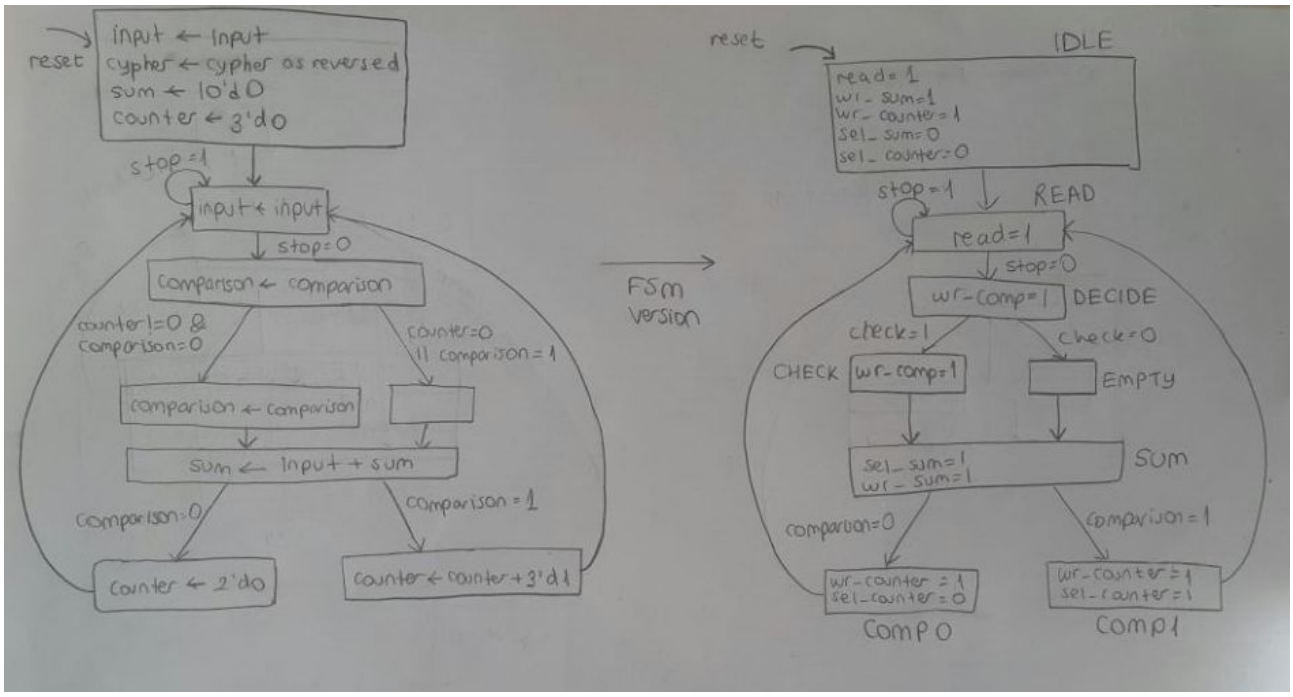
CSE 331 - Bonus HW
Sena Özbelen - 1901042601

FSM of the Program

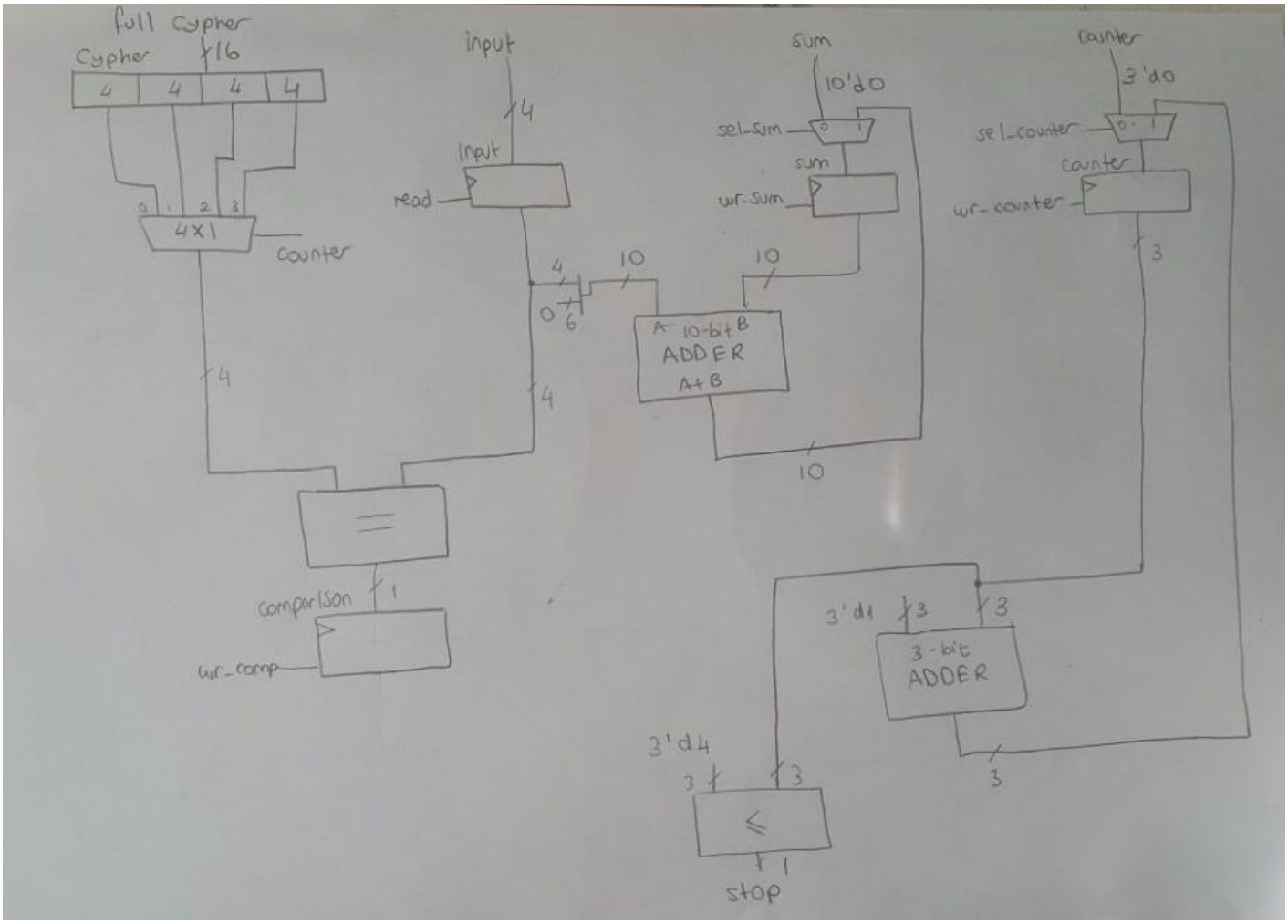


States

-IDLE: The first state is initialization part. Before that, program reverses the full cypher to keep track of it in the input sequence. In this state, sum and counter are initialized to 0. First input is initialized to the first 4 bits of the sequence.
-READ: To update the input, this state updates the input with incoming input. (reg_input <= seq_input)
-DECIDE: This state compares the first 4 bits of reversed full cypher to the input. If they are the same, comparison becomes 1 and counter increases by 1 to compare to the second part of cypher. Normally, if the counter is not 0 which means the sequence is the same as the cypher, and now they are not the same anymore, comparison becomes 0 but this input might be the same as the first 4 bits of cypher and we should be aware of the re-start of the cypher in the input. Therefore, I created another state (CHECK) to re-check if the current input is not equal to the some parts of cypher (not first part) but equal to the first part since this is an indicator of the re-start of the cypher sequence. Otherwise, we miss the first part of cypher while comparing the input to another part of the cypher.
-CHECK: This state compares the input to the first part of the cypher. If they are the same, then comparison becomes 1 again. Before that, it resets the counter and if comparison is 1, then counter increases in the next steps.

-EMPTY: This is an empty state. If there is no need to re-check, it comes to this state. This is for time balance otherwise test-bench delays cannot coincide with the program.
-SUM: This state use an adder to find total sum of the input.
-COMP0: If comparison is 0, then it resets counter.
-COMP1: If comparison is 1, then counter increases by 1.

Datapath Design of the Program



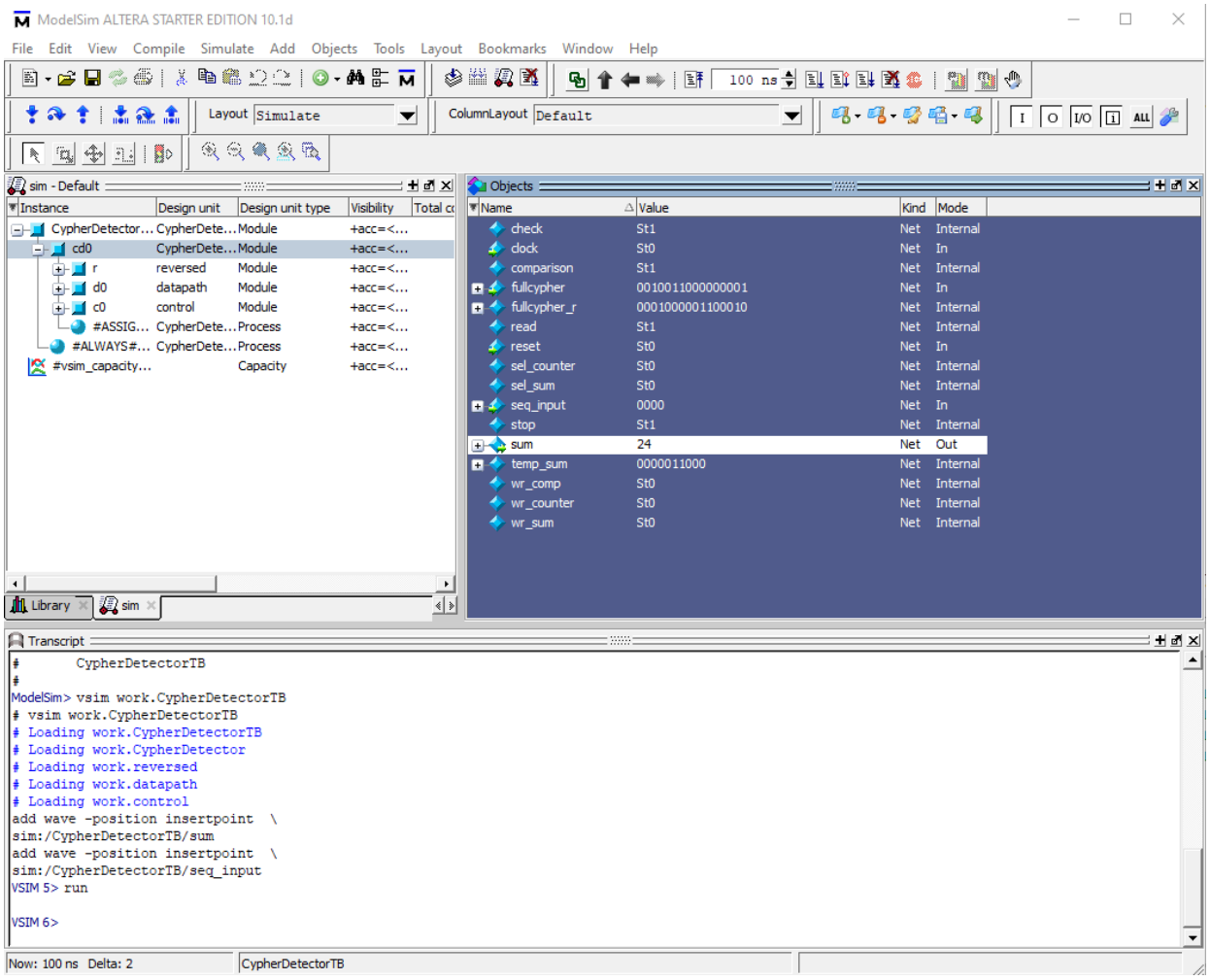
The first input is 16 bit reversed full cypher. There is a mux to get 4-bit-long corresponding cypher. This mux is controlled by counter. If counter is 0, then it gives the 15:12 bits of the cypher, 1 gives 11:8, 2 gives 7:4 and 3 gives 3:0. There is a comparator to compare the input to the corresponding part of cypher. The result is stored in comparison register.

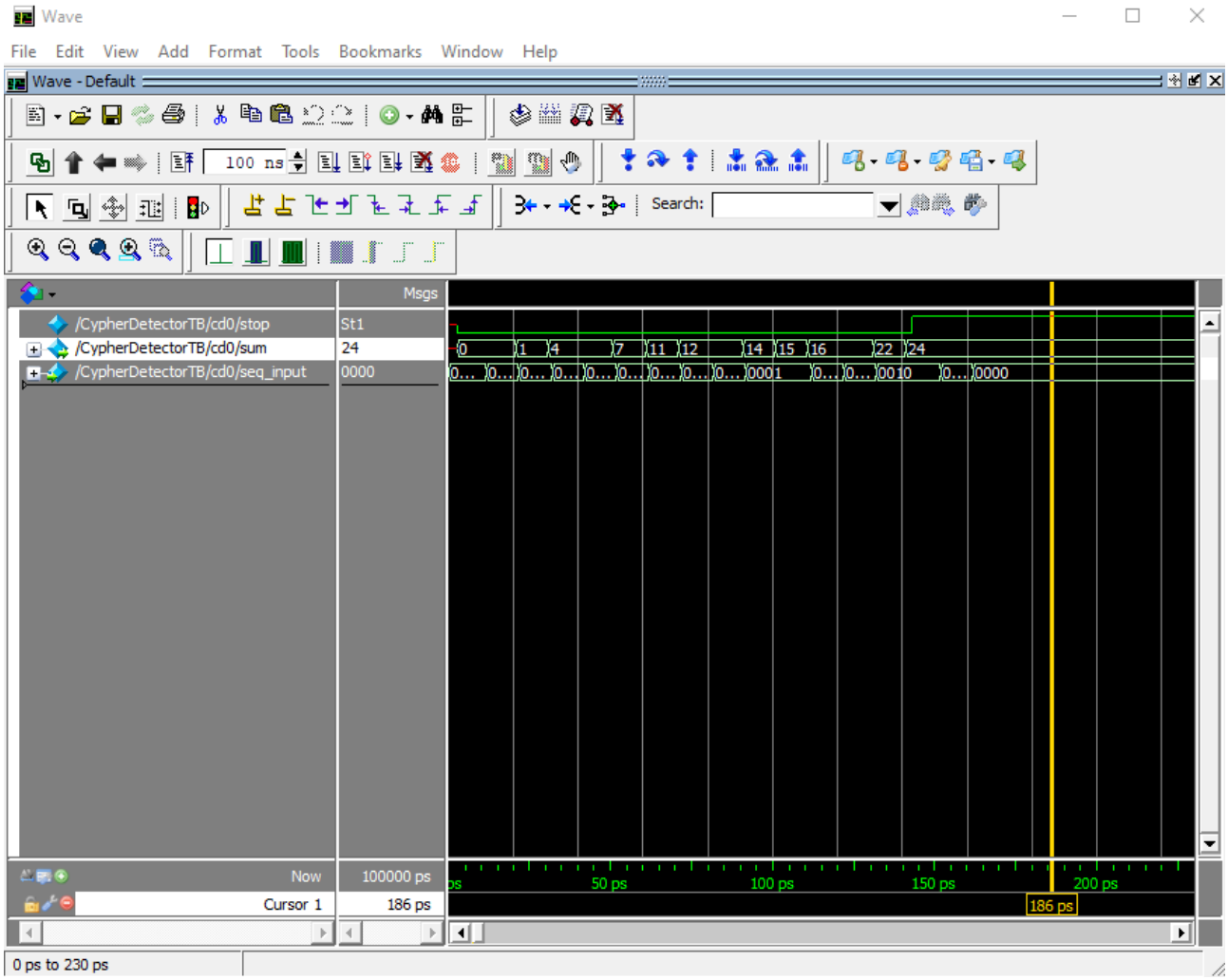
The second input is the 4 bit input. It is stored in input register. There is an adder to calculate the total sum (the second output) of the sequence until it encounters the full cypher. To sum 4 bit input and 10 bit total sum, we should extend the input to 10 bit so we add zeros to the beginning. Sum has

another mux. 0 is used to initialize it in IDLE state. Also, we have counter to keep track of full cypher. It also has a mux for initialization with 0. There is another adder for counter to increase it by 1. It also has a comparator to find out if we find the full cypher in the sequence. This produces the first output, stop signal to indicate that program found the full cypher and program should stop.

There are other control signals such as read, wr_sum, wr_counter to write and sel_counter, sel_sum for muxes.

Simulation Results for the Given Example in the PDF




Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
100 ns
Search:
Msgs
/CypherDetectorTB/cd0/stop
/CypherDetectorTB/cd0/sum
/CypherDetectorTB/cd0/seq_input
St1
24
0000
0 1 4 7 11 12 14 15 16 22 24
0001
0010
Now
100000 ps
Cursor 1
186 ps
50 ps
100 ps
150 ps
200 ps
0 ps to 230 ps