# 4CS015 – Workshop #5 TO BE SUBMITTED

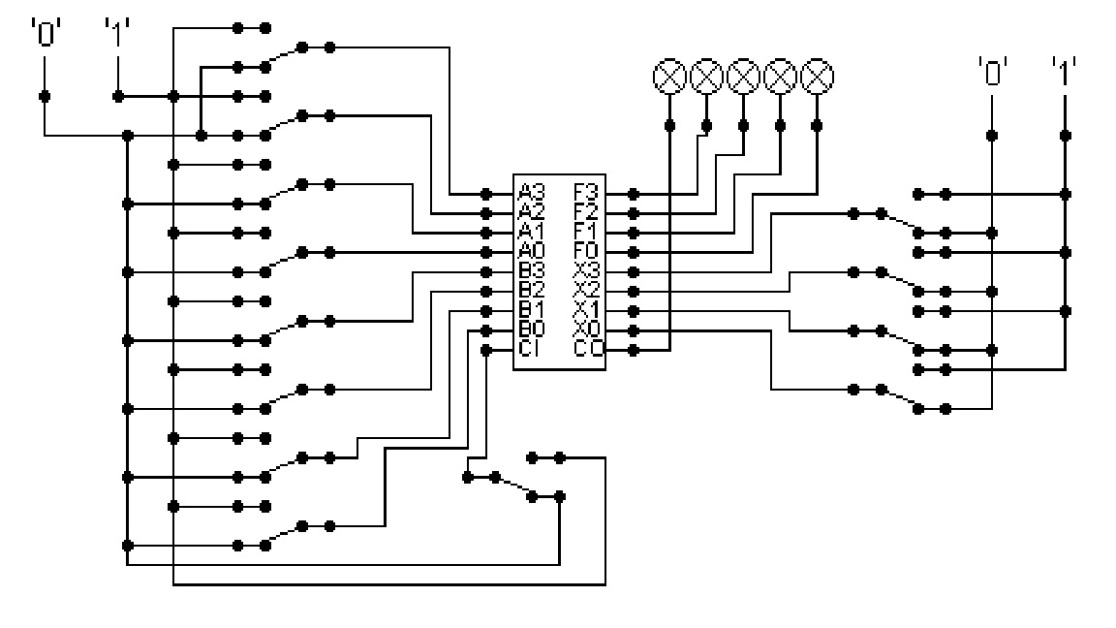
Name: Alan Shah

Student ID: 2329767

This is a marked workshop. It forms the second part of your portfolio. You will need to complete the workshop and then submit a copy of this document with a title that follows the following format (“DENNETT 1234567 wsp5.docx”), via CANVAS, by the deadline.

**Workshop tasks:**

Arithmetic Logic Unit:

Load the LogSim Arithmetic Logic Unit Circuit **alu.cct** from inside the logsim application (You'll find it in the logsim folder) (***You may need to right-click on the link to download the file instead of opening it in the browser)***. It should look like this:  
  
  
  
The circuit behaves like a simple arithmetic logic unit. The inputs A0-A3 represent a 4 bit binary number. Inputs B0-B3 represent another binary number. A0 and B0 are the least significant bits respectively. The following table details the functions supported by the chip. All other control lines = 0.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function | AND | OR | XOR | NAND | NOR | NOT A | ADD | SUBTRACT |
| X3 – X0 | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 1010 | 1011 |

Use A= 11 B=4, and complete the following table in binary ***(15 marks)***:

A = 11, Binary Value = 1011

B = 4, Binary Value= 0100

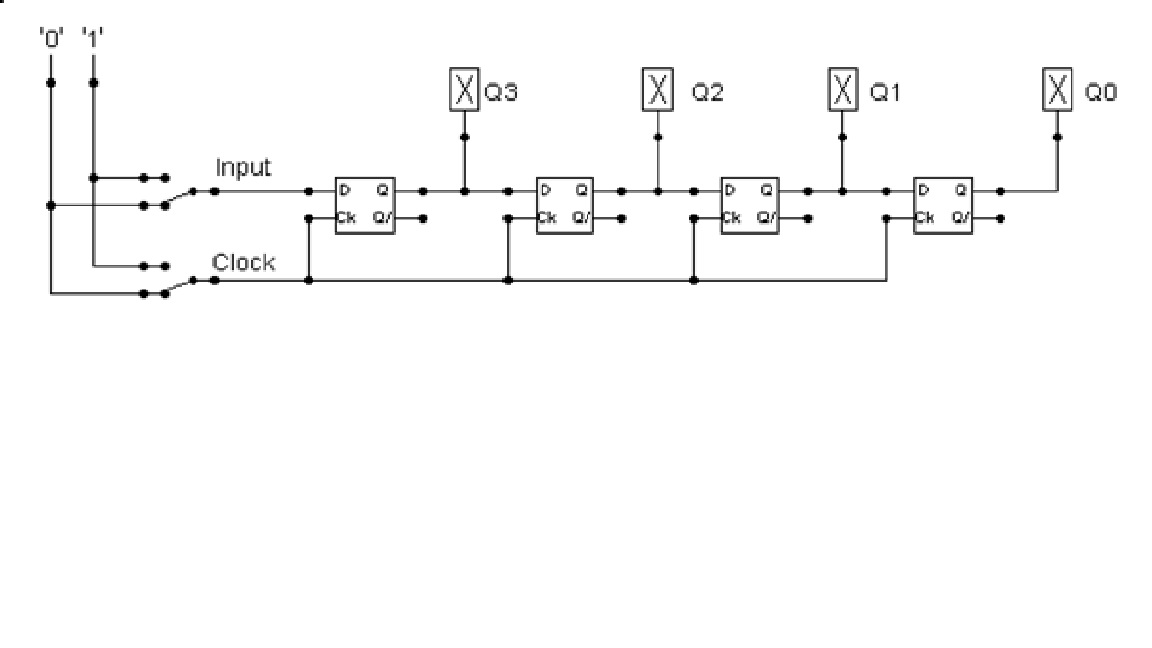
|  |  |
| --- | --- |
| FUNCTION | OUTPUT |
| AND (0000) |  |
| OR (0001) |  |
| XOR (0010) |  |
| NAND (0011) |  |
| NOR (0100) |  |
| NOT A (0101) |  |
| ADD (1010) |  |
| SUBTRACT (1011) |  |

The logical operations are bitwise. Manually prove each operation has returned the correct result by  ***(15 marks)***:  
Example:  1 0 1 1  
                 1 0 1 0 AND OPERATION  
                 1 0 1 0 RESULT

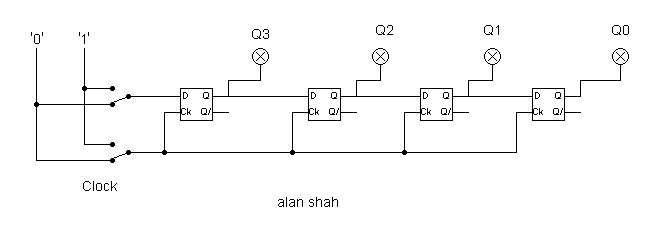
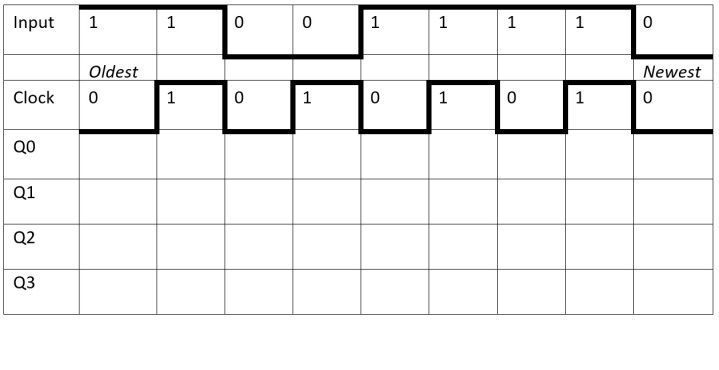
|  |  |  |
| --- | --- | --- |
| AND | OR | XOR |
| 1011  0100 AND operation  0000 | 1011  0100 OR operation  1111 | 1011  0100 XOR operation  1111 |

|  |  |  |
| --- | --- | --- |
| NAND | NOR | NOT A |
| 1011  0100 NAND operation  1111 | 1011  0100 NOR operation  0000 | 1011  0100 NOT A operation  0100 |

|  |  |
| --- | --- |
| ADD | SUBTRACT |
| 1011  0100 ADD operation  1111 | 1011  0100 SUBTRACT operation  0110 |

Serial to Parallel Decoder ***(30 marks)***:  


Build the circuit above and complete the following timing diagram by filling in the table spaces with ‘1’ or ‘0’. ***(15 marks)***

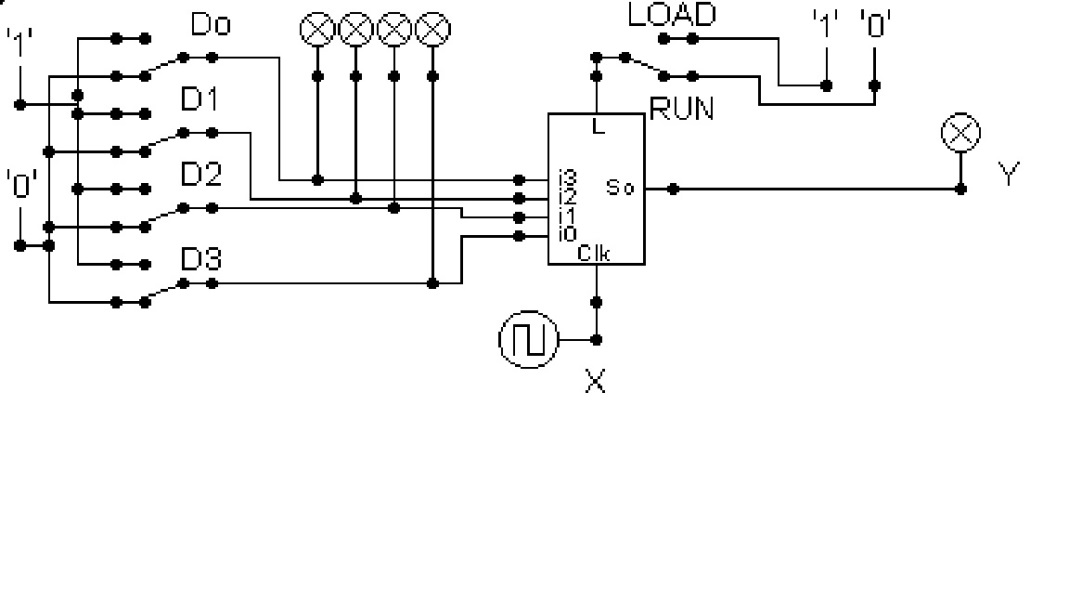
  


Table

Description automatically generated

Describe what the circuit does. ***(15 marks)***  
A shift register known as a SIPO is used in the circuit schematic that is shown. This kind of shift register receives input in a serial method and generates outputs in a parallel fashion. A sequence of digital data must be fed into a set of flip-flops or other storage components for them to operate properly. To produce the parallel data output, each storage component's output is next linked to a line inside the SIPO circuit. The input data is initially stored in a single memory cell, transmitted one bit at a time, and then outputted in parallel through the shift register's output pins. Applications for SIPO shift registers include digital storage, data collecting, and communication systems and logic circuits. Data is entered serially and outputted simultaneously in the sample provided. The circuit receives an input value and outputs Q0, Q1, Q2, and Q3 in accordance with that value throughout each clock cycle. After each instance, the clock value in the circuit changes from 0 to 1, and while the clock value is 0, the most recent input value is 0. Then, in accordance with the instructions provided by the question, two pairs of 1s are utilized as inputs, followed by two 0s and a 1 in a cycle of two 0s and two 1s. All four output values must equal 0 for the clock and input values to be equal. If both the clock value and the input value are 1, then all values every time, except for Q3, which has a value of 1, they finally approach 0. Each time the input values change, one bit is produced. Q3 and Q2 turn on when the input is set to 1, while Q1 and Q0 stay off when the clock is set to 0. In addition, only Q2 and Q1 are activated when the input value is 0 and the clock value is 1. With the exception of Q2, all outputs are switched on when the output value is 1 and the clock value is 0.

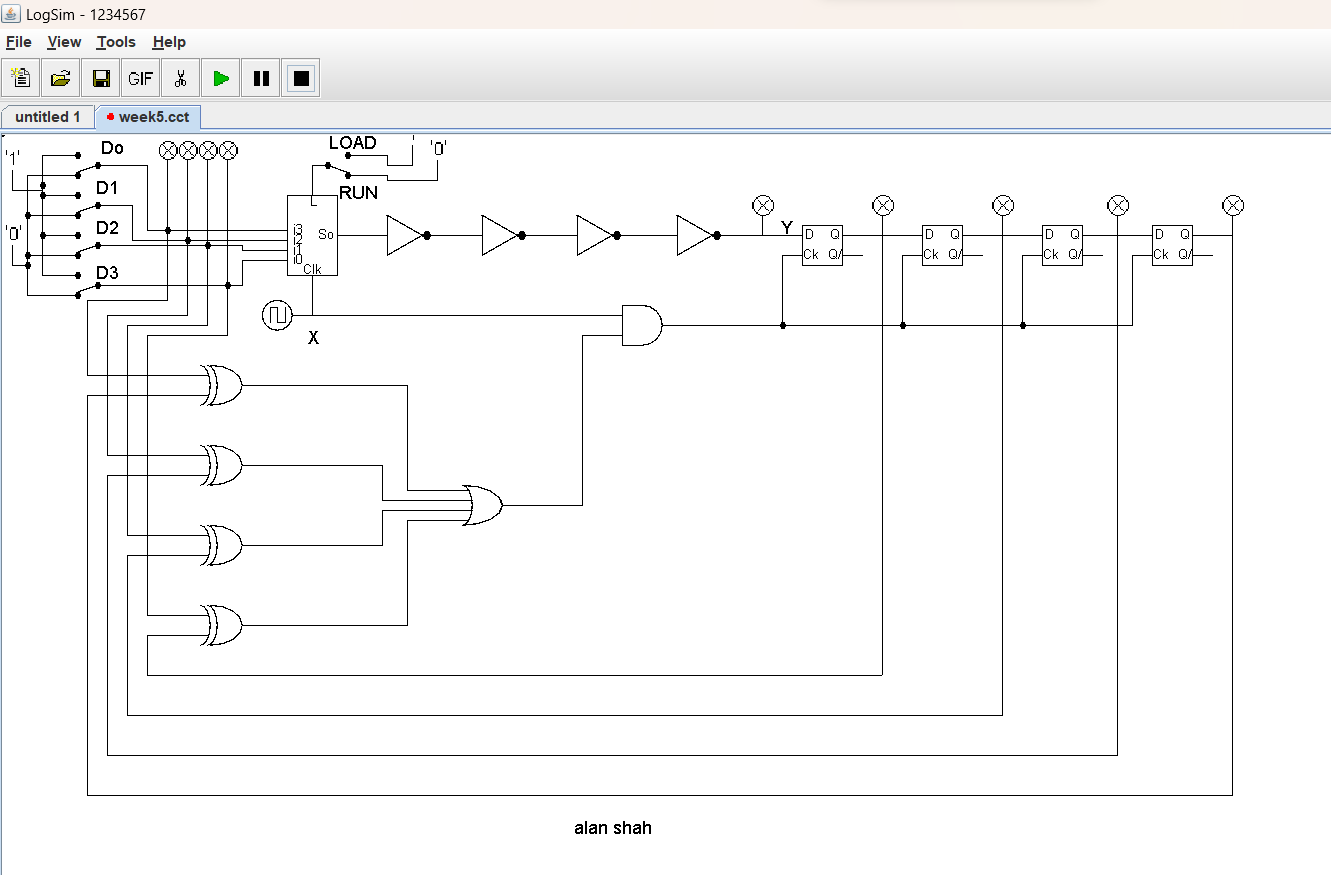
Parallel to Serial converter

Open the LogSim circuit **week5.cct** from the Logsim folder. It should look like this:  
  
  
  
Describe what this circuit does. ***(15 marks)***

An electronic circuit created for data storage is the PISO shift register. As binary values are moved from one step to the next, it has flip-flops that keep them constant. These flip-flops are arranged in a chain with just one input and one output. When the final flip-flop in the chain is reached, the data may be executed after being put into the first flip-flop and moving from one to the next.

A logic circuit with a string of four D flip-flops is used to represent a PISO register in the figure above. Even though the flip-flops are near to one another, they are each connected to the clock input by a different multiplexer. Both the output of and the multiplexer's input are linked to The output of the multiplexer is linked to the subsequent flip-flop, with the parallel inputs and preceding flip-flop serving as its parallel inputs. Since the flip-flops are linked in series, they all get the same clock signal. Specifically, the circuit in issue transforms the input-output Y into four output signals that are identical to the inputs D0 to D3 that come after them.

Design and add to the above circuit an additional circuit that takes the Clock X and the Output Y and decodes Y into 4 output indicators so that they match D0 – D3. Insert the LogSim GIF output of your design in the space below.



The highest marks will go to those who design the circuit such that it **AUTOMATICALLY** stops (not pauses) when the input to the circuit matches the output to the circuit

*Note: Save your GIF image when your output indicators match the input D0 - D3*. (35 marks)

Diagram, schematic

Description automatically generated