Experiment No. 7

Utilization of on-board resources of BASYS-3 FPGA Kit

OBJECTIVE: To realize digital logic through Verilog programs targeting the on-board peripherals of Basys-3 FPGA kit.

THEORY: The Basys-3 FPGA board has the following features:

- 1,800 Kbits of fast block RAM
- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
- Five clock management tiles, each with a phase-locked loop (PLL)
- 90 DSP slices
- Internal clock speeds exceeding 450MHz
- On-chip analog-to-digital converter (XADC)
- 16 user switches
- 16 user LEDs
- 5 user pushbuttons
- 4-digit 7-segment display
- Three Pmod connectors
- Pmod for XADC signals
- 12-bit VGA output
- USB-UART bridge
- Serial Flash
- Digilent USB-JTAG port for FPGA programming and communication
- USB HID Host for mice, keyboards and memory sticks

EXAMPLE 7.1: Write a sequential Verilog code for an 8-bit ALU, verify the design by simulation, and then implement it on the Basys-3 FPGA kit.

Solution: **Truth Table of 8-bit ALU**

Input A	Input B	Select lines ALU_Sel	Operation	Output ALU_Out	CarryOut
00000011	00000011	000	Addition	00000110	0
00000011	00000011	001	Subtraction	00000000	0
00000011	00000011	010	Multiplication	00001001	0
00000011	00000011	011	Division	00000001	0
00000011	00000011	100	Logical left shift of A	00000110	0
00000011	00000011	101	Logical right shift of A	00000001	0
00000011	00000011	110	Rotate left of A	00000110	0
00000011	00000011	111	Rotate right of A	10000001	0

Verilog Code:

```
module ALU( input [7:0] A,B,              // ALU 8-bit Inputs
      input [2:0] ALU_Sel,                // ALU Selection
      output [7:0] ALU_Out,               // ALU 8-bit Output
      output CarryOut);                   // Carry Out Flag
    reg [7:0] ALU_Result;
    wire [8:0] tmp;
    assign ALU_Out = ALU_Result;
    assign tmp = {1'b0,A} + {1'b0,B};
    assign CarryOut = tmp[8];
    always @(*)
    begin
        case(ALU_Sel)
        4'b000: // Addition
           ALU_Result = A + B ;
        4'b001: // Subtraction
           ALU_Result = A - B ;
        4'b010: // Multiplication
           ALU_Result = A * B;
        4'b011: // Division
           ALU_Result = A/B;
        4'b100: // Logical shift left
           ALU_Result = A<<1;
         4'b101: // Logical shift right
           ALU_Result = A>>1;
         4'b110: // Rotate left
           ALU_Result = {A[6:0],A[7]};
         4'b111: // Rotate right
           ALU_Result = {A[0],A[7:1]};
         default: ALU_Result = A + B ;
        endcase
    end
endmodule
```

Simulation Results:

Input: A [7:0] = 0000 0011, B [7:0] = 0000 0011, ALU_Sel[2:0]=010

Output: ALU_Out[7:0] = 0000 0001, CarryOut=0

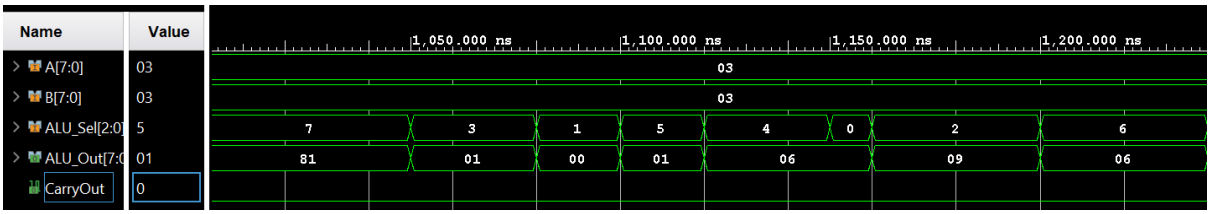


Fig. 7.1: Simulation results of example 7.1

IO Pin Assignment:

Bit	Input A	Input B	ALU_Sel	Output ALU_Out	CarryOut
0	V17	V2	T17	V13	U16
1	V16	T3	U18	V3	
2	W16	T2	W19	W3	
3	W17	R3		U3	
4	W15	W2		P3	
5	V15	U1		N3	
6	W14	T1		P1	
7	W13	R2		L1	

Hardware Results:

Input: A[7:0]

SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

Input: B[7:0]

SW15	SW14	SW13	SW12	SW11	SW10	SW9	SW8
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

Input: ALU_Sel[2:0]

BTNL	BTNC	BTNR
ON	OFF	ON

Output: ALU_Out[7:0]

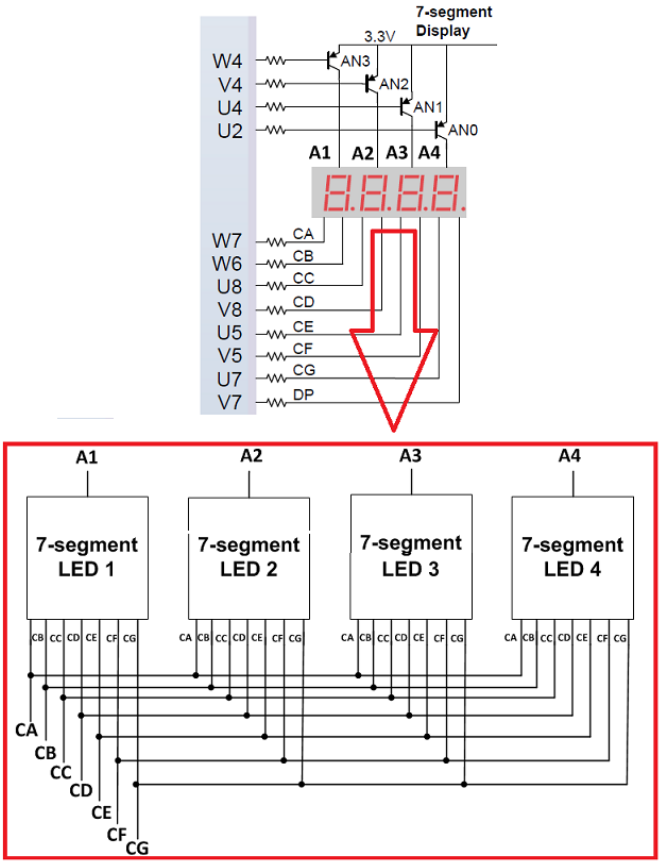
LD15	LD14	LD13	LD12	LD11	LD10	LD9	LD8
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON

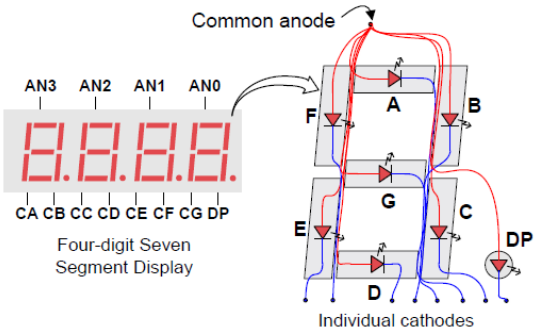
Output: CarryOut

LD0
OFF

EXAMPLE 7.2: Write a Verilog code to display the hexadecimal equivalent of the 16-bit binary input, given through the user switches, on the seven-segment display of the Basys-3 FPGA kit.

Solution:





Verilog Code:

```
module Seven_segment(
    input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
    input reset, // Reset
    input [15:0] switch, // Binary input
    output reg [3:0] Anode_Activate, // anode signals of  7-segment LED display
    output reg [6:0] LED_out   // cathode patterns of the 7-segment LED display
    );

    reg [3:0] LED_BCD;
    reg [19:0] refresh_counter;
    wire [1:0] Digit_activator;

   always @(posedge clock_100Mhz or posedge reset)
    begin
        if(reset==1)
            refresh_counter <= 0;
        else
            refresh_counter <= refresh_counter + 1;
    end

    assign Digit_activator = refresh_counter[19:18];

    always @(*)
    begin
        case(Digit_activator)
        2'b00: begin
                  Anode_Activate = 4'b0111;
                                    // activate display1 and deactivate display2, 3, 4
                  LED_BCD = switch[15:12];
                                    // the first digit of the 16-bit number
               end
        2'b01: begin
                  Anode_Activate = 4'b1011;
                                    // activate display2 and deactivate display1, 3, 4
                  LED_BCD = switch[11:8];
                                    // the second digit of the 16-bit number
               end
        2'b10: begin
                  Anode_Activate = 4'b1101;
                                    // activate display 3 and deactivate display2, 1, 4
                  LED_BCD = switch[7:4];
                                    // the third digit of the 16-bit number
                end
```

```
        2'b11: begin
                  Anode_Activate = 4'b1110;
                                  // activate LED4 and Deactivate LED2, LED3, LED1
                  LED_BCD = switch[3:0];// the fourth digit of the 16-bit number
                end
        endcase
    end
    // Cathode patterns of the 7-segment LED display
    always @(*)
    begin
        case(LED_BCD)
        4'b0000: LED_out = 7'b0000001; // "0"
        4'b0001: LED_out = 7'b1001111; // "1"
        4'b0010: LED_out = 7'b0010010; // "2"
        4'b0011: LED_out = 7'b0000110; // "3"
        4'b0100: LED_out = 7'b1001100; // "4"
        4'b0101: LED_out = 7'b0100100; // "5"
        4'b0110: LED_out = 7'b0100000; // "6"
        4'b0111: LED_out = 7'b0001111; // "7"
        4'b1000: LED_out = 7'b0000000; // "8"
        4'b1001: LED_out = 7'b0000100; // "9"
        4'b1010: LED_out = 7'b0001000; // "A"
        4'b1011: LED_out = 7'b1100000; // "b"
        4'b1100: LED_out = 7'b0110001; // "C"
        4'b1101: LED_out = 7'b1000010; // "d"
        4'b1110: LED_out = 7'b0110000; // "E"
        4'b1111: LED_out = 7'b0111000; // "F"
        default: LED_out = 7'b1111111;
        endcase
    end
endmodule
```

Simulation Results:

Input: switch[15:0] = 0100 0101 0110 0011, reset = 1→0, clock=100MHz

Output:

Clock_pulses	Digit_activator	LED_BCD	LED_out (7-segment code)
at 1^{st} pulse	0 (1^{st} 7-seg display)	4	1001100
at $(1x2^{18})^{th}$ pulse	1 (2^{nd} 7-seg display)	5	0100100
at $(2x2^{18})^{th}$ pulse	2 ($3r^{d}$ 7-seg display)	6	0100000
at $(3x2^{18})^{th}$ pulse	3 (4^{th} 7-seg display)	3	0000110

The above sequence keeps repeating at an interval of 2^{18} clock pulses

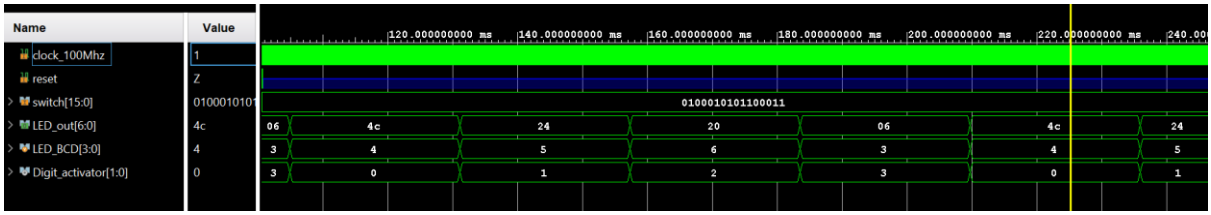


Fig. 7.2: Simulation results of example 7.2

Hardware Results:

Input: switch[15:0]

SW 15	SW 14	SW 13	SW 12	SW 11	SW 10	SW 9	SW 8	SW 7	SW 6	SW 5	SW 4	SW 3	SW 2	SW 1	SW 0
OFF	ON	OFF	OFF	OFF	ON	OFF	ON	OFF	ON	ON	OFF	OFF	OFF	ON	ON

Output:

EXAMPLE 7.3: Write a Verilog code to display the decimal equivalent of the 4-bit binary number, input through the user switches, on the seven-segment display of the Basys 3 kit.

Solution:

Binary Input	Decimal digit - 1	Decimal digit - 0
0000	0	0
0001	0	1
0010	0	2
0011	0	3
0100	0	4
0101	0	5
0110	0	6
0111	0	7
1000	0	8
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

Verilog Code:

```
module Bin_2_Dec(
    input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
    input reset, // reset
    input [3:0] switch, //4-bit Binary input
    output reg [3:0] Anode_Activate,
                                        // anode signals of the 7-segment LED display
    output reg [6:0] LED_out    // cathode patterns of the 7-segment LED display
);

reg Z;
reg [3:0] LED_BCD;
reg [19:0] refresh_counter;
wire [1:0] Digit_activator;

always @(posedge clock_100Mhz or posedge reset)
 begin
     if(reset==1)
         refresh_counter <= 0;
     else
         refresh_counter <= refresh_counter + 1;
 end

assign Digit_activator = refresh_counter[18];

always @(*)
 begin
     Z = (switch>4'b1001)?1:0;
     case(Digit_activator)
     2'b0: begin
             Anode_Activate = 4'b1101;
             LED_BCD = {3'b000,Z};
           end
     2'b1: begin
         Anode_Activate = 4'b1110;
             if(Z==1)
                 LED_BCD = switch-4'b1010;
             else
                 LED_BCD = switch;
             end
     endcase
 end
```

```
    // Cathode patterns of the 7-segment LED display
    always @(*)
    begin
        case(LED_BCD)
        4'b0000: LED_out = 7'b0000001; // "0"
        4'b0001: LED_out = 7'b1001111; // "1"
        4'b0010: LED_out = 7'b0010010; // "2"
        4'b0011: LED_out = 7'b0000110; // "3"
        4'b0100: LED_out = 7'b1001100; // "4"
        4'b0101: LED_out = 7'b0100100; // "5"
        4'b0110: LED_out = 7'b0100000; // "6"
        4'b0111: LED_out = 7'b0001111; // "7"
        4'b1000: LED_out = 7'b0000000; // "8"
        4'b1001: LED_out = 7'b0000100; // "9"
        default: LED_out = 7'b1111111; // "0"
        endcase
    end
endmodule
```

Simulation Results:

Input: switch[3:0]=1100, reset = 1→0, clock=100MHz

Output:

Clock_pulses	Digit_activator	LED_BCD	LED_out (7-segment code)
at 1st pulse	0 (3rd 7-seg display)	1	1001111
at $(1x2^{18})^{th}$ pulse	1 (4^{th} 7-seg display)	2	0010010

The above sequence keeps repeating at an interval of 2^{18} clock pulses

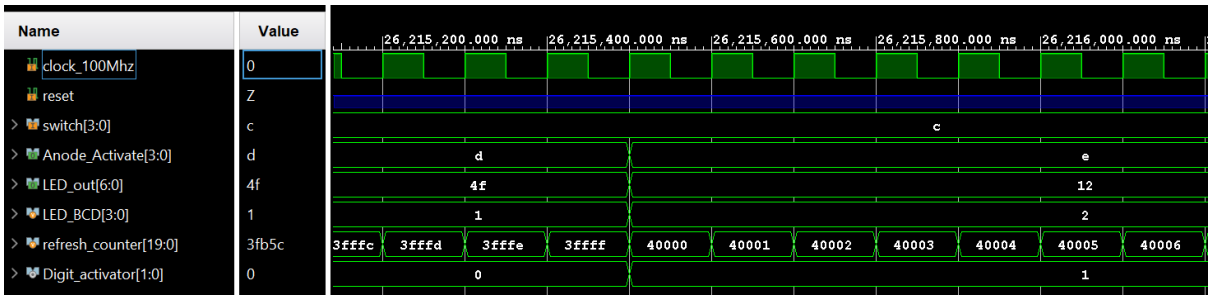


Fig. 7.3: Simulation results of example 7.3

IO Pin Assignment:

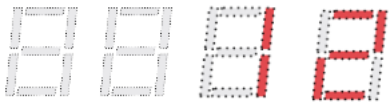
Switch[3:0]		Anode_Activate [3:0]		LED_out [6:0]			
Bit 3	W17	Bit 3: Display 1	W4	Bit 6: Seg a	W7	Bit 2: Seg e	U5
Bit 2	W16	Bit 2: Display 2	V4	Bit 5: Seg b	W6	Bit 1: Seg f	V5
Bit 1	V16	Bit 1: Display 3	U4	Bit 4: Seg c	U8	Bit 0: Seg g	U7
Bit 0	V17	Bit 0: Display 4	U2	Bit 3: Seg d	V8		
Reset: T17				**Clock:** W5			

Hardware Results:

Input: switch[3:0]

SW 3	SW 2	SW 1	SW 0
ON	ON	OFF	OFF

Output:



EXAMPLE 3.4: Write a Verilog code to perform the 4-bit BCD addition and display the results on the seven-segment display of the Basys 3 FPGA kit.

Solution:

BCD inputs (Input1+Input2+Carry)	BCD sum	Binary sum
0000+0101+0	Carry=0, sum=0101	Carry=0, sum=0101
0001+0101+0	Carry=0, sum=0110	Carry=0, sum=0110
0010+0101+0	Carry=0, sum=0111	Carry=0, sum=0111
0011+0101+0	Carry=0, sum=1101	Carry=0, sum=1000
0100+0101+0	Carry=0, sum=1001	Carry=0, sum=1001
0101+0101+0	Carry=1, sum=0000	Carry=0, sum=1010
0110+0101+0	Carry=1, sum=0001	Carry=0, sum=1011
0111+0101+0	Carry=1, sum=0010	Carry=0, sum=1100
1000+0101+0	Carry=1, sum=0011	Carry=0, sum=1101
1001+0101+0	Carry=1, sum=0100	Carry=0, sum=1110

On the 4-digit segment display of the Basys 3 kit, the binary sum is programmed to display on the leftmost two digits, and the BCD sum to be displayed on the rightmost two digits.

Verilog Code:

```
module BCD_Addition(
        input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
        input reset, // reset
        input [3:0] a,b, // two BCD inputs
        input carry_in, // Initial carry
        output reg [3:0] Anode_Activate,
                                    // anode signals of the 7-segment LED display
        output reg [6:0] LED_out // cathode patterns of the 7-segment LED display
    );
```

```
 reg [3:0] sum;
 reg carry;
 reg bin_carry;
 reg [3:0] bin_sum;
 reg [3:0] LED_BCD;
 reg [19:0] refresh_counter;
 wire [1:0] Digit_activator;

 always @(a,b,carry_in)
 begin
     {bin_carry, bin_sum} = a+b+carry_in; //add all the inputs
     if(bin_sum > 9)
      begin
          carry = 1;                      //set the carry output
          sum = bin_sum+6;                //add 6, if result is more than 9.
      end
      else
      begin
          carry = 0;
          sum = bin_sum[3:0];
     end
 end

always @(posedge clock_100Mhz or posedge reset)
 begin
     if(reset==1)
         refresh_counter <= 0;
     else
         refresh_counter <= refresh_counter + 1;
 end

assign Digit_activator = refresh_counter[19:18];

always @(*)
 begin
     case(Digit_activator)
     2'b00: begin
             Anode_Activate = 4'b1110;
             LED_BCD = sum;
           end
     2'b01: begin
         Anode_Activate = 4'b1101;
         LED_BCD = {3'b000,carry};
         end
```

```
        2'b10: begin
                Anode_Activate = 4'b1011;
                LED_BCD = bin_sum;
              end
        2'b11: begin
            Anode_Activate = 4'b0111;
            LED_BCD = {3'b000,bin_carry};
            end
        endcase
    end
```

// Cathode patterns of the 7-segment LED display

```
    always @(*)
    begin
        case(LED_BCD)
        4'b0000: LED_out = 7'b0000001; // "0"
        4'b0001: LED_out = 7'b1001111; // "1"
        4'b0010: LED_out = 7'b0010010; // "2"
        4'b0011: LED_out = 7'b0000110; // "3"
        4'b0100: LED_out = 7'b1001100; // "4"
        4'b0101: LED_out = 7'b0100100; // "5"
        4'b0110: LED_out = 7'b0100000; // "6"
        4'b0111: LED_out = 7'b0001111; // "7"
        4'b1000: LED_out = 7'b0000000; // "8"
        4'b1001: LED_out = 7'b0000100; // "9"
        4'b1010: LED_out = 7'b0001000; // "A"
        4'b1011: LED_out = 7'b1100000; // "b"
        4'b1100: LED_out = 7'b0110001; // "C"
        4'b1101: LED_out = 7'b1000010; // "d"
        4'b1110: LED_out = 7'b0110000; // "E"
        4'b1111: LED_out = 7'b0111000; // "F"
        default: LED_out = 7'b1111111; // "0"
        endcase
    end
endmodule
```

Simulation Results:

Input: a[3:0] = 1000, b[3:0]=0101, carry_in=0, reset = 1→0, clock=100MHz

Output:

Clock_pulses	**Digit_activator**	**LED_BCD**	**LED_out (7-segment code)**
at 1^{st} pulse	0 (1^{st} 7-seg display)	0	0000001
at $(1x2^{18})^{th}$ pulse	1 (2^{nd} 7-seg display)	d	1000010

at $(2x2^{18})^{th}$ pulse	2 ($3r^{d}$ 7-seg display)	1	1001111
at $(3x2^{18})^{th}$ pulse	3 (4^{th} 7-seg display)	3	0000110

The above sequence keeps repeating at an interval of 2^{18} clock pulses

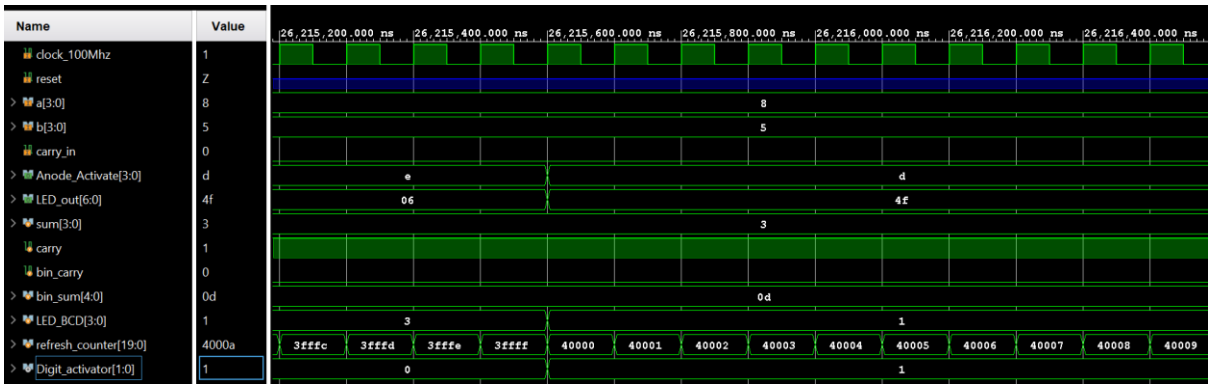


Fig. 7.4: Simulation results of example 7.4

IO Pin Assignment:

a[3:0]	B[3:0]	Anode_Activate [3:0]		LED_out [6:0]			
W17	W13	Bit 3: Display 1	W4	Bit 6: Seg a	W7	Bit 2: Seg e	U5
W16	W14	Bit 2: Display 2	V4	Bit 5: Seg b	W6	Bit 1: Seg f	V5
V16	V15	Bit 1: Display 3	U4	Bit 4: Seg c	U8	Bit 0: Seg g	U7
V17	W15	Bit 0: Display 4	U2	Bit 3: Seg d	V8		
Reset: T17		**carry_in:** R2		**Clock:** W5			

Hardware Results:

Input:

carry_in	b[3:0]				a[3:0]			
SW 15	**SW 7**	**SW 6**	**SW 5**	**SW 4**	**SW 3**	**SW 2**	**SW 1**	**SW 0**
OFF	**OFF**	**ON**	**OFF**	**ON**	**ON**	**OFF**	**OFF**	**OFF**

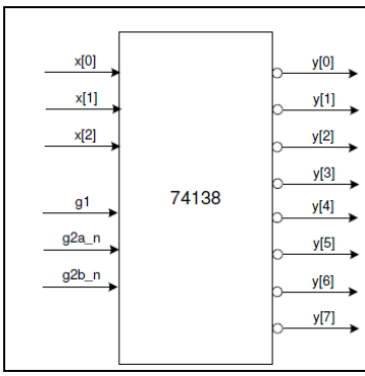
Output:

EXERCISE PROBLEMS:

1. Write the structural Verilog code to realize a 3x1 multiplexer using 2x1 multiplexers, synthesize, and implement it through the onboard user switches and LEDs of the Basys 3 FPGA kit.
2. Write a Verilog code to display the push button's status on the seven-segment display of the Basys 3 kit as follows:

Pressed Push Button	Character to be displayed on the seven segment display
BTNR	P
BTNU	U
BTNL	L
BTND	d
BTNC	C

3. Write a data flow Verilog code to create a 4-bit ripple carry adder, synthesize, and implement it on Basys 3 hardware.
4. Design and implement a popular IC, 74138, functionality using dataflow modeling of Verilog. The IC symbol and truth table are given below. Verify the functionality on Basys 3 hardware.



g_1 g_{2a_n} g_{2b_n}	x_0 x_1 x_2	y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7
0 x x	x x x	1 1 1 1 1 1 1 1
x 1 x	x x x	1 1 1 1 1 1 1 1
x x 1	x x x	1 1 1 1 1 1 1 1
1 0 0	0 0 0	0 1 1 1 1 1 1 1
1 0 0	0 0 1	1 0 1 1 1 1 1 1
1 0 0	0 1 0	1 1 0 1 1 1 1 1
1 0 0	0 1 1	1 1 1 0 1 1 1 1
1 0 0	1 0 0	1 1 1 1 0 1 1 1
1 0 0	1 0 1	1 1 1 1 1 0 1 1
1 0 0	1 1 0	1 1 1 1 1 1 0 1
1 0 0	1 1 1	1 1 1 1 1 1 1 0

5. Design a comparator that compares two 2-bit numbers (A and B) and assert outputs indicating whether the decimal equivalent of word A is less than, greater than, or equal to that of word B. Realize it on Basys 3 hardware.