Gareth Peters (260678626)
Peter Quinn (260689207)

November 10, 2017
ECSE 323

Lab 3: stack52, debouncer, testbed

Gareth Peters (260678626)
Peter Quinn (260689207)

November 10, 2017
ECSE 323

g07_stack52

Description

This circuit was designed to implementation a stack of size 52 where each memory location of the stack holds 6 bits of data. The stack will be used to imitate a card deck or a card hand. Card values are represented using an integer in the range 0 to 51. The stack can be initialized to have one of every card, simulating a full deck of cards. There are 4 possible operations for the stack circuit NOOP, PUSH, POP and INIT (see Table 1 for descriptions). There is also an asynchronous reset signal that resets the value of each stack element to 0. There are 6 inputs and 4 outputs for this circuit.

Inputs and Outputs

MODE is a 2 bit control input that determines the action to be taken on the rising edge of the clock. These 4 control modes are NOOP (00), PUSH (01), POP (10) and INIT(11). The description of each operation is per Table 1.

00	NOOP	● Nothing occurs
01	PUSH	● Puts the DATA input on top of the stack ● Pushes all data that was already in the stack down one slot ○ Data on the bottom of the stack is lost ● Nothing happens if FULL = 1 (NUM = 52) ● Increments NUM if NUM!=52
10	POP	● Removes the data at the stack location corresponding to the ADDR input. ● Pushes all data below this location up one using g07_pop_enable ● Decrements NUM if NUM!= 0 ● Does nothing if EMPTY = 1
11	INIT	● Initiates each memory location to be its corresponding ADDR value (location 0 is set to 0, location 1 is set to 1, etc) ● NUM is set to 52

Table 1: Modes of operation for g07_stack52

DATA is 6 bits wide and contains the data that is to be pushed onto the stack.
ADDR is a 6 bit wide address input that sets the location to be viewed as well as the location to perform the POP operation.

ENABLE is a 1 bit control input that enables the MODE operation. If ENABLE is low then NOOP occurs, otherwise MODE occurs.
RESET is a 1 bit control input that resets all values in the stack to 0, and sets NUM to be 0.
CLK is a 1 bit clock input.

VALUE is a 6 bit output that outputs the value pointed to by the ADDR input.
NUM is a 6 bit output that indicates the number of elements in the stack.
EMPTY is a 1 bit output that is set to high when there are no elements in the stack (i.e. NUM=0).
FULL is a 1 bit output that is set to high when there are 52 elements in the stack (i.e. NUM=52).

Opcode Section

In order to obtain the several signals required to perform the MODE operations, a set of operation codes were designed. Each bit of the Opcode is as in Table 2.

Bit	Description
5 downto 0	● Represents the address that is to be viewed from the stack ● Set to 1's for NOOP and INIT so as not to enable the LPM_FFs. ● Set to all 0's for PUSH since every LPM_FF must shift their data down 1.
6	● Is the select bit for the lpm_counter_stack and BUSMUX circuit ● Shifts all data in the stack down if 1, and shifts data up if 0 ● Is set to 1 when NUM is supposed to increase, and is otherwise 0
7	● A control bit that enables the counter. ● Is set to 1 for POP and PUSH only
8	● The set bit for the LPM_FF circuits. ● Is set to 1 only for INIT

Table 2: The bit representations in the Opcode

Four lpm_constant circuits were used to set the initialize the Opcodes for each of the modes. lpm_constant_NOOP, lpm_constant_PUSH and lpm_constant_INIT are initialized with 9 bits, while lpm_constant_POP is initialized with 3 bits. The output of the POP constant is concatenated with the ADDR input.

NOOP and PUSH are fed into a 2 input multiplexer with select bit the is the FULL output bit. If FULL is 1, then NOOP occurs. NOOP and POP are fed into a second 2 input multiplexer with EMPTY as the select bit, similarly to the first multiplexer. The output of these two multiplexers, as well as the INIT and NOOP constants are fed into a 4 input multiplexer with MODE as the

select bit. The output of the 4 input multiplexer is as in Table 1. Lastly, the output of the 4 input multiplexer and NOOP constant are fed into a 2 input multiplexer with ENABLE as the select bit.

The outputs of bits 6, 7 and 8 are each fed into a D flip-flop to compensate for the fact that there is an extra clock cycle due to the g07_pop_enable circuit. The D flip flops ensure that all signals reach the stack at the appropriate (i.e. same) time, so the operation is executed correctly.

Stack Circuit

The actual circuitry of the stack uses the LPM_FF and BUSMUX circuits. Each LPM_FF holds 6 bits of data. For each LPM_FF, the output is fed to the BUSMUX of the next and previous LPM_FF. The select bit of each BUSMUX is the 6th bit from the Opcode and is set based on the description per Table #2. The RESET input is given as input to each of the aclr inputs for the LPM_FFs. The CLK input is given as input for the clock of the LPM_FFs.

This circuit makes use of g07_pop_enable [1], a component designed and tested in lab 2 (see lab 2 report). The address obtained from the Opcode circuit is given as input to the pop enable circuit. The 52 output bits are given as an input to their corresponding LPM_FF enable input.

Two lpm_compare circuits were used; one for EMPTY and one for FULL. The first lpm_compare compares NUM with 0 and outputs EMPTY = 1 if NUM = 0. Similarly for the second lpm_compare, but instead NUM is compared with 52 and outputs FULL = 1 if NUM = 52.

Finally, each LPM_FF output is given as input to a lpm_mux_valueSelect circuit that takes the 52 outputs of each LPM_FF and selects which the correct output based on the ADDR input. The output of this multiplexer is the VALUE output of the g07_stack52 circuit.

Schematic

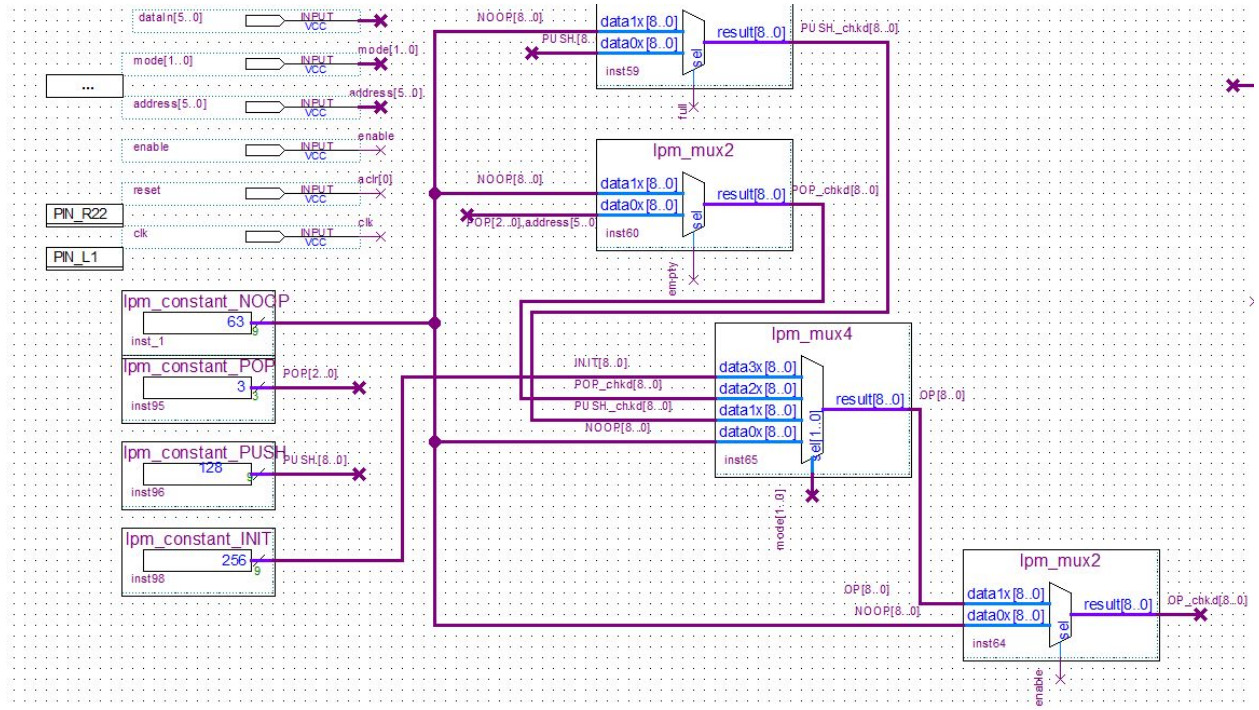


Fig. 1: Schematic diagram for the Opcode section of the circuit

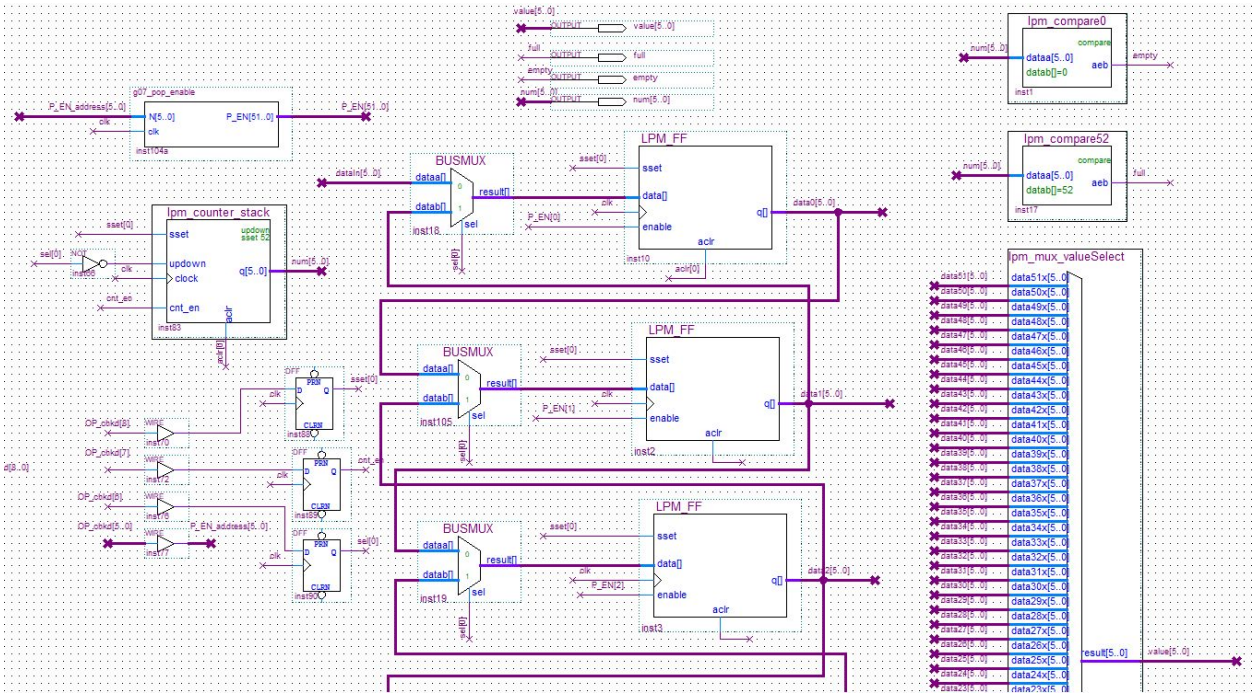


Fig. 2: Schematic diagram for the Stack section of the circuit. The circuit continues below the figure with 49 other LPM_FFs.

Gareth Peters (260678626)
Peter Quinn (260689207)

November 10, 2017
ECSE 323

Testing and Simulation

The circuit was simulated using a timing simulation in Quartus with a 50MHz clock input. Each mode was tested individually. Various inputs and outputs were tested while each mode was tested.

INIT was tested by setting mode to 11 and then switching the ENABLE signal to 1 once at the beginning of the test. The address input was incremented to have the values at each flip flop be shown. EMPTY and FULL outputs were a byproduct to this test, as the circuit first starts off empty and then goes to full when the stack is initiated. DATA does not matter in this test since INIT does not depend on the data being inputted. RESET was also never used in this test. The timing simulation for INIT can be viewed in Fig. 3.

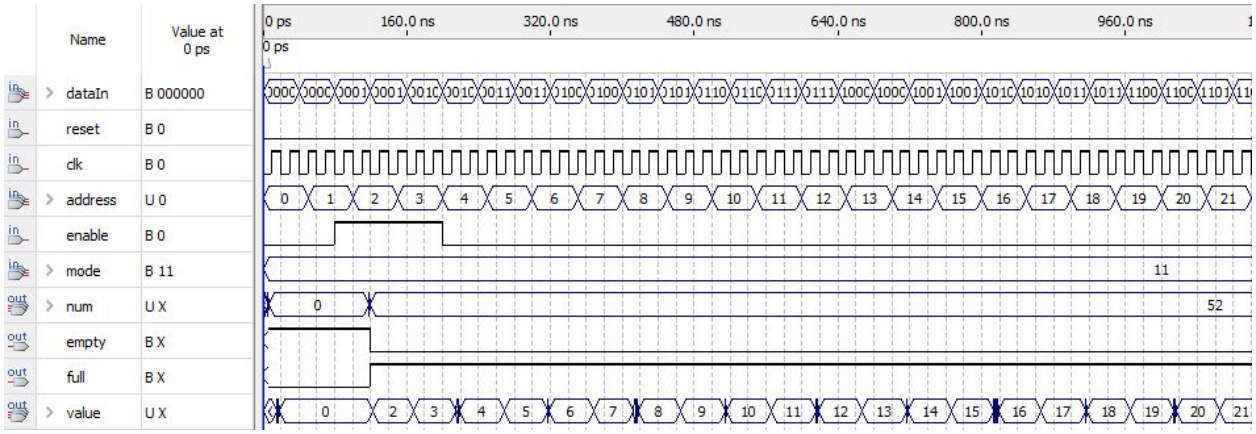


Fig. 3: Timing Simulation testing the INIT mode

PUSH was tested by first initiating the stack, and then switching the mode to PUSH to first test to see if anything can be pushed onto the stack while it is full. Then the stack was reset using the RESET input in order to properly test to see if PUSH was pushing data onto the stack. DATA was selected arbitrarily to be 42. DATA is switched at one point to be 43 during the test to see if the value viewed at the address specified is eventually 43 after continuously pushing items. ADDR is kept at a constant 5 arbitrarily. It can also be noted that NUM increments every time an item is pushed onto the stack. The timing simulation for PUSH can be viewed in Fig. 4.

POP was tested by first initiating the stack, and then setting mode to POP while the address value stayed at a constant. This was to observe the VALUE output change at that specific address. The address was selected to be 0 so as to also test popping on an empty stack. RESET is set to 1 at the end of the test to observe that POP is impossible when the stack is empty. Testing for NOOP mode was snuck into this simulation to quickly observe the that mode does not change any of the outputs. Notice that NUM decrements every time POP occurs. The timing simulation for POP can be viewed in Fig. 5.

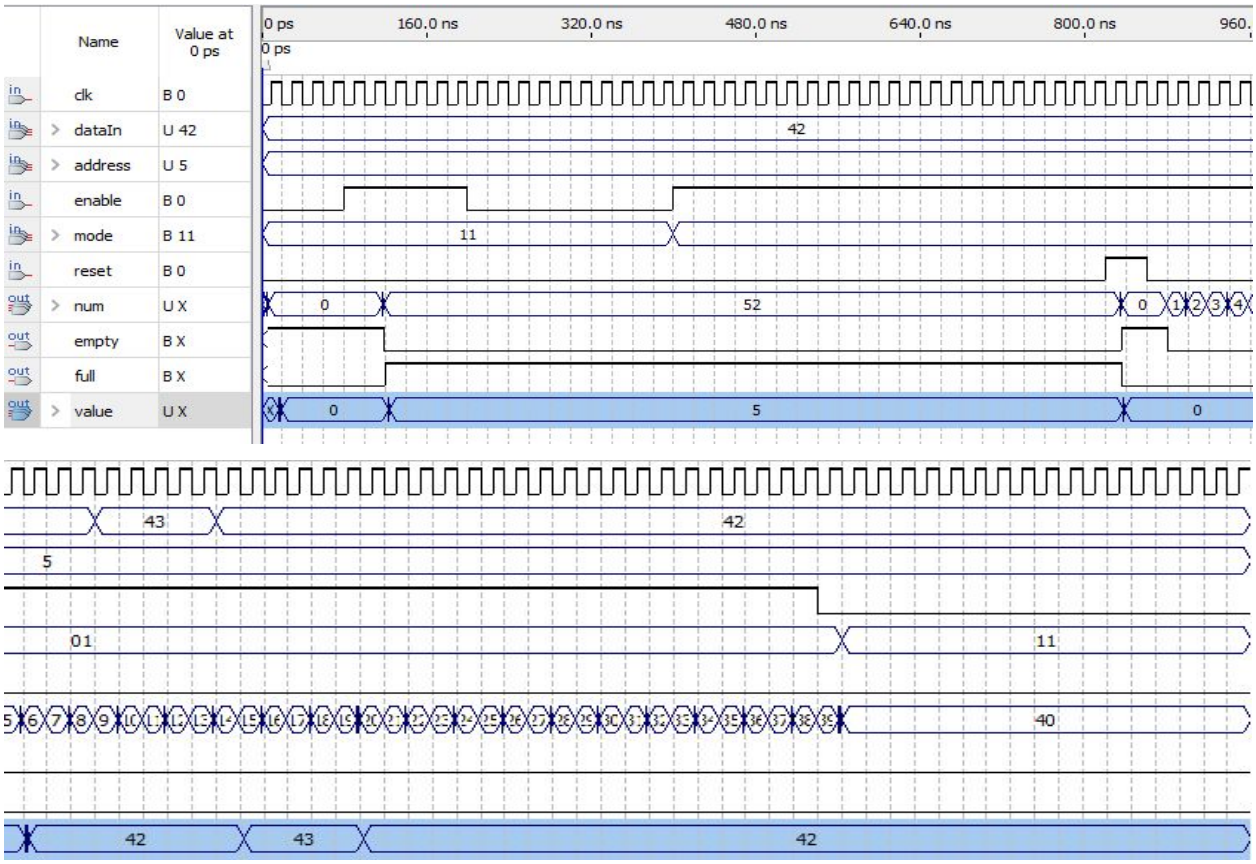
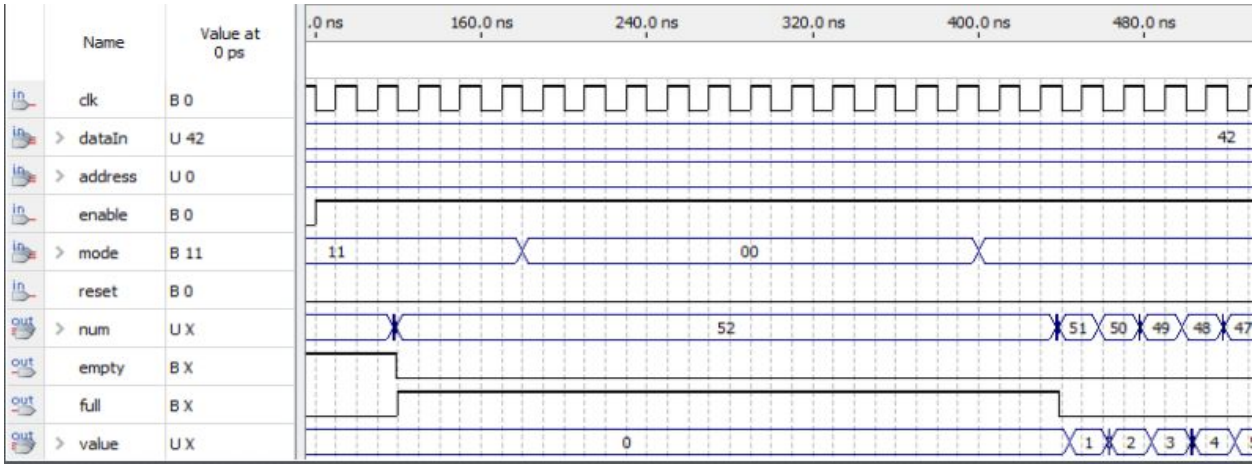


Fig. 4: Timing Simulation testing the PUSH mode



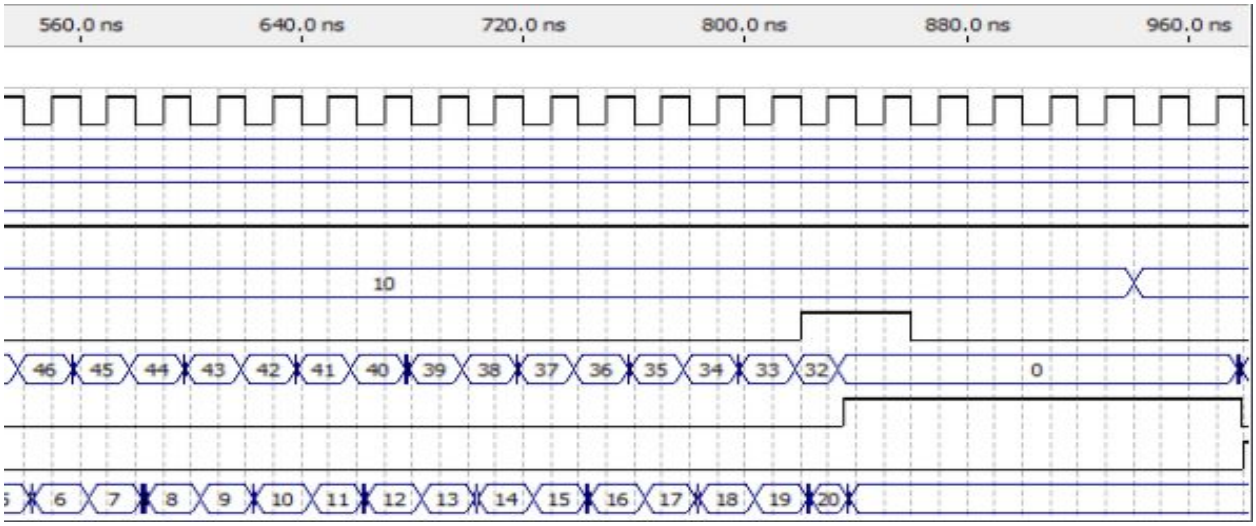


Fig. 5: Timing Simulation testing the POP and NOOP mode

Advantages and Limitations

This circuit faces some limitations in terms of timing. The circuit requires two clock cycles to process the inputs given and provide outputs. This then causes greater delay in the output of VALUE. This is caused by the addition of the g07_pop_enable circuit and was needed to be compensated for by adding flip flops in the middle of the circuit. This can be fixed creating a different circuit that performs a similar function to the pop enable that does not require an extra clock cycle. A costly way to do this would be to use logic gates and decoders as a replacement for the pop enable circuit .

The advantage to the circuit is that it is relatively straight forward and simple to implement. This method is also relatively cost efficient when compared to other methods of implementation. An example of such an alternative implementation would be a different type of linear structure, such as a linked list or queue. The Opcode section of the circuit is simple to implement as well as comprehend compared to any other method, such as using logic gates to obtain the mode of operation and actions to be performed while in such a mode.

Gareth Peters (260678626)
Peter Quinn (260689207)

November 10, 2017
ECSE 323

g07_debouncer

Description

This circuit is designed to monitor input coming from a switch. When the switch is closed, this circuit produces a single pulse that is synchronized with the clock and is high for one complete clock cycle. After this single pulse is generated, any input from the switch being monitored is ignored for certain length of time. This circuit is also known as a single pulse generator. It is used to create an enable signal to be used for other components that is one clock cycle long, so as to ensure that only a single operation is performed.

This circuit intended to deal with “switch bounce”. Switch bounce occurs when a switch is being closed and it does not cleanly go from open to closed. The transition can be noisy and signal produced by the switch can go from low to high several times before it settles. This noise is due to the mechanical nature of the switch. In order for the circuit to operate properly, a delay of 10 million clock cycles was chosen. This value was determined through testing with the FPGA board. With a 50 MHz clock, this amounts to 200 ms of ignore period for the button.

There are two 1 bit inputs into this circuit. One is the synchronous clock signal, clk. The other is the signal directly from the push button switch, btnPshd.

The output of the circuit sig is a single wire/bit that carries the single pulse.

This circuit makes use of a SR-flip flop, a modulo-counter and a comparator to achieve its function. The counter and the comparator were implemented using the Altera LPM functions

1. The signal from the push button switch is inverted and fed into the S port of the SR flip flop. The raw signal from the push button switch is fed to the R port.
    a. This has the effect of setting the the value of the SR flip flop to 1 when the button is pressed (i.e. value of btnPshd goes low, because this switch is active low).
    b. While the button is not pressed the value of btnPshd stays high, and the SR flip flop is set to 0.
    c. The SR flip is rising edge triggered by the common clock.

2. The output of the SR flip flop is sent to the output sig

3. The counter and the comparator are responsible for blocking the input during the input ignore period of the operation
    a. When the output of the flip flop goes to 1, the counter is set to 1 using sset
    b. While the value of the counter is non zero, it is enabled to count up by inverting the signal coming from the comparator that compare the value of the counter with 0

i. The rate of the counting is determined by the clock speed

c. When the counter reaches its limit and loops back around to zero, the counter is disabled by the comparator
   i. The counter stays at 0 until the SR flip flop passes a high signal to sset, to move the counter off of 0

d. While the counter is counting up (i.e. its value is non zero) a signal is passed from the comparator to the CLRN port of the SR flipflop that clears its output (sets it to 0)
   i. The continual clearing of the flipflop holds its output low regardless of what S and R are.
   ii. The delay between the SR flip flop being set to high by the btnPshd signal and then the value being cleared by the comparator is exactly one clock cycle
   iii. Once the counter reaches zero again, the clear signal is unasserted, and the SR values are monitored again

Schematic

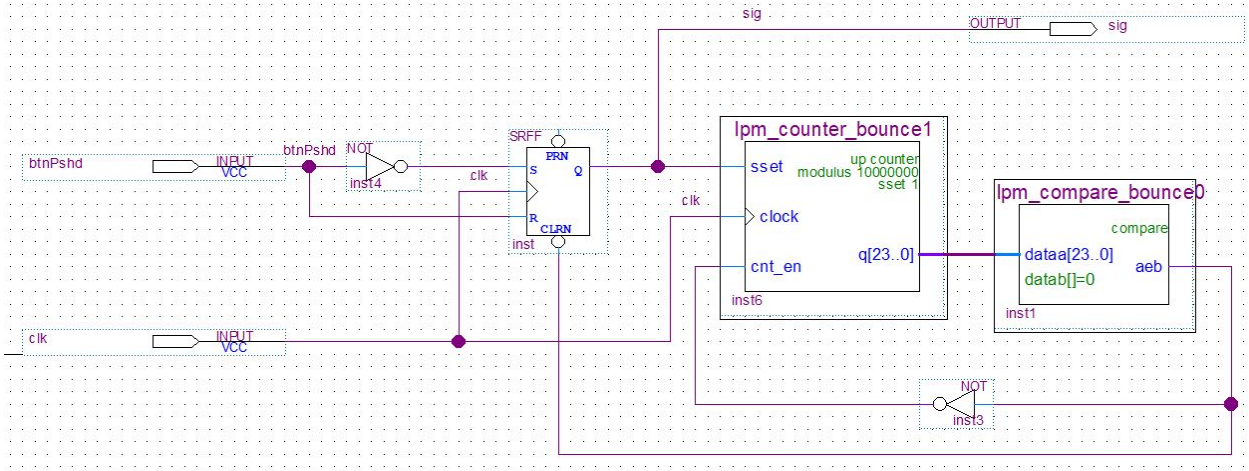


Fig. 6: Gate level schematic of g07_debouncer

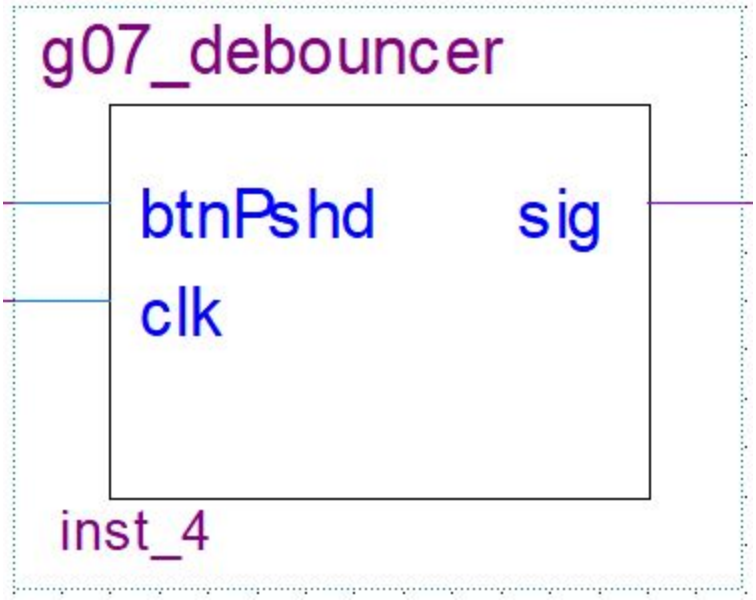


Fig. 7: Symbol of g07_debouncer

Testing and Simulation

For the testing of this circuit, a clock signal as well as the btnPshd signal were provided as inputs. Note that the btnPshd signal start high and then goes to low for a period of time before returning to high. This is because the buttons on the FPGA are active low, so this waveform simulates the pushing and the release of the button. Some simulated noise is also included after the initial low portion is also included in the testing waveform. If the length of the simulation is extended longer than the ignore period of the debouncer, it can be observed that the circuit resets and has the same behaviour as when it is first initialized. This can also be observed by making the duration of the of the ignore period very short for simulation purposes.

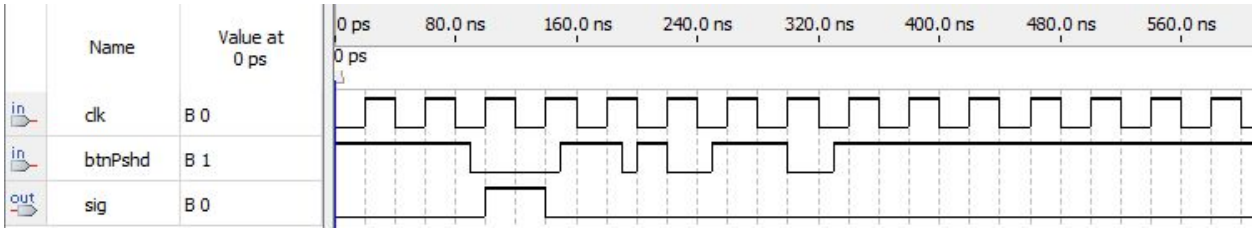


Fig. 8: Functional simulation of g07_debouncer. It can be observed that the high period of the output sig corresponds exactly to one clock cycle.

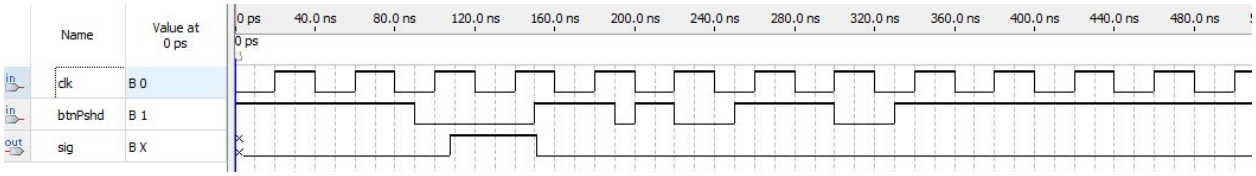


Fig. 9: Timing simulation of g07_deouncer. The high portion of the output sig does not correspond exactly to the rising and falling of the clock signal due to timing delays. However, the pulse is still one clock cycle in duration.

Advantages and Limitations

This circuit is quite vulnerable to mechanical noise. Any closing of the switch will produce a pulse. This pulse could be produced in error if the board or the switch is bumped in such a way that causes contact to be made inside in the switch, without the switch being truly depressed. This could be improved by introducing a circuit that require the switch to be depressed for a certain length of time before the single pulse is generated. This circuit also limits the maximum rate of input, and therefore the frequency of operations that can be done.

The advantage of this circuit is that it is very simple and does not require many resources, while reliably producing the correct output. Through testing, it was determined that the potential vulnerability to noise did not affect the circuit to a significant degree.

Gareth Peters (260678626)
Peter Quinn (260689207)

November 10, 2017
ECSE 323

g07_lab3_testbed

Description

This is a complete circuit that is intended to be uploaded to the FPGA. The FPGA used was a Cyclone II EP2C20F484C7. The objective is to test on actual hardware the the functionality of the circuits that have been designed in this and previous labs. This circuit incorporates several different elements. The primary block is g07_stack52 (see above). Other blocks include g07_Modulo_13 (see lab 1) and g07_7_segment_decoder (see lab 2). Two units of g07_debouncer (see above) are also used.

This circuit monitors two push button switches using two units of g07_debouncer. One of the buttons triggers the reset function of a unit of g07_stack52. The other button triggers the stack operation specified by the mode input of the stack. The 2 bit mode input is selected using two of the dip switches on the FPGA. The address input of the stack is selected using six dip switches on the board. The value of the card at the selected address is displayed using two seven segment LED displays. One LED display corresponds to the mod13 of the value (the card number). The other LED displays the floor13 of the value (the “suit” of the card). The value for mod13 and floor13 are computed using g07_Modulo_13. Amod13 and floor13 are converted from binary numbers to the display codes for the 7 segment displays using g07_7_segment_decoder. The mode port of the units of g07_7_segment_decoder were left unattached, so the outputs on the LED will be hexadecimal values (rather than card glyphs). A constant value of 42 (created using LPM_constant [3]) is attached to the dataIn input of the stack. This value is pushed onto the stack when the push operation of the stack is triggered. A constant value was chosen because there were not enough physical input devices on the board to select a range of values. The number 42 was chosen arbitrarily. All the synchronous (clock dependent) modules are controlled using the same clock input, which was chosen to be the 50 MHz clock on the FPGA.

The various inputs and outputs of this circuit are given in the tables below. All inputs and outputs are mapped to physical pins on the FPGA (see “Appendix 1: Pin Assignments”) unless otherwise noted. See also “Appendix 2: Input/Output Mapping on the Board” for details regarding the assignment of inputs and outputs to displays/buttons/LEDs/switches on the board.

Inputs

Name	Use
reset	● The signal from the reset button ● Mapped to a push button switch
button	● The signal from the button that triggers the operation specified by mode[1..0] ● Mapped to a push button switch
clk	● The synchronous clock for the system ● Mapped to the 50 MHz clock on the FPGA
adr[5..0]	● Used to set address that is used for stack operations and viewing values at locations the stack ● Each bit is mapped to a dip switch
mode[1..0]	● Used to set the stack operation to be performed ● Each bit is mapped to a dip switch

Table 3: The inputs, their function and mapping details for g07_lab3_testbed

Outputs

Name	Use
LED1[6..0]	● Contains the display code for a 7 segment display that corresponds to the value of floor13 (the suit of the card) ● Each bit mapped to the corresponding segment of a 7 segment display
LED2[6..0]	● Contains the display code for a 7 segment display that corresponds to the value of Amod13 (the value of the card) ● Each bit mapped to the corresponding segment of a 7 segment display
LED3[6..0]	● Held high (off) ● Keeps one of the displays permanently off
LED4[6..0]	● Held high (off) ● Keeps one of the displays permanently off
rled	● High (on) when the stack is empty ● Mapped to a red LED

gled	• High (on) when the stack is full • Mapped to a green LED
activeLED	• High (on) when a stack operation is triggered • Mapped to a green LED • Note: when being used on the board, the turning off and on of the LED happens so fast it is not noticeable
value_test[5..0]	• The value (card) the address is pointing to in the stack • Used only for testing in simulations, not mapped to any pins
num_test[5..0]	• The value of num (the number of cards in the stack) • Used only for testing in simulations, not mapped to any pins

Table 4: The outputs, their function and mapping details for g07_lab3_testbed

Schematic

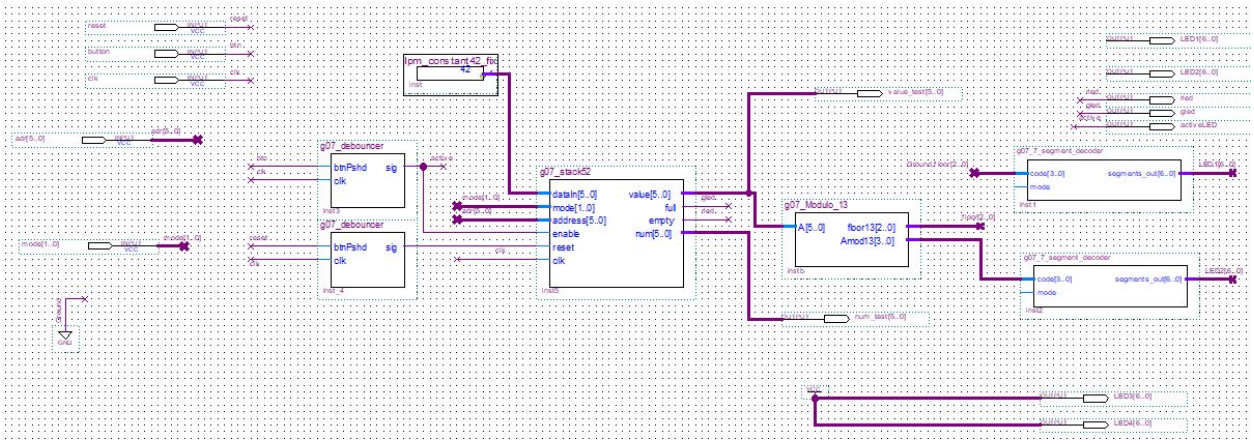


Fig. 10: Schematic of g07_lab3_testbed

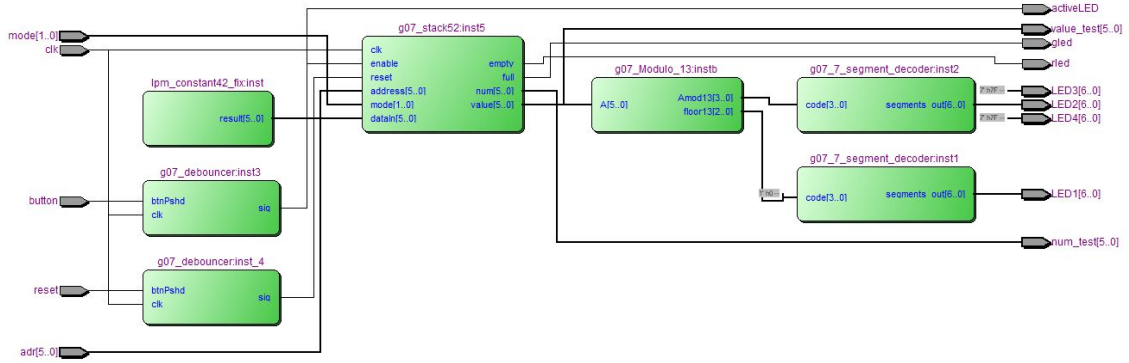


Fig. 11: Block diagram for g07_lab3_testbed

Testing and Simulation

The various operations of the testbed circuit were simulated. The input mode was set to determine the stack operation to execute (see the section on g07_stack52 for the codes for mode). The trigger for the operation is the input “button” going low since the buttons are active low on the FPGA.

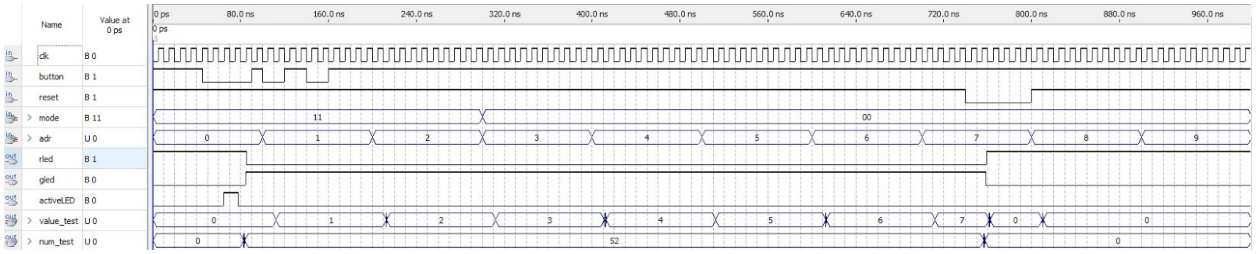


Fig. 12: Timing simulation for g07_lab3_testbed initialization operation (mode=11) and reset. rled (empty) goes low and gled (full) goes high immediately following the one clock pulse long enable signal (activeLED) which is generated by the debouncer circuit connected to button. The reset signal clears the stack, so rled (empty) goes high and all the values go to 0. It can also be observed that noise on the button input is ignored for a period of time after the initial triggering.

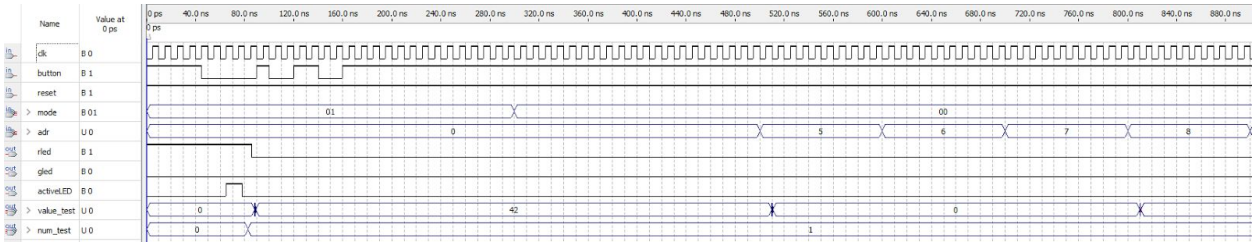


Fig. 13: Timing simulation for g07_lab3_testbed push operation (mode=01). rled (empty) goes low after the value 42 is pushed onto the top of the stack and num goes to 1. Address 0 is 42 while other addresses remain 0

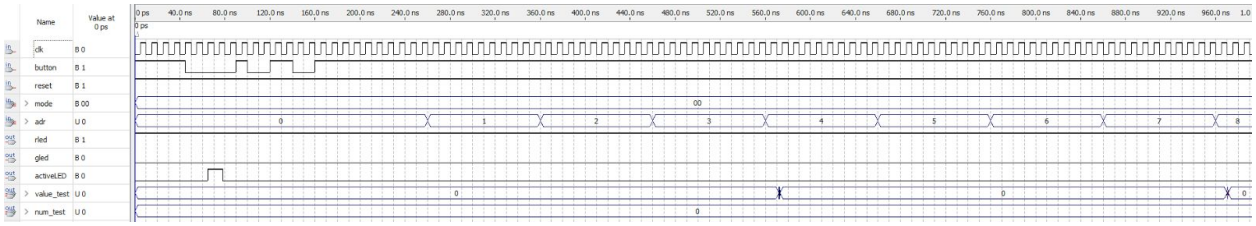


Fig. 14: Timing simulation for g07_lab3_testbed noop operation (mode=00). Nothing changes when noop is done

It was impractical to test the operation POP with a simulation. It would be necessary to either initialize the stack or push a value before being able to pop. This would require two operations being done and the long ignore period of the debouncer attached to the operation triggering button input would require an unreasonably long simulation. The proper functioning of the pop operation in g07_stack52 was previously verified, as well as the proper operation of the g07_debouncer circuit. These two facts combined with the fact that all the other operations were working as intended made it quite likely the pop function would also work correctly. As a result, the pop function was tested after the code was uploaded to FPGA. The pop function was found to be working properly in all test cases.

After testing through simulation, the circuit was uploaded to the board. The full functionality of it was tested and found to be correct for all operations. The proper decoding of the numbers sent to g07_7_segement_decoder were verified by producing all possible outputs.

The longest path in the testbed circuit is the adr[0] input to LED2[5] output. The delay is 9.879 ns in the fast model and 24.555 ns in the slow model.

The resource utilization of the testbed circuit was checked and is summarized in the table below.

Total logic elements	1,414 / 18,752 (8 %)
Total combinational functions	1,099 / 18,752 (6 %)
Dedicated logic registers	956 / 18,752 (5 %)
Total registers	956
Total pins	54 / 315 (17 %)
Total virtual pins	0
Total memory bits	6,400 / 239,616 (3 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Table 5: Resource utilization of the FPGA when g07_lab3_testbed is compiled

Advantages and Limitations

This circuit allows for a fully functioning system to be uploaded to the FPGA and tested on the physical hardware. This is necessary to ensuring that the simulation accurately reflect the behaviour of the circuit when implemented on the hardware.

However, a limitation of testing on the FPGA is that only certain parts/values of the system can be observed (i.e. the values connected to a physical output on the board). This makes it difficult to debug the circuit should it not function correctly on the board, as it is difficult to determine exactly where bugs are since the majority of the values are not viewable. However, this limitation can be overcome by using a monitoring software such as SignalTap.

Appendix 1: Pin Assignments

The proper pin assignments were found using the Altera DE1 Development and Education Board User Manual [2].

Schematic Pin	Physical Pin
LED1[1]	PIN_J1
LED1[0]	PIN_J2
LED1[2]	PIN_H2
LED1[3]	PIN_H1
LED1[5]	PIN_F1
LED1[4]	PIN_F2
LED1[6]	PIN_E2
LED2[0]	PIN_E1
LED2[1]	PIN_H6
LED2[2]	PIN_H5
LED2[3]	PIN_H4
LED2[4]	PIN_G3
LED2[6]	PIN_D1
LED2[5]	PIN_D2
clk	PIN_L1
reset	PIN_R22
button	PIN_R21
gled	PIN_U22
rled	PIN_R20
activeLED	PIN_Y21

adr[5]	PIN_L2
adr[0]	PIN_W12
adr[4]	PIN_M1
adr[3]	PIN_M2
adr[1]	PIN_U12
adr[2]	PIN_U11
mode[0]	PIN_L22
mode[1]	PIN_L21
LED3[0]	PIN_G5
LED3[6]	PIN_D3
LED3[5]	PIN_E4
LED3[4]	PIN_E3
LED3[3]	PIN_C1
LED3[2]	PIN_C2
LED3[1]	PIN_G6
LED4[6]	PIN_D4
LED4[5]	PIN_F3
LED4[4]	PIN_L8
LED4[3]	PIN_J4
LED4[2]	PIN_D6
LED4[1]	PIN_D5
LED4[0]	PIN_F4

Gareth Peters (260678626)
Peter Quinn (260689207)

Appendix 2: Input/Output Mapping on Board

7 Segment Displays

LED3	LED2	LED1	LED0
		Mod (card number)	Floor (suit)

Dip Switches

SW9	SW8	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
adr[5]	adr[4]	adr[3]	adr[2]	adr[1]	adr[0]			mode[1]	mode[0]

Push Button Switches

KEY3	KEY2	KEY1	KEY0
		Set mode	reset

LEDs

LEDR0	LEDG0
empty	full

References

[1] J. Clark, Lab #3–Sequential Circuit Design. Montreal: McGill, 2017.

[2] "DE1 Development and Education Board User Manual", Altera.com, 2017. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-4904342209-de1-usermanual.pdf. [Accessed: 06- Nov- 2017].

[3] "LPM Quick Reference Guide", Altera.com, 2017. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/catalogs/lpm.pdf. [Accessed: 09- Nov- 2017].