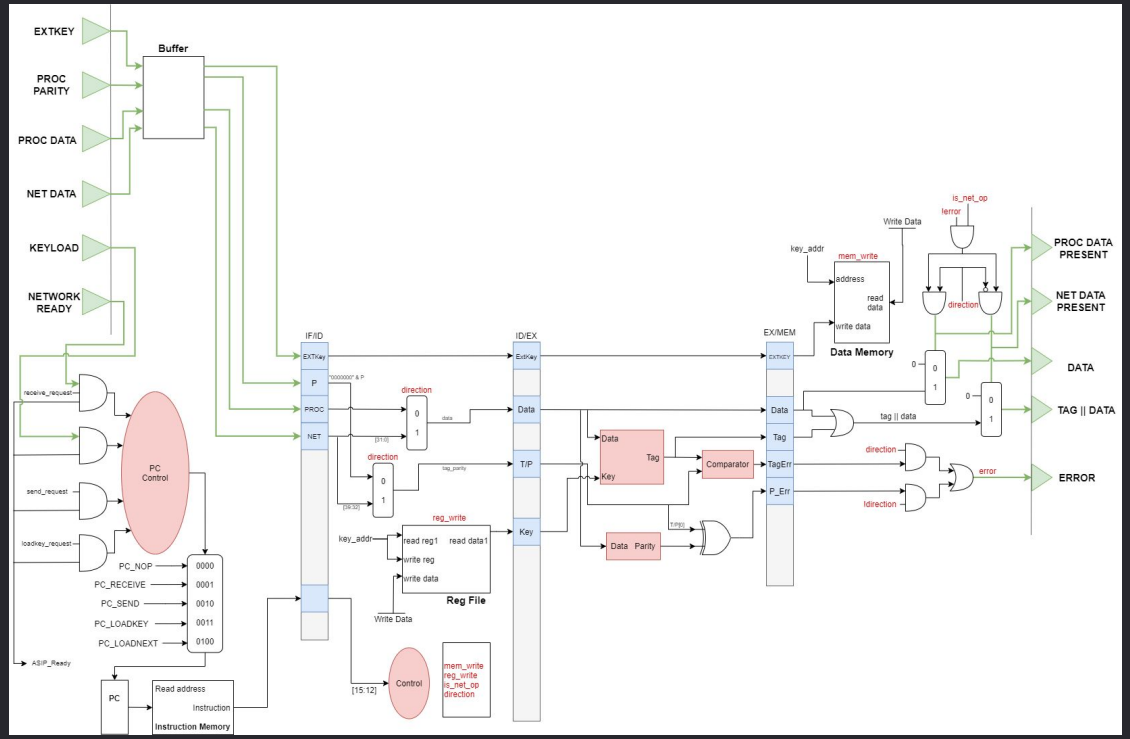
COMP3211 PROJECT

ANDREW WONG (z5206677) | GABRIEL TAY WEI CHERN (z5216632) | LINDSAY SMALL (z5232920) | MALAVIKA PASUPATI (z5209390) | QUYNH BOI PHAN (z5205690)

1.
SCHEMATIC

ASIP DESIGN

BLOCK DIAGRAM



I/O REGISTERS

Input

- External Key [7:0]
- Processor Data [31:0]
- Network Data [39:0]
- Processor Parity Bit

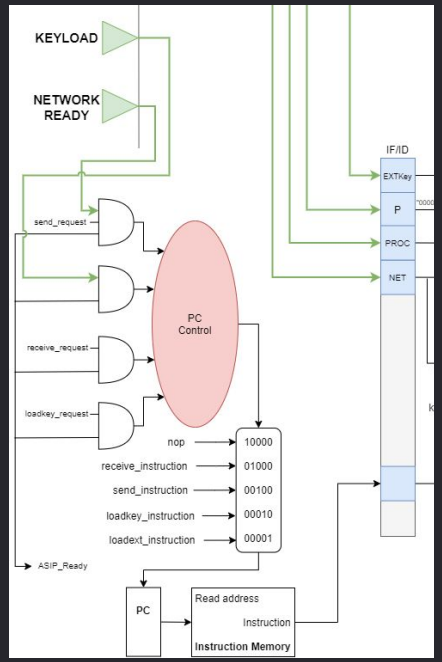
Output

- Tag || Data [39:0]
- Data [31:0]

Flags

- ASIP Ready
- Network Ready
- Load Key (Loadext Request)
- Send Request
- Receive Request
- Error
- Net Data Present
- Proc Data Present

IF STAGE - OTHER COMPONENTS



PC

Sends the address to read in the *instruction memory*.

MUX

Changes PC value based *PC Control*.

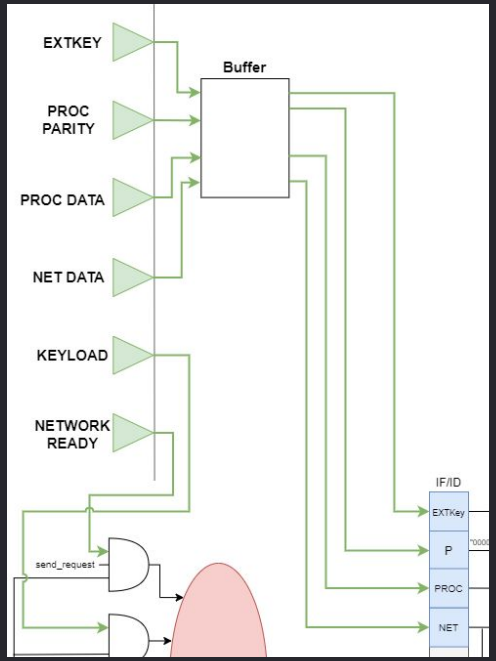
PC Control

Alters PC based on incoming requests and ready states of ASIP and network.

Instruction Memory

Memory of the instructions that needs to be executed.

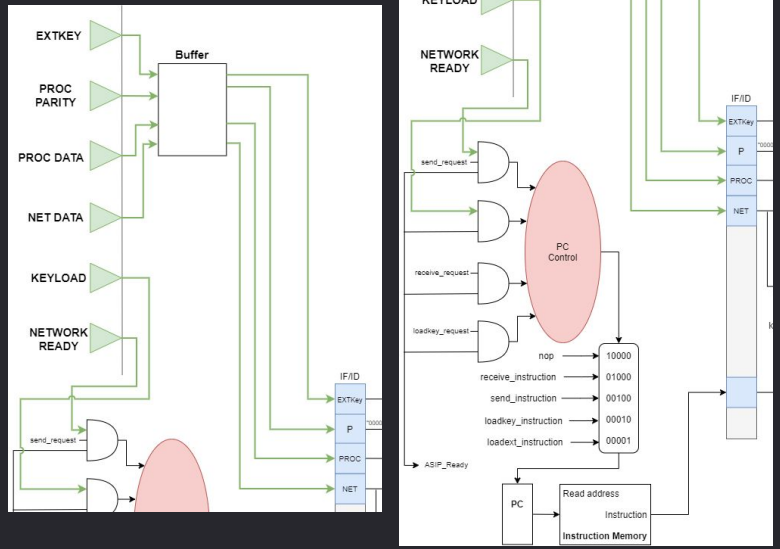
IF STAGE - INPUTS AND BUFFER REGISTER



Buffer Register

Store incoming data to account for jump delay.

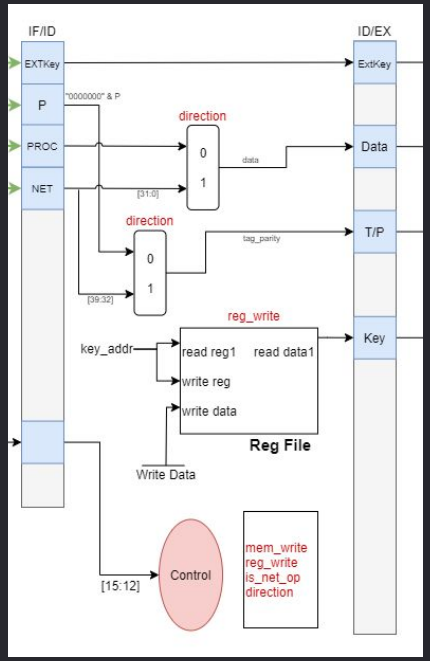
IF/ID PIPELINE INPUTS



IF/ID Pipeline Inputs

- External Key
- Parity Bit
- Processor and Network Data
- Instruction Memory

ID STAGE - COMPONENTS



Control

Sends control signals to the ASIP.

RegFile

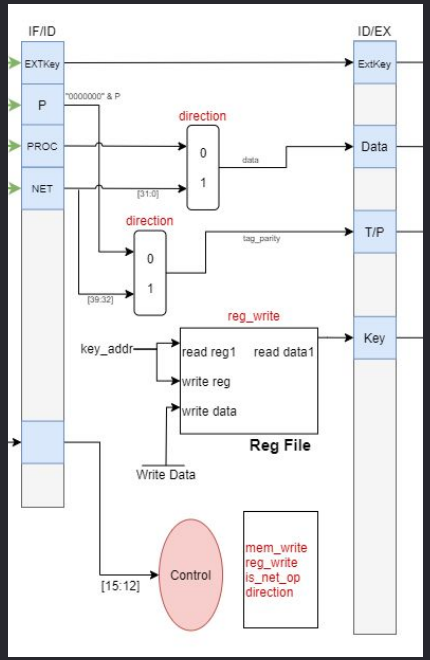
Reads the constant address *key_addr* to output the key data. The same address is used when a new key data is given.

Direction MUXes

Selects data to use for the process

- When LOW: use processor data and parity
- When HIGH: use network data

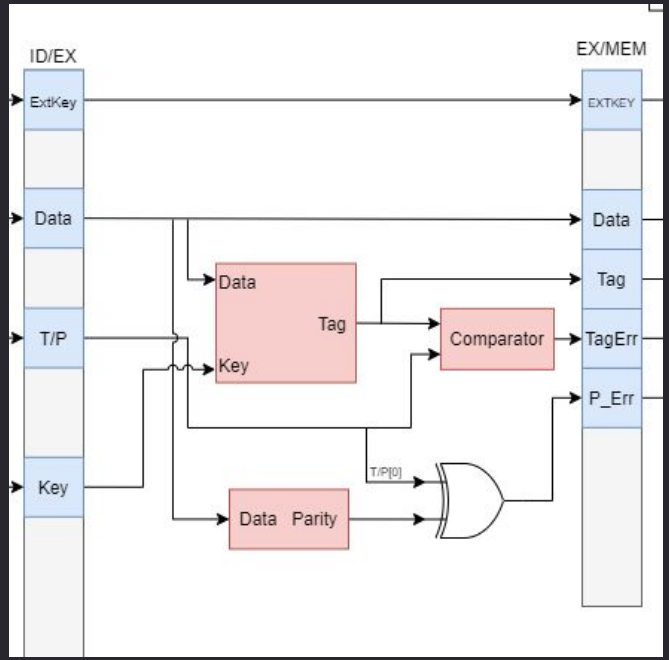
ID/EX PIPELINE INPUTS



ID/EX Pipeline Inputs

- Control Signals:
  - mem_write
  - reg_write
  - is_net_op
  - direction
- Data
- Tag + Parity bit
- External Key (From IF/ID)
- Secret Key

EX STAGE - COMPONENTS



Tag Generator

Generate a tag from the key and data given.

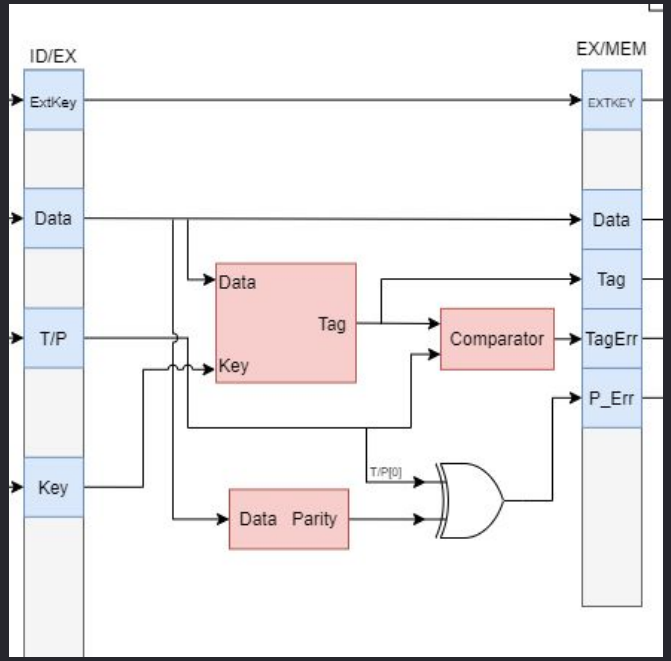
Parity

Get the parity bit from the data.
This is then compared with the received parity bit from the processor to see if they are the same.

Comparator

Compares the tag generated by the tag generator to the tag given by the network.

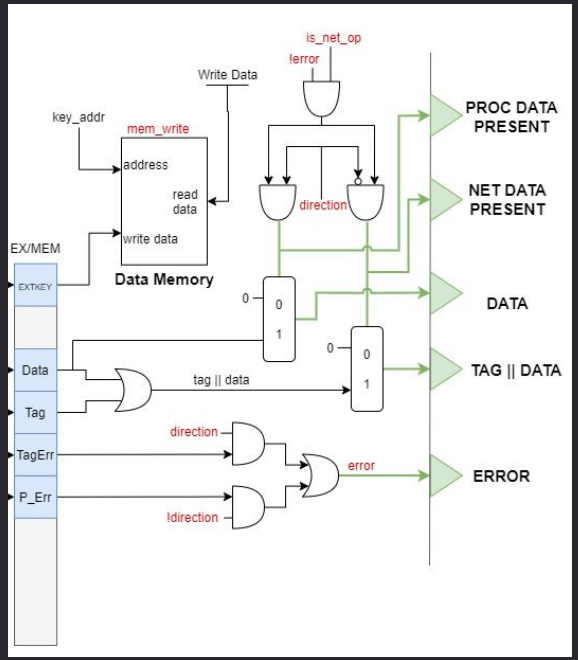
EX/MEM PIPELINE INPUTS



EX/MEM Pipeline Inputs

- Control Signals
  - mem_write
  - reg_write
  - is_net_op
  - direction
- Tag
- TagErr (Tag error)
- P_err (Parity bit error)
- P (Parity bit)
- Data (From ID/EX)
- External Key (From ID/EX)

MEM STAGE - SIGNALS



Direction

Indication of the data's direction.

- When LOW: Network → Processor System
- When HIGH: Processor System → Network

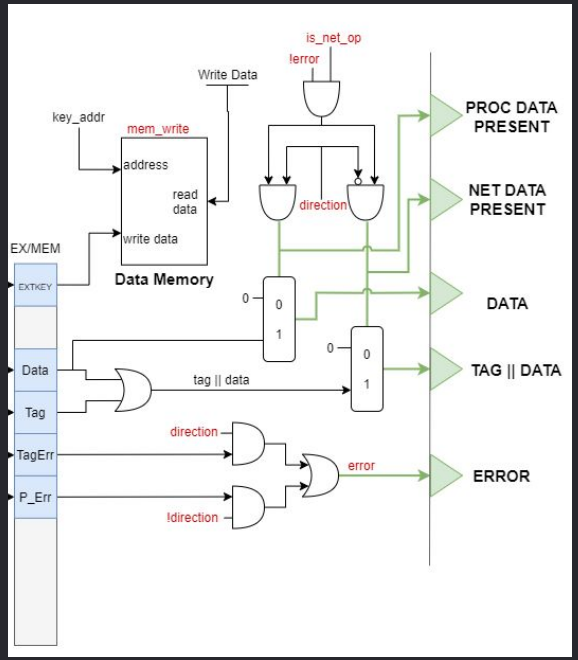
Error

The *error* signal is HIGH when the parity bits or the tags are not the same, indicating that there is a software error or an integrity attack.

is_net_op

The *is_net_op* signal is HIGH if the current operation is a send or receive operation.

MEM STAGE - COMPONENTS AND OUTPUTS



Data Memory

Outputs data from *key_addr*. Data from *key_addr* can only be overwritten if *mem_write* is HIGH.

TAG || DATA MUX

Allows data to be sent only if conditions are met by the AND operation (*is_net_op*, *!direction*, *and !error*)

2.
ISA

Instructions and Signals

ARGUMENT FORMULATION

INSTRUCTIONS

OP_SEND

Send data and tag to network

OP_RECEIVE

Receive data from network

OP_LOADKEY

Load key from data memory to secret register.

Hardcoded to load from data #0

OP_LOADEXT

Load key from external port into data memory

Hardcoded to store to data #0

OPERANDS

OP_SEND

Net Data

OP_RECEIVE

Proc Data

OP_LOADKEY

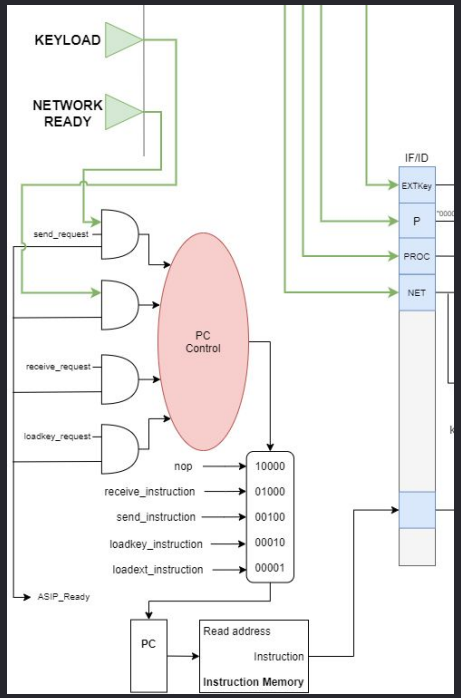
N/A

OP_LOADEXT

EXTKEY

PC CONTROL

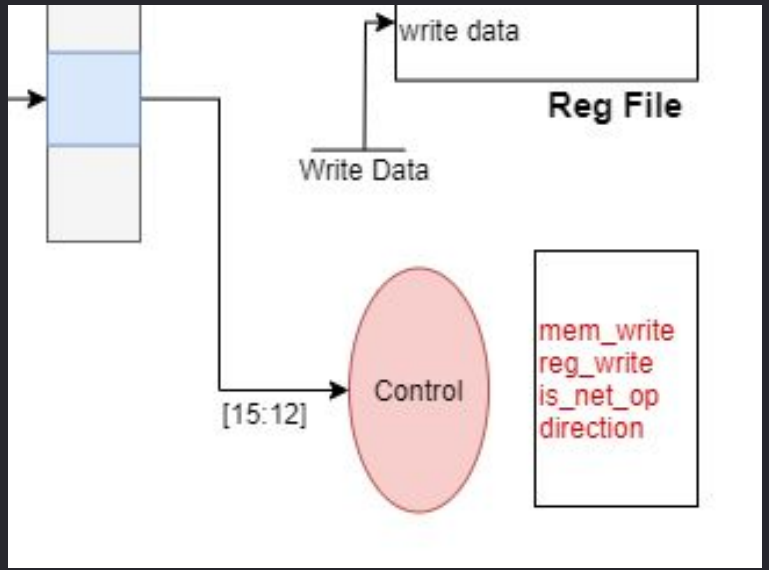
- PC will jump to the instruction needed to be executed instead of using PC + 1
- PC Control determines which instruction to jump to:
  - Utilises request signals to determine jump location
  - ASIP needs to be ready before any jumps is made



CONTROL SIGNAL INPUTS

opcode (instruction [15:12])

Contains the command needed to be executed by the ASIP



CONTROL SIGNALS - OUTPUTS

mem_write

Overwrites the data of the original secret key used in the ASIP when HIGH.

reg_write

Writes the data of the secret key onto the register of *key_addr*.

is_net_op

Indicates if the current operation is a network (send data/receive data) operation or not.

direction

Indication of the data's direction.

LOW: Network → Processor System

HIGH: Processor System → Network

3.
HDL MODEL

Runthrough of the code

4.
TESTBENCH

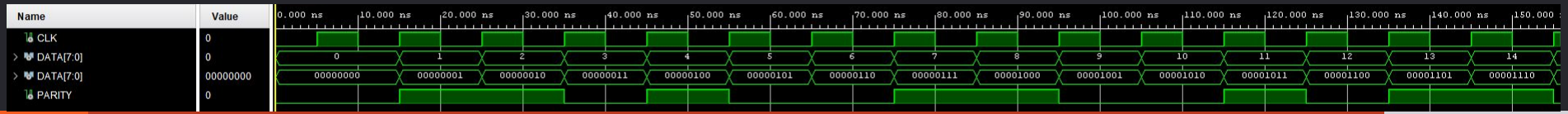
Testbenches created to ensure validity of components and hence system

PARITY CHECKING

Each bit of the input port is XOR'd together. The resulting bit is used as the parity bit.

e.g. 00001011
0 ^ 0 ^ 0 ^ 0 ^ 1 ^ 0 ^ 1 ^ 1 = 1, so the parity bit is 1

e.g. 10100101
1 ^ 0 ^ 1 ^ 0 ^ 0 ^ 1 ^ 0 ^ 1 = 0, so the parity bit is 0



BLOCK FLIP - CASES TESTED

No bits flipped (0000)

- D should remain the same throughout

All bits flipped (1111)

- All bits of D should be flipped throughout

Flipping parts of D (0001, 0011, 0111, 1111, 0101, 1010)

- Only sections of D should be flipped e.g. in 0101 case, D[23:16] and D[7:0] should be flipped whereas D[31:34] and D[15:8] remains the same.

ROTATE LEFT SHIFT - CASES TESTED

No shift (000 000 000 000)

- D should remain the same throughout

Rotate shift left by 1 (001 001 001 001)

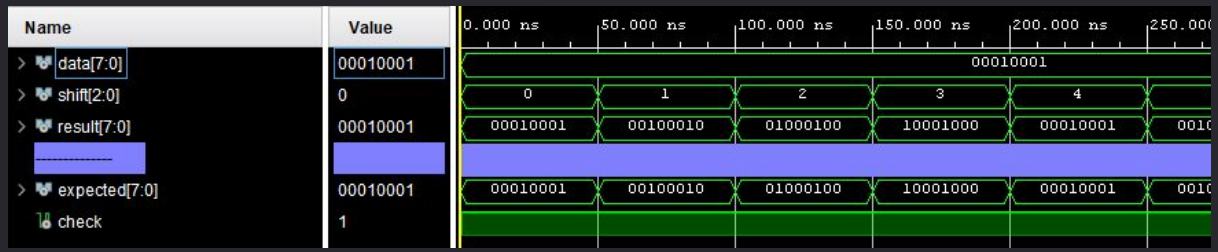
- All bits of D should be shifted to the left by 1.

Rotate shift left by 4 (100 100 100 100)

- All bits of D should be shifted to the left by 4.

Rotate shift left by 7 (111 111 111 111)

- All bits of D should be shifted to the left by 7.



TAG GENERATION

Test different data inputs without bit flip and rotation:

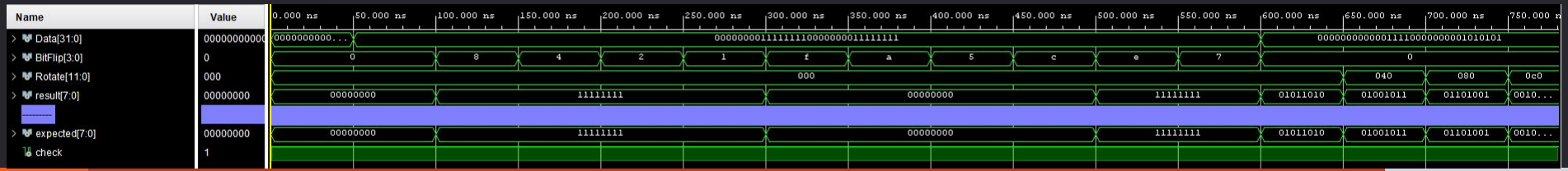
- All 0's
- 0's in D[31:34] and D[15:8], 1's in D[23:16] and D[7:0]

Test different bit flip combinations without rotation:

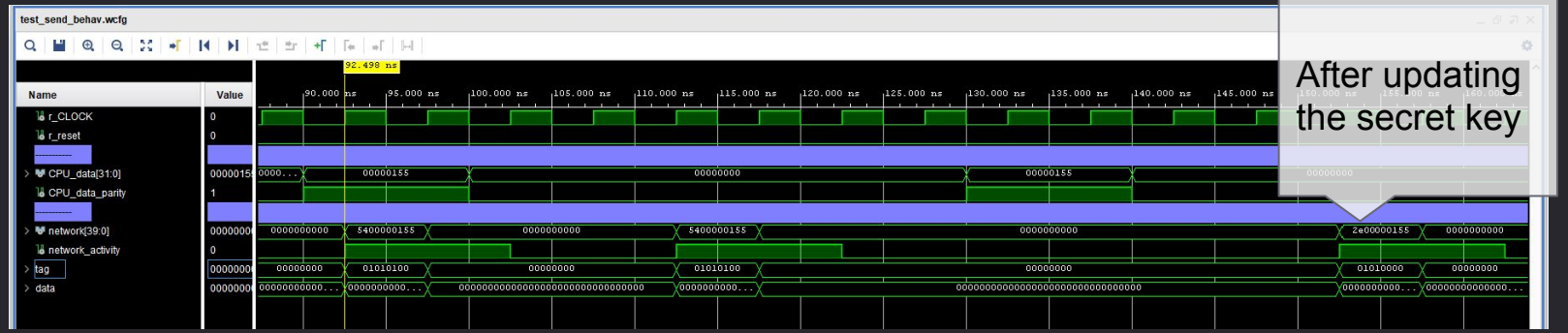
- Cases are similar as before

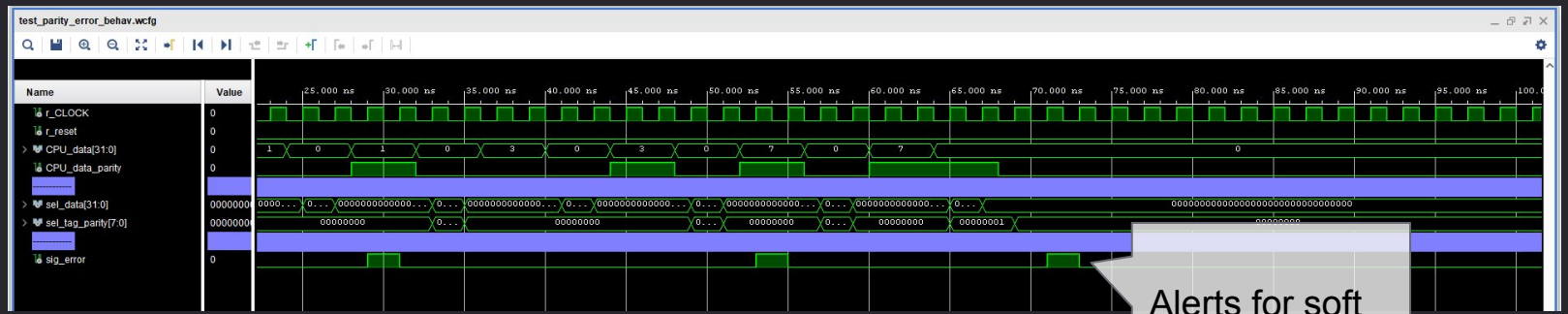
Test different rotate left shift combinations without bit flip:

- Cases are similar as before

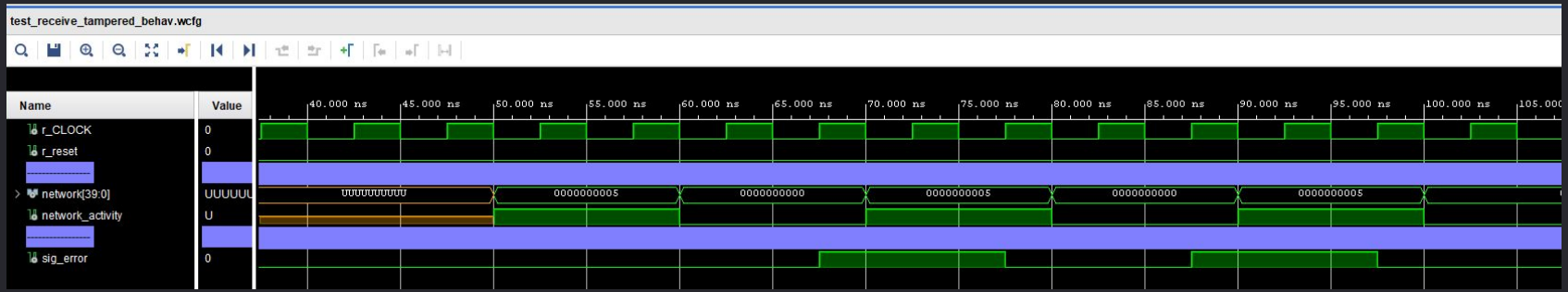


ASIP Integration Test - Sending, Secret, Parity Soft Error

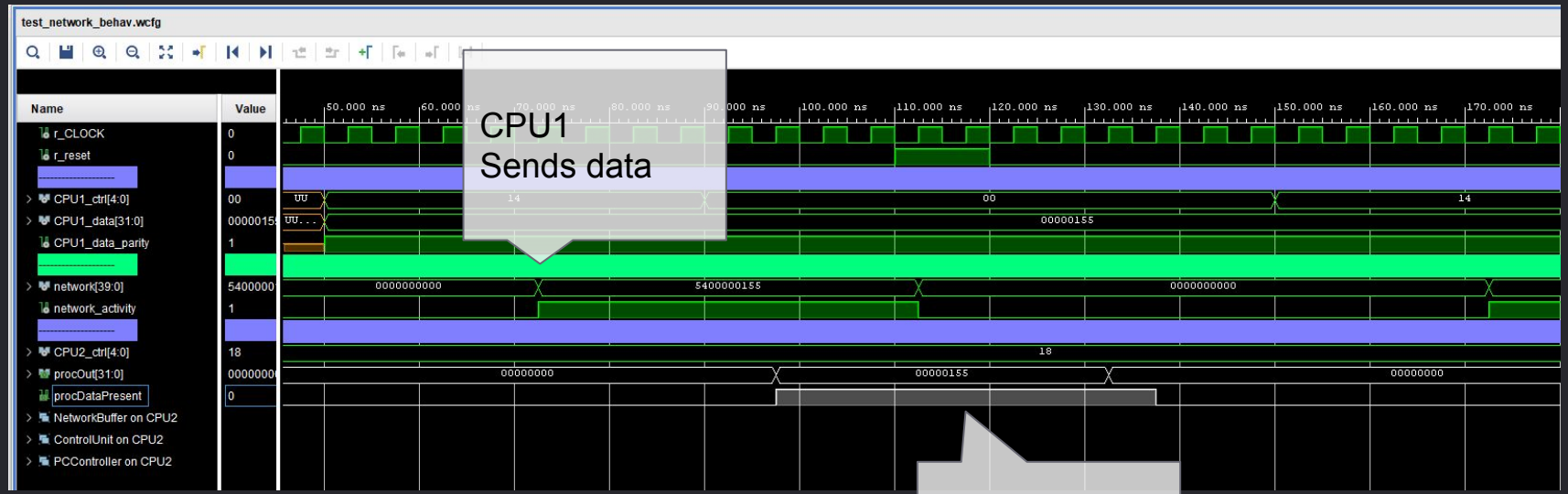




ASIP Integration Test - Receiving tampered data (tag mismatch)



ASIP Integration Test - Basic Simulated Network (Send / Receive)

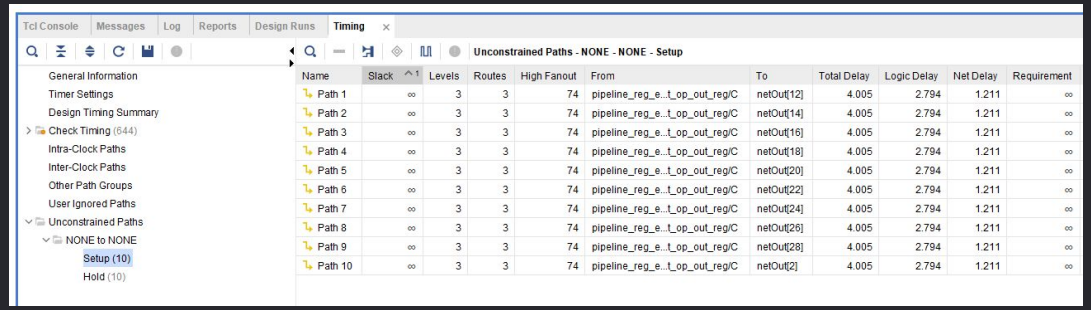


5.
PERFORMANCE

Performance of the system

PERFORMANCE

Latency



Component Count

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP
synth_1	constrs_1	synth_design Complete!								95	241	0.0	0	0
impl_1	constrs_1	route_design Complete!	NA	NA	NA	NA	NA	5.574	0	95	241	0.0	0	0

Netlist Properties

network_coprocessor_ASIP

Primitive Statistics

Primitive type	Count
FLOP_LATCH	241
LUT	135
IO	156
CLK	1

6.
IMPROVEMENTS

Improvements to be made with our current system

IMPROVEMENTS

PROBLEM: The chip of the processor system must be changed if the tag formula changes.

- Break structure down to more basic reusable instructions for a more flexible design.
- Only firmware would be changed if the tag generation algorithm were to change.

PROBLEM: User is assumed to load the key before doing any other commands

- Improve the PC control system and instruction memory - on system starts up, an *init* command will initialise the data required before the user can use the system.

7.
REFLECTIONS

Reflecting on the process of building the ASIP

Reflections

- Task delegation and overall collaboration
  - Delegate task evenly so the ASIP could be consistently worked on and improved.
  - Design several ASIP structures, discussed and compared. This can be a collaborative effort but implementation is difficult to distribute.
- Communication
  - Difficulties communicating online.
  - Many had commitments outside the project.
  - More frequent meetings to address each other's queries.
- Was confused with parts, needed clarification from the tutor.
  - Took away time to work on project.

THANKS!

ANY QUESTIONS?