LAB #3 REPORT

November 10, 2017

McGill University
ECSE 323

Group #27
Matthew Lesko 26069352; Romain Nith 260571471

Contents

Group #27
Matthew Lesko 26069352; Romain Nith 260571471

52 – Element Stack

Description of Circuit's Function

The goal is to create a stack data structure to hold a deck of 52 playing cards. The user can perform 4 operations on the stack:

- NOP: no operation (nothing happens)
- INIT: initializes all 52 cards in the stack from value 0 to 51 and sets the NUM of the counter to 52
- PUSH: pushes the DATA value to the top of the stack. If the stack is full, nothing happens
- POP: pops the value of the stack located to the specified ADDR address. If the stack is empty, nothing happens.

Design and Implementation

Inputs and Outputs

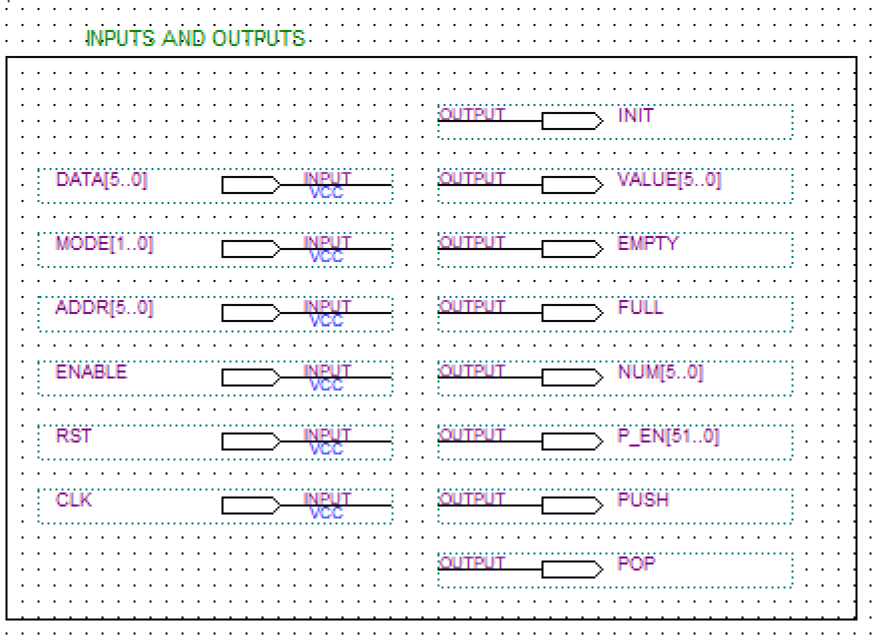


Figure 1 - Input and Output Pins

The input descriptions are:

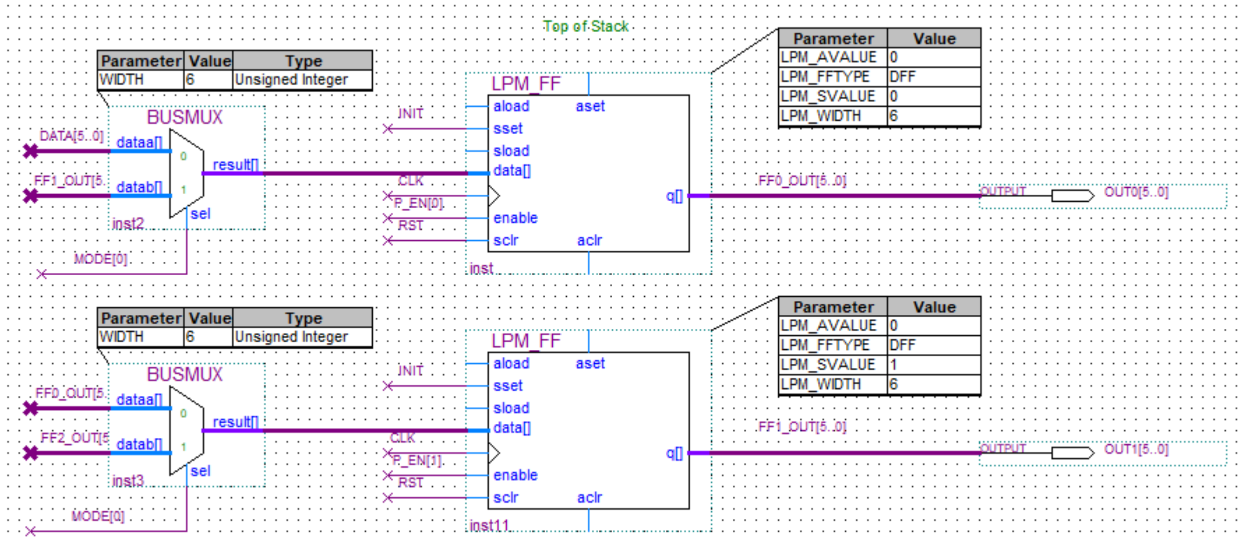
- DATA[5..0]: A 6-bit input to be pushed on top of the stack
- ADDR[5..0]: 6-bit input that acts as the address of the element in the stack that the user's wishes to pop out
- MODE[1..0]: A 2-bit input describing the stack's operation's mode
    - 00: No operation
    - 01: Initialize stack
    - 11: Pop element
    - 10: Push element on top of stack
- ENABLE: 1-bit input that acts as a pulse to enable the operation desired on the stack
- RST: 1-bit input that resets the whole stack, removing all elements

- CLK: 1-bit synchronous clock

The output descriptions are:

- VALUE[5..0]: 6-bit output that describes the number of the element popped out of the stack
- EMPTY: 1-bit output describing if the stack is empty (1 if empty, 0 if not)
- FULL: 1-bit output describing if the stack is full (1 if full, 0 if not)
- P_EN[51..0]: 52-bit output describing which flip flops are enabled for an operation
    - ADDR[5..0] = N, bits P_EN(0) to P_EN(N-1) are 0 bits.
- PUSH: 1-bit output signaling if the operation is a push
- POP: 1-bit output signaling if the operation is a pop
- NUM[5..0]: 6-bit output describing the count of elements within the stack
- INIT: 1-bit output signaling if the operation is in initialization mode

Flip Flop Design



Parameter	Value	Type
WIDTH	6	Unsigned Integer

Parameter	Value
LPM_AVALUE	0
LPM_FFTYPE	DFF
LPM_SVALUE	0
LPM_WIDTH	6

Parameter	Value	Type
WIDTH	6	Unsigned Integer

Parameter	Value
LPM_AVALUE	0
LPM_FFTYPE	DFF
LPM_SVALUE	1
LPM_WIDTH	6

Figure 2 - First two Flip Flops in the Stack

The flip flop design contains:

- LPM_FF (LPM Flip Flop) – An Altera pre-designed and pre-implemented Flip Flop
    - Each Flip Flop has the following configurations:
        - LPM_WIDTH: 6, defining the output length
        - LPM_FFTYPE: DFF, D Flip Flop type
        - LPM_SVALUE: #N of Flip Flop, this is the number set to the element in the flip flop during the initialization operation
        - LPM_AVALUE: 0, the value set to the element when RST is enabled
- BUSMUX – An Altera pre-designed and pre-implemented Multiplexer
    - The first flip flop takes as input DATA[5..0] if the operation is PUSH
    - Every flip flop takes as input the output of the next flip flop if the operation is POP

Push, Pop, Init, and Pop_Enable Variables

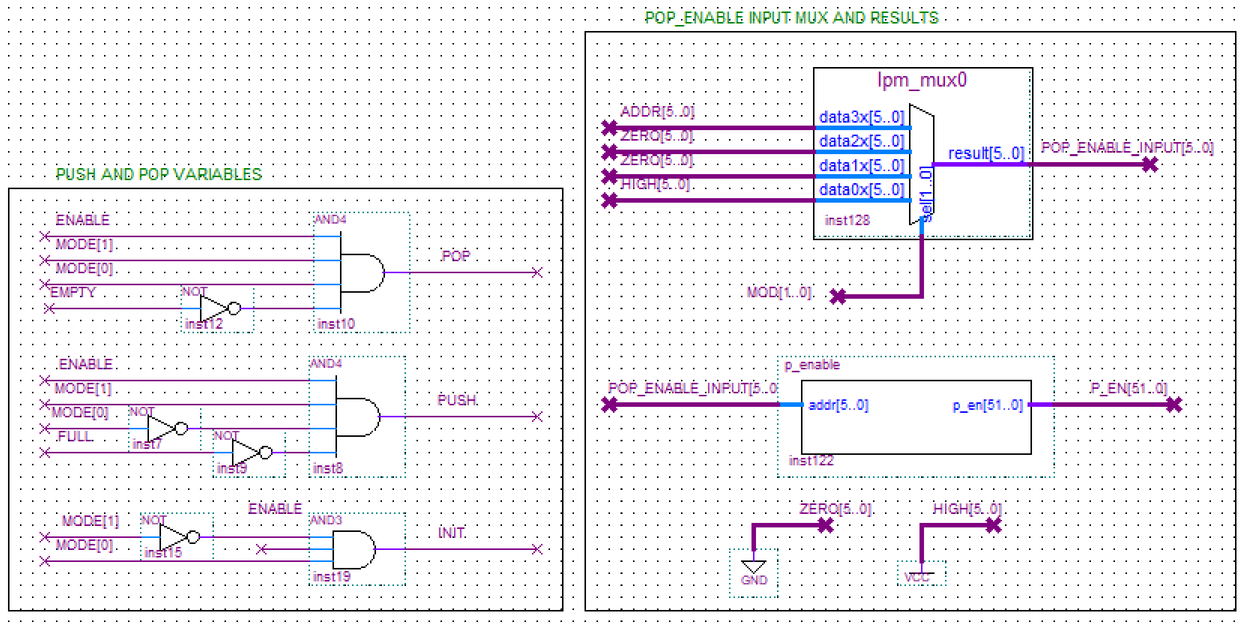


Figure 3 - Push, Pop, Init, and Pop_Enable Logic

Here is the Boolean logic for the Push, Pop, Init, and Pop_Enable variables. Important to take note that the pop_enable component takes as input a CLK signal and a POP_ENABLE_INPUT[5..0] 6-bit signal from the lpm_mux0 (multiplexer). The output of the lpm_mux is selected by a MOD[1..0] 2-bit signal which is explained in further detail in the next section.

A modified version of the Pop_Enable was used in this lab after noticing that the Lab #2's version [2] was taking 2 clock cycles which causes a problem since the ENABLE from the Test-Bed (explained in detail in the next sections) only goes high for a clock cycle. The stack would then not modify its values under any modes. The solution to this problem was to hardcode a map in VHDL just like the rom and create its circuit. For more information or details on the pop_enable component, please refer to Lab #2 Report [2].

MOD Logic

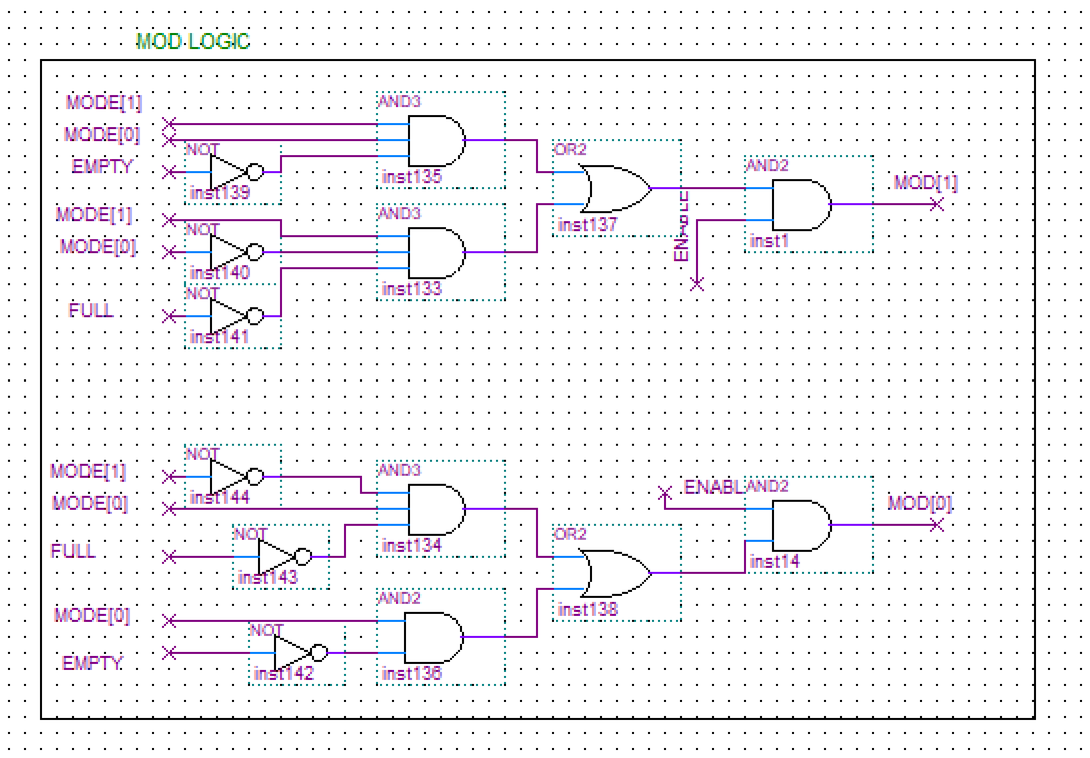


Figure 4 - MOD Logic, the Logic for lpm_mux0 Select

The MOD Logic was produced from a minimized PoS Boolean Function. The truth table is the following:

FULL	EMPTY	MODE[1]	MODE[0]	MOD[1]	MOD[0]
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	1	1

Group #27
Matthew Lesko 26069352; Romain Nith 260571471

K-maps:

MOD[1]:

FE\MODE	00	01	11	10
00	0	0	1	1
01	1	0	0	1
11	0	0	0	0
10	0	0	1	0

MOD[0]:

FE\MODE	00	01	11	10
00	0	1	1	0
01	0	1	0	0
11	0	0	0	0
10	0	1	1	0

From the K-maps, MOD has the following Boolean expression:

MOD[1] = MODE[1]*MODE[0]*notEMPTY + notMODE[1]*notMODE[0]*notFULL

MOD[0] = MODE[0]*notEMPTY + notMODE[1]*MODE[0]*notFULL

Up Down Counter

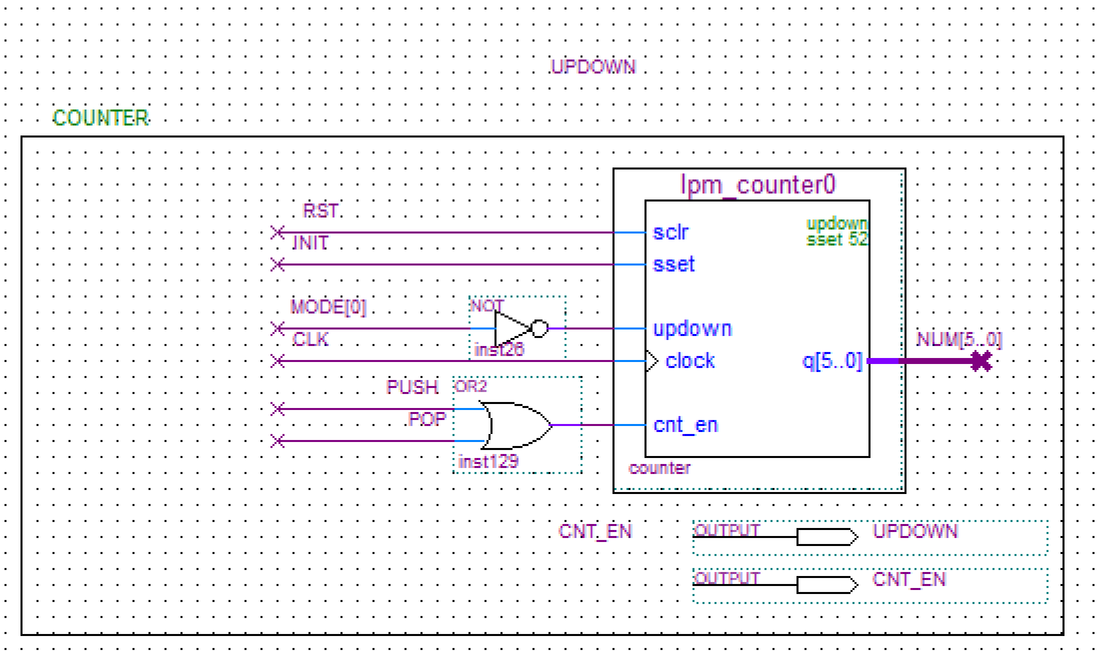


Figure 5 - Up Down Counter Block Diagram Design

The up down counter is enabled when the operation is either a pop or a push. The counter increments when the operation is a push (MODE[0] = 0) and decrements when the operation is a pop (MODE[0] = 1). The counter sets to a count of 52 when the operation is INIT, and clears to 0 if the RST signal is high.

Group #27
Matthew Lesko 26069352; Romain Nith 260571471

Empty and Full States

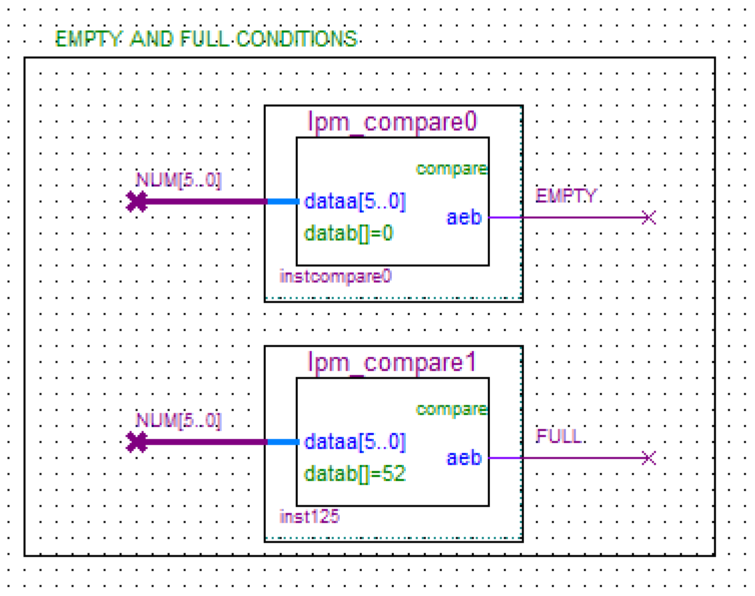


Figure 6 - Empty and Full Counters Block Diagram Design

When NUM[5..0] is compared to and equal to 52, the FULL signal is high.

When NUM[5..0] is compared to and equal to 0, the EMPTY signal is low.

Functional Simulation

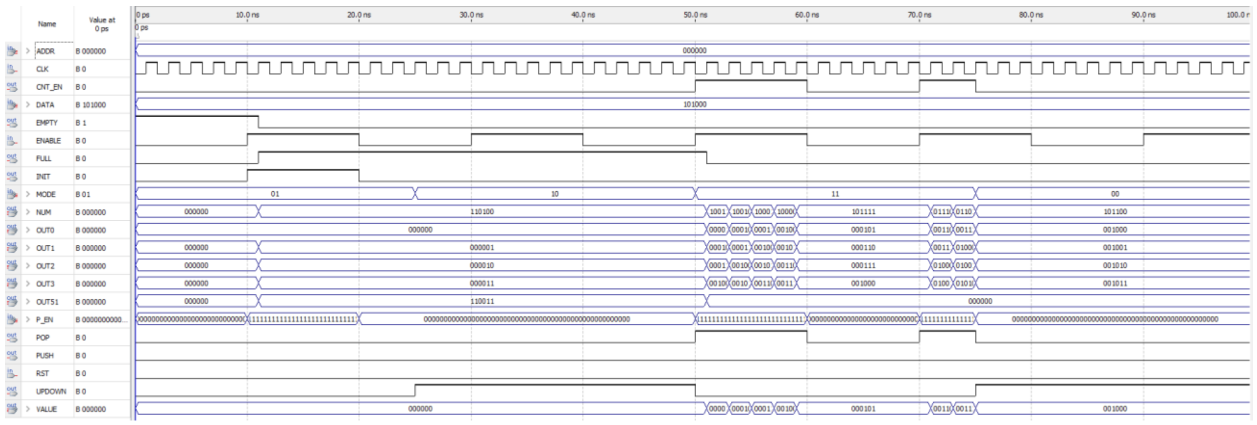


Figure 7 - Functional Simulation of Stack 52 Circuit

One can see that the functional simulation of the circuit is correct. When the operation is in INIT mode, the values of the first three flip flops are 0,1,2. The counter is set to 52. When the operation is in PUSH mode and is enabled, no element can be pushed since the circuit is full and the counter does not increment. When the operation is in POP mode and is enabled, the element at ADDR[5..0] is popped out of the circuit and the counter decrements and is hence 51.

Limitations and Advantages

Limitations of the circuit include:

- To increase space on the stack, flip flops must be copied and if changes are made to one flip flop, they must be changed for 51 others. Hence the maintenance of the circuit is non-ideal.
- Since the sequential circuit components are synchronous to the same clock signal and they several components depend on the output of others, there can be hazardous situations if one or several components experience significant delay

Advantages of the circuit include:

- The design and implementation of the stack is straight-forward and simple to implement
- Many components are already designed and implemented within the Altera LPM library, hence design and implementation design is decreased.

Test-Bed for 52 – Element Stack

Description of Circuit's Function

The Test-Bed for the 52 – Element stack is meant to implement and test the 52-stack with the goal to run it on the Altera FPGA. It records a user input with specified parameters such as the *MODE* or the *ADDR* and outputs the values in the stack pointed by the address.

Design and Implementation

The design of the Test-Bed is based off the diagram provided in the Lab 3 instructions[3].

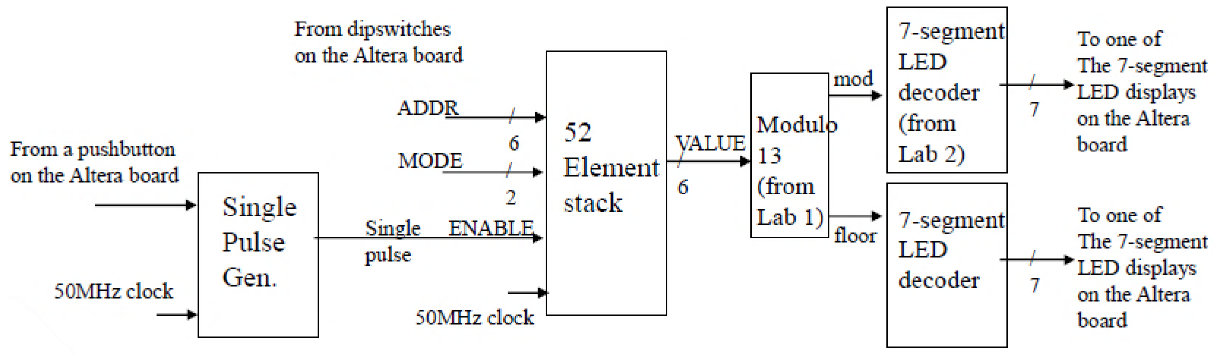

From dipswitches on the Altera board
ADDR
6
MODE
2
From a pushbutton on the Altera board
Single Pulse Gen.
50MHz clock
Single pulse
ENABLE
50MHz clock
52 Element stack
VALUE
6
Modulo 13 (from Lab 1)
mod
floor
7-segment LED decoder (from Lab 2)
7-segment LED decoder
7
7
To one of The 7-segment LED displays on the Altera board
To one of The 7-segment LED displays on the Altera board

Figure 8 - Test Bed Design

Inputs and Outputs

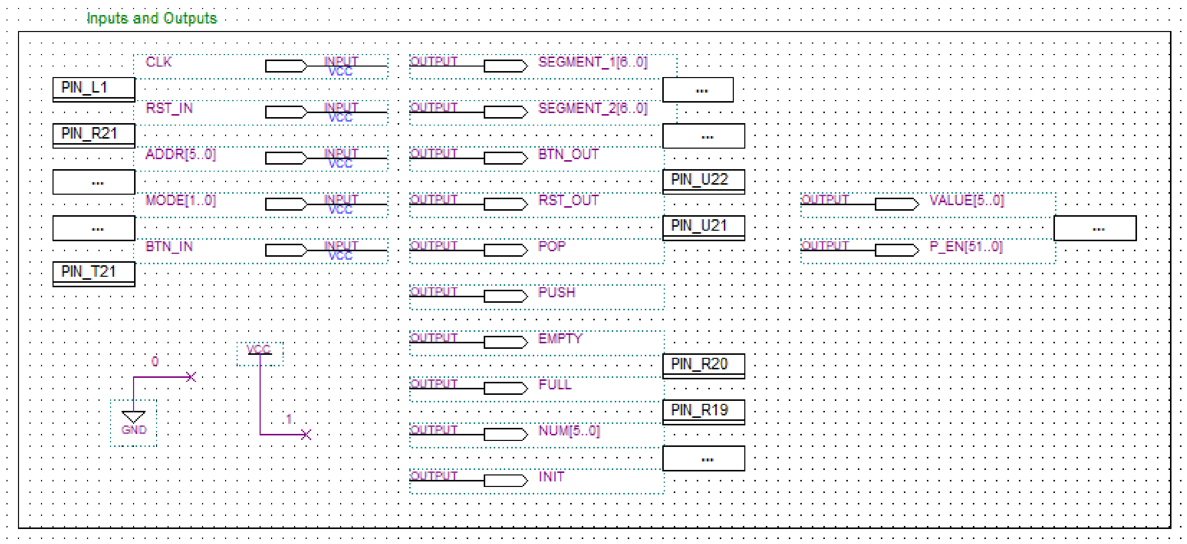


Figure 9 - Test Bed Input and Output Ports

The important I/O of the Test-Bed are the following:

- BTN_IN and RST_IN: both signals are physical switches triggered on the board. BTN_IN enables the stack whereas RST_IN rests it.
- MODE: controls the *MODE* of the stack.
- ADDR: tells which address the stack should look at.
- DATA: is the value pushed on top of the stack. For testing purposes, the value has been set to 1.
- SEGMENT_1 and SEGMENT_2: are the outputs for the two 7-Bits Segment Displays.

The other outputs are probes meant for monitoring the circuit in the simulations.

Single Pulse Generator

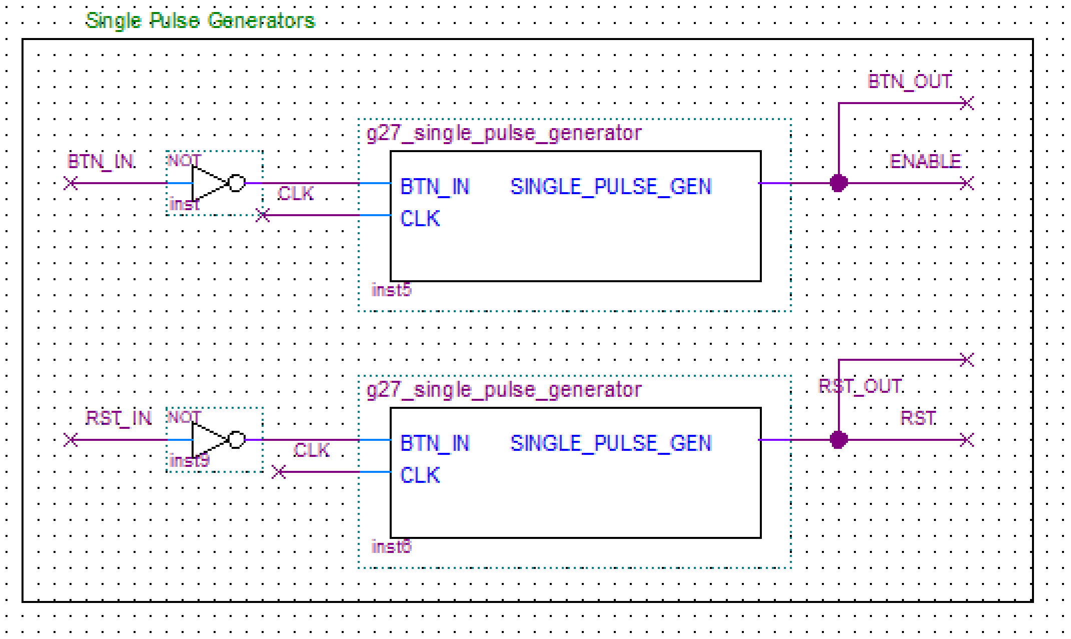


Figure 10 - Single Pulse Generator for RST and ENABLE

The stack's *ENABLE* and *RESET are* triggered by physical buttons on the board. Pressing the buttons once can trigger several voltage peaks due to mechanical properties. Therefore, the signals are being parsed to Single Pulse Generators which output a single pulse for any high input within 20'000'000 clock cycles.

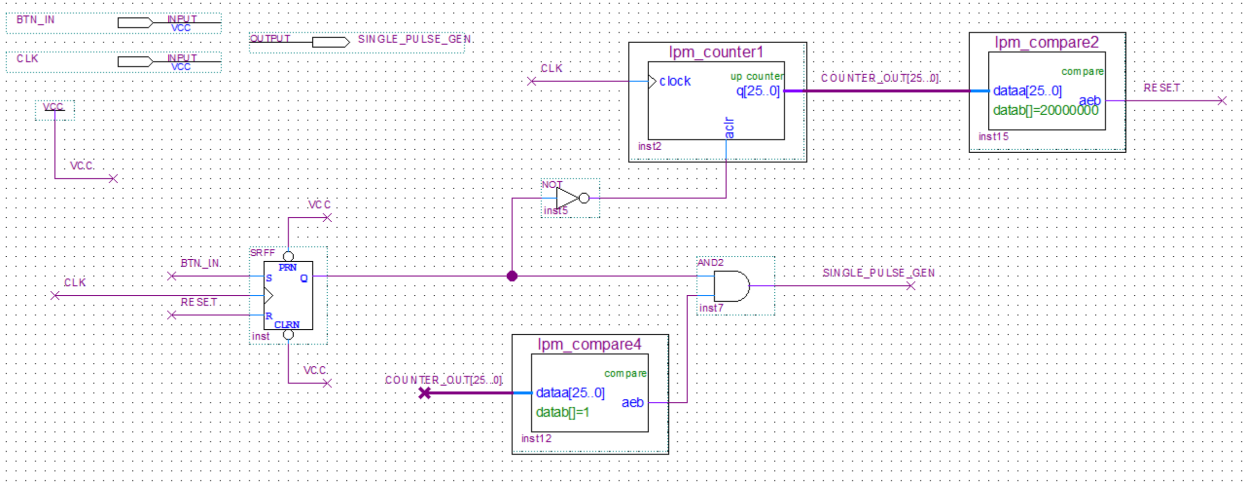


Figure 11 - Single Pulse Generator for RST and ENABLE

In the Generator, the input signal is fed to a SR FF as the signal. In parallel, a counter is counting to 20'000'000 clock cycles before resetting the SR FF. When the output of the counter is equal to 1, it is added to the FF's Q to output the Single Pulse.

52 Stack

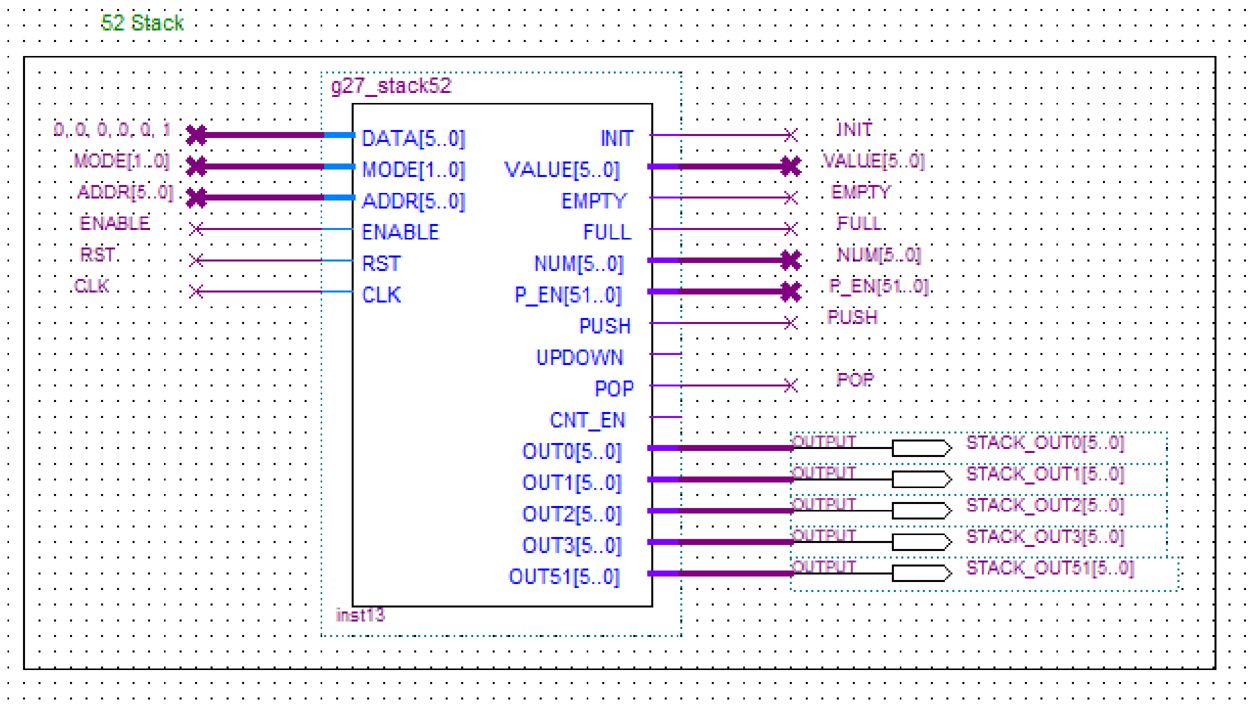


Figure 12 - 52-Stack Component in Test Bed

This is the 52-stack from the previous section but shown as a component in the test-bed. For Debugging purposes, there are outputs for the values of the stack elements: 0, 1 ,2 ,3 and 51.

Modulo 13 and 7 – Segment Decoder Displays

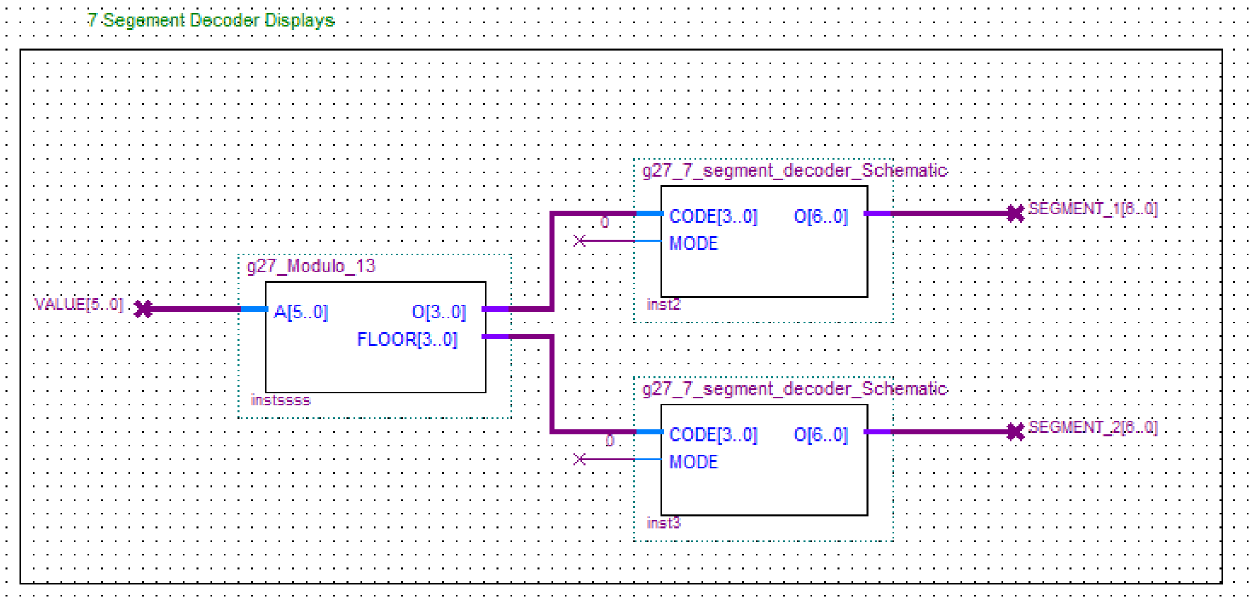


Figure 13 - Modulo and 7-Segment Decoder Components in Test Bed

Finally, the last components of the Test-Bed are circuits made in the previous labs where more detailed information can be found in Lab 1[1] and Lab 2 reports[2]. The 2 Decoders take the value outputted by the stack and display a base 13 value of it. Modes are set to 0 to display only numbers.

The *Modulo_13* in addition to the original design, outputs the floor of INPUT/13 which gets redirected to the second 7-Bit Decoder.

Timing Simulation

```
+--------------------------------------------------------+
; Slow Model Fmax Summary                                ;
+------------+-----------------+-------------+------+
; Fmax       ; Restricted Fmax ; Clock Name ; Note ;
+------------+-----------------+-------------+------+
; 69.76 MHz ; 69.76 MHz       ; clk        ;      ;
+------------+-----------------+-------------+------+
```

Figure 14 - Test Bed Slow Model Fmax Timing Analysis Summary

This exported Slow Model Fmax Summary describes the maximum clock speed, the higher it is the faster the circuit is. In this case it can be 69.76 MHz.

Functional Simulation

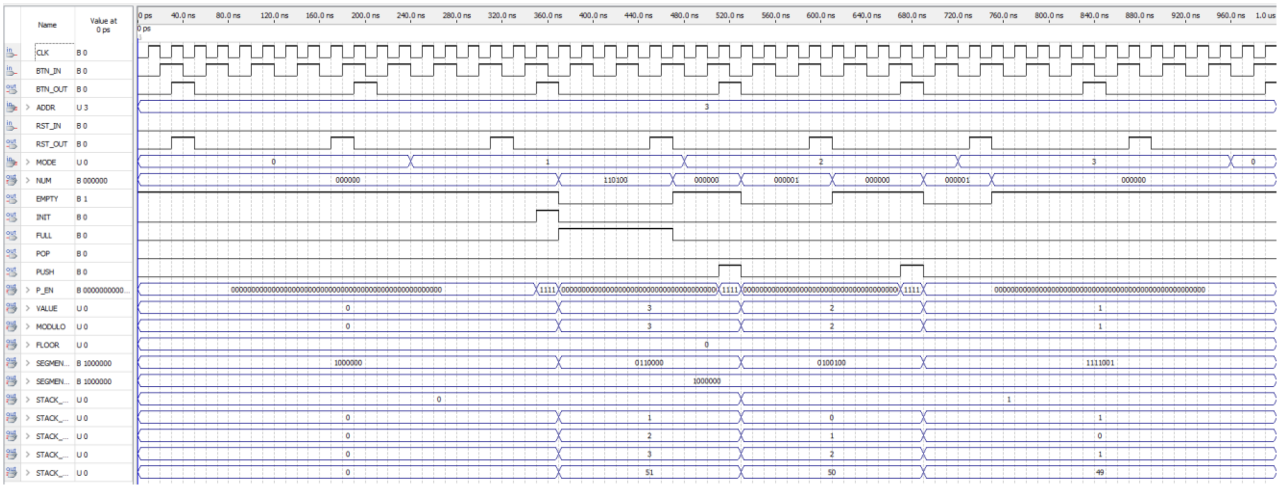


Figure 15 - Functional Simulation Results of Test Bed

In the *Figure 15* simulation, the Pulse Generators' counter are set so after a high *BTN_IN* only 1 pulse is outputted for 240ns so it is readable in the software. *MODES* are cycled every 240ns for the *ENABLE* to act on the stack.

At the falling edge of 360ns, the entire stack is initiated and the *P_EN* only parse the address for 1 clock.

Further down, when the *MODE* push is called at 680ns, the value 1 is properly pushed to the top of the stack.

Finally, the *POP* function extract the value at address 3 and the bottom rest of the stack gets push to the top to fill the empty stack.

Test-Bed on Altera Board

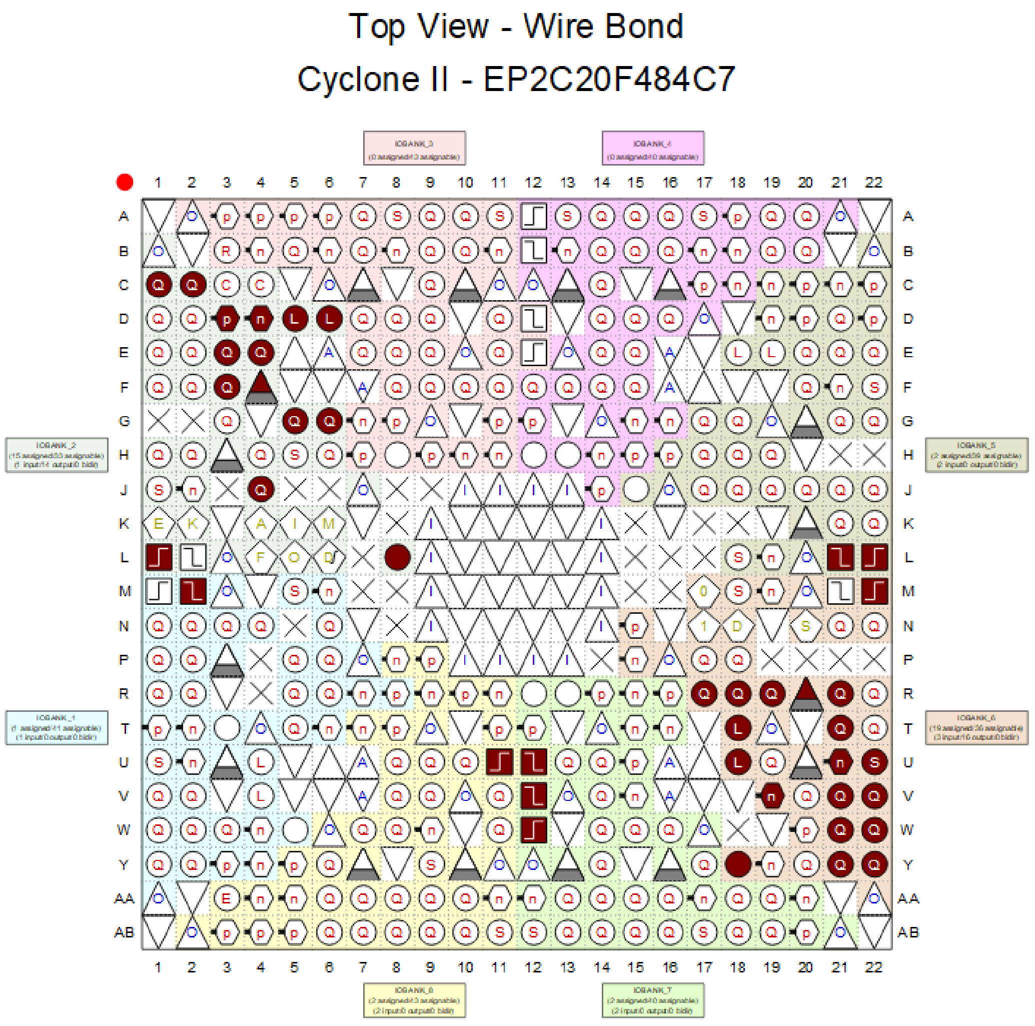


Figure 16 - Test Bed Pin FPGA Mapping

This figure includes the pin output/input mapping of the circuit to the FPGA board's pins.

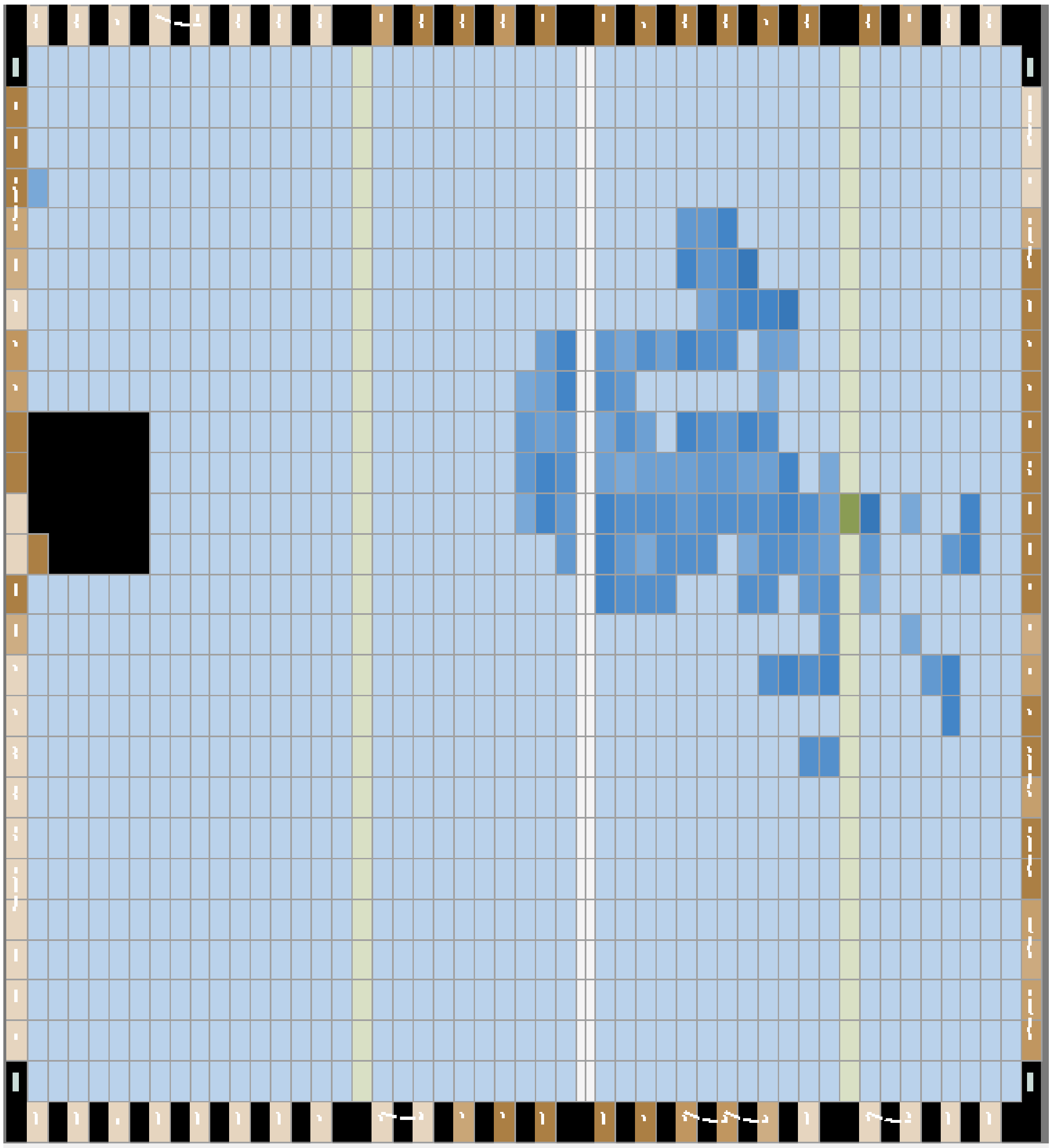
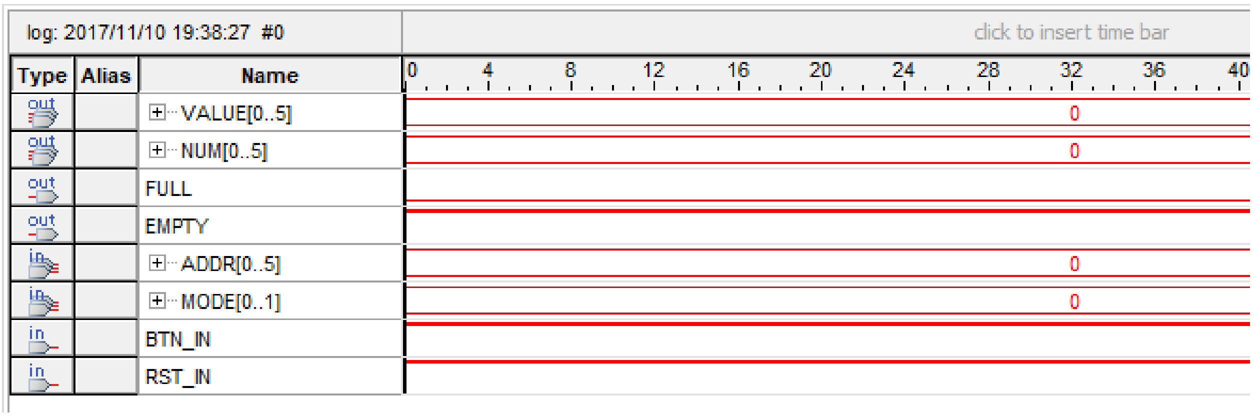


Figure 17 - Test Bed Circuit FPGA Mapping

This figure includes the test-bed mapping on the FPGA board.

On-Chip Testing with Signal Tap II Logic Analyzer

No Operation

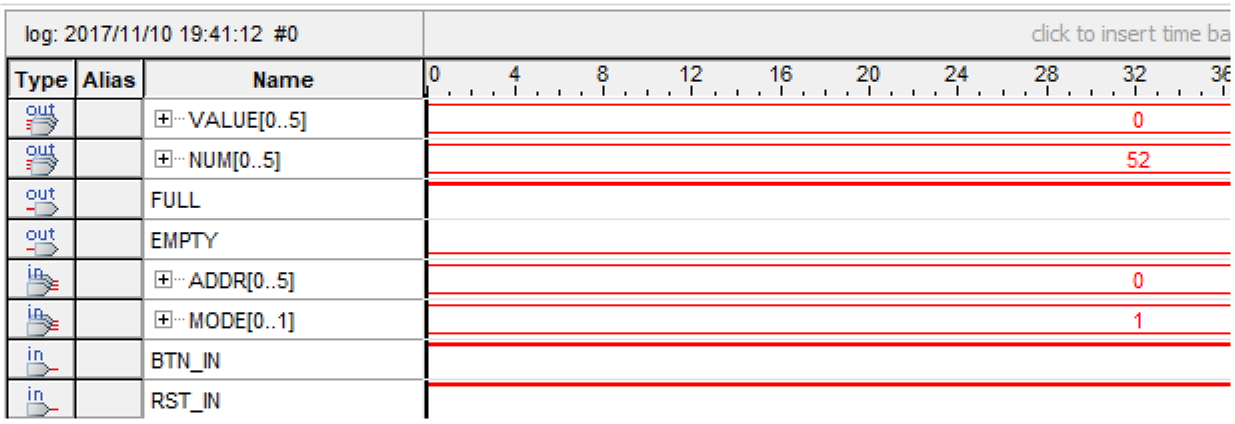

log: 2017/11/10 19:38:27 #0

Type	Alias	Name	Value
out		VALUE[0..5]	0
out		NUM[0..5]	0
out		FULL	
out		EMPTY	
in		ADDR[0..5]	0
in		MODE[0..1]	0
in		BTN_IN	
in		RST_IN	

Figure 18 - Signal Tap II FPGA Testing - No operation

One can see that when MODE[1..0] = 0, which is the no operation mode, nothing happens.

Initialize

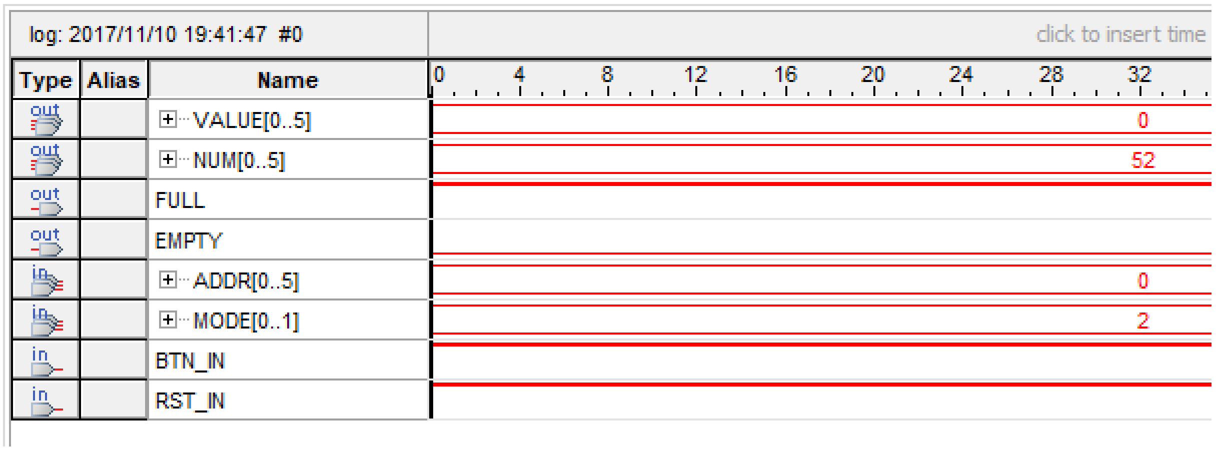

log: 2017/11/10 19:41:12 #0

Type	Alias	Name	Value
out		VALUE[0..5]	0
out		NUM[0..5]	52
out		FULL	
out		EMPTY	
in		ADDR[0..5]	0
in		MODE[0..1]	1
in		BTN_IN	
in		RST_IN	

Figure 19 - Signal Tap II FPGA Testing - Initialize Operation

One can see that when MODE[1..0] = 1, which is the initialize operation mode, the count is set to 52.

Push if Full



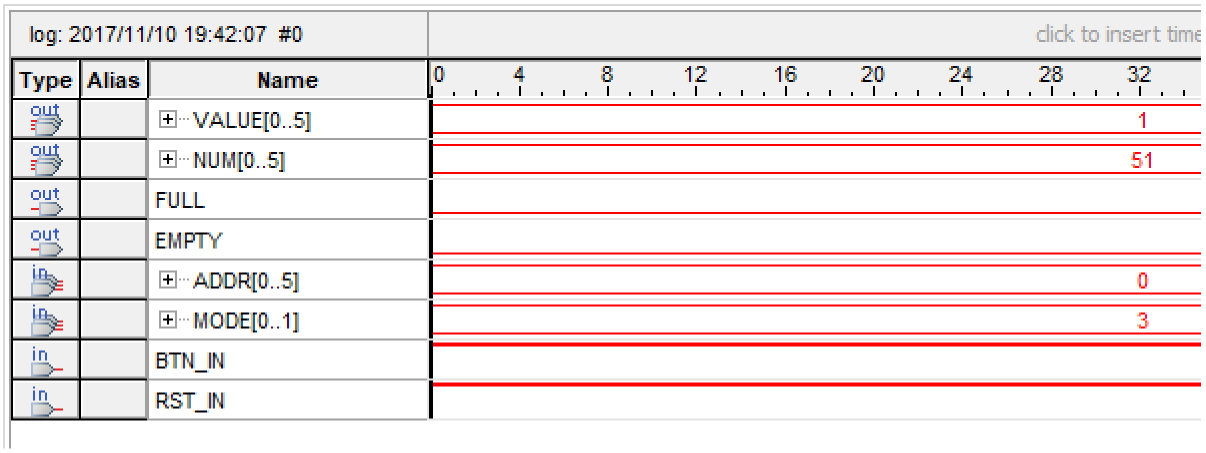
log: 2017/11/10 19:41:47 #0

Type	Alias	Name	Value
out		VALUE[0..5]	0
out		NUM[0..5]	52
out		FULL	
out		EMPTY	
in		ADDR[0..5]	0
in		MODE[0..1]	2
in		BTN_IN	
in		RST_IN	

Figure 20 - Signal Tap II FPGA Testing - Push Operation When Full

One can see that when MODE[1..0] = 2, which is the push operation mode, but the stack is full, then it is not possible to push an element onto the stack. The counter does not increment.

Pop Element 0 (Hard coded to Value of 1) if Full



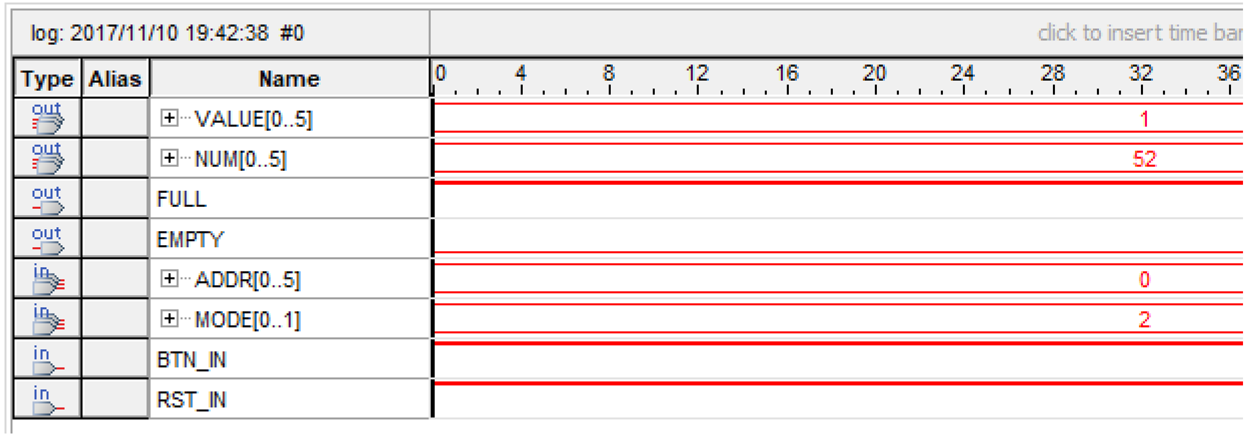
log: 2017/11/10 19:42:07 #0

Type	Alias	Name	Value
out		VALUE[0..5]	1
out		NUM[0..5]	51
out		FULL	
out		EMPTY	
in		ADDR[0..5]	0
in		MODE[0..1]	3
in		BTN_IN	
in		RST_IN	

Figure 21 - Signal Tap II FPGA Testing - Pop Element 0 when Full

One can see that when MODE[1..0] = 3, which is the pop operation mode, the count decrements by 1, and the value returned is that of element 0 since ADDR[5..0] = 0. VALUE[5..0] = 1 since it was hard coded to that value for debugging purposes.

Push Value of 1 on Element 0 if Not Full



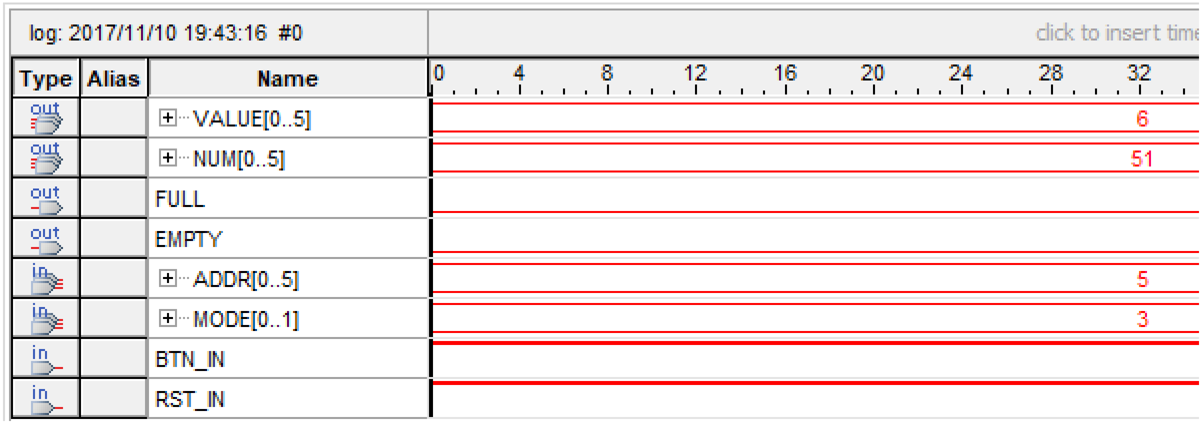
log: 2017/11/10 19:42:38 #0

Type	Alias	Name	Value
out		VALUE[0..5]	1
out		NUM[0..5]	52
out		FULL	
out		EMPTY	
in		ADDR[0..5]	0
in		MODE[0..1]	2
in		BTN_IN	
in		RST_IN	

Figure 22 - Signal Tap II FPGA Testing - Push Operation of Value 1

One can see that when MODE[1..0] = 2, which is the push operation mode, the count increments to 52, and the element is pushed onto the stack.

Pop Element 5



log: 2017/11/10 19:43:16 #0

Type	Alias	Name	Value
out		VALUE[0..5]	6
out		NUM[0..5]	51
out		FULL	
out		EMPTY	
in		ADDR[0..5]	5
in		MODE[0..1]	3
in		BTN_IN	
in		RST_IN	

Figure 23 - Signal Tap II FPGA Testing - Pop Operation of Element 5

One can see that when MODE[1..0] = 3, which is the pop operation mode, and ADDR[5..0] = 5, the value of the popped element is 6, since the stack has been previously pushed on so the value at element 5, which was previously 5 is now a 6. The counter decrements by 1.

Reset

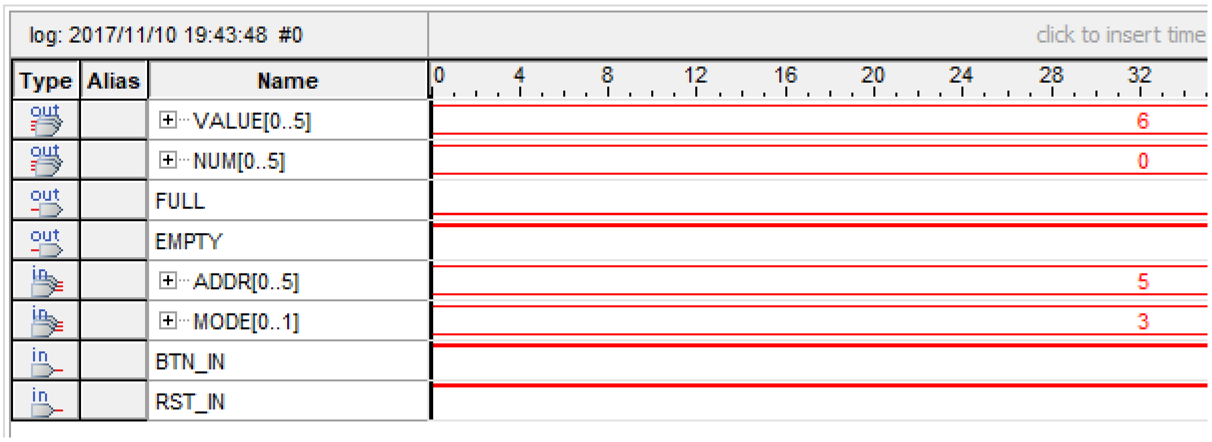


Figure 24 - Signal Tap II FPGA Testing - Reset Feature

One can see that no matter the value of ADDR[5..0] or MODE[1..0], when the RST button is enabled, the counter is set to 0 and all flip flop elements are 0.

Limitations and Advantages

The limitations of the test-bed include:

- A single pulse generator is needed for the enable and reset features.
- Clock signals can be tedious to deal when trying to do testing.

The advantages of the test-bed include:

- The design and implementation of the test-bed is straight-forward and simple to do.

References

1. Lab #1 Report. (2017). 1st ed. [ebook] Montreal: ECSE 323, pp 1-11. Available at https://mycourses2.mcgill.ca/d2l/lms/dropbox/user/folder_submit_files.d2l?db=71371&grpid=0&isprv=0&bp=0&ou=274765 [Accessed 10 Nov. 2017].
2. Lab #2 Report. (2017). 1st ed. [ebook] Montreal: ECSE 323, pp 1-14. Available at https://mycourses2.mcgill.ca/d2l/lms/dropbox/user/folder_user_view_feedback.d2l?db=72527&grpid=0&isprv=0&bp=0&ou=274765 [Accessed 10 Nov. 2017].
3. Lab_3_17F. (2017). 1st ed. [ebook] Montreal: ECSE 323, pp 1-61. Available at https://mycourses2.mcgill.ca/d2l/le/content/274765/viewContent/3504890/View [Accessed 10 Nov. 2017].