EE533 NETWORK PROCESSOR DESIGN & PROGRAMMING

LAB#3: (Mini-Intrusion Detection Engine Design)

Instructor: Prof. Young Cho, PhD

CREATED AND COMPILED BY:

SARTHAK JAIN

(MS EE, UNIVERSITY OF SOUTHERN CALIFORNIA)

GITHUB LINK FOR MY REPOSITORY:

You'll find all the codes and the executables on this repository.

https://github.com/SARTHAK-JAIN-ASIC/EE533/tree/main/LAB3

QUESTIONS:

1. Look at the created Verilog. Do they make sense? Which do you think easier: entering the schematics or writing Verilog? Why? In which cases might you do the other?

A1:

Generated code is gibberish to us as it is generated by the tool using very specific instance names, wire names, module names etc. While the simulation would perfectly fine, making heads or tails of the generated code is a challenge.

It would personally be way easier to write rtl logic for a huge circuit like this as code logic can be reused easily. Behavioral model of HDL utilizes much less effort to design logic than it would take to create a schematic. The debugging process in case something goes wrong is also easier.

2. What does the testbench do?

A2:

For a mini intrusion system, we need an input data stream, a pattern matcher logic that detects certain intrusion patterns in the input data stream and a output FIFO wrapper to drop the intrusive pattern matched packets and write the rest of the data into a dual port FIFO (which essentially decouples the packets (intrusion free) from the 'to be processed circuit')

The test bench has 4 components for this:

1. fallthrough fifo

Essentially this is the fifo we use to send in the packets to our pattern detection logic (again, this is a decoupler that decouples our input data stream from the pattern matching logic)

2. detect7B matcher

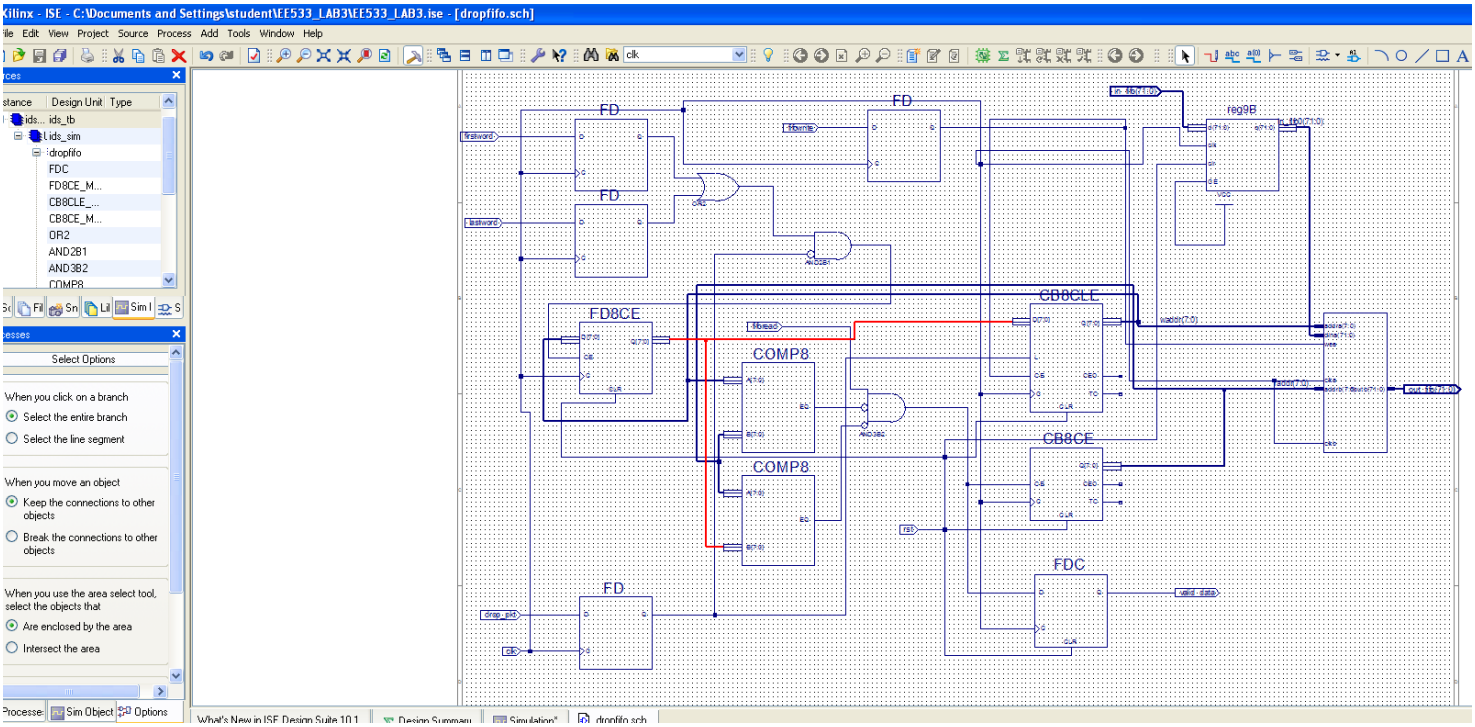
This is the logic that detects the "intrusive data bit pattern" with the input data stream from the fallthrough fifo.

3. dropfifo

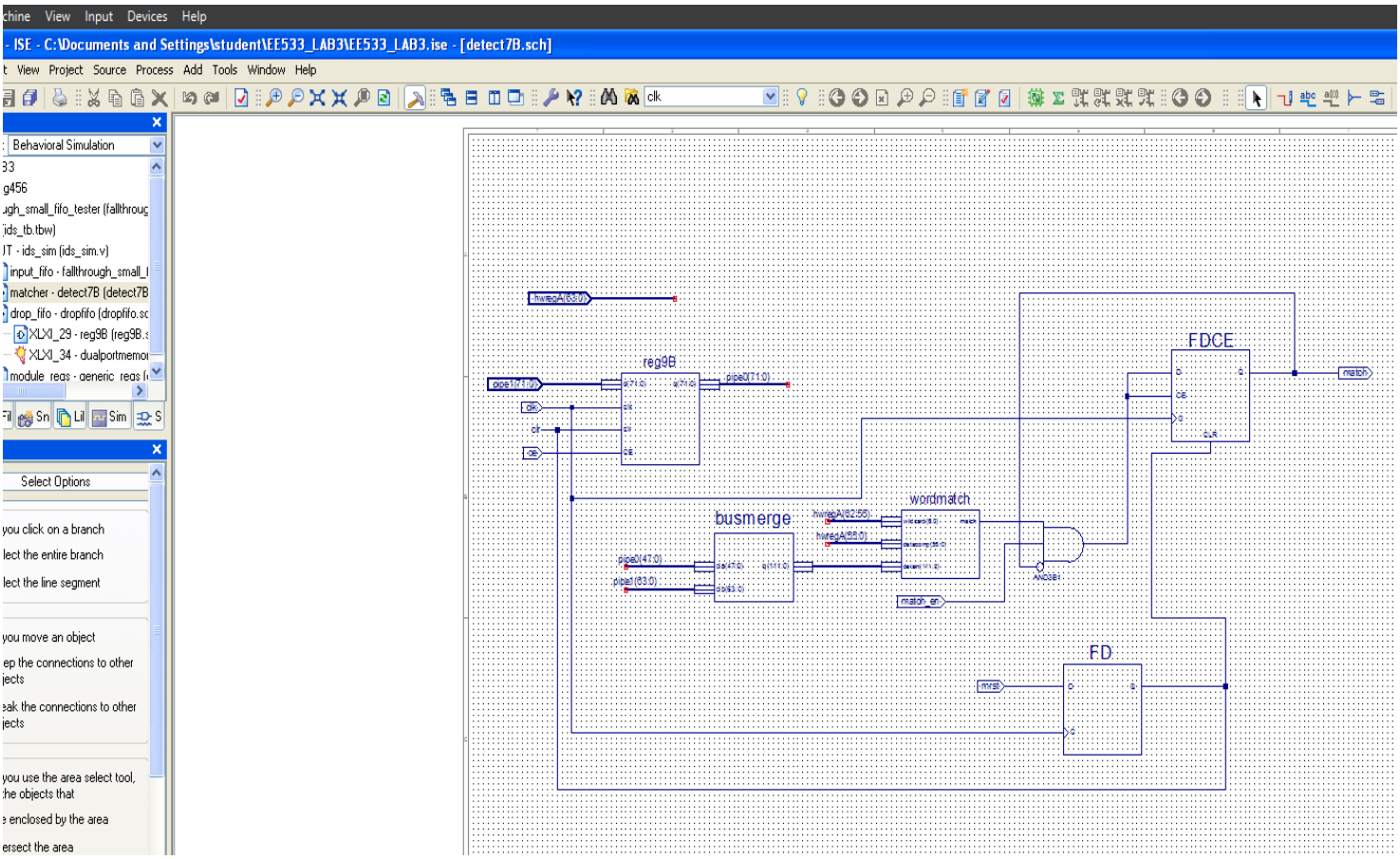
This is the fifo wrapper that has the logic to drop "pattern matched" intrusive packets from the data stream

4. Packet separation logic

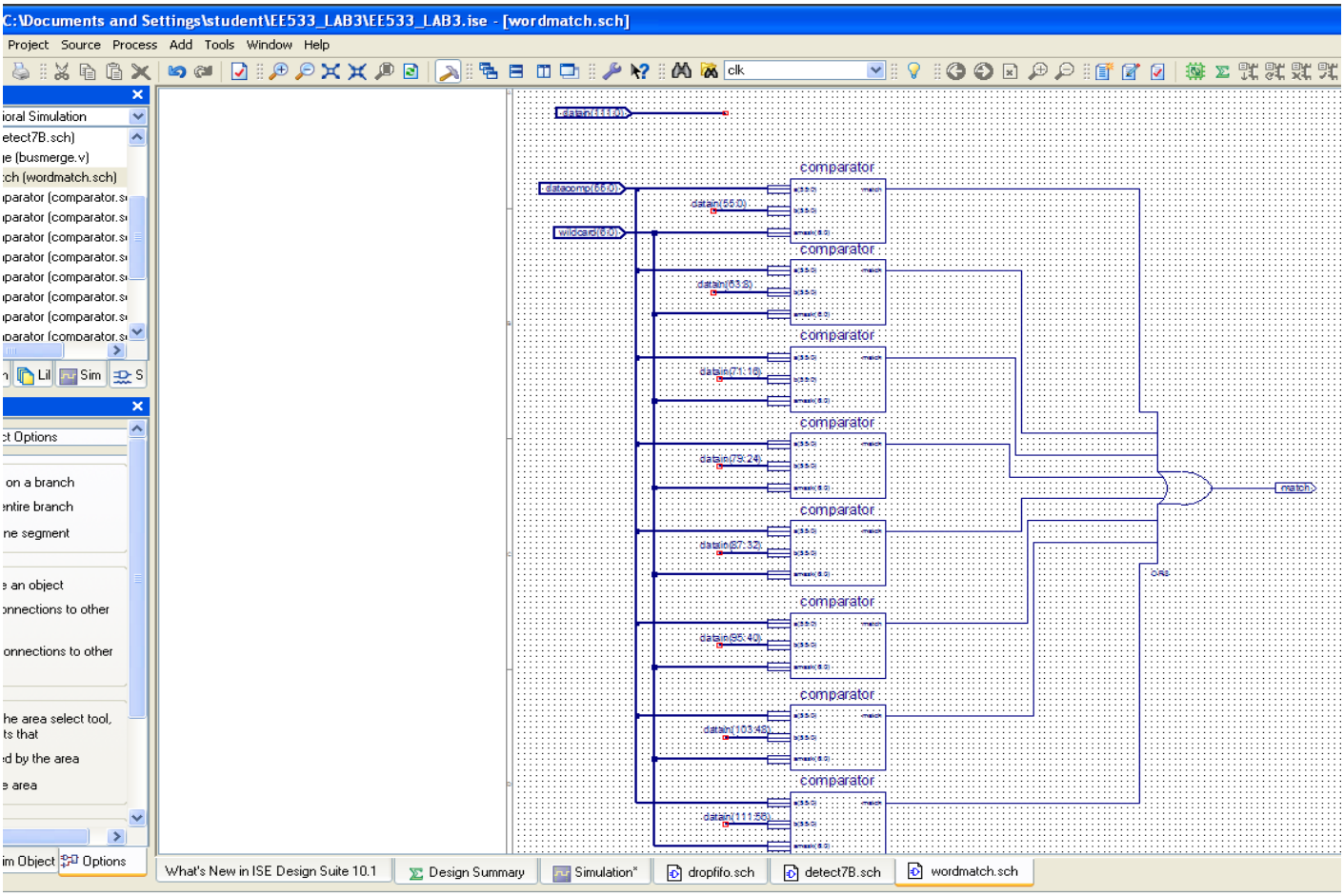
This state machine basically separates the packet into header and the payload which then allows us to test each byte of the input packet bytewise with the intrusion pattern.



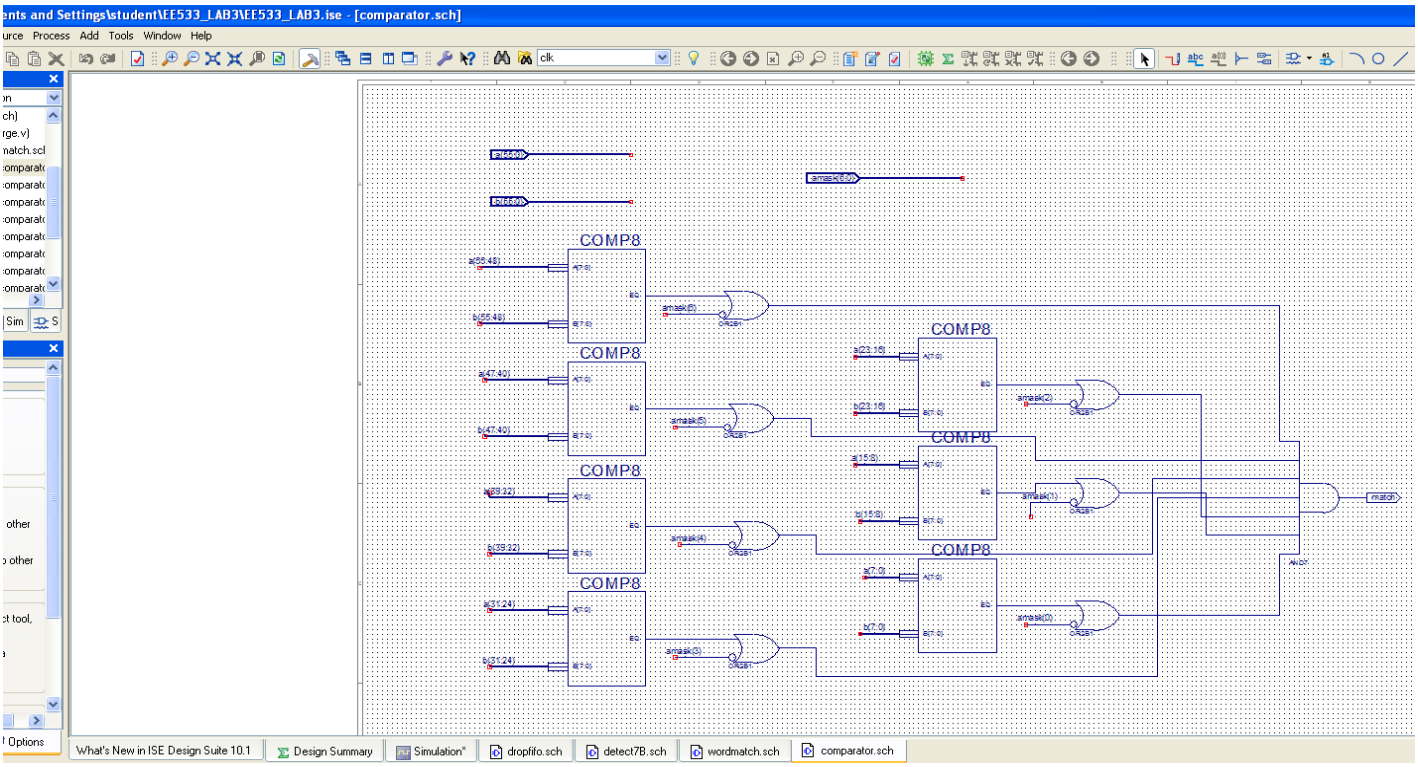
Dropfifo



Detect7B

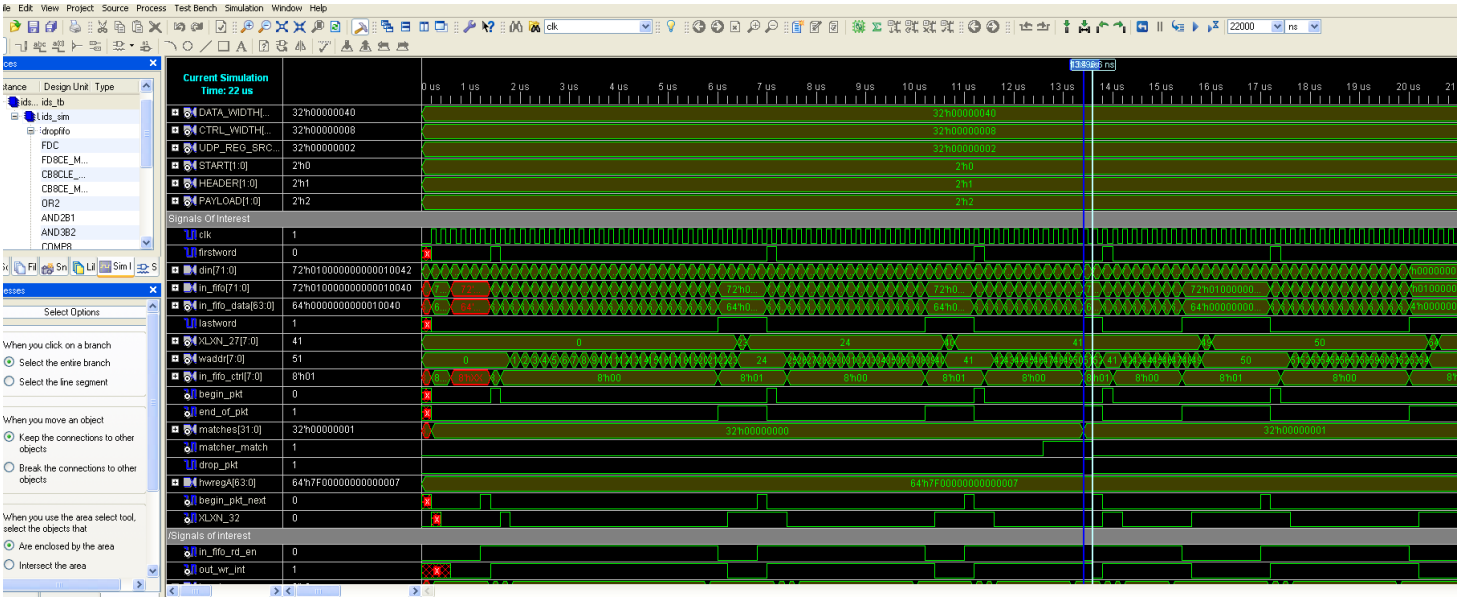


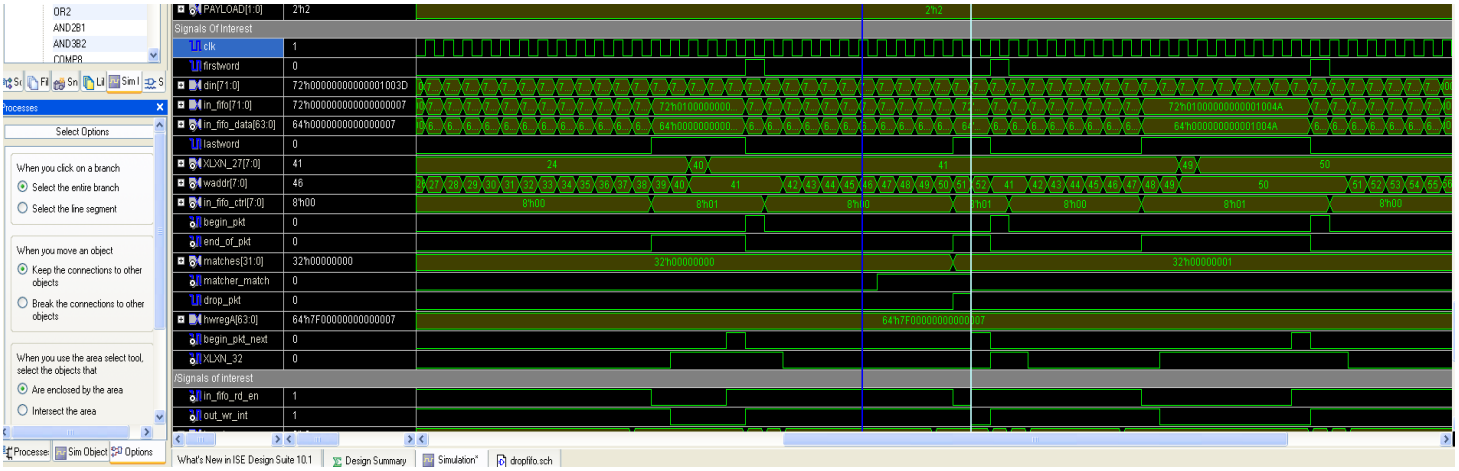
wordmatch



comparator

WAVEFORM AND GENERATED OUTPUT SCREENSHOT:

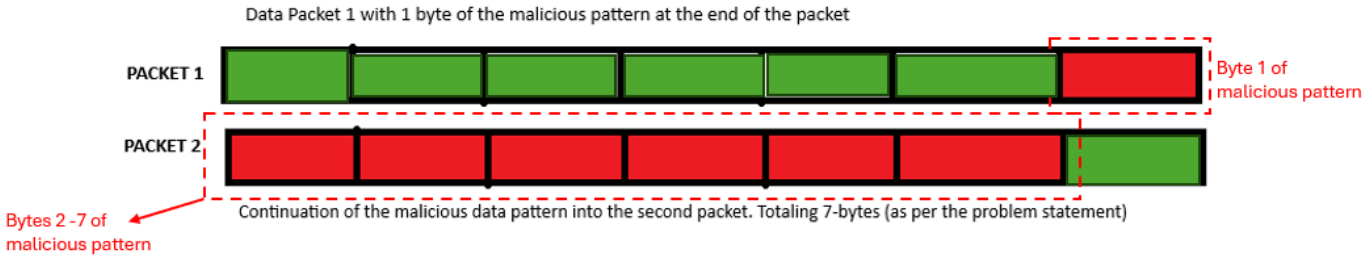




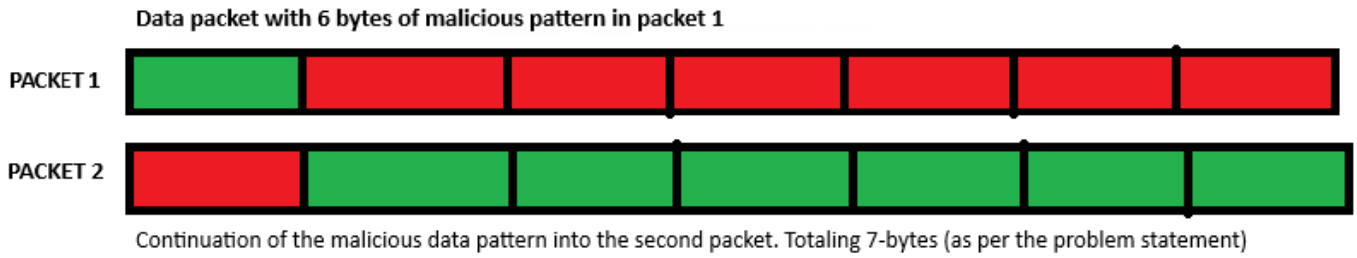
a)

i) The idea of this intrusion detection system is to detect a 7-byte malicious intrusive pattern in the incoming network data steam. The input stream is 8-byte data packet + 1 Byte control signal. Totaling 9-byte data. This data goes into the fallthrough fifo and then into the pattern matcher 7B module.
Now, we need to consider the possibility that the malicious pattern is hidden in the packets a variety of ways.

To consider all corner cases we'll take this example:



Apart from this we certainly can have other combinations as well (such as 3 bytes in packet 1 and 4 bytes in packet 2) but the above should cover the worst-case scenario. Or it could the other way around as well.



For a mini-intrusion system, we need an input data stream, a pattern matcher logic that detects certain intrusion patterns in the input data stream and a output FIFO wrapper to drop the intrusive pattern matched packets and write the rest of the data into a dual port FIFO (which essentially decouples the packets (intrusion free) from the 'to be processed circuit')

The test bench has 4 components for this:

1. fallthrough fifo

Essentially this is the fifo we use to send in the packets to our pattern detection logic (again, this is a decoupler that decouples our input data stream from the pattern matching logic)

2. detect7B matcher

This is the logic that detects the "intrusive data bit pattern" with the input data stream from the fallthrough fifo.

3. dropfifo

This is the fifo wrapper that has the logic to drop “pattern matched” intrusive packets from the data stream

4. Packet separation logic

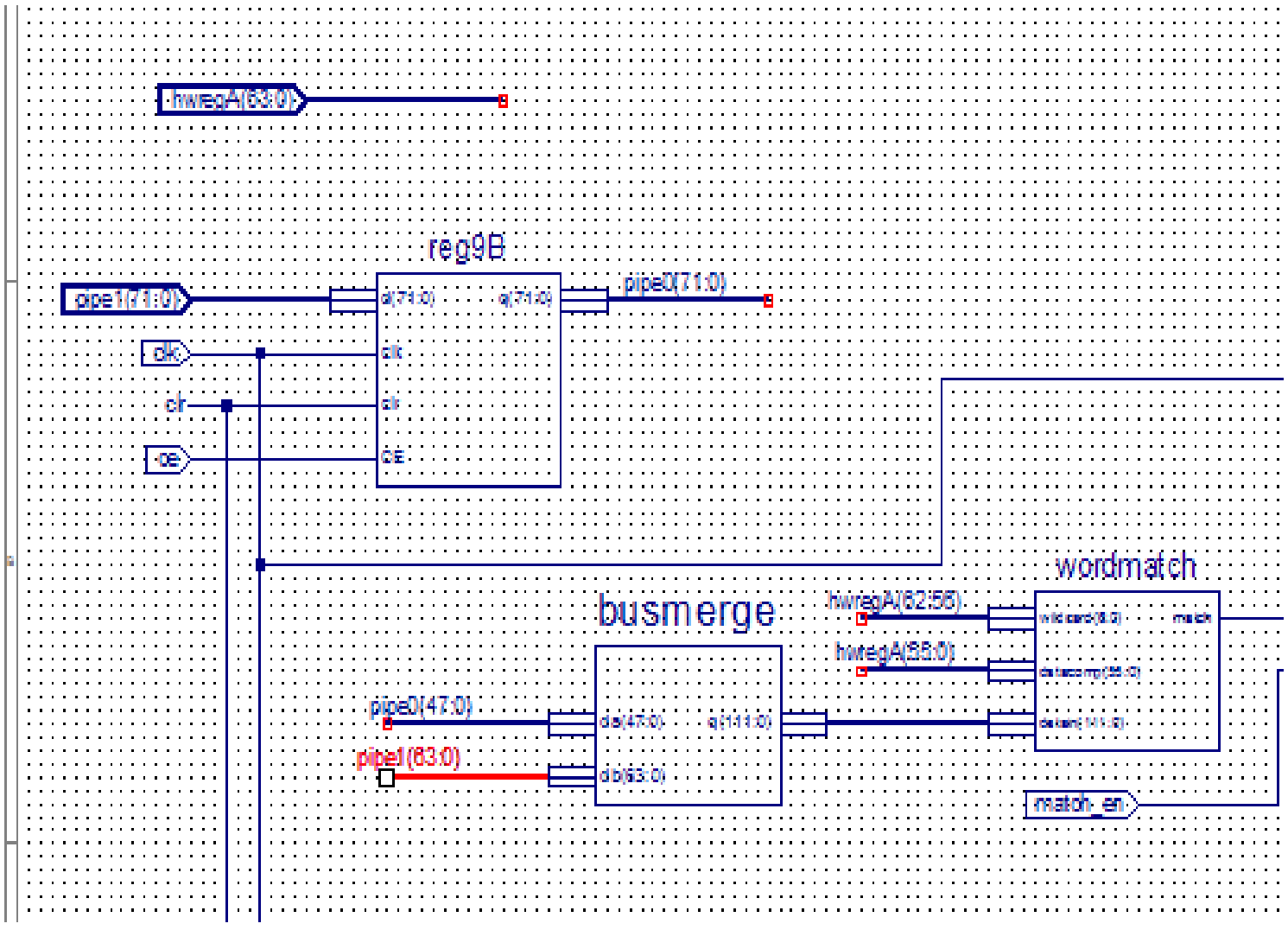
This state machine basically separates the packet into header and the payload which then allows us to test each byte of the input packet bytewise with the intrusion pattern.

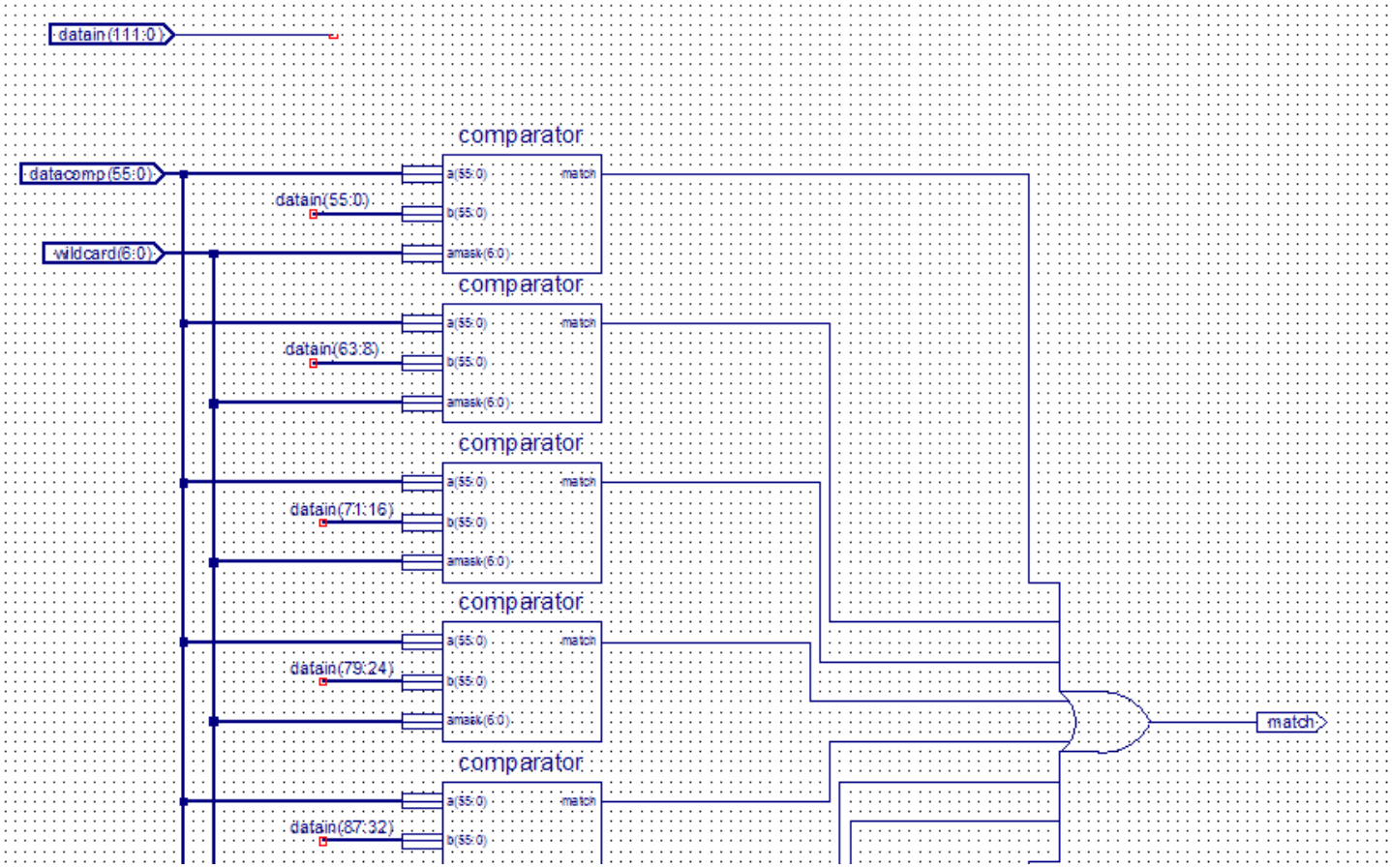
b)

i) *What is the purpose of AMASK[6:0]?*

AMASK[6:0] is used for deciding which byte of the pattern will be compared with the input data stream. So, 7’b1111111 means that all 7 bytes of the pattern will be matched against incoming packet bytes. If any bit in unset (i.e. set to 0), that byte won’t be compared with the incoming data packet and will essentially be considered a don’t care. This way our matching becomes smarter and not just blindly compares inputs to the same pattern irrespective of the situation

ii) *What exactly does busmerge.v do?*





Since the comparison that we need to make is 7-bytes (continuous), we need 2 chunks of 7-byte input data. This is because what if one chunk has a few bytes of matching pattern and the next incoming data chunk has the rest of the pattern. Therefore, what the BUSMERGE.v does is, concatenate 6-bytes of 'previous data' to the 8-bytes of 'newer data' (i.e. data from previous clock appended with data from latest clock), and send it for pattern matching and comparison.

iii) ***What do the comp8 modules do in this schematic?***

The comparator's job is to compare bytes of incoming data packet and spit out whether there is a match or an equality between the reference data byte (the pattern stream byte) and the incoming data packet.

iv) ***What is the purpose of dual9Bmem in dropfifo.sch?***

The job of the dual port 9B memory is to not let the pattern matched malicious data in and only let the non-malicious data in. This is 8B of data + 1B of control signal bits (ie. Total 9B). This fifo (which only lets in good data) decouples the data from the consumer.

GENERATED VERILOG SCREENSHOT: