Group 30
Gabriel Chootong 260 637 105
Sean Stappas 260 630 512

October 7, 2016
ECSE-323

Report of the g30_16_4_Encoder Circuit

1. Circuit Description

This encoder circuit takes the 16-bit signal “BLOCK_COL” as input. It outputs the 4-bit signal “CODE” as well as the 1-bit signal “ERROR”. The “CODE” signal corresponds to a 4-bit binary number that indicates the index of a bit that is set to '1' in the “BLOCK_COL” signal. If more than one bit is set to '1' in “BLOCK_COL”, the circuit will output the index of the least significant bit. If the 16 bits of “BLOCK_COL” are set to '0', the “ERROR” signal will be set to '1'. Input and output signals are shown on Figure 1.

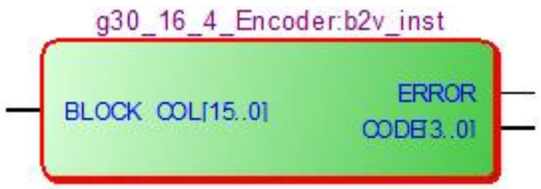

Figure 1. Pinout Diagram of Encoder Circuit

The circuit was designed with VHDL using one conditional assignment. A gate level schematic diagram was then generated, shown in Figure 2. Because of the conditional assignment, the encoder is implemented with several multiplexers in series using a priority system. Each multiplexer has 2 inputs, and the “select” signal is the input signal of the encoder (more specifically, it is one bit of the 16-bit BLOCK_COL signal for each multiplexer). If the select signal is '1', the multiplexer will choose to output a predefined 5-bit signal (4 bits for the encoded output, 1 bit for the error signal) that specifically corresponds to that select bit. Else, if the select bit is '0', it will choose the output of the previous multiplexer, and so on.

In Figure 2, the signals shaded in gray shown as inputs to the multiplexers are the predefined encoded signals. A more detailed gate level schematic diagram of the circuit is shown in Figure 3.

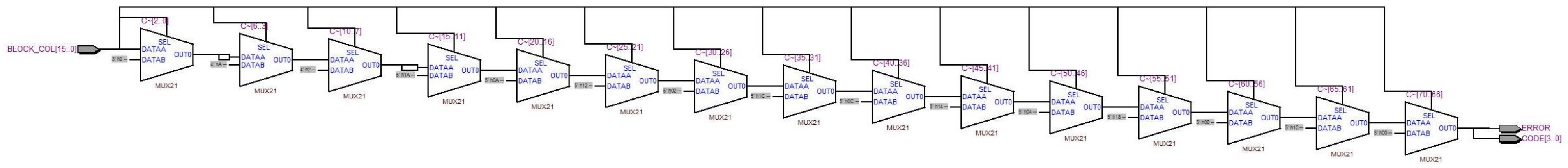

Figure 2. Gate Level Schematic Diagram of Encoder Circuit.

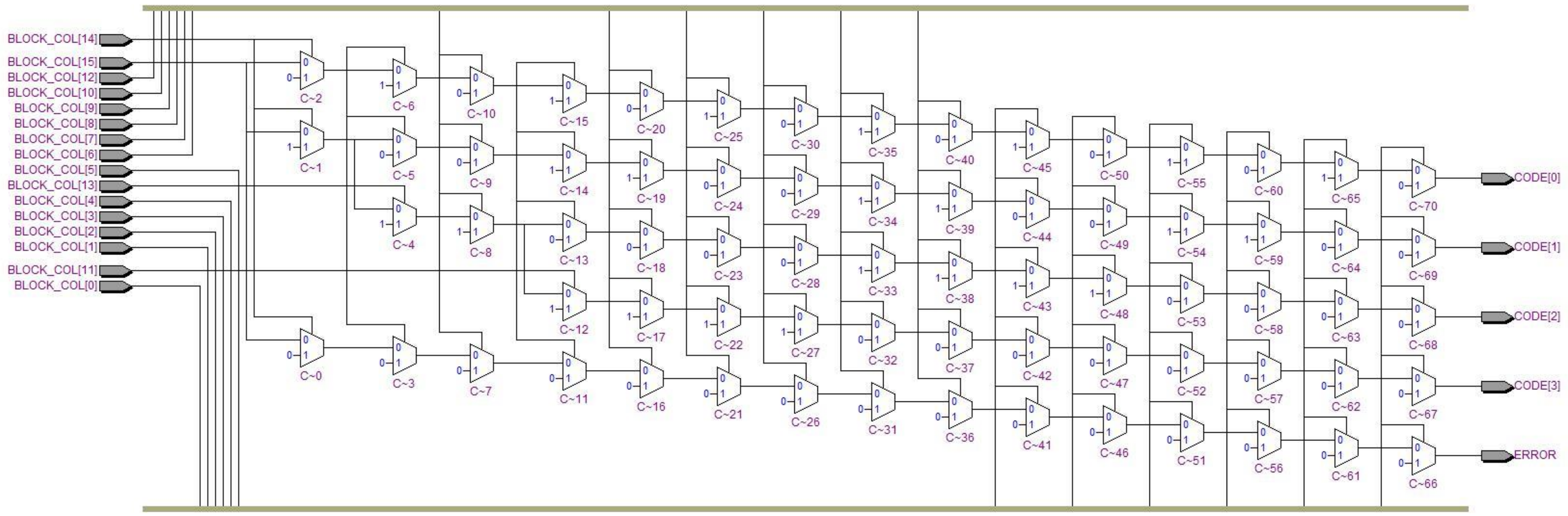

Figure 3. Detailed Gate Level Schematic Diagram

2. Testing

Figure 4 shows the main process code used to test the circuit. The full code can be found in the accompanying zip file.

```
always : PROCESS
   type TEST_ARRAY is array (0 to 9) of integer;
   constant ARBITRARY_INPUT : TEST_ARRAY := (3, 7, 29, 301, 452, 1502, 12304, 20433, 30001, 32000);
BEGIN
   --- test 16 valid input patterns (1 to 2^15)
   for i in 0 to 15 loop
      BLOCK_COL <= (i => '1', others => '0');
      wait for 10 ns;
   end loop;

   --- test 0-input (error condition)
   BLOCK_COL <= (15 downto 0 => '0');
   wait for 10 ns;

   --- test some arbitrary input (set in above array)
   for i in 0 to 9 loop
      BLOCK_COL <= std_logic_vector(to_unsigned(ARBITRARY_INPUT(i), BLOCK_COL'length));
      wait for 10 ns;
   end loop;

   wait;
END PROCESS always;
```

Figure 4. Main Test Process.

As can be seen from the code, first the 16 valid input patterns were tested. This corresponds to inputs where only one of the bits is high, i.e. inputs corresponding to unsigned integers 2^0 to 2^{15}. The error input was then tested, corresponding to BLOCK_COL = ‘0’. This is the first part of the resulting waveform:

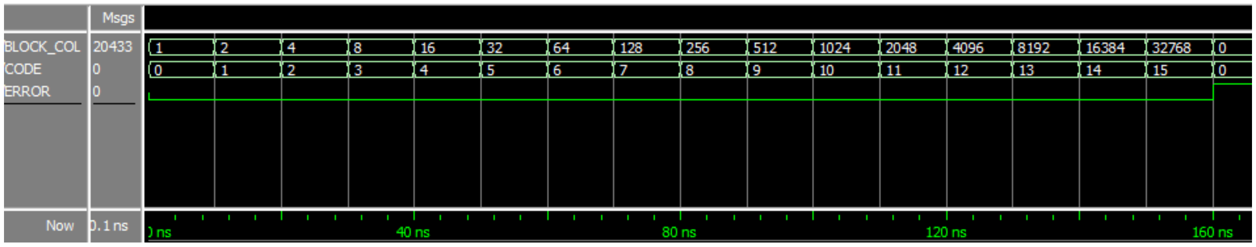

Figure 5. Resulting Waveform with All 'Valid' Inputs and Error Condition.

Note that the BLOCK_COL and CODE values are their unsigned integer values. As can be seen above, the behaviour is as expected. For inputs 2^0 to 2^{15}, CODE corresponds to the index of the input bit that is high (0 to 15), and ERROR is low. When the input BLOCK_COL is ‘0’, the CODE output is ‘0’ (our design choice), and, more importantly, the ERROR output is high.

Afterwards, some arbitrary input was tested. The unsigned values of these inputs were put into an array for convenience, and were then iterated over. Table 1 summarizes the input (in unsigned and binary form) with corresponding expected output (least significant ‘1’ bit in bold for emphasis). The unsigned value of the input is given for easy recognition in the resulting waveform (Figure 6).

Table 1. Test Input and Output.

BLOCK_COL (unsigned)	BLOCK_COL (binary)	Expected CODE (unsigned)	Expected ERROR
3	0000 0000 0000 001**1**	0	0
7	0000 0000 0000 011**1**	0	0
29	0000 0000 0001 110**1**	0	0
301	0000 0001 0010 110**1**	0	0
452	0000 0001 1100 0**1**00	2	0
1502	0000 0101 1101 11**1**0	1	0
12304	0011 0000 000**1** 0000	4	0
20433	0100 1111 1101 000**1**	0	0
30001	0111 0101 0011 000**1**	0	0
32000	0111 110**1** 0000 0000	8	0

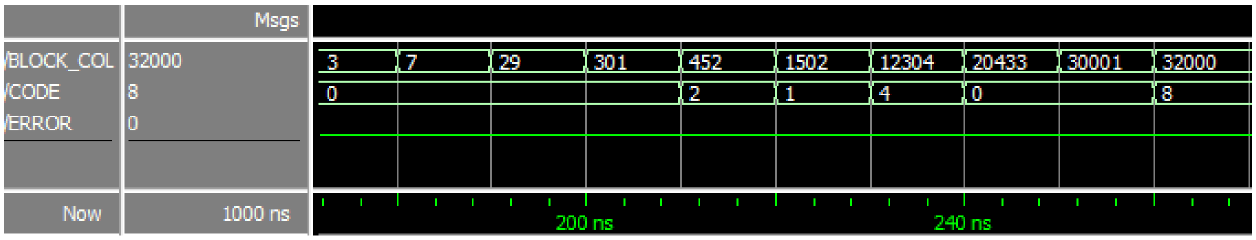

Figure 6. Resulting Waveform with Arbitrary Input.

As can be seen by comparing Table 1 and Figure 6, the behaviour is as expected. Our tests did not cover all possible input cases, but we have covered enough that we are confident that the circuit performs correctly.

Grade Sheet for Lab #1

Fall 2016.

Group Number: 30

Group Member Name: Sean Staffas   Student Number: 260 639 512

Group Member Name: Gabriel Chootong   Student Number: 260637105

Marks			TA Signatures
2	1.	Schematic diagram for the 6-bit comparator	[illegible]
2	2.	VHDL file for the 6-bit comparator	
2	3.	Initial partial simulation results for the 6-bit comparator	[illegible]
2	4.	Complete simulation results for the 6-bit comparator	[illegible]
2	5.	VHDL description for the 16:4 encoder circuit	
2	6.	Simulation Testbench VHDL for the 16:4 encoder circuit	
7	7.	Simulation results for the 16:4 encoder circuit	

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.