**Lab 1. Stopwatch**

* **Objectives:**
* Be familiar with the Logisim software.
* Learn how to design combinational and sequential circuits with Logisim.
* **Resources:**
* Download Logisim 2.7.1 from canvas. **Highly recommended to install in Windows system. Students reported that its MAC or Linux version does not work well.**
* Download “Lab1 - Stopwatch.circ” from Canvas. Inputs, outputs, and most of the necessary electronic components have been given in each circuit. **Do not make any changes on the given Inputs and outputs in the circuits, and Do not change the appearances of electronic components.**
* **Requirements**
* Write down your answers to the questions in the following “4. Tasks” section in your lab report. Do not forget to show your demo to TA when finish all the project.
* This lab is separated into three parts and have three deadlines. Please submit the report for each part on time.

**<1> Part I: tasks 4.1.1 – 4.1.6 (30 pints), the deadline is 2/9.**

**<2> Part II: tasks 4.2.1 – 4.2.6 (36 pints), the deadline is 2/16.**

**<3> Part III: tasks 4.2.7 – 4.4 (34 pints), the deadline is 2/23.**

* **Tasks**:

Design a stopwatch that can realize the following functions:

* Start: reset counter to zero and start counting from zero.
* Stop: stop counting and display the time.
* Store: update the best (shortest time) record and display the result.
* Reset: reset the counter to 00.00 and the best record to 99.99.

To realize this stopwatch design, you need to build the following combinational and sequential circuits:

Combinational circuits: 2-channel 16-bit multiplexer, 16-bit unsigned comparator, and display control circuit.

Sequential circuits: 4-digit counter, 16-bit register for best record storage, stopwatch control unit

**4.1 Combinational circuits design**

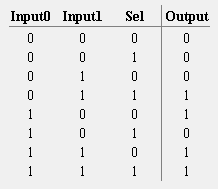
For combinational circuits, the output depends only on the current input, as shown in the following figure:

To design a combinational circuit, you can input the truth table that reflects the relation between inputs and outputs to Logisim, then Logisim will design the circuit for you automatically.

To make the circuit design easier, we can design simple circuit units first, then use simple circuit units to build complex circuits.

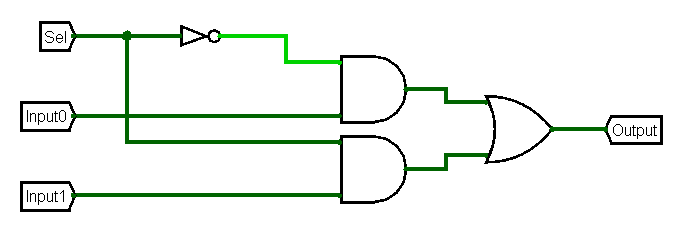
**4.1.1 Design a 2-channel 1-bit multiplexer (5 points)**

Truth table:

Generated circuit by Logisim:

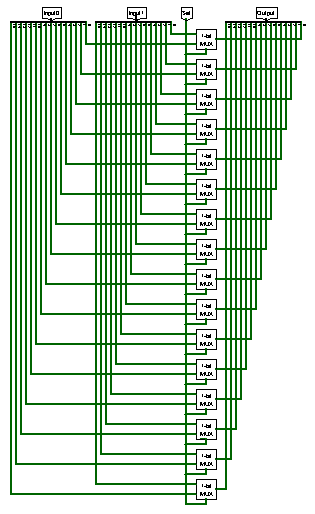
**4.1.2 Design a 2-channel 16-bit multiplexer (5 points)**

**Requirement:** Only 1-bit multiplexers designed in 4.1.1 and basic logic gates are allowed.

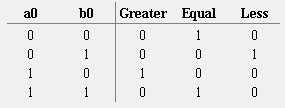
Explain your design method:

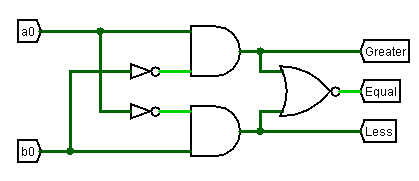
Make sixteen 2c\1b MUXs

Circuit:

**4.1.3 Design a 1-bit unsigned comparator (5 points)**

Truth table:

Generated circuit by Logisim:

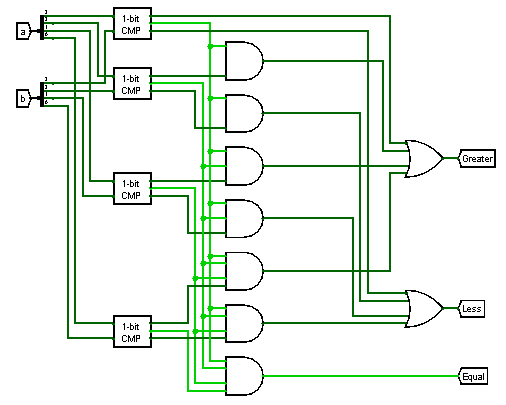
**4.1.4 Design a 4-bit unsigned comparator (5 points)**

**Requirement:** only1-bit unsigned comparators designed in 4.1.3 and basic logic gates are allowed.

Explain your design method:

I put each bit pair through a comparator and connected the outputs into a series of logic gates corresponding to its output.

Circuit:

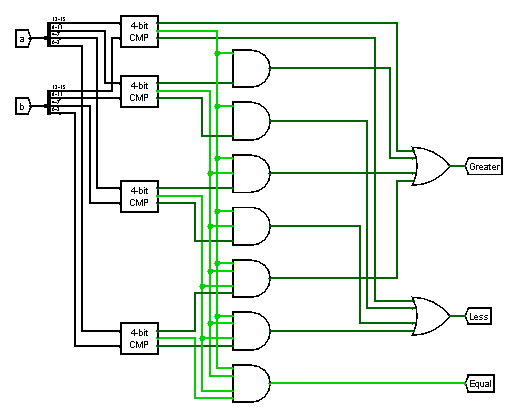
**4.1.5 Design a 16-bit unsigned comparator (5 points)**

**Requirement:** only4-bit unsigned comparators designed in 4.1.4 and basic logic gates are allowed.

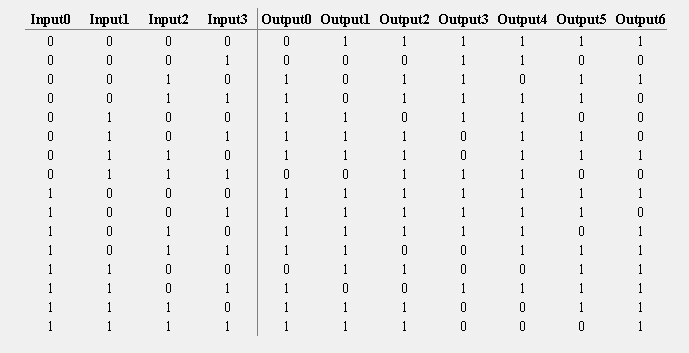
Explain your design method:

I used the same basic format as the 4-bit comparator, but replaced 1 bit comparators with the 4 bit comparators.

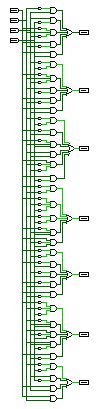
Circuit:

**4.1.6 Design a control unit for 7-segment display (5 points)**

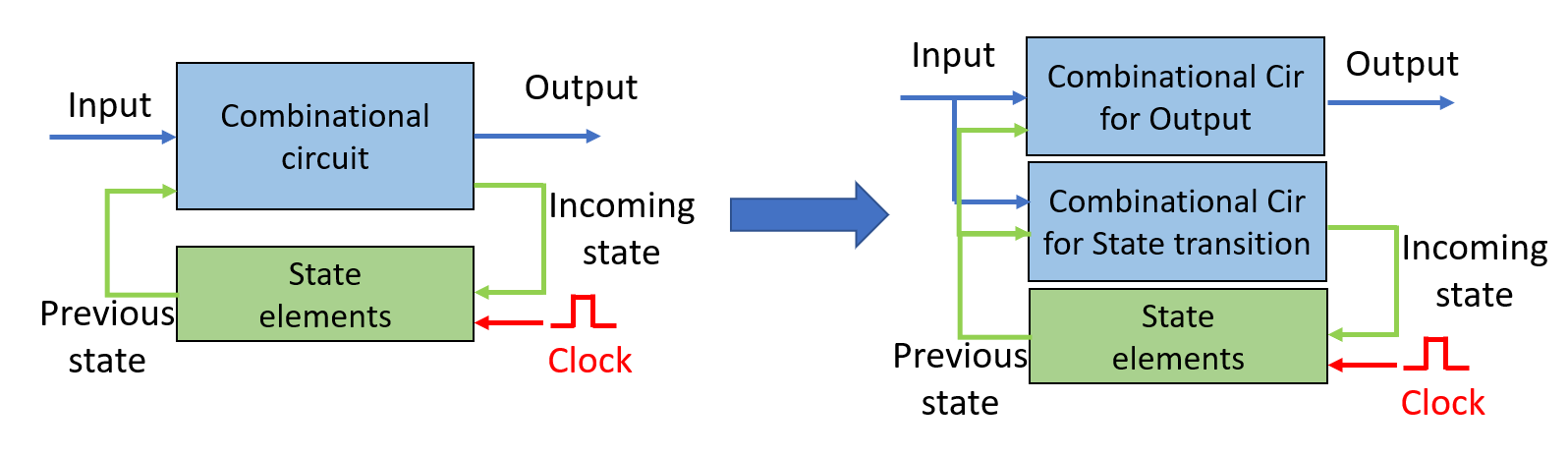
Truth table:



Generated circuit by Logisim:

* **Sequential circuits design**

Sequential circuits are with memory elements that contain state. The outputs depend on both their inputs and the contents of the internal state. As shown in the following figure, the left figure shows how a representative sequential circuit. To make it easy to design a sequential circuit, we can separate the combinational circuit part into two parts (as shown in the right diagram): the combinational circuit to generate output and the combinational circuit to generate new states. To design the combinational circuit that generates “Output” only, we only need to find the corresponding “Input”, “Previous state”, and “Output”, then build a truth table in Logisim. Finally, Logisim can build the circuit automatically. Similarly, to design the combinational circuit that generates “Incoming states”, we only need to find the corresponding “Input”, “Previous state”, and “Incoming states”, then build a state-transition table in Logisim. Logisim can build the state transition circuit for us automatically.



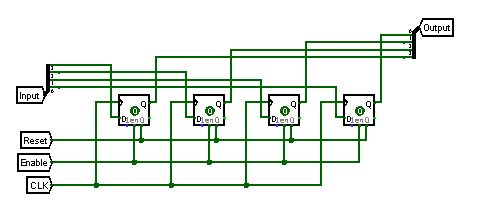
To make the circuit design easier, we can design simple circuit units first, then use simple circuit units to build complex circuits.

**4.2.1 Design a 4-bit parallel-load register (use D-Flip-Flop) (6 points)**

Explain your design method:

Put each bit through a d flip-flop

Circuit:

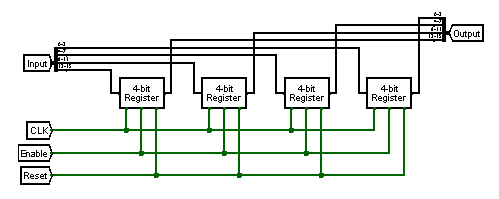
**4.2.2 Design a 16-bit parallel-load register (6 points)**

**Requirement:** only4-bit parallel-load register designed in 4.2.1 and basic logic gates are allowed.

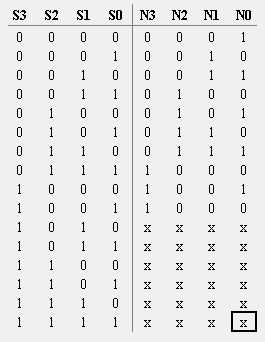
Explain your design method:

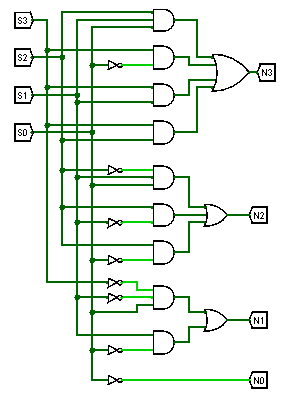
Make the same circuit but replace the d flip-flops with 4-bit Registers

Circuit:

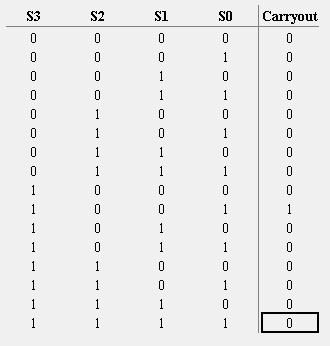
**4.2.3 Design a combinational circuit for state-transition of the 4-bit BCD counter (6 points)**

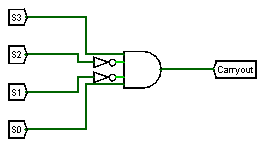
Truth table:

Generated circuit by Logisim:

**4.2.4 Design a combinational circuit for the output of the 4-bit BCD counter (6 points)**

Truth table:

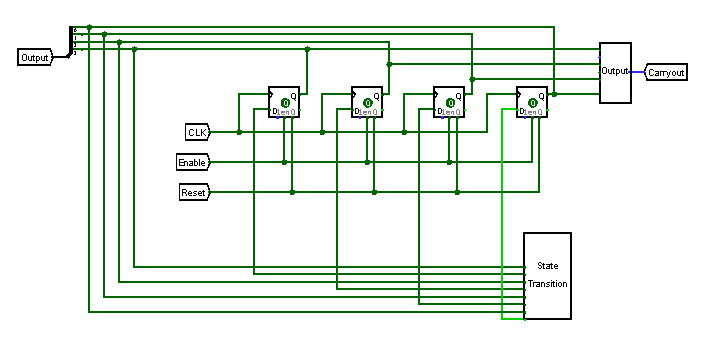
Generated circuit by Logisim:

**4.2.5 Design a 4-bit BCD counter (states are stored with D-Flip-Flops) (6 points)**

Explain your design method:

Clock, Enable, and Reset connected to each corresponding input on each D-flip-flop. The input is fed from the output of the state transition circuit. The carryout is connected to the output of the output circuit.

Circuit:

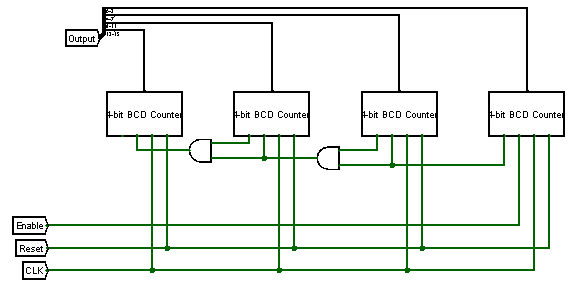


**4.2.6 Design a 4-digit BCD counter (6 points)**

**Requirement:** only4-bit BCD counters designed in 4.2.5 and basic logic gates are allowed.

Explain your design method:

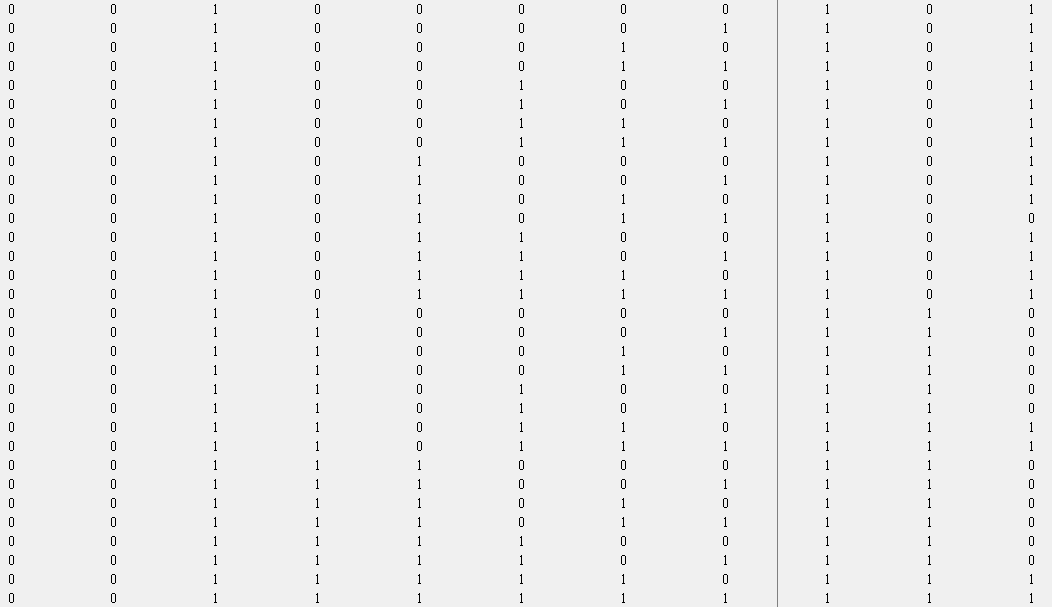
Connect clock and reset to each 4-bit BCD counter. The enable input is only fed to the first 4-bit BCD counter. The other enables are from ANDing the less-significant carryout bits. The last carryout bit on the most-significant digit is left unconnected. The outputs are connected to the output.

Circuit:

**4.2.7 Design a combinational circuit for state-transition of the Stopwatch control unit (8 points)**

Truth table:

A picture containing text, chain, metalware, several

Description automatically generatedA picture containing chain, metalware, swing

Description automatically generated

A picture containing text, chain, metalware, several

Description automatically generatedA picture containing chain, metalware, swing

Description automatically generatedA picture containing text, chain, metalware

Description automatically generated

A picture containing shape

Description automatically generatedA picture containing text, chain, metalware

Description automatically generated

Generated circuit by Logisim:

**Diagram, schematic

Description automatically generated**

**4.2.8 Design a combinational circuit for the output of the Stopwatch control unit (6 points)**

Table

Description automatically generatedTruth table:

Generated circuit by Logisim:**Diagram, schematic

Description automatically generated**

**4.2.9 Design a Stopwatch control unit (states are stored with D-Flip-Flops) (6 points)**

Explain your design method:

Every matching output is connected to one another, and the unmatched ones go into the registers

Circuit:

* Diagram, schematic

  Description automatically generated**Build the stopwatch circuit (6 points)**

Explain your design method:

Diagram

Description automatically generatedConnect The matching outputs together and have the stored value MUX’d with the BCD counter

Circuit:

**4.4 Demo your design to TA (8 points)** It doesn’t work