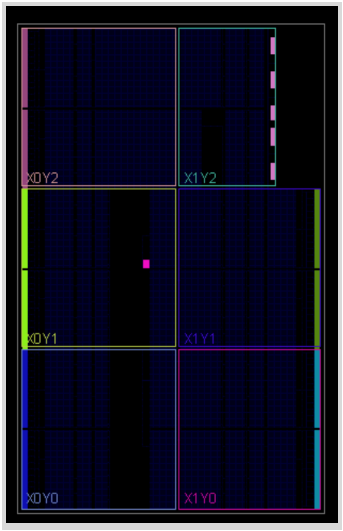
EECS 112L Organization of Digital Computers Lab
Lab 5 - Time, Power, & Area for MIPS Pipelined Processor

Date: March 15, 2024
Student ID: XXXXXXXX

Objective & Procedure

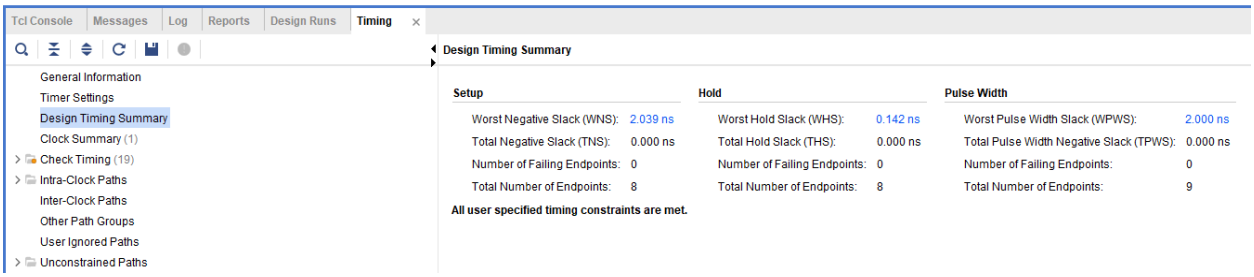
In this lab, we aim to time and calculate the power & area for pipelined MIPS processors. This lab will involve learning and implementing practical ways to collect data points that can determine the efficiency of our processor design and development. Modifications can be made after these values are collected and compared to new iterations, overall aiding the development of processors. The procedure of this lab involves running the behav_counter file with the Basys 3 FPGA selected for synthesis and the MIPS pipelined processor designed in Lab 4. We will be following the procedure outlined in the lab manual to report.

2 Behavior Counter



2.2 Timing Constraints (P3)

All timing constraints have been met as we can see in the figure below.



The worst negative slack (WNS) was 2.039 ns for the setup. The hold WHS 0.142 ns for the hold period. Finally, pulse width (WPWS) was 2.000 ns for this design. The final statement shows all user specified timing constraints have been met for this design.

2.3 Report Utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
behav_counter	12	8	20	1

Figure 2: Hierarchy Utilization

This figure shows the slice LUTs and registers used by the behav_counter design.

Resource	Utilization	Available	Utilization %
LUT	12	20800	0.06
FF	8	41600	0.02
IO	20	106	18.87

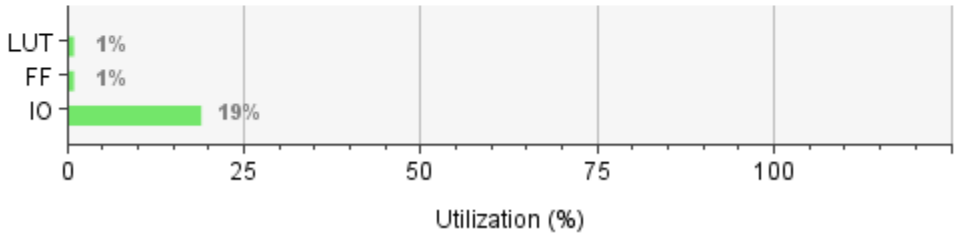


Figure 3: Hierarchy Summary

(1) For this counter, we have used 12 LUTs of the available above, 8 FFS, and 20 IOs.
(2) With this consumption in mind, considering the least available resource (IO) - number per counter requirements, we can fit approximately 106/20 ~ or **5 counters** on this board. We know that we will have certain resources like LUT and FF abundantly available as a fraction of percentage of the total available are required per counter, yet we must consider the availability of the least available resource – or in other words – the most consumed resource, to find the max number we can allocate on a given board.

2.4 Report Power

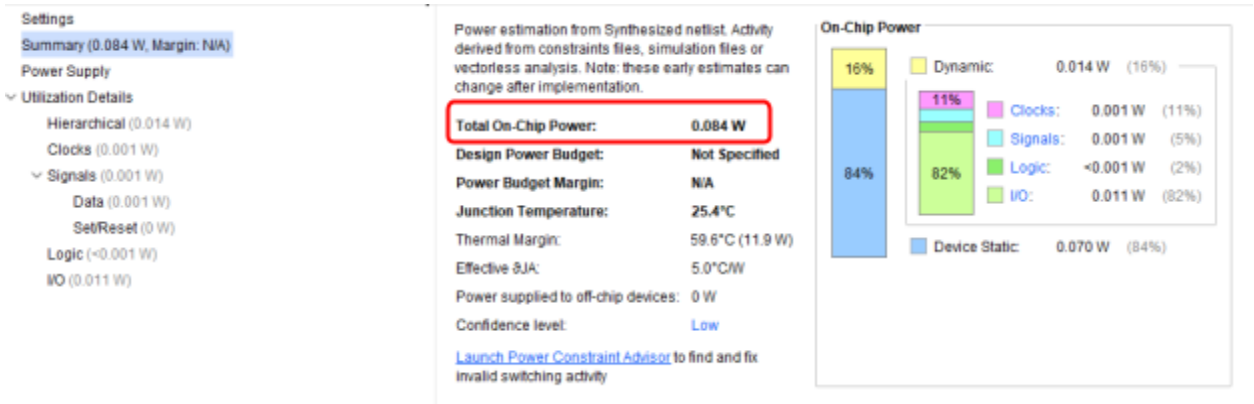


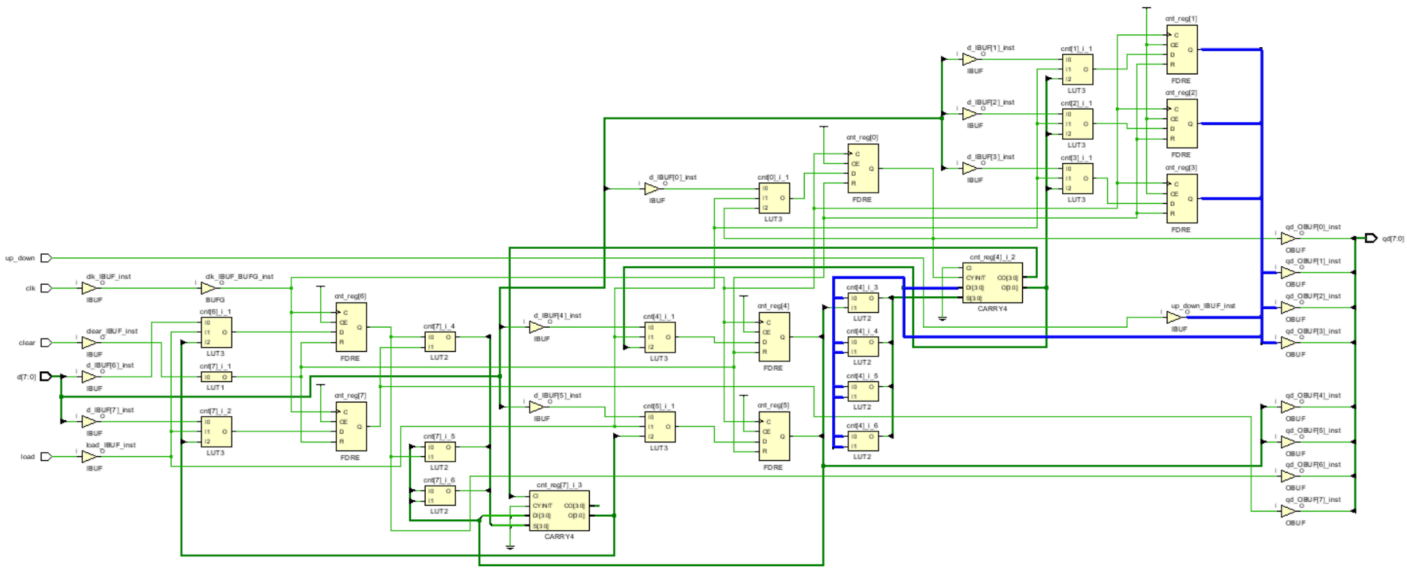
Figure 4: Power Summary

(1) From the figure above, we can conclude that the total On-Chip Power is approximately 0.084 W.
(2) When we compare static and dynamic power, static power is 84% of the total on-chip power, while dynamic power is 15%. Therefore, static power is greater than dynamic power. Importantly, this is because of their properties and meanings. Static power by definition is the power used by the chip regardless of the tasks running. It is commonly due to transistor leakage. Dynamic power adjusts to the speed of programs being run, workload (comparing a heavy usage with multiple programs, and sleep mode of the chip). Consequently, dynamic power is less in proportion to the representation of static power in the compositions of Total On-Chip Power.
(3) According to the power report, a massive 82% of the dynamic power composing Total On-Chip Power, is represented by I/O alone. Relative to other components, I/O consumes more power. The equation for power consumption: P = C * V^2 * (a * f), where P = power consumption, C = capacitance, V = Voltage, a = activity / switching rate, f = switching frequency can be used to understand why this is the case.

Part 3 Explanation:
I/O components consume more power for dynamic power consumption because of important factors. One of which is due to high capacitance: since they drive signals across far distances, they need to consume more energy in the process compared to logic gates that perform the operation being given the inputs at execution. Additionally, due to frequent transitions, switching factor is high for I/O operations, multiplying the power consumption. Driving loads taken into consideration by the same token will also have power consumption increase because of the need for more power for driving external loads. To keep everything functional across great distances (comparatively), voltage sources would also need to be higher compared to voltage of internal circuits. Finally, I/O would need additional features like shifting and buffering, increasing the overall circuitry for connections between components. All in all, these components, directly proportional in the equation for power consumption, would increase power by a factor, thereby making this component of dynamic power the most representative.

2.5 Schematic Diagram

(1) Diagram



(2) Explanation of Parts:

NOTE: For these definitions, I utilized the textbook as reference because of the specificity of the components.

FDRE: (Flip-Flop with Data Retention Enable) - used for sequential logic operations is a D Flip-Flop with the additional feature of Data Retention Enable – it stores a bit of data and is clocked to transfer the stored value to output on the rising edge of the clock cycle.

LUT: (Look-Up Table) – building blocks for implementing combinational logic functions. LUT's are configurable memory blocks that store truth tables for logic functions. The truth table maps all possible input combinations to output values and when an input is applied to the LUT, it fetches the corresponding output value from its memory unit.

CARRY4: used for arithmetic operations, like addition within FPGAs logic fabric. Used to propagate carry signals across multiple slices (or logic cells) within FPGA. enables fast addition operations by cascading the carry signals through chains of logic elements.

IBUF: (Input Buffer) – designed to buffer and condition external input signals before they enter the FPGA fabric. Ensures that signals meet the required voltage levels, timing specifications and other electrical characteristics specified by FPGA. It can perform voltage shifting to translate voltage levels of external signals to appropriate voltage levels used with the FPGA, allowing external devices to operate at different voltage levels.

OBUF: (Output Buffer) – designed to buffer + condition internal signals from FPGA fabric before they are driven out of the device to external devices or other internal components. Ensures that signals meet the required voltage levels, timing specifications, and other characteristics specified by the FPGA.

3 MIPS Processor Specification

3.2 Timing Constraints

(1) Screenshot

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-5.777 ns	Worst Hold Slack (WHS):	0.074 ns	Worst Pulse Width Slack (WPWS):	1.250 ns
Total Negative Slack (TNS):	-770.725 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	1178	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	3603	Total Number of Endpoints:	3603	Total Number of Endpoints:	1411

Timing constraints are not met.

(2) Explanation: Constraints were not met given the setup ran inefficiently. The clk component was set up inefficiently and resulted in the diagram above. The clock summary for the timing below is given as follows:

Name	Waveform	Period (ns)	Frequency (MHz)
clk	{0.000 2.500}	5.000	200.000

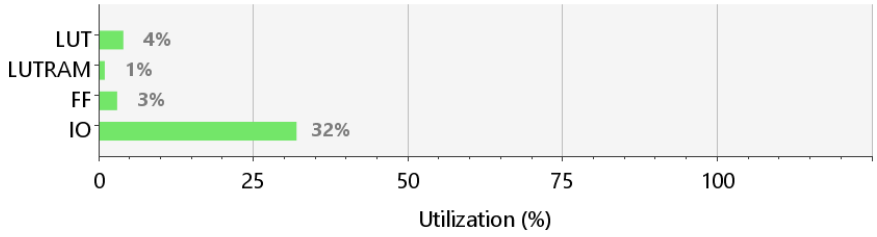
3.3 Report Utilization

```
Report Cell Usage:
+------+----------+------+
|      |Cell      |Count |
+------+----------+------+
|1     |BUFG      |     1|
|2     |CARRY4    |    10|
|3     |LUT1      |     4|
|4     |LUT2      |    48|
|5     |LUT3      |    80|
|6     |LUT4      |    52|
|7     |LUT5      |   108|
|8     |LUT6      |   480|
|9     |MUXF7     |   184|
|10    |MUXF8     |    28|
|11    |RAM256X1S |    32|
|12    |FDCE      |   290|
|13    |FDRE      |   992|
|14    |LDC       |     7|
|15    |IBUF      |     2|
|16    |OBUF      |    32|
+------+----------+------+
```

(For reference)

(1) Summary

Resource	Utilization	Available	Utilization %
LUT	877	20800	4.22
LUTRAM	128	9600	1.33
FF	1289	41600	3.10
IO	34	106	32.08



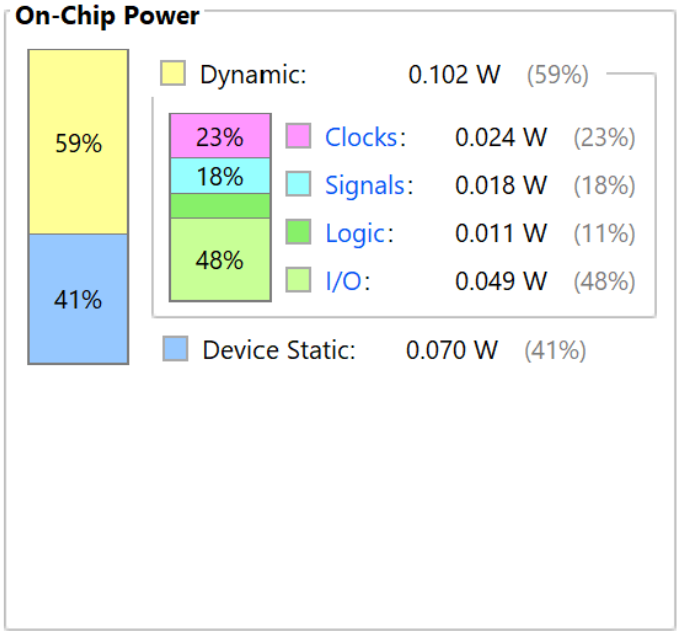
(1) From this design, 877 LUTs, 1289 FFs, 34 IOs were used by the design. More details are above.
(2) From this resource consumption, approximately 106/34 or 3 designs could fit on the FPGA. The FPGA could allocate its resources for IO approximately evenly to 3 units to fit and function. Resources like LUTs, LUTRAMs, and FFs are of more abundance and will be used to determine how many can fit, given I/O is more scarce given its availability and usage per unit. Theory into why is described above when analyzing the behav_counter design.

3.4 Report Power

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

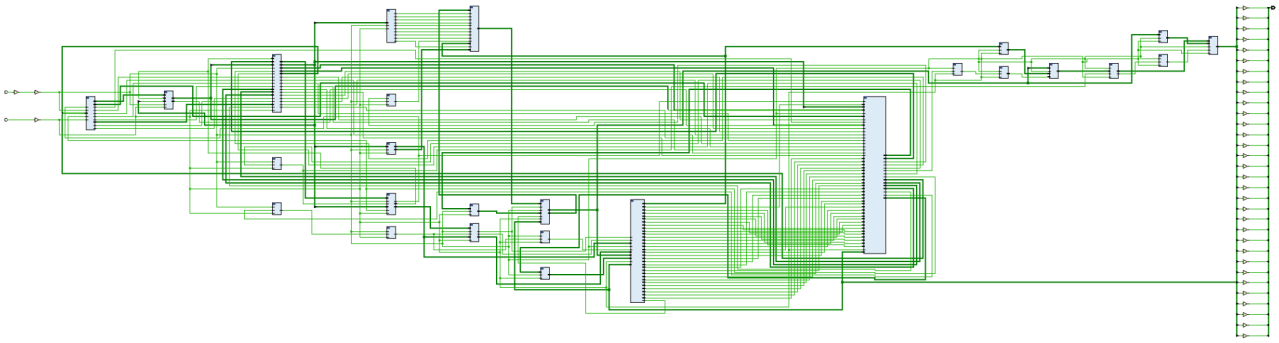
Total On-Chip Power:	**0.173 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.9°C**
Thermal Margin:	59.1°C (11.8 W)
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity



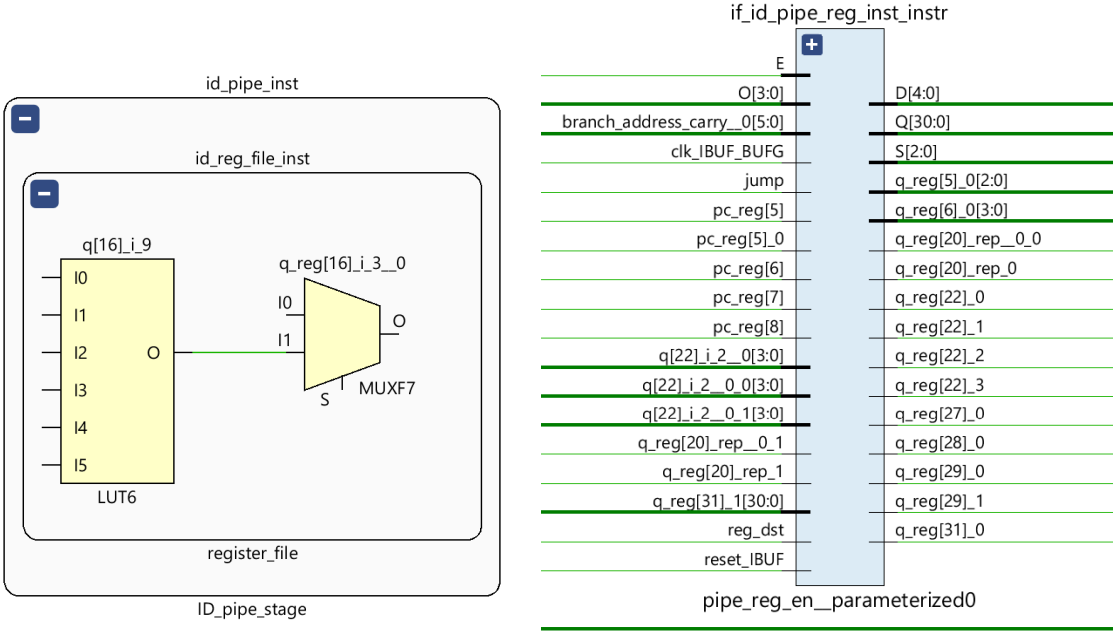
This image shows the power consumption diagram. Details regarding why one component is more than another is explored and explained more thoroughly in previous sections of the lab for the behav_counter. In essence, however, we see that Dynamic and Device Static are far different ratios from before, and represent a better description of power.

3.5 Schematic



The image is a zoomed-out view of the design. The design contains many small important components, such as register banks at each stage of the processor, connections from one component to another, and more that can be inspected.

As an example, this is the ID of the pipe-stage and the image of the pipe_reg_en_parameterized.



4 32-bit ALU Implementation on Basys 3 (with Engineers Michael Phieffer & Canting Zhu)

Case	A	B	ALU Sel	Output	Zero	Carry out	Overflow
i.	111	001	0000	001	0	0	0
ii.	100	000	0000	000	1	0	0
iii.	110	101	0001	111	0	0	0
iv.	000	000	0001	000	1	0	0
v.	001	010	0010	011	0	0	0
vi.	100	111	0010	011	0	1	1
vii	110	110	0010	100	0	1	0
viii.	011	010	0110	001	0	0	0
ix.	100	111	0110	101	0	0	0
x.	001	011	0111	001	0	0	0
xi.	100	000	0111	001	0	0	0
xii.	111	001	1100	000	1	0	0
xiii.	001	101	1100	010	0	0	0
xiv.	000	000	1111	001	0	0	0
xv.	100	001	1111	000	1	0	0

Figure 1: 32-Bit ALU Operations Test Cases

Procedure: Since there were multiple steps to this portion of the lab, some notes regarding the set up will be provided. The instructions in the lab manual included running synthesis, then implementation, generating bitstream, loading the files into the board, and manually testing each case with the switches provided.

Most importantly, in testing, noting which are input/output/select and carryout output overflow is important. Here is a screenshot of the details from the lab manual to aid checking if the test cases are met by the design loaded into Basys 3.

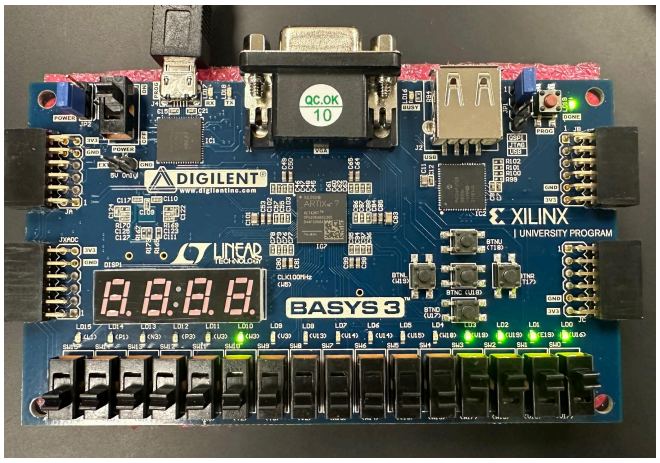
INPUT(Switch): 3bits Input A: SW2-SW0; 3bits Input B: SW5-SW3; 4bits Input Sel: SW9-SW6;

OUTPUT(LED): 3bits Output Result: P3 U3 W3; 1bit Output CarryOut: N3; 1bit Output Zero: P1; 1bit Output Overflow: L1;

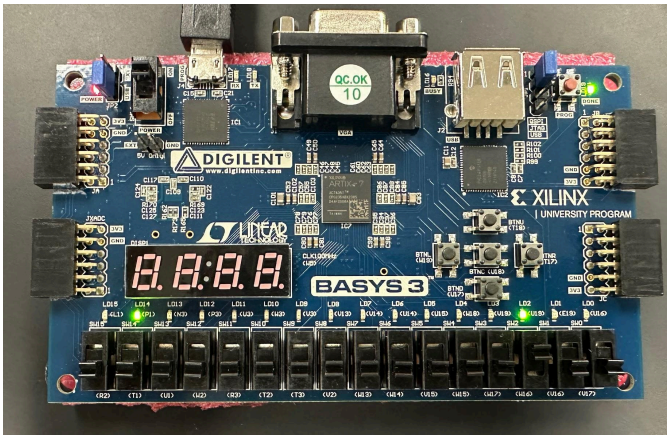
This is the input/output layout provided for the test cases.

Please look up exact values for the inputs and outputs in the table above. Omitted inputting each input value by hand for ease of explanation.

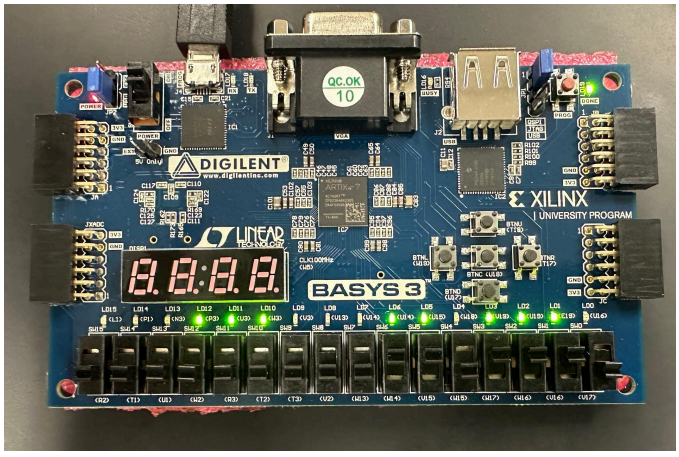
Test Case 1: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise AND operation is working as intended.



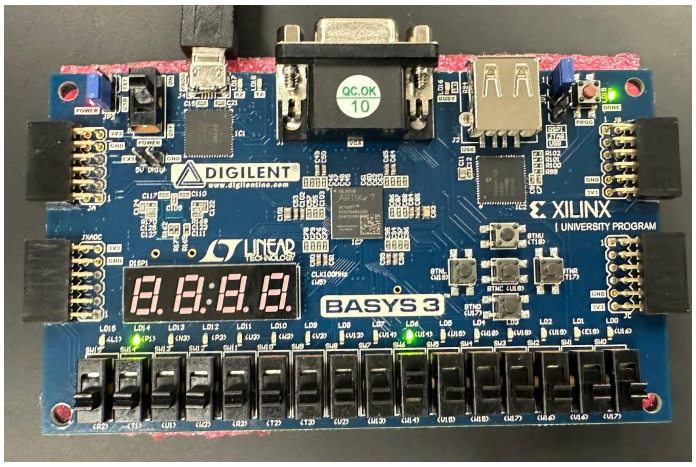
Test Case 2: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise AND operation is working as intended.



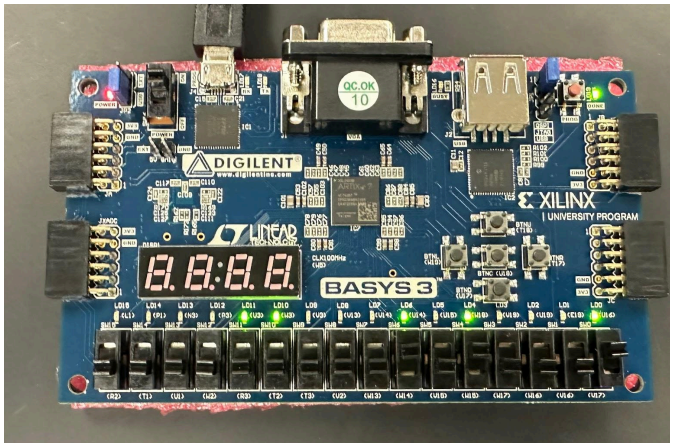
Test Case 3: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise OR operation is working as intended.



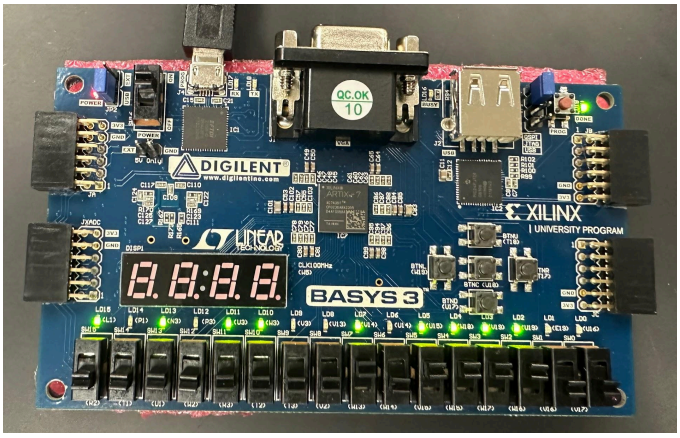
Test Case 4: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise OR operation is working as intended.



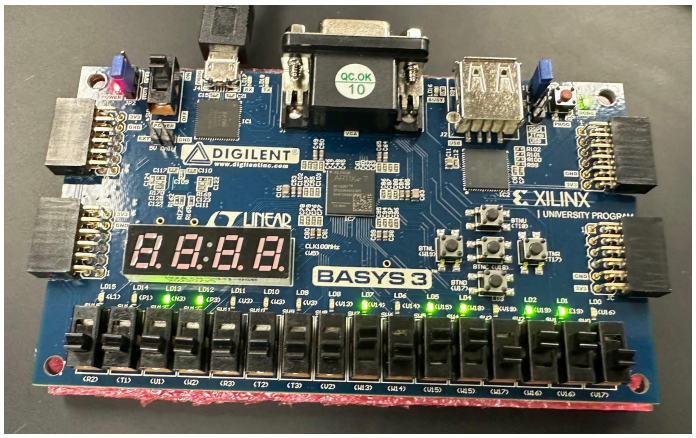
Test Case 5: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed ADD w/out carry out operation is working as intended.



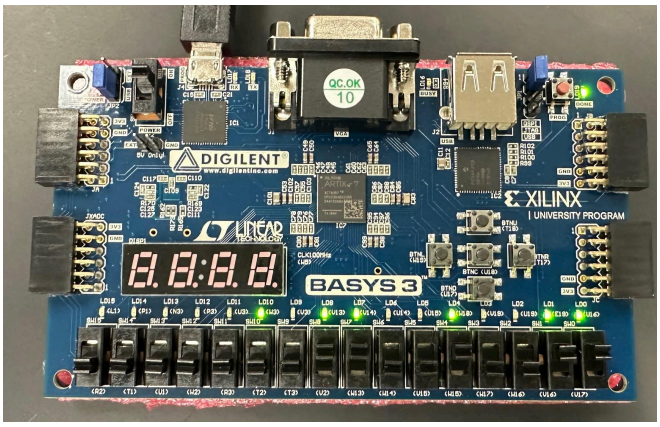
Test Case 6: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed ADD w carry out and overflow operation is working as intended.



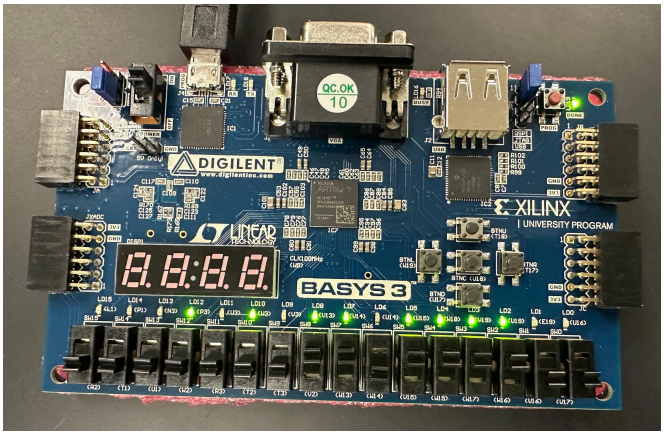
Test Case 7: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed ADD w carry out and no overflow operation is working as intended.



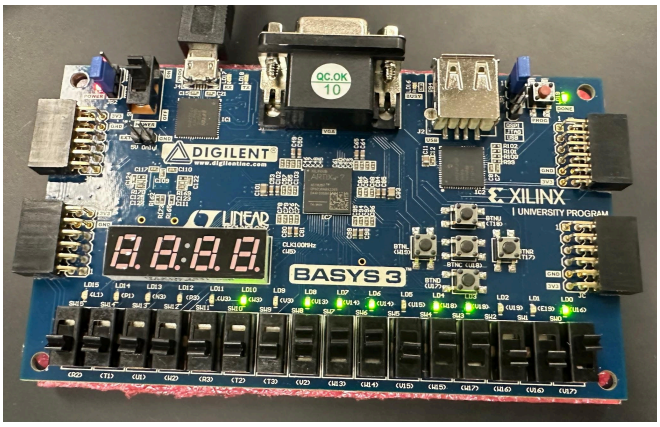
Test Case 8: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed SUB with overflow checking operation is working as intended.



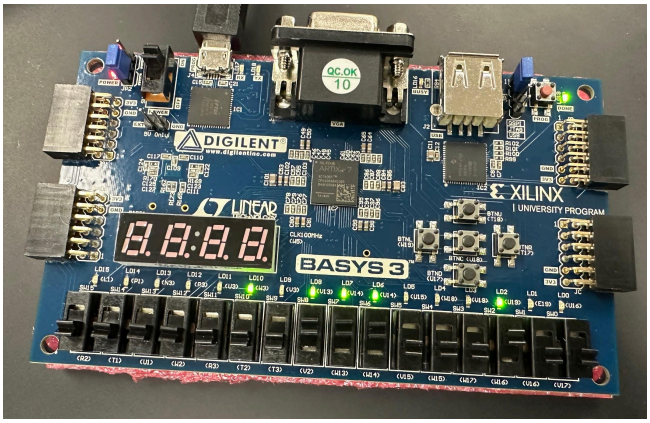
Test Case 9: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed SUB with overflow checking operation is working as intended.



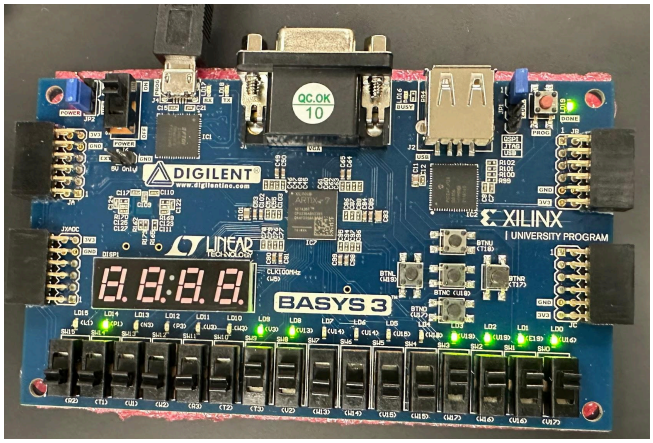
Test Case 10: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed LT comparison checking operation is working as intended.



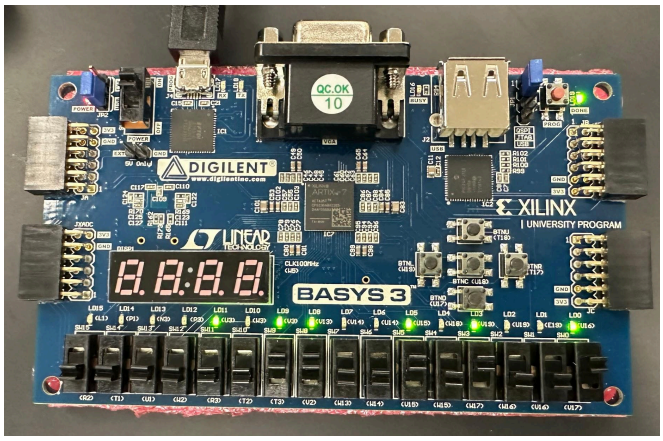
Test Case 11: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The signed LT comparison checking operation is working as intended.



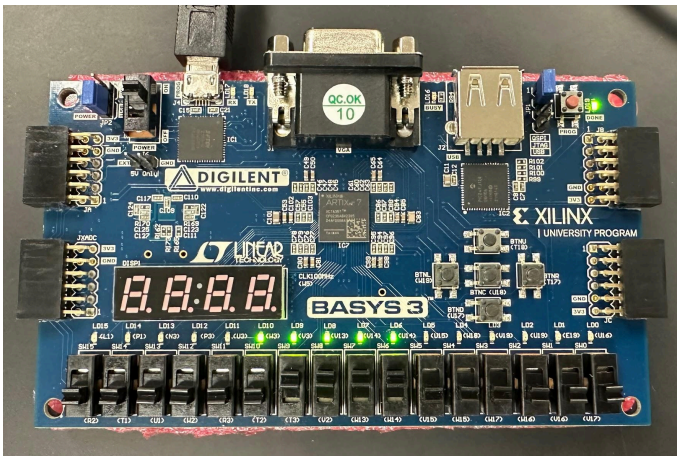
Test Case 12: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise NOR operation is working as intended.



Test Case 13: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The bitwise nor operation is working as intended.



Test Case 14: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The equal to operation is working as intended.



Test Case 15: A, B, ALU sel as inputs return the exact outputs in the test case diagram above. The equal to operation is working as intended.