Design of cyclic code decade counter using K-Maps, Schematic capture and simulation of counter in Quartus-Altera software

LAB #1 EE 2160
SEPTEMBER 20, 2004

BY:
Narayanan Krishnamurthy

Purpose:
To design and simulate a 4-bit decade counter, that implements a cyclic code encoding for the 0-9 count (see table1 for the mapping of count to cyclic code). And to use the 2 such counters to count 0-99.

Table 1 Cyclic Code Encoded Decade Count

Decade Count	Cyclic Code	
0	0000	0x0
1	0001	0x1
2	0101	0x5
3	0111	0x7
4	0110	0x6
5	1110	0xE
6	1100	0xC
7	1101	0xD
8	1001	0x9
9	1000	0x8

Procedure:
The assignment required the use of basic logic gates and flip-flops, and hence the first step was to obtain the state table based on the flip-flop implementation. T-flip-flop and D-flip-flop were chosen and two different designs were built. Then k-maps were drawn using the state table to obtain logic equations. Based on the logic equations, basic logic gates and were used to implement the schematic. The schematic was then compiled and simulated to obtain waveforms that verified the functionality of the decade counter.

Specifications:

- ❖ To design a sequential logic that implements a state machine (to take care of mapping of decade count onto the cyclic code).
- ❖ The sequential logic circuit contains storage elements – flip-flops, and combinational logic that together implements an event based state-machine.
- ❖ The event in the case of the counter is the raising edge of the clock. The next state of the storage element depends on its current state.
- ❖ The design can be implemented using any kind of storage element, and the decision to use any particular implementation rests on simplicity of design, chip area of implementation etc. Two different designs were implemented to verify the operation of the counter. The first design used T-Flip-Flop (TFF) while the second design used D-Flip-Flop (DFF). Both TFF and DFF were chosen for their simplicity in design. The operation truth table of the TFF and DFF are given in table 2 and 3 respectively. The output of the TFF toggles if its T-input is asserted, and remains unaltered if the T-input is low. The input of the DFF transfers to its output during the rising edge of the clock.

Table 2 Truth Table of a T-Flip-Flop

T (Input)	Q (Cur. State)	Q (Next State)
0	0	0
0	1	1
1	0	1
1	1	0

Table 3 Truth Table of a D-Flip-flop

Clock	D (Input)	Q (Output)
0	0	Q
Rising	1	1
0	0	Q
Rising	0	0

Design:

The state table based on the flip-flop implementation is the first step in the design of the counter. Please see table 4 and 5 for TFF and DFF implementation.

Table 4 State table TFF implementation

Q4Q3Q2Q1 (Output)		T4T3T2T1 (Input)
(Cur. State) **dcba**	(Next State)	To trigger state chg
0000	0001	0001
0001	0101	0100
0010	XXXX	XXXX
0011	XXXX	XXXX
0100	XXXX	XXXX
0101	0111	0010
0110	1110	1000
0111	0110	0001
1000	0000	1000
1001	1000	0001
1010	XXXX	XXXX
1011	XXXX	XXXX
1100	1101	0001
1101	1001	0100
1110	1100	0010
1111	XXXX	XXXX

Table 5 State table DFF implementation

(Current State) Q3Q2Q1Q0	(Next State) D3D2D1D0
0000	0001
0001	0101
0101	0111
0111	0110
0110	1110
1110	1100
1100	1101
1101	1001
1001	1000
1000	0000

The crucial part of the design problem is determining a suitable combinational logic to generate an input that would effect the next-state transition on the raising edge of the clock.

K-Map for Q (1-4) also named ‘dcba’ current state to T-input(1-4) is given in table 5.
K-Map for D (0-4) for Q input (0-4) is given in table 6.

Table 5. K-map for T-Flip-Flop implementation.

T4	00	01	11	10
00	0	0	X	X
01	X	0	0	1
11	0	0	X	0
10	1	0	X	X

T3	00	01	11	10
00	0	1	X	X
01	X	0	0	0
11	0	1	X	0
10	0	0	X	X

T2	00	01	11	10
00	0	0	X	X
01	X	1	0	0
11	0	0	X	1
10	0	0	X	X

T1	00	01	11	10
00	1	0	X	X
01	X	0	1	0
11	1	0	X	0
10	0	1	X	X

From the adjacent elements of the 'ON' T-input bit we get the following logic equations for the T(1-4):

LOGIC EQUATIONS FOR T-INPUTS OF TFF

T4 = a'd'c + a'dc' = a'(d'c + dc') = a'(d xor c)
T3 = ad'c' + adc = a(d'c' + dc) = a(D xnor A)
T2 = d'cb'+ dba'
T1 = cb'a' + cba + c'd'a' + c'da = c(b'a'+ba) +c'(d'a'+da) = c(b xnor a) + c'(d xnor a)

Table 6. K-map for D-Flip-Flop implementation.

D0	00	01	11	10
00	1	1	X	X
01	X	1	0	0
11	1	1	X	0
10	0	0	X	X

D1	00	01	11	10
00	0	0	X	X
01	X	1	1	1
11	0	0	X	0
10	0	0	X	X

D2	00	01	11	10
00	0	1	X	X
01	0	1	1	1
11	1	0	X	1
10	0	0	X	X

D3	00	01	11	10
00	0	0	X	X
01	X	0	0	1
11	1	1	X	1
10	0	1	X	X

LOGIC EQUATIONS FOR D-flip-flop

D0 = Q3’Q2’Q1’ + Q3’Q1’Q0 + Q3Q2Q1’
D1 = Q3’Q2Q0 + Q3’Q2Q1Q0’
D2 = Q3’Q1’Q0 + Q3’Q2Q1 + Q3Q2Q0’
D3 = Q3Q2Q1’ + Q3Q1’Q0 + Q2Q1Q0’

Schematics:

Figure 1: 4-Bit Decade Counter TFF implementation

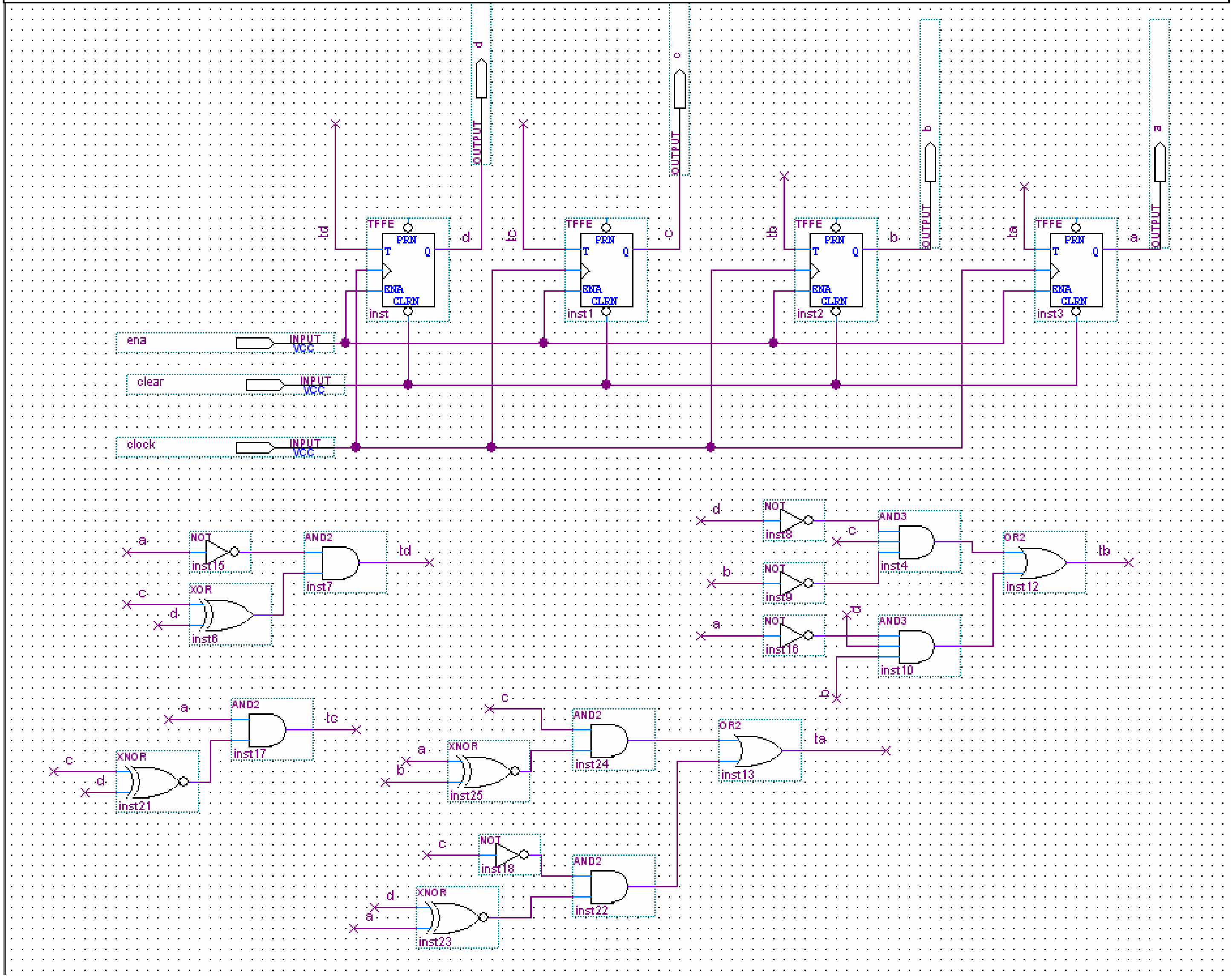
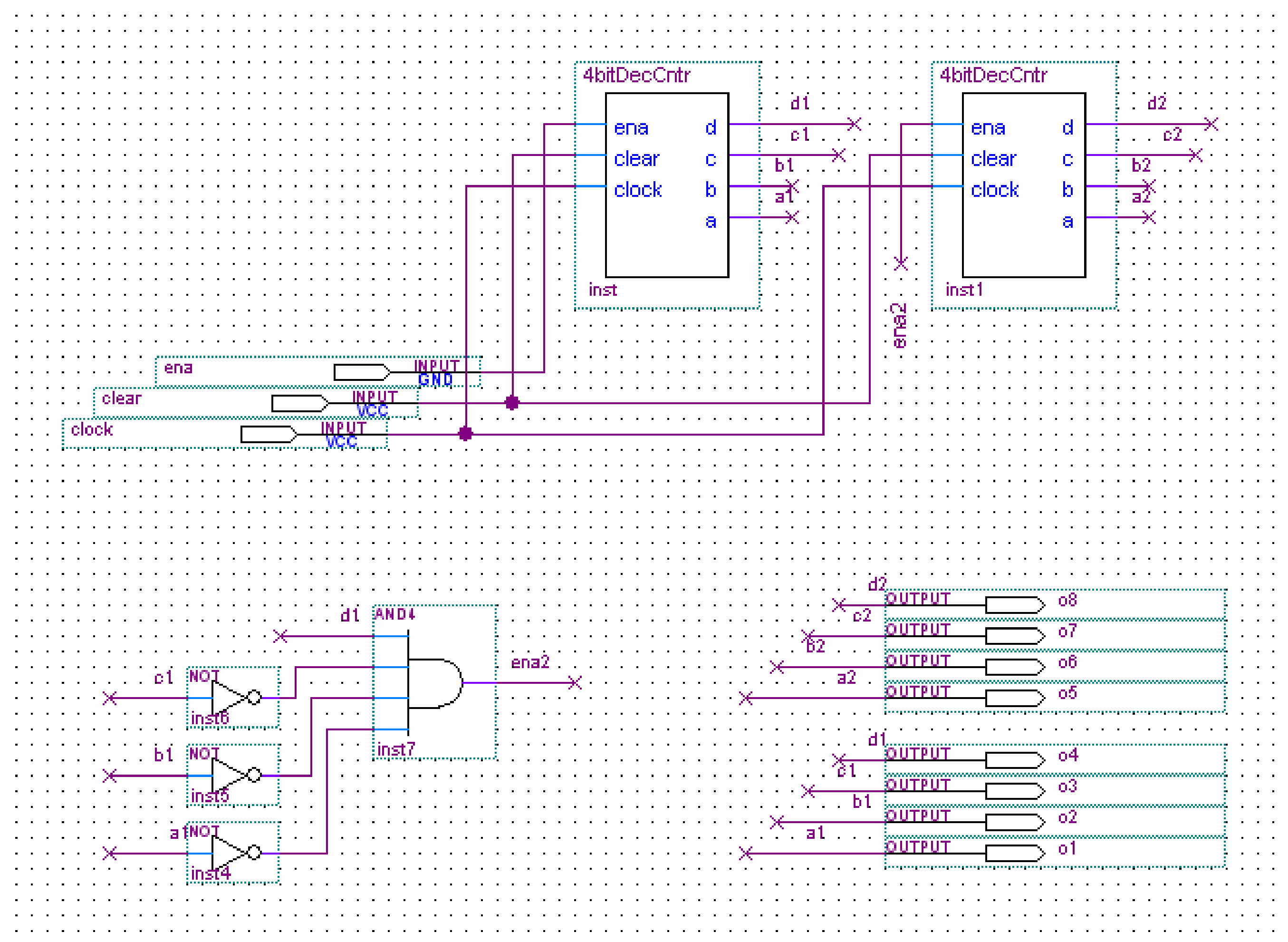


Figure 2: 4-Bit Decade Counter TFF implementation



Every nine cycles the cascaded 4-bit counters enable is asserted by the logic equation ena = d1c1'b1'a1', thus the cascaded counter counts once every nine counts of first counter, implementing a 0-99 count.

Figure 3. 4-Bit Decade Counter DFF implementation

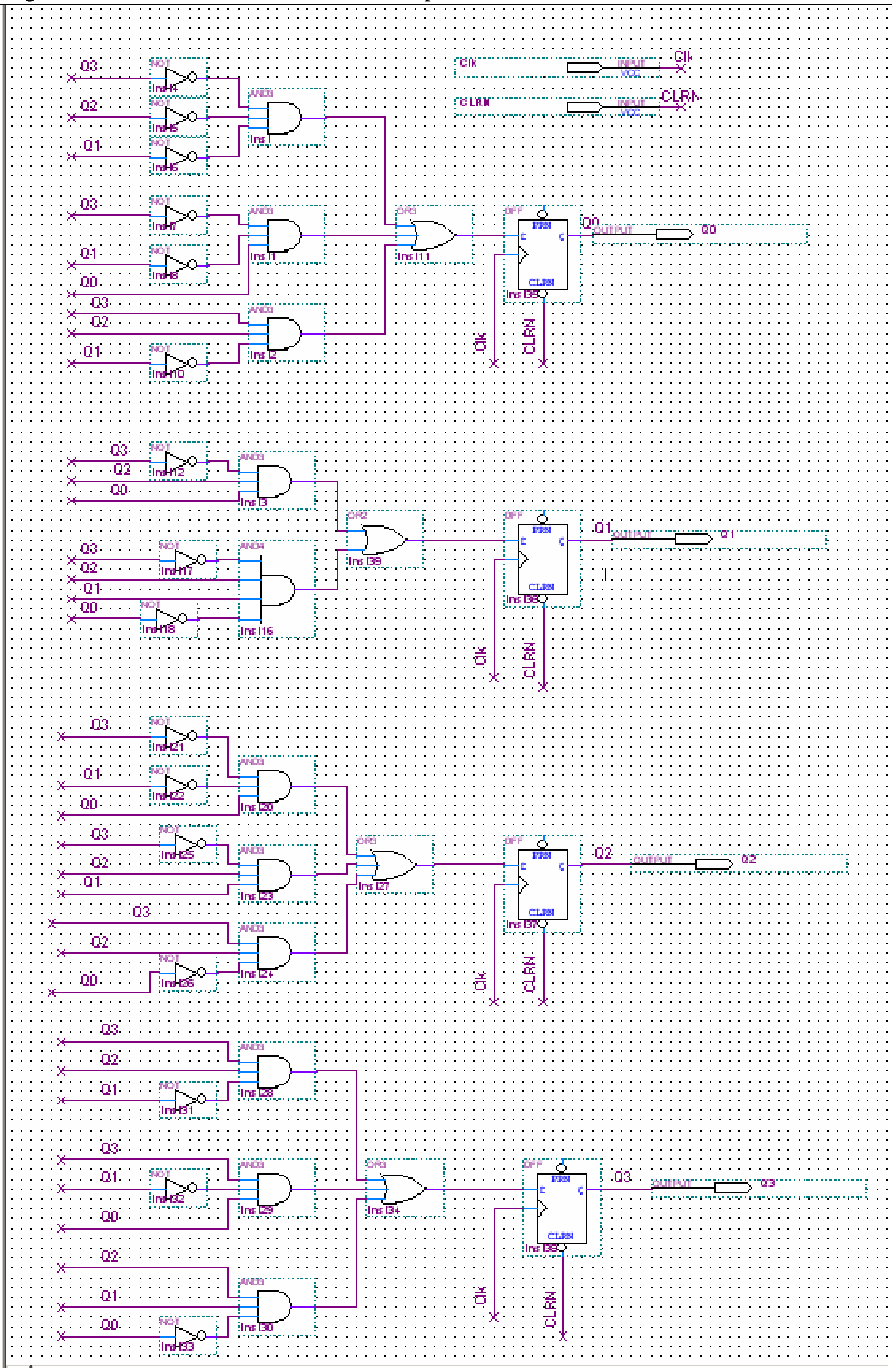
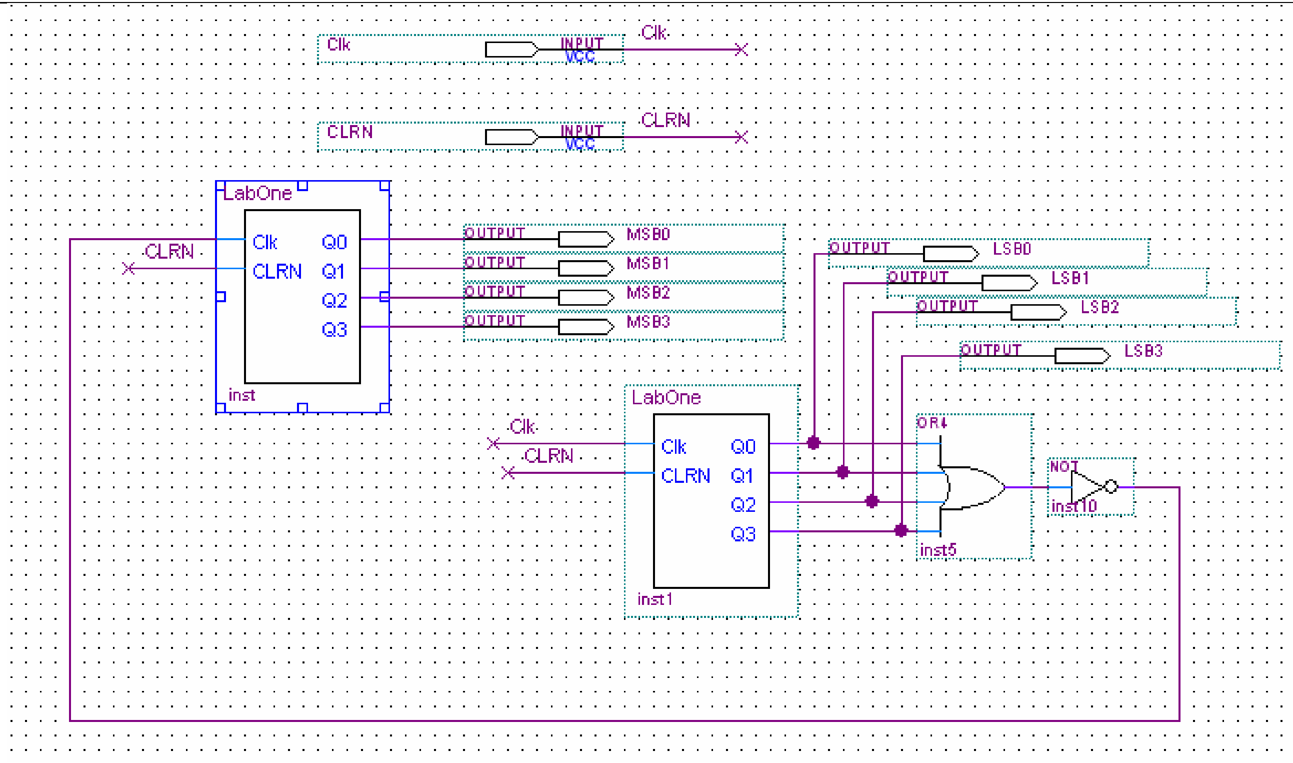


Figure 4. 4-Bit Decade Counter DFF implementation

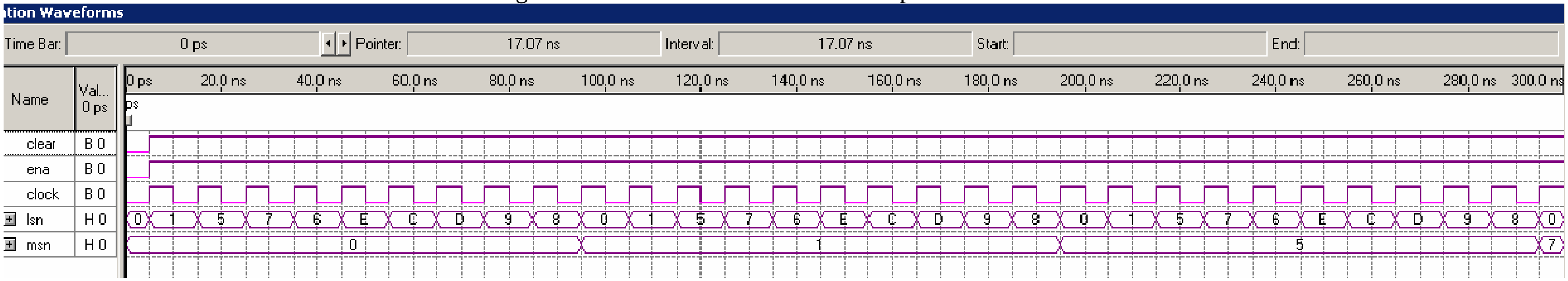


The clock of the cascaded counter is derived from the output of the first counter, thus it counts once for every 9cycle count of the first counter, implementing a 0-99 count.

Results

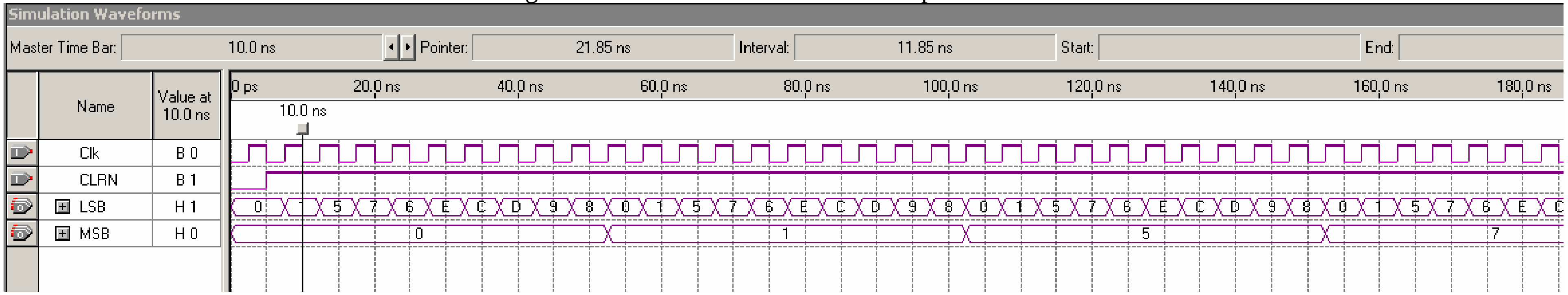
The output waveform of the 0-99 counter shows the implementation of the cyclic code (see the hex equivalent), the least significant nibble (lsn) implements a 4-bit decade counter. And as per our design the most significant nibble (msn) – the cascaded counter counts every 9 counts of the lsn, and it also implements the cyclic code. See figure 5 and 6 for waveforms from TFF and DFF implementation.

Figure 5. Waveform results for TFF implementation.



ation Waveforms
Time Bar: 0 ps
Pointer: 17.07 ns
Interval: 17.07 ns
Start:
End:
Name
Val... 0 ps
0 ps 20.0 ns 40.0 ns 60.0 ns 80.0 ns 100.0 ns 120.0 ns 140.0 ns 160.0 ns 180.0 ns 200.0 ns 220.0 ns 240.0 ns 260.0 ns 280.0 ns 300.0 ns
clear B 0
ena B 0
clock B 0
lsn H 0
0 1 5 7 6 E C D 9 8 0 1 5 7 6 E C D 9 8 0 1 5 7 6 E C D 9 8 0
msn H 0
0 1 5 7

Figure 6. Waveform results for DFF implementation.



Simulation Waveforms
Master Time Bar: 10.0 ns
Pointer: 21.85 ns
Interval: 11.85 ns
Start:
End:
Name
Value at 10.0 ns
0 ps 20.0 ns 40.0 ns 60.0 ns 80.0 ns 100.0 ns 120.0 ns 140.0 ns 160.0 ns 180.0 ns
10.0 ns
Clk B 0
CLRN B 1
LSB H 1
0 1 5 7 6 E C D 9 8 0 1 5 7 6 E C D 9 8 0 1 5 7 6 E C D 9 8 0 1 5 7 6 E C
MSB H 0
0 1 5 7

Conclusion

We verified the functionality of a cascaded decade counter that counted values from 0 through 99. We went a step further and implemented the counter using two different designs based on a TFF and a DFF. The results of both designs were the same. This assignment was very useful in teaching us how to design, compile and simulate on Quartus II software.

FPGA implementation of 0-99 Cyclic code decade counter

LAB #2 EE 2160
SEPTEMBER 27, 2004

BY:
Narayanan Krishnamurthy

Purpose:

To implement the schematic of the 8bit 0-99 counter on the Excalibur target board, with on board FLEX20K200EFC484-2X FPGA chip and in the process learn pin assignments procedure. The final goal is to display the output on the on-board 7-segment LED display.

Procedure:

In order to properly display the count from 0-99 on the seven segment LED displays, the output from lab 1 was converted from the BCD sequence 0, 1, 3, 2, 6, 14…. to the sequence 0, 1, 2, 3, 4, 5, 6, …. To achieve this, a VHDL code converter logic block was developed using the VHDL code from the given lab assignment. The input to this code converter logic block was the output from the decade counter.

The next step was to connect the output from the code converter to the seven segment displays. For this, a 74247 decoder was used. Each of the decade counters was connected to its own 74247 decoder.

The next step was to modify the 33.3 MHz clock signal from the Excalibur board. The clock signal needs to be modified since it will update the counters too quickly for the human eye to detect. For this the signal was converted to a 2 Hz signal. At this rate, the seven segment display will update twice every 1 second. In order to implement this, a 24 bit counter was implemented. Using the "MegaWizard Plug-In Manager", a 24 bit counter logic block was generated. Using the following calculations, the 24th bit was chosen to obtain a 2 Hz clock frequency:

$$33.3 \text{ MHz} / (2\text{^}24) = 2 \text{ Hz}$$

The 24th bit output from the counter was then used as input to the clock signal on the DFF.

The next step was to assign input and output pins from the Quartus design to the Excalibur board. For this, the Excalibur Pin Assignments datasheet was used. The seven output pins from each 74247 BCD were mapped to the seven segment displays on the Excalibur board. The clock signal was mapped to pin L6 and the reset signal was mapped to the push button SW4. Please see the design section of this lab report for a complete list of the pin assignments.

The final step was to download the compiled design on the Excalibur board. The JTAG cable was connected from the parallel port of the computer to the JP3 terminal on the board. The power adapter was connected from the board to the power supply. After giving the "start programming" command on the Quartus menu, the results were seen on the 7 segment LED displays.

Design:

Creation of the CodeConverter.vhd:

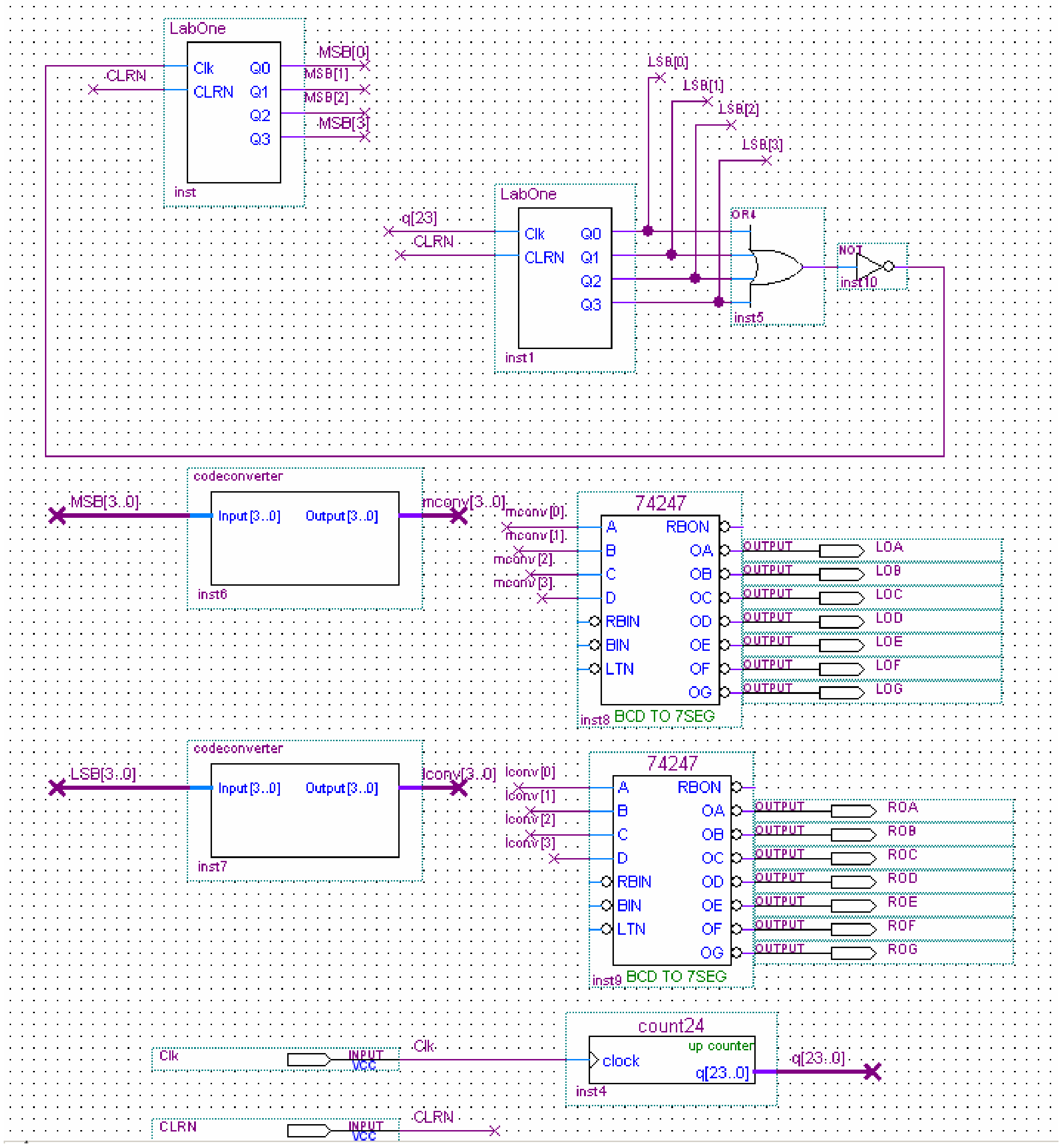
The VHDL code to convert the cyclic code count to a 0-9 binary count is a simple case statement that maps every cyclic code to its equivalent binary output. The symbol thus created is used in the schematics between the decade counter and the 74247 decoder, which expects a binary increment to display 0-9 on the seven segment LED.

```
********************************************************
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY CodeConverter IS
     PORT(
          Input       : IN    std_logic_vector(3 DOWNTO 0);
          Output  : OUT  std_logic_vector(3 DOWNTO 0)
     );
END CodeConverter;


ARCHITECTURE dataflow OF CodeConverter IS
BEGIN
Output <=                "0000" when Input = "0000" else
               "0001" when Input = "0001" else
               "0010" when Input = "0101" else
               "0011" when Input = "0111" else
               "0100" when Input = "0110" else
               "0101" when Input = "1110" else
               "0110" when Input = "1100" else
               "0111" when Input = "1101" else
               "1000" when Input = "1001" else
               "1001" when Input = "1000" else
               "1110"; -- "1110" signifies an error condition
END dataflow;
********************************************************
```

The lab1 design was modified to include the Cyclic to Binary converter, 74247 decoder, and the step down clock 24 bit up counter. The schematic shown in Figure1 gives the interconnection of the various elements.

Note: A-D of 74247 decoder maps to [0-3] of the 4 bit counter

Figure 1: Schematic of FPGA implementation of 0-99 Counter



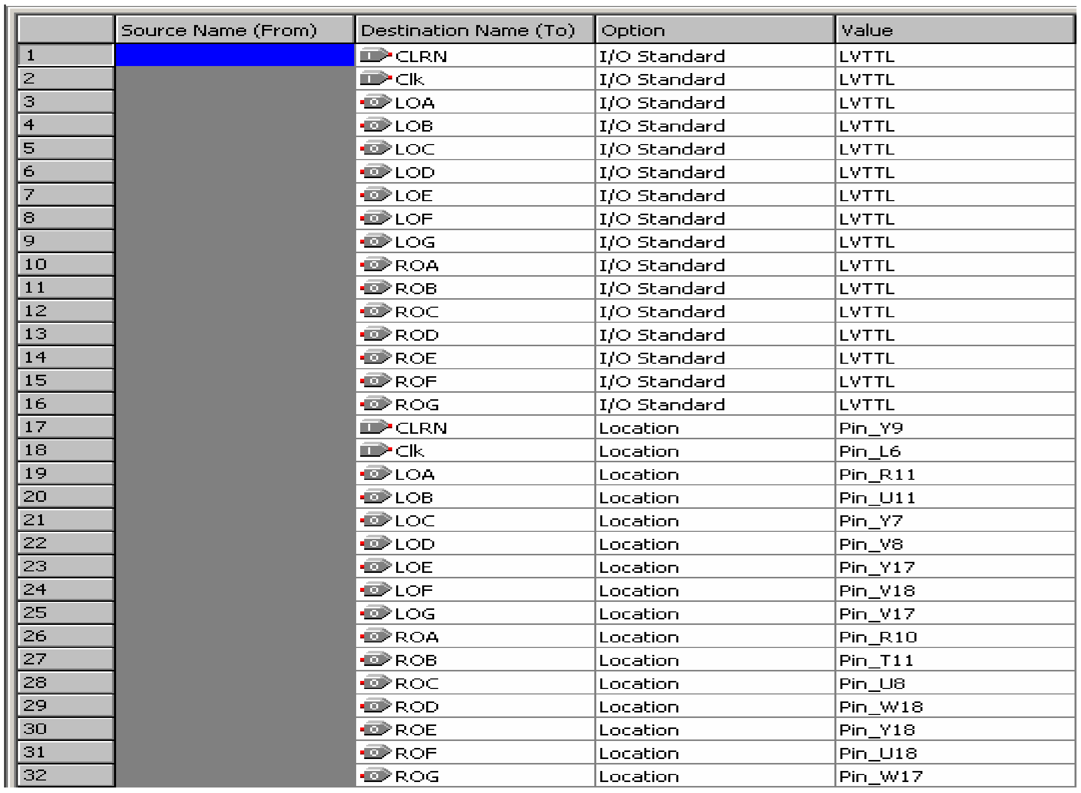
The pin assignment procedure is the process of mapping the logical pins of the schematic to the physical wires on the Excalibur target board. The Pin mapping from the reference excel sheet used for this lab are :

Excalibur Pin Assignments (External I/O)

Seven Segment Left		**Seven Segment Right**		**Push Buttons**	
Name	***APEX Pin***	***Name***	***APEX Pin***	***Name***	***APEX Pin***
a	R11	a	R10	SW7	W9
b	U11	b	T11	SW6	Y8
c	Y7	c	U8	SW5	T9
d	V8	d	W18	SW4	Y9
e	Y17	e	Y18	**Clock**	
f	V18	f	U18	***Name***	***APEX Pin***
g	V17	g	W17	Clock	L6

The left seven segment pins LOA-LOG maps to R11, U11,Y7, V8,Y17,V18,V17, the left segment, displays the most significant nibble of the 8 bit counter. The right segment pins ROA-ROG maps to R10, T11, U8, W18, Y18, U18, W17, the right segment, displays the least significant nibble of the 8 bit counter. (See Figure 2).

Figure 2: Table of Pin Assignments



	Source Name (From)	Destination Name (To)	Option	Value
1		CLRN	I/O Standard	LVTTL
2		Clk	I/O Standard	LVTTL
3		LOA	I/O Standard	LVTTL
4		LOB	I/O Standard	LVTTL
5		LOC	I/O Standard	LVTTL
6		LOD	I/O Standard	LVTTL
7		LOE	I/O Standard	LVTTL
8		LOF	I/O Standard	LVTTL
9		LOG	I/O Standard	LVTTL
10		ROA	I/O Standard	LVTTL
11		ROB	I/O Standard	LVTTL
12		ROC	I/O Standard	LVTTL
13		ROD	I/O Standard	LVTTL
14		ROE	I/O Standard	LVTTL
15		ROF	I/O Standard	LVTTL
16		ROG	I/O Standard	LVTTL
17		CLRN	Location	Pin_Y9
18		Clk	Location	Pin_L6
19		LOA	Location	Pin_R11
20		LOB	Location	Pin_U11
21		LOC	Location	Pin_Y7
22		LOD	Location	Pin_V8
23		LOE	Location	Pin_Y17
24		LOF	Location	Pin_V18
25		LOG	Location	Pin_V17
26		ROA	Location	Pin_R10
27		ROB	Location	Pin_T11
28		ROC	Location	Pin_U8
29		ROD	Location	Pin_W18
30		ROE	Location	Pin_Y18
31		ROF	Location	Pin_U18
32		ROG	Location	Pin_W17

Note: The pin type 'LVTTL' is chosen for all pin assignments, and all unconnected inputs tristated. The Clk was assigned to L6 and CLRN – the clear to the flip-flops was assigned to the SW4 –(push button).

Results

The seven segment LED display, counted 0-99 and looped over, thus validating our design. The clock input was changed from 2hz to higher speeds to see a faster counter implementation. Higher clock speeds obtained by tapping not the 24th bit but lower order bits (for instance the 20th bit would give a clock of 33.3 MHz / (2^20) = 32 Hz).

Both the TFF and DFF implementations of the decade counter were programmed in FPGA and – the 0-99 count display on the seven segment LED was verified.

Conclusion

The cyclic code decade counter that was designed using flip-flops and logic gates in lab1 was tested on the Altera Excalibur FPGA target board. The seven segment LED display was used to verify the 0-99 count, of the cascaded 4 bit cyclic decade counter.

This assignment was very useful in teaching us about pin-assignments (mapping between the logical nodes of the schematic to the physical nodes on the target board) and to program the target board using the JTAG cable.