EE314 Digital Circuits Laboratory

Experiment 2 Preliminary

1)

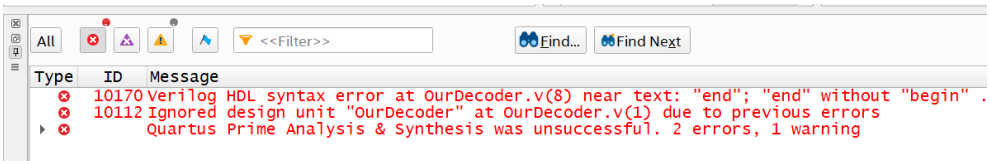

Figure 1: Error Messages for OurDecoder.v

As it can be seen from the error message in Figure 1, “begin-end” structure is used without “begin” statement. In order to solve this problem, we simply add “begin” statement as shown in Figure 2 below.

```
    always @(*)
    begin // added the missing "begin" statement
        outdecoder = (16'b1 << indecoder); // Shift a single '1' to the position given by indecoder
    end

endmodule
```

Figure 2: The Correction of Errors in OurDecoder.v

2)

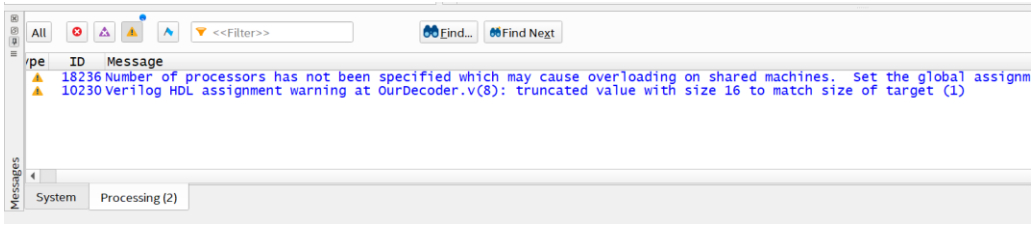

Figure 3: Warning Messages for OurDecoder.v

The second warning message shows that the 16 bit of information is truncated to 1 bit of output register since we have not mentioned the size of the register in the beginning. We will just add the size information ([15:0]) of the output register “outdecoder”:

```
1 module OurDecoder (
2     input [3:0] indecoder,      // 4-bit input
3     output reg [15:0] outdecoder // 16-bit output - Added the size of outdecoder
4 );
```

Figure 4: The Correction of Warnings in OurDecoder.v

3)

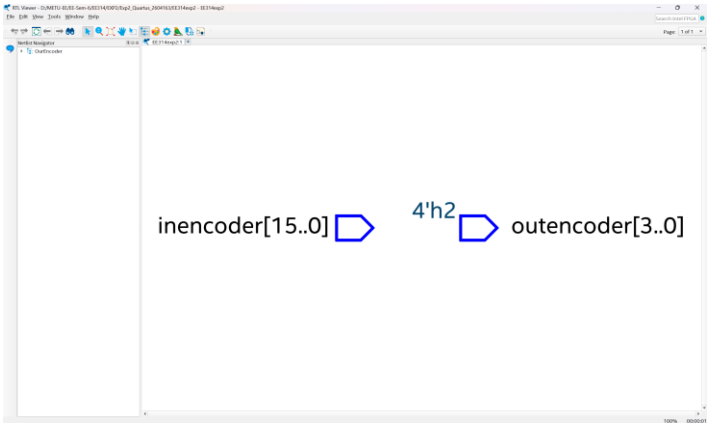

Figure 5:RTL Schematic of OurEncoder.v

In general, it is obvious from Figure 5 that our inputs and outputs are not connected to anything as if they are floating. When we examine the code, we can see that the expression inside case() is given wrong and needs to be changed to “inencoder” since our output is changing depending on the input:

```
6   always @(*) begin
7       case (inencoder) // corrected the expression inside case()
8           16'b0000_0000_0000_0001: outencoder = 4'd0;
9           16'b0000_0000_0000_0010: outencoder = 4'd1;
10          16'b0000_0000_0000_0100: outencoder = 4'd2;
11          16'b0000_0000_0000_1000: outencoder = 4'd3;
12          16'b0000_0000_0001_0000: outencoder = 4'd4;
13          16'b0000_0000_0010_0000: outencoder = 4'd5;
14          16'b0000_0000_0100_0000: outencoder = 4'd6;
15          16'b0000_0000_1000_0000: outencoder = 4'd7;
16          16'b0000_0001_0000_0000: outencoder = 4'd8;
17          16'b0000_0010_0000_0000: outencoder = 4'd9;
18          16'b0000_0100_0000_0000: outencoder = 4'd10;
19          16'b0000_1000_0000_0000: outencoder = 4'd11;
20          16'b0001_0000_0000_0000: outencoder = 4'd12;
21          16'b0010_0000_0000_0000: outencoder = 4'd13;
22          16'b0100_0000_0000_0000: outencoder = 4'd14;
23          16'b1000_0000_0000_0000: outencoder = 4'd15;
```

Figure 6: The corrected OurEncoder.v

4)

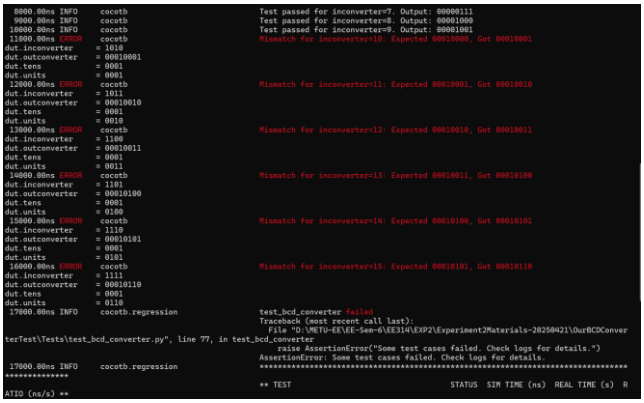

Figure 7: Errors in Testbanch of OurBCDConverter.v

As the errors in testbench explains, we face with errors for cases where input is greater than or equal to 10 so we will check the Verilog code for this case. When we examine the Verilog code, we see that for “units” in if statement instead of “10”, “9” is subtracted from the input:

```
    always @(*) begin
        // Use if-else logic for conversion
        if (inconverter >= 4'b1010) begin
            tens = 4'b0001;          // Add 1 to tens for values >= 10
            units = inconverter - 4'b1010; // (Corrected): Subtract 10 from units
        end else begin
            tens = 4'b0000;          // No tens for values < 10
            units = inconverter;     // Units remain the same
        end

        // Combine tens and units into BCD output
        outconverter = {tens, units};
    end

endmodule
```

Figure 8: The Corrected OurBCDConverter.v

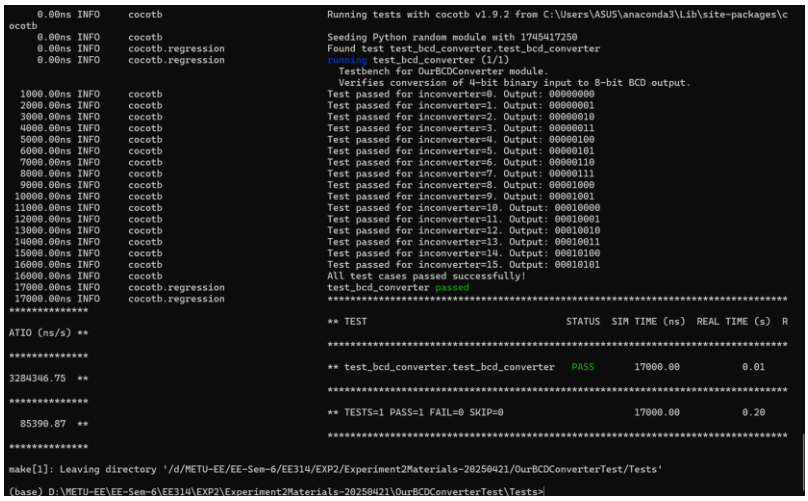

Figure 9: The Result of Test Cases for OurBCDConverter.v

5)

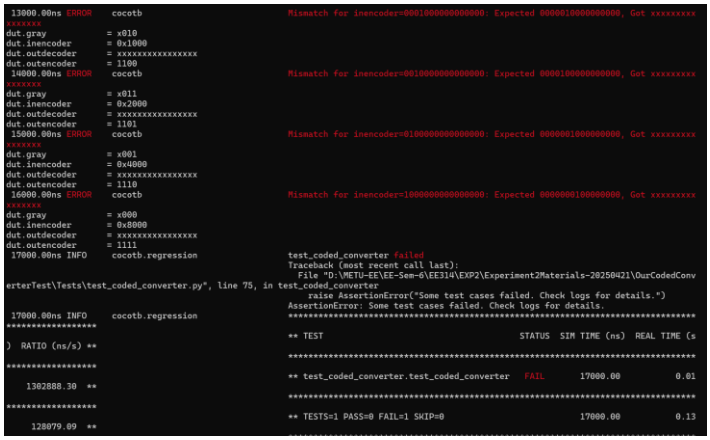

Figure 10: Error for Test of OurCodedConverter.v

In the error messages we see that “outdecoder” is considered as don’t care.

```
  8000.00ns INFO     cocotb                         Test passed for binary=7. Output: 0100
  9000.00ns INFO     cocotb                         Test passed for binary=8. Output: 1100
 10000.00ns INFO     cocotb                         Test passed for binary=9. Output: 1101
 11000.00ns INFO     cocotb                         Test passed for binary=10. Output: 1111
 12000.00ns INFO     cocotb                         Test passed for binary=11. Output: 1110
 13000.00ns INFO     cocotb                         Test passed for binary=12. Output: 1010
 14000.00ns INFO     cocotb                         Test passed for binary=13. Output: 1011
 15000.00ns INFO     cocotb                         Test passed for binary=14. Output: 1001
 16000.00ns INFO     cocotb                         Test passed for binary=15. Output: 1000
 16000.00ns INFO     cocotb                         All test cases passed successfully!
 17000.00ns INFO     cocotb.regression              test_binary_to_gray passed
 17000.00ns INFO     cocotb.regression              ****************************************************************
**********************************

                                                    ** TEST                                STATUS  SIM TIME (ns

)  REAL TIME (s)  RATIO (ns/s) **

                                                    ****************************************************************

**********************************

                                                    ** test_binary_to_gray.test_binary_to_gray   PASS       17000.00

           0.01    2322049.30  **

                                                    ****************************************************************

**********************************

                                                    ** TESTS=1 PASS=1 FAIL=0 SKIP=0                         17000.00

           0.14     121822.45  **

                                                    ****************************************************************

**********************************

make[1]: Leaving directory '/d/METU-EE/EE-Sem-6/EE314/EXP2/Experiment2Materials-20250421/OurBinaryToGrayConverterTest/Te
sts'

(base) D:\METU-EE\EE-Sem-6\EE314\EXP2\Experiment2Materials-20250421\OurBinaryToGrayConverterTest\Tests>
```

Figure 11: Test Results for OurBinaryToGrayConverter.v

```
 12000.00ns INFO     cocotb                         Test passed for inencoder=0000100000000000. Output: 010000000000
0000
 13000.00ns INFO     cocotb                         Test passed for inencoder=0001000000000000. Output: 000001000000
0000
 14000.00ns INFO     cocotb                         Test passed for inencoder=0010000000000000. Output: 000010000000
0000
 15000.00ns INFO     cocotb                         Test passed for inencoder=0100000000000000. Output: 000000100000
0000
 16000.00ns INFO     cocotb                         Test passed for inencoder=1000000000000000. Output: 000000010000
0000
 16000.00ns INFO     cocotb                         All test cases passed successfully!
 17000.00ns INFO     cocotb.regression              test_coded_converter passed
 17000.00ns INFO     cocotb.regression              ****************************************************************
**********************************

                                                    ** TEST                                  STATUS  SIM TIME (

ns)  REAL TIME (s)  RATIO (ns/s) **

                                                    ****************************************************************

**********************************

                                                    ** test_coded_converter.test_coded_converter   PASS       17000.

00           0.00          inf  **

                                                    ****************************************************************

**********************************

                                                    ** TESTS=1 PASS=1 FAIL=0 SKIP=0                         17000.

00           0.17      98243.64  **

                                                    ****************************************************************

**********************************

make[1]: Leaving directory '/d/METU-EE/EE-Sem-6/EE314/EXP2/Experiment2Materials-20250421/OurCodedConverterTest/Tests'

(base) D:\METU-EE\EE-Sem-6\EE314\EXP2\Experiment2Materials-20250421\OurCodedConverterTest\Tests>
```

Figure 12: Test Results for OurCodedConverter.v

Appendix

```python
import cocotb
from cocotb.triggers import Timer

def binary_to_gray(binary_val):
    binary_to_gray_dict = {
    # Binary (4-bit) : Gray Code (4-bit)
    '0000': '0000',  # 0 → 0
    '0001': '0001',  # 1 → 1
    '0010': '0011',  # 2 → 3
    '0011': '0010',  # 3 → 2
    '0100': '0110',  # 4 → 6
    '0101': '0111',  # 5 → 7
    '0110': '0101',  # 6 → 5
    '0111': '0100',  # 7 → 4
    '1000': '1100',  # 8 → 12
    '1001': '1101',  # 9 → 13
    '1010': '1111',  # 10 → 15
    '1011': '1110',  # 11 → 14
    '1100': '1010',  # 12 → 10
    '1101': '1011',  # 13 → 11
    '1110': '1001',  # 14 → 9
    '1111': '1000'   # 15 → 8
    }
    binary_str = format(binary_val, '04b')
    gray_str = binary_to_gray_dict[binary_str]
    return int(gray_str, 2)

@cocotb.test()
async def test_binary_to_gray(dut):
    """
    Testbench for OurBinaryToGrayConverter module.
    Verifies conversion of 4-bit binary input to 8-bit gray code output.
    """
    test_failed = False  # Flag to track overall test failure

    for binary_val in range(16):  # Loop through all possible 4-bit binary values
        # Apply the binary input
        dut.binary.value = binary_val

        # Wait for a short delay to allow combinational logic to propagate
        await Timer(1, units='us')
```

```
        # Calculate expected Gray Code output
        expected_gray = binary_to_gray(binary_val)

        # Check if the output matches the expected value
        if dut.gray.value != expected_gray:
            test_failed = True  # Mark overall test as failed
            cocotb.log.error(
                f"Mismatch for binary={binary_val}: Expected
{bin(expected_gray)[2:].zfill(4)}, "
                f"Got {bin(dut.gray.value)[2:].zfill(4)}"
            )
            #Log_Design(dut)
        else:
            cocotb.log.info(f"Test passed for binary={binary_val}. Output:
{dut.gray.value}")

    # Final assertion for the overall result
    if test_failed:
        raise AssertionError("Some test cases failed. Check logs for details.")
    else:
        cocotb.log.info("All test cases passed successfully!")
```