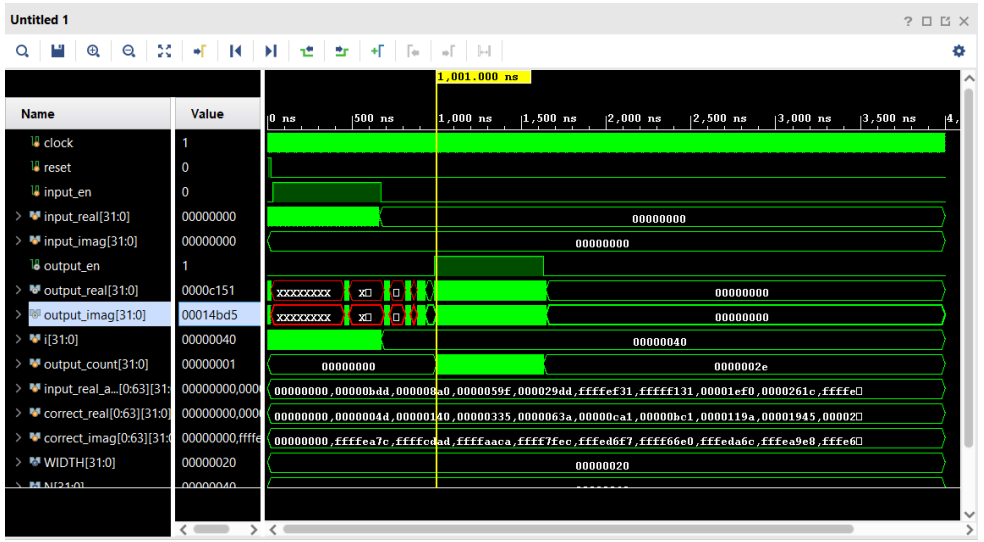
- **Unrolled with memory controller:** $T = \frac{2N}{k} + \log_2(N)$
  (k is the rate of transmission)

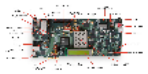
- **SDF:** $T = N + 1 + \sum_{i=1}^{\log_2 N - 1} \left(2^i + 1\right) = 2N - 2 + \log(N)$


Clock Cycles for Different FFT Architectures
Radix-2
Radix-4
Radix-2 SDF
Clock Cycles
0
100
200
300
400
500
36
34
34
134
131
132
518
16
64
256
FFT Size (N)

TOOLS USED

- We used Vivado for behavioral, post Implementation simulations
- The implementation was based on the ZedBoard FPGA
- All code is written with Verilog
- Algorithms were tested in software with C++
- Python for synthetic data (input, output, twiddles) and analysis (similarity scores, plots etc.)



Display Name	Preview	Vendor	File Ver...	Part
Nexys A7-100T		digilentinc.com	1.3	xc7a100tcsg324-1
ZedBoard Zynq Evaluation and Development Kit Add Daughter Card Connections		em.avnet.com	1.4	xc7z020clg484-1
Artix-7 AC701 Evaluation Platform Add Daughter Card Connections		xilinx.com	1.4	xc7a200tfbg676-2
Kintex UltraScale+ KCU116 Evaluation Platform		xilinx.com	1.3	xcku5p-ffvb676-2-e

```
module fft_64_top #(
    parameter WIDTH = 32,
    parameter N = 64
)(
    input                   clock,
    input                   reset,
    input                   input_en,
    input       [WIDTH-1:0] input_real,
    input       [WIDTH-1:0] input_imag,
    output                  output_en,
    output      [WIDTH-1:0] output_real,
    output      [WIDTH-1:0] output_imag
);

// Internal signals between stages
wire stage1_output_en;
wire [WIDTH-1:0] stage1_output_real;
wire [WIDTH-1:0] stage1_output_imag;

wire stage2_output_en;
wire [WIDTH-1:0] stage2_output_real;
wire [WIDTH-1:0] stage2_output_imag;
```

PROBLEMS FACED

- Memory bandwidth bottlenecks in the unrolled versions forced us to implement a memory controller to feed the data. This decision changed the algorithm's complexity from O(log N) to O(N). Theoretically, using an AXI master interface could improve the data rate (bigger k), but even then, the unrolled algorithm remains not scalable for small FPGAs
- We encountered difficulties implementing the SDF algorithm for the Radix-4 FFT.


END OF PRESENTATION
CLAP NOW!!