256 Point Fast Fourier Transform using Avalon Bus

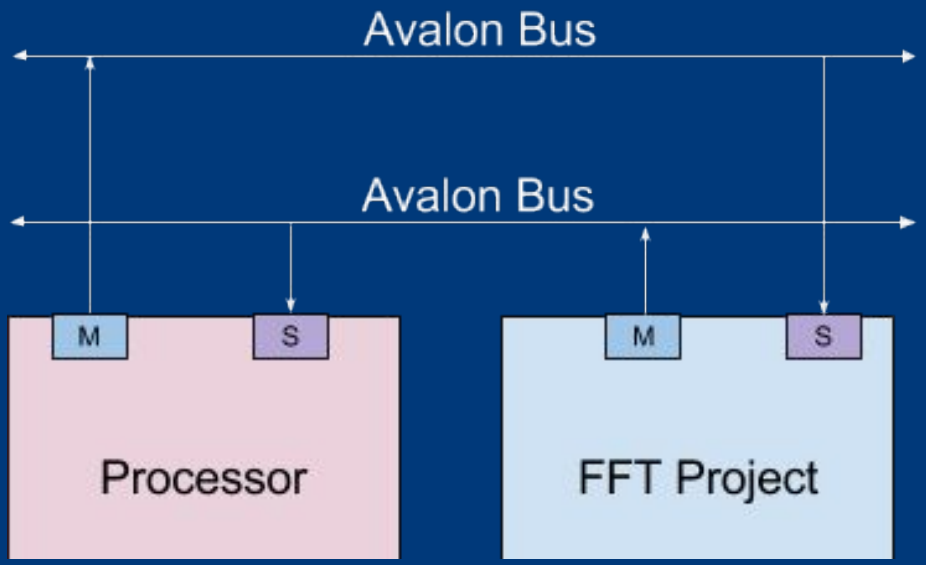
Team Members:
Erin Hill
Charles Li
Luke McBee
Thao Nguyen

Project Overview

- Intended Function
  - 256 point time domain sequence to the frequency domain
  - Cooley–Tukey FFT algorithm
  - Reading and writing the samples on an Avalon bus
  - Optimize both chip size and operation time
- Why someone would use it
  - The Fourier Transform is fundamental to modern day DSP
  - Speed of FFT
- Why ASIC?
  - AISCs are much faster and more efficient than a microprocessor
  - Needs to be fast for live sound processing

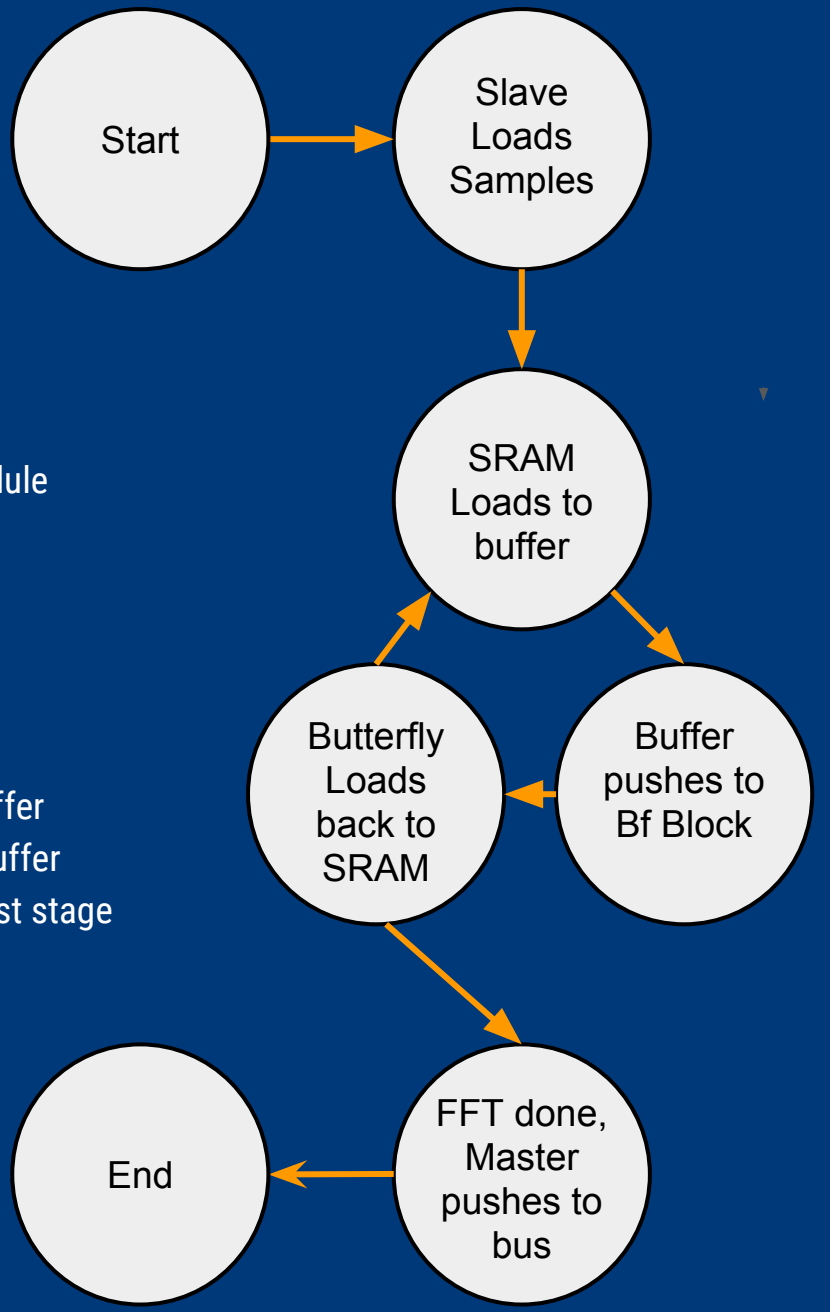
Interfacing to External Devices

- No peripherals necessary
  - Automatically processes the samples
  - Responds to a write to 0x1FF
- Avalon Buses
  - One master and one slave
  - Can be implemented on FPGA
  - Loads the SRAM, then writes the output to another bus

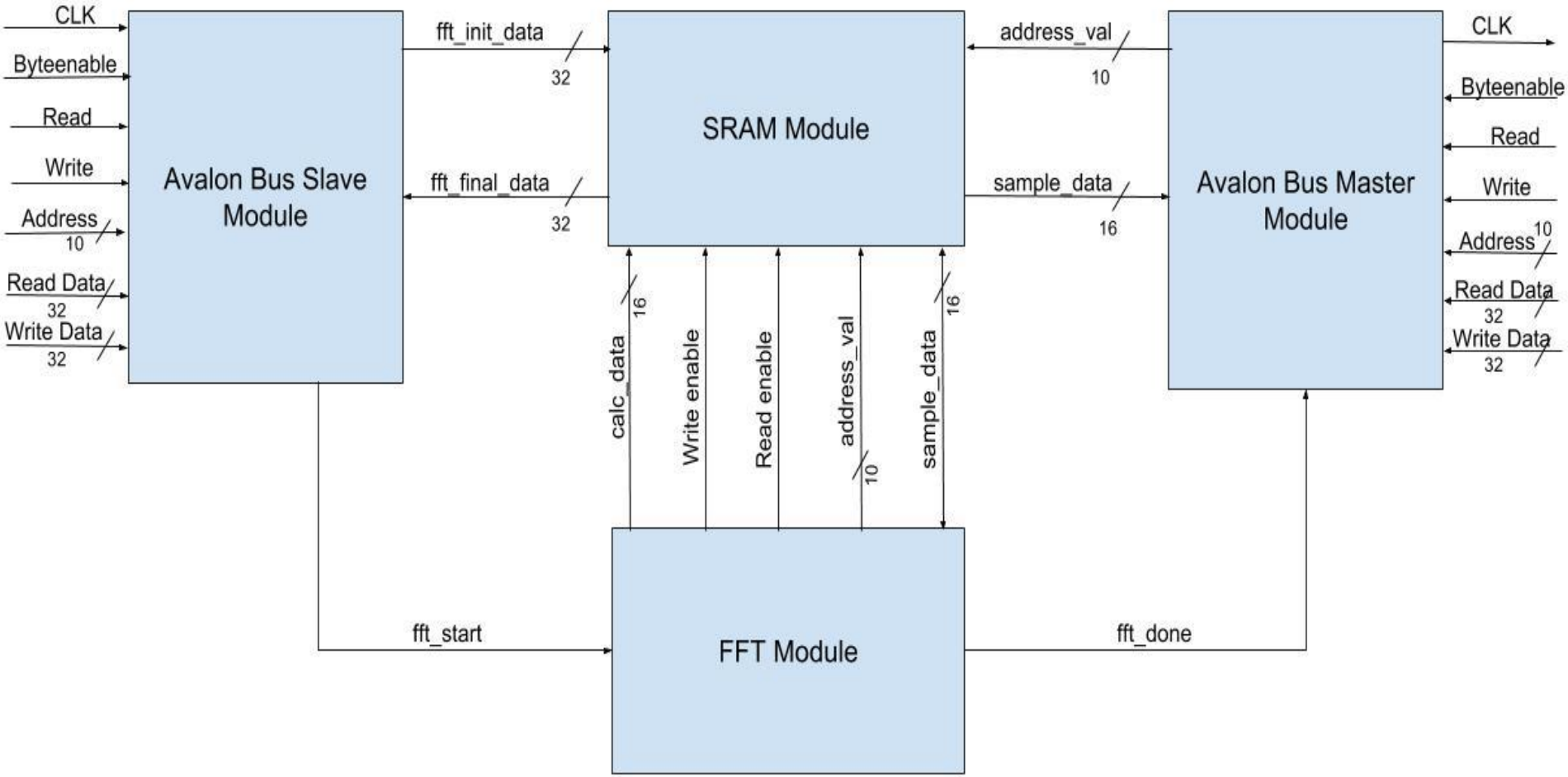


Sequence of Operations

- Avalon Slave -> SRAM
  - External master loads samples, then begins operation
  - Master has access to all memory addresses in the module
  - Half real, half imaginary
  - Initiates the algorithm
- SRAM -> FFT, FFT -> SRAM repeating cycles
  - Process is broken up into groups and stages
  - Samples to be loaded based on address block
  - Proper samples+twiddle values loaded into register buffer
  - Register buffer feeds into butterfly block, output into buffer
  - Buffer is fed back into SRAM, process repeated until last stage
- SRAM -> Avalon Master
  - Upon completion, writes values to external module
  - Dumps memory in consecutive order



High-Level Architecture Diagram



Major Design Decisions and Rationale

Number of Butterfly Blocks

- Decision Made : 8 blocks
- Rationale :
  - Fast yet space efficient
  - Must be less than 16 mm^2

# of Butterfly Blocks	Total # of groups	Area per bfb (um^2)	Area BFB (mm^2)	Flip-flops (mm^2)	Total Area (mm^2)
1	2048	349500	0.3495	0.384	0.7335
2	1024	699000	0.699	0.768	1.467
4	512	1398000	1.398	1.536	2.934
8	256	2796000	2.796	3.072	5.868
16	128	5592000	5.592	6.144	11.736
32	64	11184000	11.184	12.288	23.472
64	32	22368000	22.368	24.576	46.944
128	16	44736000	44.736	49.152	93.888
256	8	89472000	89.472	98.304	187.776

Using SRAM or Register blocks

- Decision Made : SRAM
- Rationale :
  - SRAM is significantly smaller
  - Neither the SRAM or flip-flops would be the critical path
  - Registers would allow for more control, but more effort to write

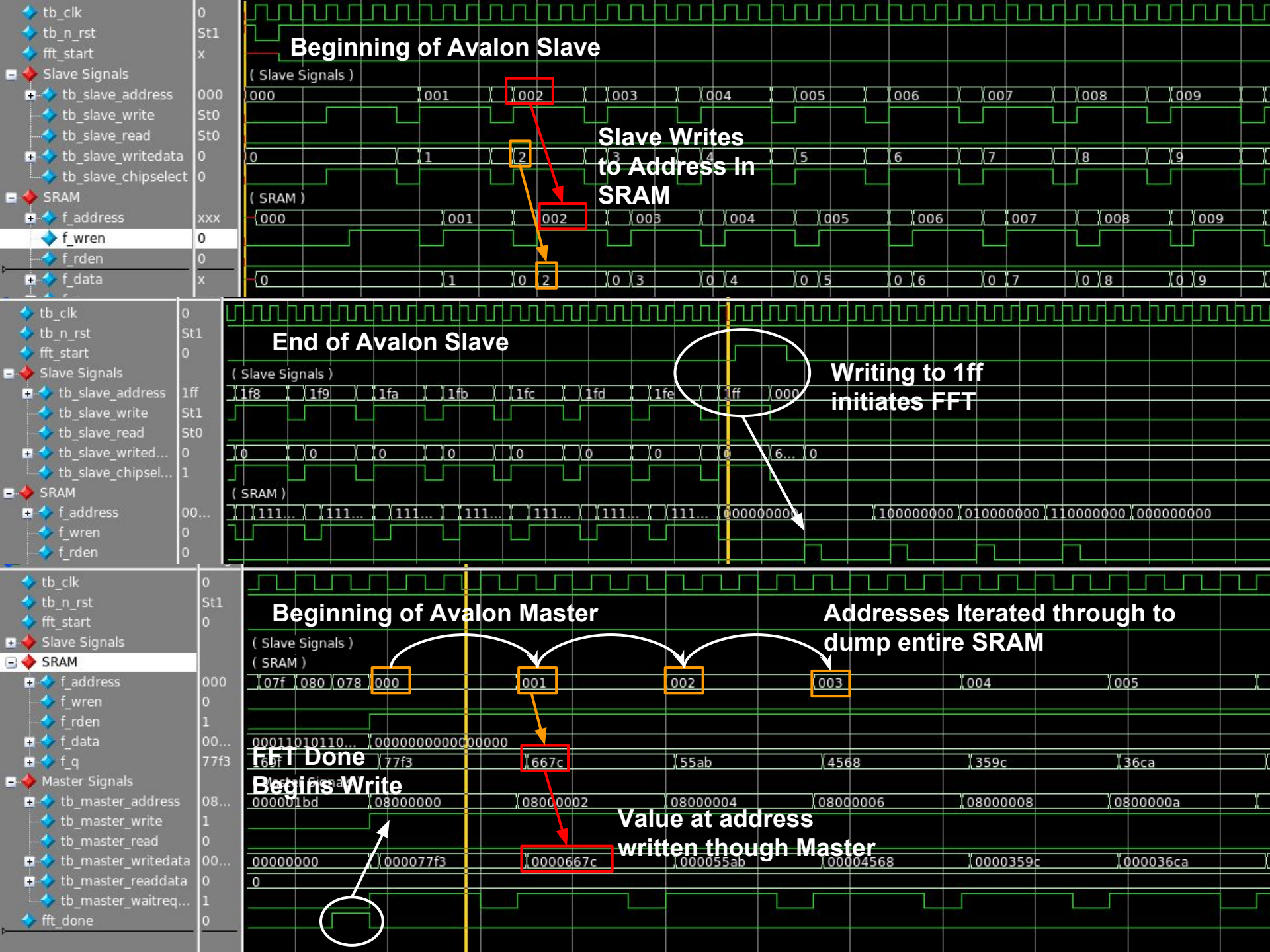
# of Points	256
# of Values	512
# of bits	8192
Area On-chip SRAM (mm^2)	0.4096
Area flip-flops (w/out reset) (mm^2)	7.3728

SRAM

Flip-flops

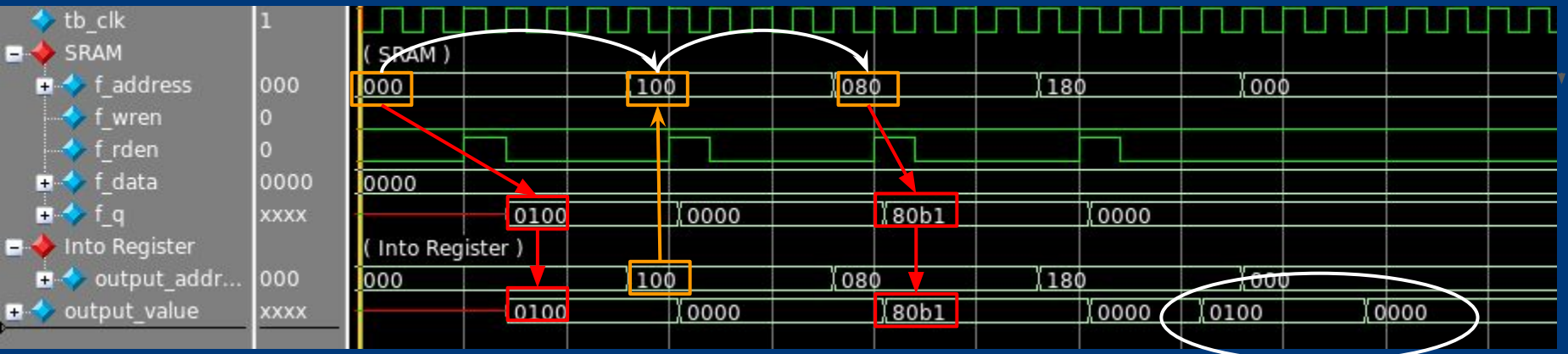
Design Specific Criteria

1. Demonstrate via System Verilog test bench the ability to responds to Avalon writes to on chip SRAM in the order required for FFT processing. (1pt)
2. Demonstrate via System Verilog test bench the ability to write the on-chip SRAM to a desired address, in the correct order, then bring the whole module to idle. (1pt)
3. Demonstrate via System Verilog test bench of separate the correct frequency output for a given time domain samples. (1 pt)
4. Demonstrate via System Verilog test bench that the Main Controller Unit produces the appropriate signals to the timer modules which will in turn signal data loads, calculations from the butterfly block, and data writes. (1 pt)
5. Demonstrate via System Verilog test bench that the combinational logic for the Butterfly block is operational and produces the expected results. (2 pts)
6. Demonstrate via System Verilog test bench that the Address Controller algorithm is operational and produces expected results. (2 pt)


tb_clk
tb_n_rst
fft_start
Slave Signals
tb_slave_address
tb_slave_write
tb_slave_read
tb_slave_writedata
tb_slave_chipselect
SRAM
f_address
f_wren
f_rden
f_data
Beginning of Avalon Slave
Slave Writes to Address In SRAM
End of Avalon Slave
Writing to 1ff initiates FFT
Beginning of Avalon Master
Addresses Iterated through to dump entire SRAM
FFT Done Begins Write
Value at address written though Master
Master Signals
tb_master_address
tb_master_write
tb_master_read
tb_master_writedata
tb_master_readdata
tb_master_waitreq...
fft_done
f_q

SRAM Addressing

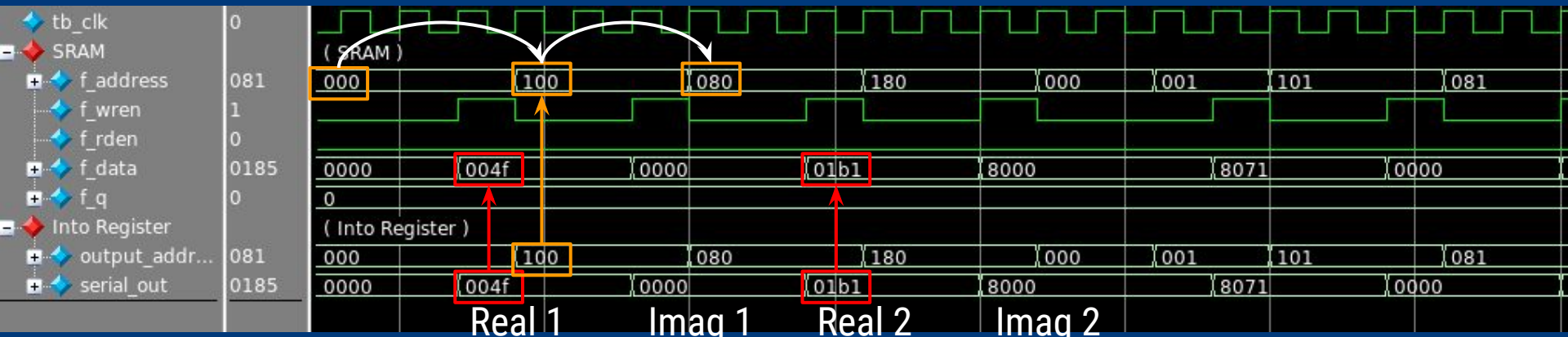
SRAM to FFT



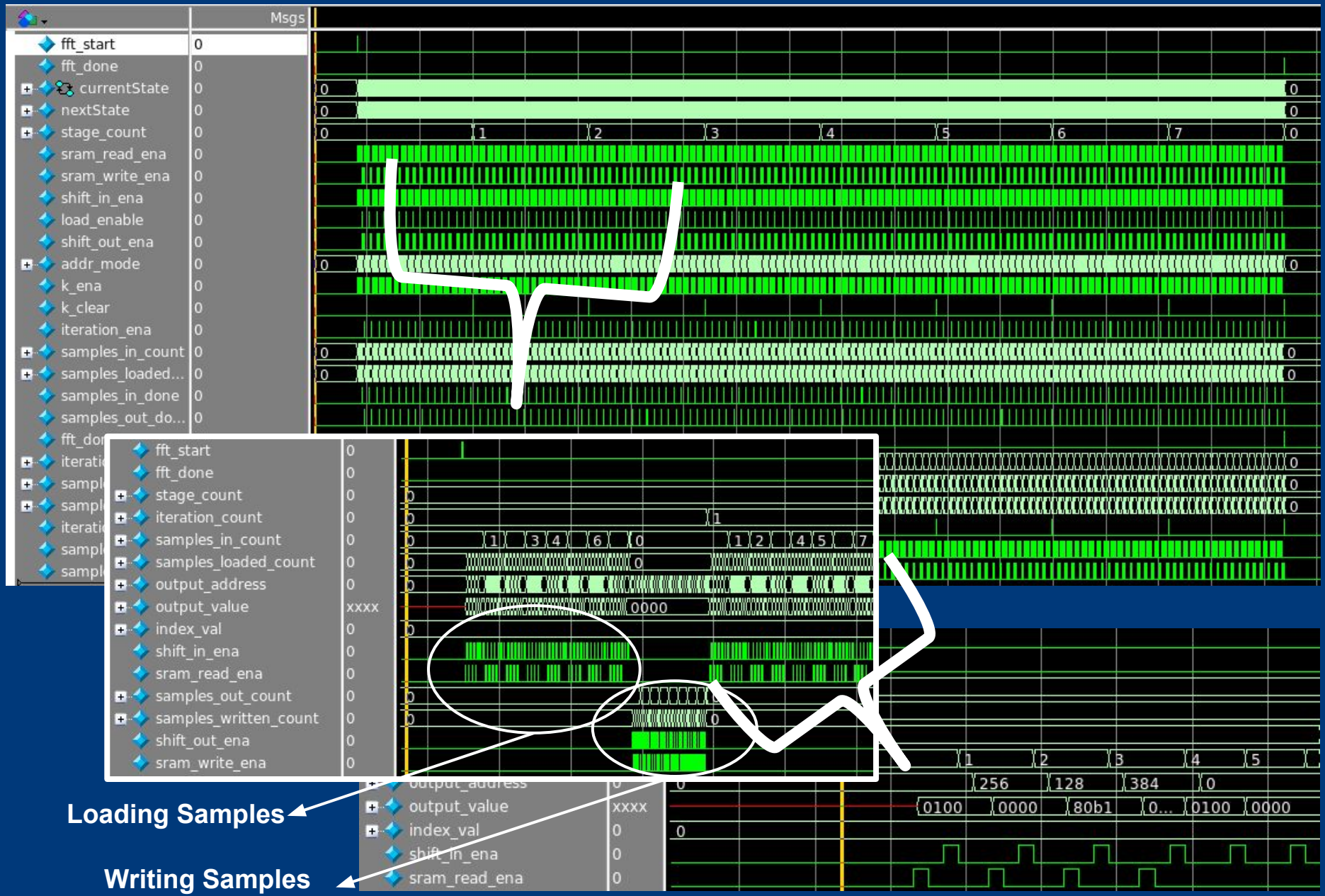
Overwrite to same values

Twiddle Values from Twiddle ROM

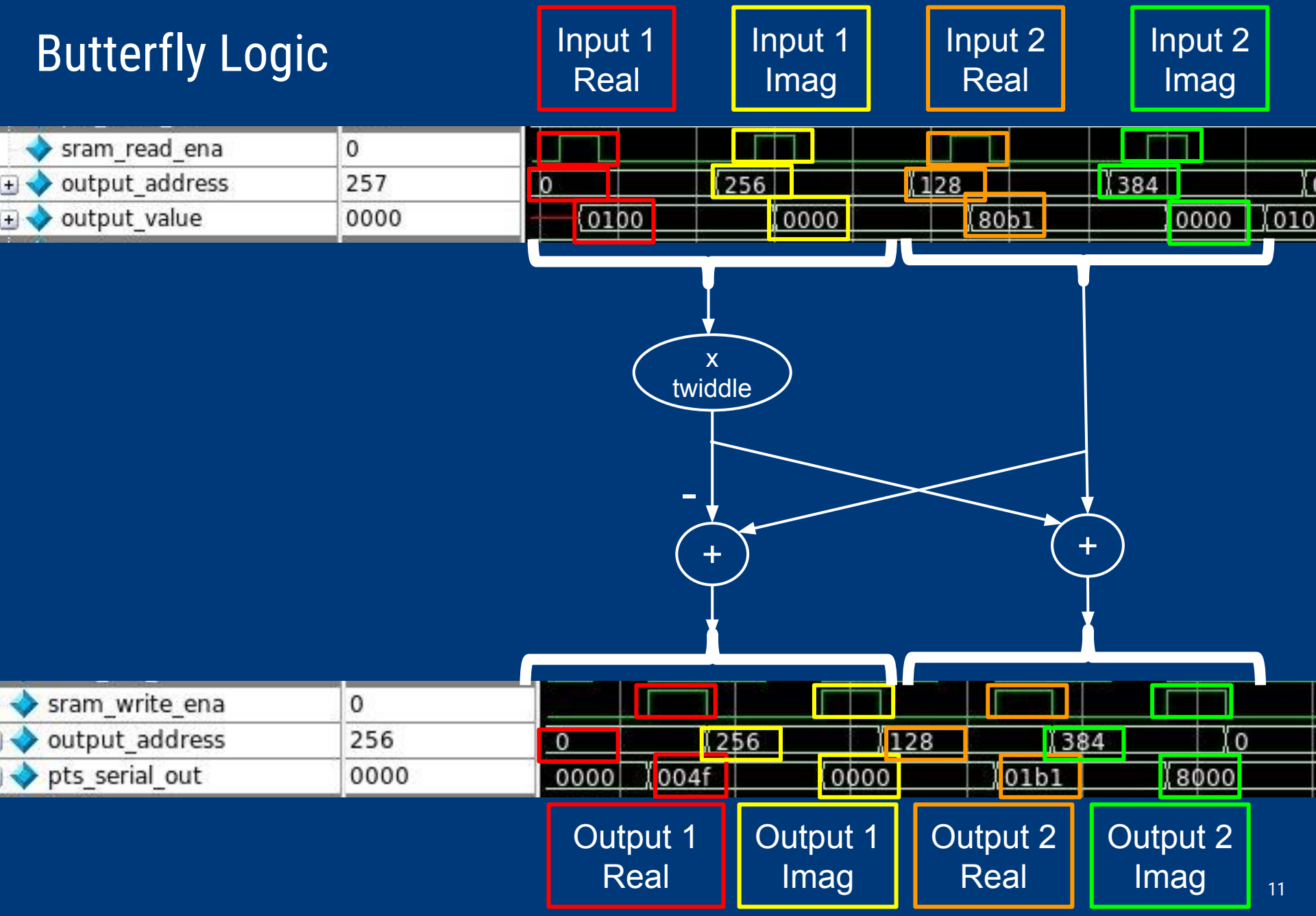
FFT to SRAM



Main Controller Unit & Timers



Butterfly Logic



Fixed Criteria

1. Test benches exist for all top level modules
2. Mapped design synthesizes completely without any inferred latches, timing arcs, and sensitivity list warnings
3. Source and mapped version of the complete design behave the same for all test cases
4. Complete IC layout is produced and passes all geometry and connectivity checks
5. Entire design satisfies
    a. Area : 4mm x 4mm
    b. Pin Count : 164

Conclusion

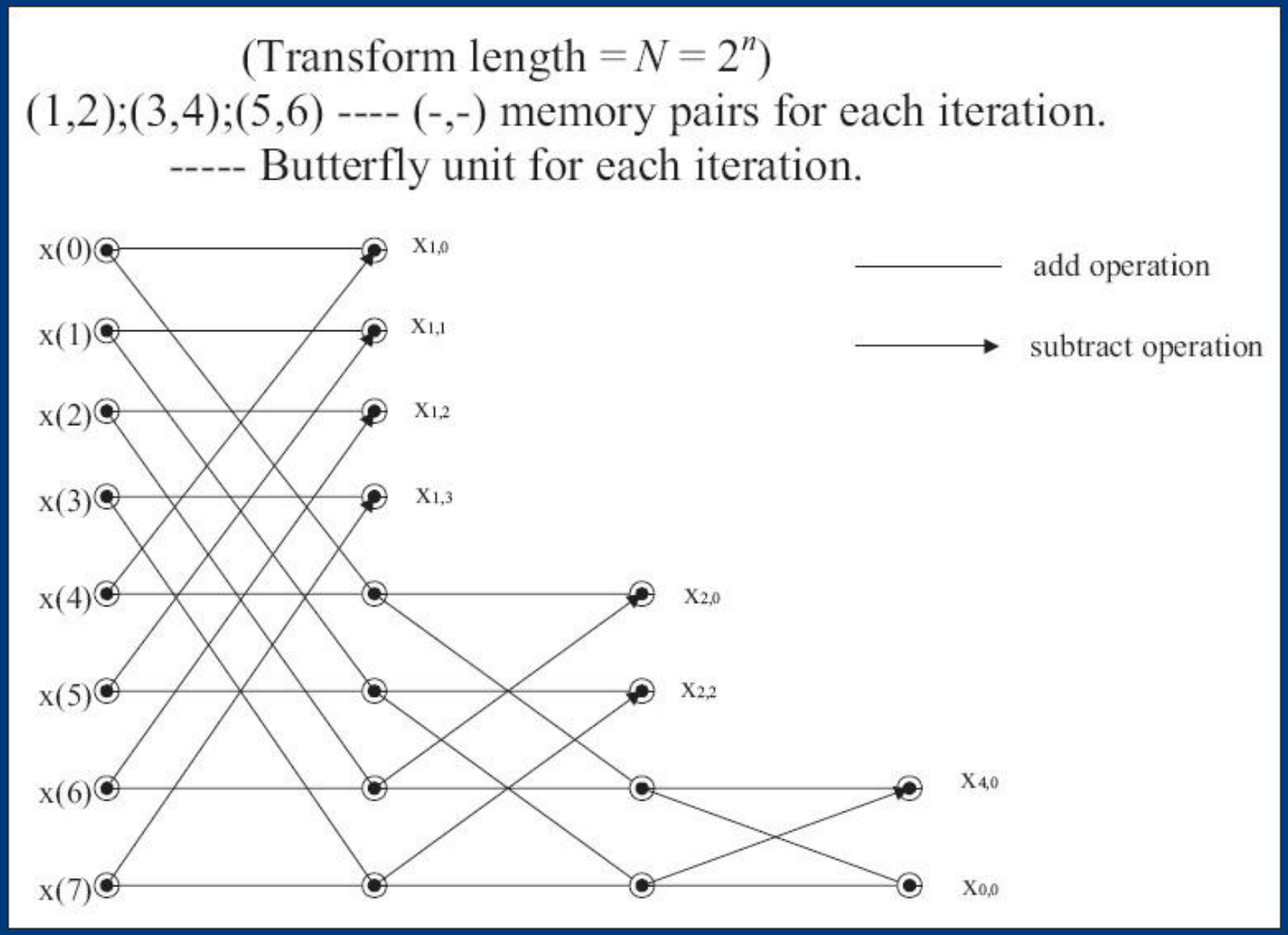
- Biggest challenges in design
  - Loss of precision when doing fixed point math
  - Figuring out how to read and write from the correct memory address
- How we would do it differently
  - Included more precision in decimal section (instead of 7, 8 , 2 ,12)
- What improvements could we make with more time
  - Increased FFT precision
  - Put on FPGA
  - Improve efficiency

Butterfly Computation

AR’ = AR + (BR * TR - BI * TI)
AI’ = AI + (BR * TI + BI * TR)
BR’ = AR - (BR * TR - BI * TI)
BI’ = AI - (BR * TI + BI * TR)

AR’ = 0100 + (80b1 * 0100 - 0000 * 0000) = 004f
AI’ = 0000 + (80b1 * 0000 + 0000 * 0100) = 0000
BR’ = 0100 - (80b1 * 0100 - 0000 * 0000) = 01b1
BI’ = 0000 - (80b1 * 0000 + 0000 * 0100) = 0000

High Level Butterfly Block Overview



http://www.onlinejet.net/article.asp?issn=0976-8580;year=2011;volume=1;issue=2;spage=83;epage=87;aulast=Ranganadh

Full Architecture Diagram