

## EE314 Digital Circuits Laboratory

### Experiment 2 Preliminary

1)

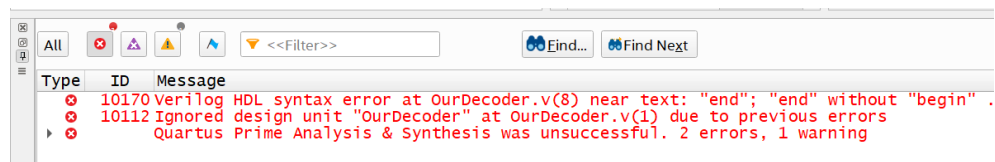


Figure 1: Error Messages for OurDecoder.v

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)

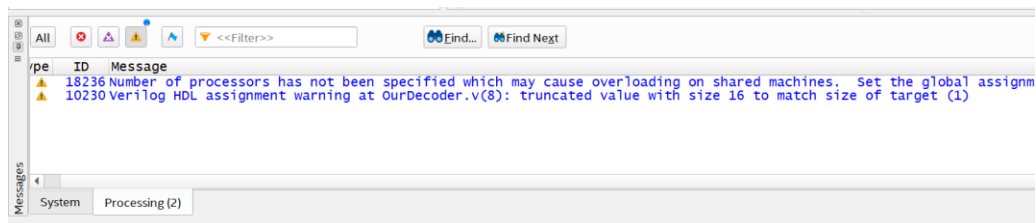


Figure 3: Warning Messages for OurDecoder.v

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)

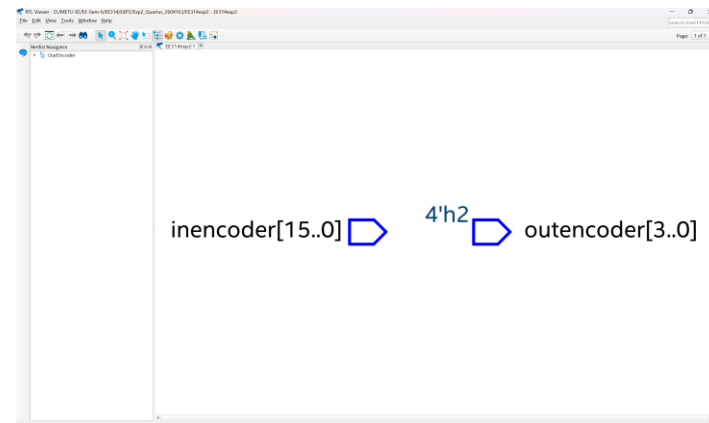


Figure 5:RTL Schematic of OurEncoder.v

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     0: outencoder = 4'd0;
9     1: outencoder = 4'd1;
10    2: outencoder = 4'd2;
11    3: outencoder = 4'd3;
12    4: outencoder = 4'd4;
13    5: outencoder = 4'd5;
14    6: outencoder = 4'd6;
15    7: outencoder = 4'd7;
16    8: outencoder = 4'd8;
17    9: outencoder = 4'd9;
18   10: outencoder = 4'd10;
19   11: outencoder = 4'd11;
20   12: outencoder = 4'd12;
21   13: outencoder = 4'd13;
22   14: outencoder = 4'd14;
23   15: outencoder = 4'd15;

```

Figure 6: The corrected OurEncoder.v

4)

[illegible]

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:

```

9      always @(*) begin
10          // Use if-else logic for conversion
11          if (inconverter >= 4'b1010) begin
12              tens = 4'b0001;           // Add 1 to tens for values >= 10
13              units = inconverter - 4'b1010; // (Corrected): Subtract 10 from units
14          end else begin
15              tens = 4'b0000;           // No tens for values < 10
16              units = inconverter;       // Units remain the same
17          end
18
19          // Combine tens and units into BCD output
20          outconverter = {tens, units};
21      end
22
23  endmodule

```

Figure 8: The Corrected OurBCDConverter.v

```

0.00ms INFO cocotb Running tests with cocotb v1.9.2 from C:/Users/VASU/anaconda3/Lib/site-packages/cocotb
scotb 0.00ms INFO cocotb Seeding Python random module with 1745417250
0.00ms INFO cocotb Found test_bcd_converter.test_bcd_converter
0.00ms INFO cocotb.regression Running test_bcd_converter (1/3)
Testbench for OurBcdConverter module.
Verifies conversion of 4-bit binary input to 8-bit BCD output.
1000.00ms INFO cocotb Test passed for Inverter=0. Output: 00000000
2000.00ms INFO cocotb Test passed for Inverter=1. Output: 00000001
3000.00ms INFO cocotb Test passed for Inverter=2. Output: 00000010
4000.00ms INFO cocotb Test passed for Inverter=3. Output: 00000011
5000.00ms INFO cocotb Test passed for Inverter=4. Output: 00000100
6000.00ms INFO cocotb Test passed for Inverter=5. Output: 00000101
7000.00ms INFO cocotb Test passed for Inverter=6. Output: 00000110
8000.00ms INFO cocotb Test passed for Inverter=7. Output: 00000111
9000.00ms INFO cocotb Test passed for Inverter=8. Output: 00010000
10000.00ms INFO cocotb Test passed for Inverter=9. Output: 00010001
11000.00ms INFO cocotb Test passed for Inverter=10. Output: 00010000
12000.00ms INFO cocotb Test passed for Inverter=11. Output: 00010001
13000.00ms INFO cocotb Test passed for Inverter=12. Output: 00010010
14000.00ms INFO cocotb Test passed for Inverter=13. Output: 00010011
15000.00ms INFO cocotb Test passed for Inverter=14. Output: 00010100
16000.00ms INFO cocotb Test passed for Inverter=15. Output: 00010101
17000.00ms INFO cocotb All test cases passed successfully!
17000.00ms INFO cocotb.regression test_bcd_converter PASS
*****
** TEST STATUS SIM TIME (ns) REAL TIME (s) R
*****
** test_bcd_converter.test_bcd_converter PASS 17000.00 0.01
*****
** TESTS:1 PASS=1 FAIL=0 SKIP=0 17000.00 0.20
*****
make[1]: Leaving directory '/d:/METU-EE/EE-Sem-6/EE314/EXP2/Experiment2Materials-20250421/OurBcdConverter/Test/Tests'
(cake) D:\METU-EE\Sem-6\EE314\EXP2\Experiment2Materials-20250421\OurBcdConverter\Test\Tests

```

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

5)

```

13000.00ns ERROR cocotb Mismatch for inenCoder=0010000000000000: Expected 0000100000000000, Got xxxxxxxx
#####
dut_gray = x010
dut_inenCoder = 0x1000
dut_outenCoder = xxxxxxxxxxxxxxxx
dut_outenCoder = 1100
14000.00ns ERROR cocotb Mismatch for inenCoder=0010000000000000: Expected 0000100000000000, Got xxxxxxxx
#####
dut_gray = x011
dut_inenCoder = 0x2000
dut_outenCoder = xxxxxxxxxxxxxxxx
dut_outenCoder = 1101
15000.00ns ERROR cocotb Mismatch for inenCoder=0100000000000000: Expected 0000010000000000, Got xxxxxxxx
#####
dut_gray = x001
dut_inenCoder = 0x4000
dut_outenCoder = xxxxxxxxxxxxxxxx
dut_outenCoder = 1110
16000.00ns ERROR cocotb Mismatch for inenCoder=1000000000000000: Expected 0000001000000000, Got xxxxxxxx
#####
dut_gray = x000
dut_inenCoder = 0x8000
dut_outenCoder = xxxxxxxxxxxxxxxx
dut_outenCoder = 1111
17000.00ns INFO cocotb.regression test_coded_converter failed
Traceback (most recent call last):
  File "D:\VLSI\HLS\See-0\SEE34\EXP2\ExperimentMaterials_20250621\OurCode\Conv
arterTest\test_coded_converter.py", line 75, in test_coded_converter
    raise AssertionError("Check logs for details.")
AssertionError: Some test cases failed. Check logs for details.
=====
17000.00ns INFO cocotb.regression
=====
# TEST STATUS SIM TIME (ns) REAL TIME (s)
=====
** test_coded_converter test_coded_converter FAIL 17000.00 0.01
=====
** TESTS=1 PASS=0 FAIL=1 SKIP=0 17000.00 0.13
=====

```

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.test_regression test_binary_to_gray passed
17000.00ns INFO cocotb.test_regression

*****
** TEST STATUS SIM TIME (ns)
*****
) REAL TIME (s) RATIO (ns/s) **
*****
0.01 2322049.30 **
*****
0.14 121822.45 **
*****
make[1]: Leaving directory '/d/METU-EE/EE-Sem-6/EE314/EXP2/Experiment2Materials-20250421/OurBinaryToGrayConverterTest/Tests'
(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.test_regression test_coded_converter passed
17000.00ns INFO cocotb.test_regression

*****
** TEST STATUS SIM TIME (
*****
ns) REAL TIME (s) RATIO (ns/s) **
*****
00 0.00 inf **
*****
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

```
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!")
```