Name: Joseph Hany **ID:900182870**

Architecture Lab 2 Report

Experiment 0:

- This experiment aims to use the UART module as the input to the board instead of the swithces. This can be done by sending serial commands to the module throuogh teraterm program which ill convert it to signal that will be used as our input in the future experiments.

Experiment 1:

- This experiment aims to implement a 4-bit ripple carry adder (RCA) which takes two 4 bit input positive numbers and returns their sum as an output.

- In order to implement such a circuit, we should do the following:

  1- write a “Full_Adder” module that adds two bits and acts as the building block of the ripple carry adder.

  2- write an “RCA4” module the instantiates n istantations of theFull_Adder module

  3- Add the modules used in lab1 to this experiment in order to be able to see the results of the experiemnt on the 7-segment displayer.

  4- write a “const” file in order to inform the software what physical pins on the Nexys A7-100T FPGA Board that we plan on using or connecting to in relation to the Verilog code that we wrote to describe the behavior of the FPGA.

- **Explaining code functionality by commenting on the code:**

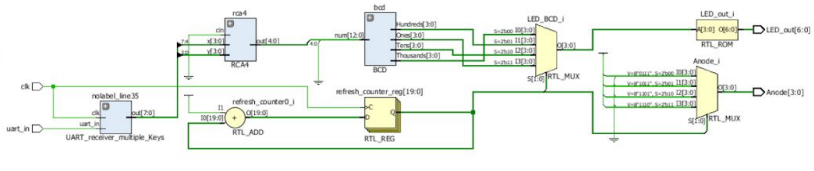
```
module Full_Adder(          // the full adder module takes three inputs: two bits and a
                            // carray if exists  and outputs the sum of these two bits
                            // and a carry out if exists (i.e. 1 bit + 1bit = 0 bit + carryout)
input x,y,cin,
output sum,carry_out
   );
   assign sum = x ^ y ^ cin;    // the sum can be computed by xoring the three inputs
   assign carry_out = x&y | (x^y) & cin;
```

```
Endmodule
```

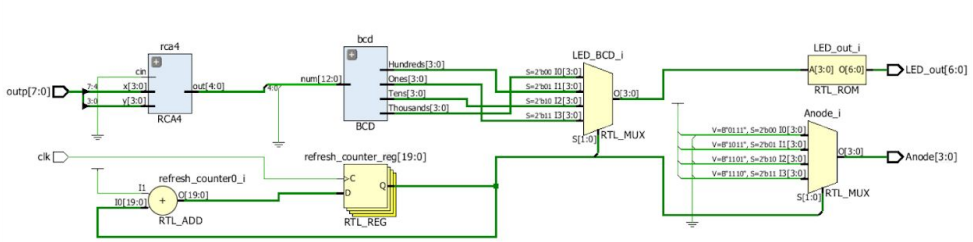
```
module RCA4 #(parameter n = 4) (input [n-1:0] x,input [n-1:0] y,input cin, output [n:0]
out,output last_carry);

// The ripple carry adder module takes 3 inputs: 2 4-bits input numbers and a carry-in if
exists and outputs the sum of these two numbers and the last carry if exists.

wire [n-1:0] sum;
wire carry[n:0];          // this bus of wires is used to store all the carry-ins
assign carry[0]=cin;

genvar i;
Generate                  // generate block to instantiate the full adder module n times
   for(i=0;i<n;i=i+1)
   begin
      Full_Adder fa(.x(x[i]),.y(y[i]),.cin(carry[i]),.sum(sum[i]),.carry_out(carry[i+1]));
   end
endgenerate

assign out={carry[n],sum};

assign last_carry=carry[n];


endmodule
```

Schematics:

With the UART module:



Without the UART module:



It would be apparent that this implementation is simpler than the circuit found in experiment three which will eventually result in the lonhger delay.

Utilization Hierarchy:

Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	LUT Flip Flop Pairs (63400)	Bonded IOB (210)	BUFGCTRL (32)
Four_Digit_Seven_Segment_Dr...	226	316	122	226	100	13	1
nolabel_line35 (UART_recei...	223	296	117	223	99	0	0

Utilization Report:

```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              |  226 |     0 |     63400 |  0.36 |
|   LUT as Logic          |  226 |     0 |     63400 |  0.36 |
|   LUT as Memory         |    0 |     0 |     19000 |  0.00 |
| Slice Registers         |  316 |     0 |    126800 |  0.25 |
|   Register as Flip Flop |  316 |     0 |    126800 |  0.25 |
|   Register as Latch     |    0 |     0 |    126800 |  0.00 |
| F7 Muxes                |    0 |     0 |     31700 |  0.00 |
| F8 Muxes                |    0 |     0 |     15850 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

Total slice LUTs used = 226
Total slice Registers used = 316

Hold time:

Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 31	∞	2	7	ref.../C	Anode[1]	2.103	1.445	0.657	-∞	sys_clk_pin
Path 32	∞	3	7	ref.../C	LED_out[4]	2.152	1.448	0.704	-∞	sys_clk_pin
Path 33	∞	2	7	ref.../C	Anode[0]	2.177	1.504	0.673	-∞	sys_clk_pin
Path 34	∞	2	7	ref.../C	Anode[3]	2.272	1.521	0.752	-∞	sys_clk_pin
Path 35	∞	3	7	ref.../C	LED_out[0]	2.362	1.492	0.870	-∞	sys_clk_pin
Path 36	∞	3	7	ref.../C	LED_out[3]	2.388	1.505	0.883	-∞	sys_clk_pin
Path 37	∞	2	7	ref.../C	Anode[2]	2.689	1.484	1.205	-∞	sys_clk_pin

Hold time= 2.689

Setup time:

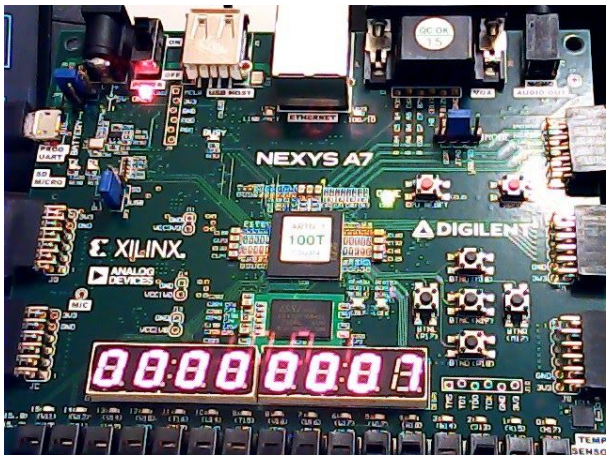
Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 21	∞	4		7 nol.../C	LED_out[5]	10.930	4.383	6.547	∞	sys_clk_pin
Path 22	∞	4		7 nol.../C	LED_out[6]	10.750	4.405	6.345	∞	sys_clk_pin
Path 23	∞	4		7 nol.../C	LED_out[1]	10.173	4.389	5.784	∞	sys_clk_pin
Path 24	∞	4		7 nol.../C	LED_out[2]	9.723	4.362	5.361	∞	sys_clk_pin
Path 25	∞	4		7 nol.../C	LED_out[3]	9.625	4.378	5.247	∞	sys_clk_pin
Path 26	∞	4		7 nol.../C	LED_out[0]	9.234	4.365	4.868	∞	sys_clk_pin
Path 27	∞	4		7 nol.../C	LED_out[4]	9.018	4.321	4.697	∞	sys_clk_pin

Setup time = 10.930

Results

- As a result of this experiment, we can now press of the keyboard from home the two input 4-bits numbers and see the addition of them on the board.
  (as shown in the screenshot below)

Camera Screenshot:



This screenshot shows the result of adding those two numbers in bits :
5 → 0101
2 → 0010

———
7 → 0111

Experiment 2:

- This experiment aims to implement A 4-digit 7-segment display that takes a number and outputs it on the 4-digit 7-segment display.

- In order to implement such a circuit, we should do the following:

  1- Copy the code found in the first lap with optimization that displays the numbers on the 7-segment display.

  2- add on it a piece of code that detects the negative number and displays it on the screen .

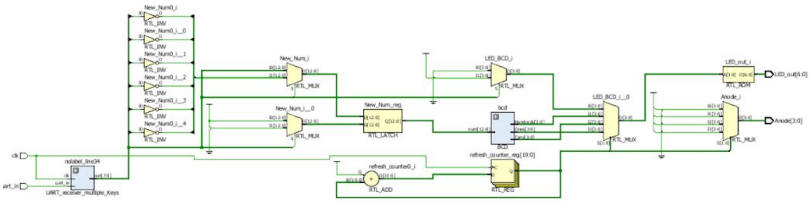
  3- write an “FDSSD_constraint” file in order to inform the software what physical pins on the Nexys A7-100T FPGA Board that we plan on using or connecting to in relation to the Verilog code that we wrote to describe the behavior of the FPGA.

- **Explaining code functionality by commenting on the code:**

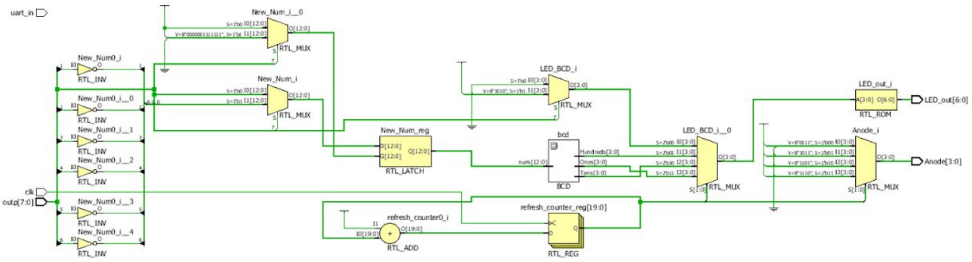
```
reg [12:0] New_Num;
reg Sign;
integer i;
always @(*)
begin
case(Num[7]) //We had to add this code that recognizes whether the input is negative or
             //not and if negative, it sets the sign register to 1 else it is 0.
 1'b0: begin
   Sign=1'b0;
   New_Num=Num;
  end
 1'b1: begin
 Sign=1'b1;
 New_Num=-Num;     // if the numebr was negative we perform two’s complement on it,
                   //where the two’s complement of the negative number= positive
                   // number
 end
endcase
end
```

Schematics:

With the UART module:



Without the UART module:



Utilization Hierarchy:

Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	LUT Flip Flop Pairs (63400)	Bonded IOB (210)	BUFGCTRL (32)
Four_Digit_Seven_Segment_Dr...	206	316	121	206	111	13	1
nolabel_line34 (UART_recei...	202	296	115	202	110	0	0

Utilization Report:

Site Type	Used	Fixed	Available	Util%
Slice LUTs	206	0	63400	0.32
LUT as Logic	206	0	63400	0.32
LUT as Memory	0	0	19000	0.00
Slice Registers	316	0	126800	0.25
Register as Flip Flop	316	0	126800	0.25
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Total slice LUTs used = 206
Total slice Registers used = 316

Hold time:

Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 31	∞	2	13	ref.../C	Anode[1]	2.012	1.422	0.589	-∞	sys_clk_pin
Path 32	∞	2	13	ref.../C	Anode[0]	2.059	1.422	0.637	-∞	sys_clk_pin
Path 33	∞	2	13	ref.../C	Anode[3]	2.208	1.504	0.704	-∞	sys_clk_pin
Path 34	∞	3	6	nol.../C	LED_out[4]	2.365	1.448	0.917	-∞	sys_clk_pin
Path 35	∞	3	13	ref.../C	LED_out[0]	2.396	1.469	0.927	-∞	sys_clk_pin
Path 36	∞	3	13	ref.../C	LED_out[3]	2.415	1.547	0.868	-∞	sys_clk_pin
Path 37	∞	3	6	nol.../C	LED_out[2]	2.529	1.488	1.040	-∞	sys_clk_pin

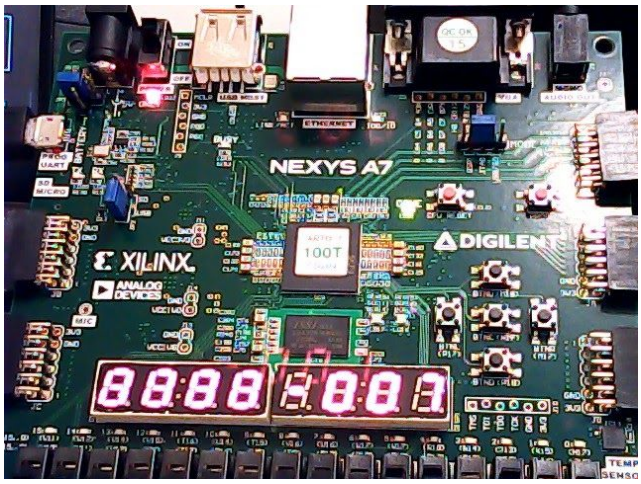
Hold time= 2.529

Setup time:

Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 21	∞	5	9	nol.../C	LED_out[5]	11.624	4.569	7.054	∞	sys_clk_pin
Path 22	∞	5	8	nol.../C	LED_out[1]	11.468	4.811	6.657	∞	sys_clk_pin
Path 23	∞	5	8	nol.../C	LED_out[6]	11.352	4.591	6.761	∞	sys_clk_pin
Path 24	∞	5	9	nol.../C	LED_out[2]	10.420	4.548	5.872	∞	sys_clk_pin
Path 25	∞	5	8	nol.../C	LED_out[3]	10.177	4.794	5.383	∞	sys_clk_pin
Path 26	∞	5	8	nol.../C	LED_out[0]	10.080	4.551	5.528	∞	sys_clk_pin
Path 27	∞	5	8	nol.../C	LED_out[4]	9.844	4.507	5.337	∞	sys_clk_pin

Setup time = 11.624

Camera Screenshot:



This screenshot shows the result of entering a negative number 7:
1001→ -7

Experiment 3:

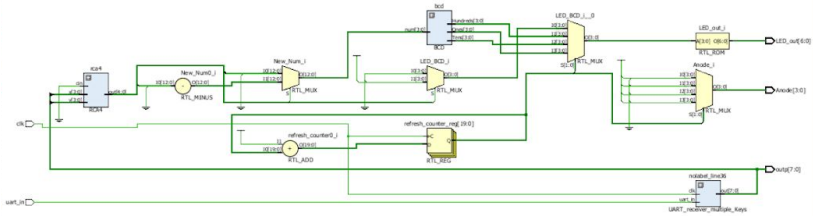
- This experiment aims to make a circuit that has two 4 bits input signed numbers and it adds them and displays the result on the 4 7-segment display with the negative sign if the number is negative.

Explaining code functionality:

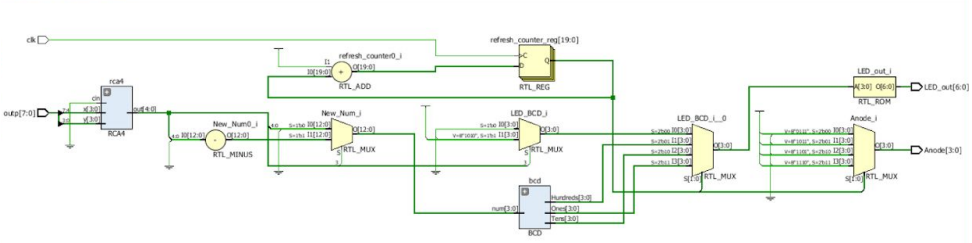
In this experiments' verilog code, we combined the code in the first experiemnt which is the 4 bits ripple carry adder and the code found in the second experiemnt which is the negative sign detection and conversion of the negative numbers to positive.

Schematic:

With the UART module:



Without the UART module:



Here, it can be clearly seen that there are more components in the implementation of this circuit than the circuit found in experiment 1, since we added another logic for recognizing a negative number and displaying the negative sign on the first 7-segment display.

Utilization Hierarchy:

Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	LUT Flip Flop Pairs (63400)	Bonded IOB (210)	BUFGCTRL (32)
Four_Digit_Seven_Segment_Dr...	197	316	117	197	103	21	1
nolabel_line36 (UART_recei...	194	296	111	194	102	0	0

Utilization Report:

Site Type	Used	Fixed	Available	Util%
Slice LUTs	197	0	63400	0.31
LUT as Logic	197	0	63400	0.31
LUT as Memory	0	0	19000	0.00
Slice Registers	316	0	126800	0.25
Register as Flip Flop	316	0	126800	0.25
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Total slice LUTs used = 197
Total slice Registers used = 316

Hold time:

Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 31	∞	1	10	nol.../C	outp[4]	1.851	1.393	0.458	-∞	sys_clk_pin
Path 32	∞	1	5	nol.../C	outp[3]	1.854	1.392	0.462	-∞	sys_clk_pin
Path 33	∞	1	7	nol.../C	outp[1]	1.855	1.400	0.455	-∞	sys_clk_pin
Path 34	∞	1	5	nol.../C	outp[2]	1.908	1.394	0.514	-∞	sys_clk_pin
Path 35	∞	1	5	nol.../C	outp[6]	1.947	1.397	0.550	-∞	sys_clk_pin
Path 36	∞	1	5	nol.../C	outp[7]	2.048	1.397	0.651	-∞	sys_clk_pin
Path 37	∞	2	10	nol.../C	LED_out[4]	2.064	1.403	0.660	-∞	sys_clk_pin

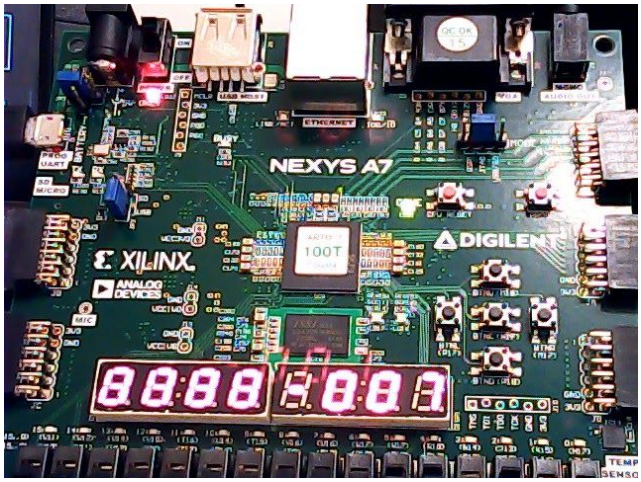
Hold time= 2.064

Setup time:

Name	...	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock
Path 21	∞	4	13	ref.../C	LED_out[5]	11.086	4.611	6.475	∞	sys_clk_pin
Path 22	∞	4	13	ref.../C	LED_out[6]	10.821	4.633	6.188	∞	sys_clk_pin
Path 23	∞	4	13	ref.../C	LED_out[3]	10.714	4.606	6.108	∞	sys_clk_pin
Path 24	∞	4	13	ref.../C	LED_out[1]	10.527	4.617	5.910	∞	sys_clk_pin
Path 25	∞	4	13	ref.../C	LED_out[4]	10.216	4.549	5.667	∞	sys_clk_pin
Path 26	∞	4	13	ref.../C	LED_out[2]	10.080	4.590	5.490	∞	sys_clk_pin
Path 27	∞	2	13	ref.../C	Anode[2]	8.937	4.154	4.783	∞	sys_clk_pin

Setup time = 11.086

Camera Screenshot:



This screenshot shows the result of adding those two numbers in bits :
-5 → 1011
-2 → 1110

_____
-7 → 1001

_____

Conclusion

	Not optimized divisor (Experiment 1)	**Optimized divisor (Experiment 3)**
Total slice LUTs used	226	197
Total slice Registers used	316	316
Hold time	2.689	2.064
Setup time	10.930	11.086

The circuit in experiment 3 clearly takes more setup time since it uses extra circuit components that are used to differentiate between a negative number and a positive number, and if the number is negative it performs two's complement on it in order to be seen as a positive number on the right with a negative sign on the left-most 7-segment displayer.