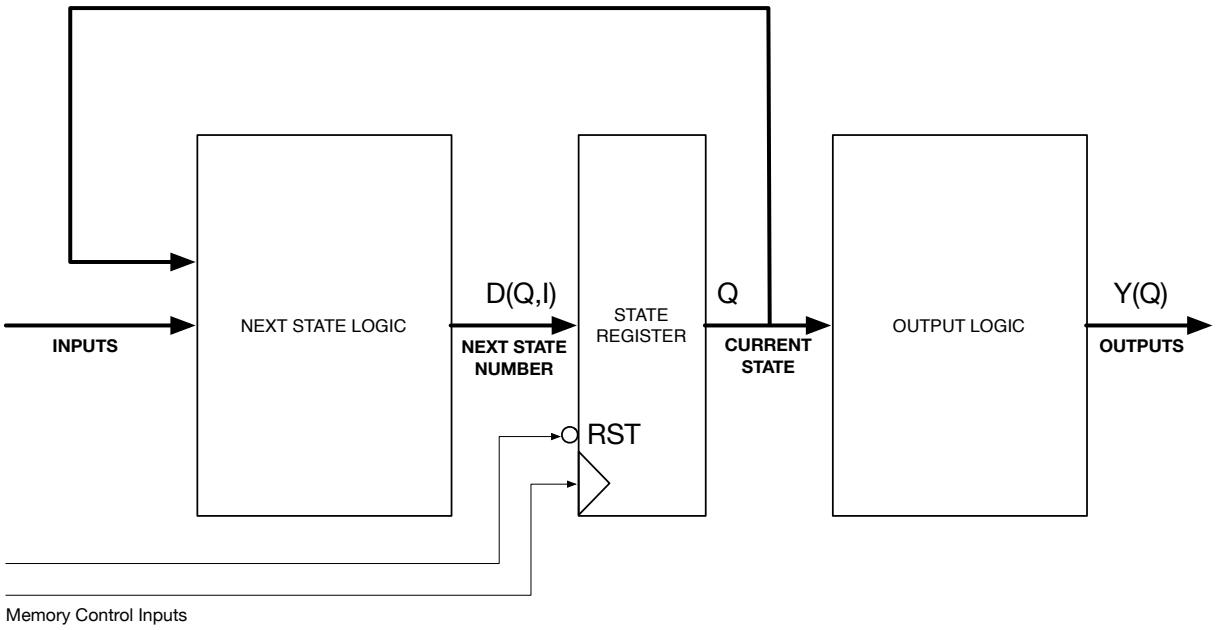
REVIEW MATERIAL

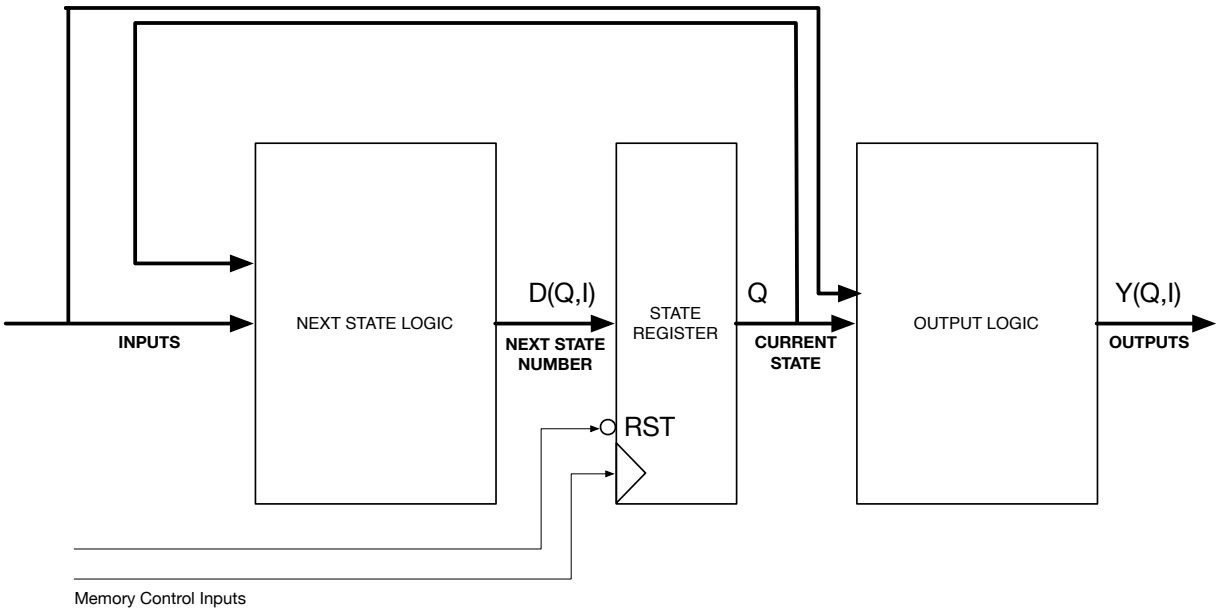
1. **Combinational circuits** do not have memory. They calculate instantaneous outputs based only on current inputs. They implement basic arithmetic and logic functions.
2. **Sequential circuits** create rich behavior by adding memory so that the circuit can have a historical record of input-output pairs through time.
3. **Machine state** is a number (or set of numbers) that represents the historical input-output transitions that have occurred since reset.
4. **State memory** stores and holds the **current machine state**.
5. **Clock signal edges** control memory storage and thus changes the stored machine state.
6. **State diagrams** allow engineers to pictorially model machine behavior as a first design step.
7. **State machines** are complete designs that implement state diagram machine behavior.
8. **Moore Machines** have outputs that are only functions of current machine state: Y(Q)

MOORE FINITE STATE MACHINE MODEL



9. **Mealy Machines** differ in one way: outputs depend on current machine state and inputs.

MEALY FINITE STATE MACHINE MODEL

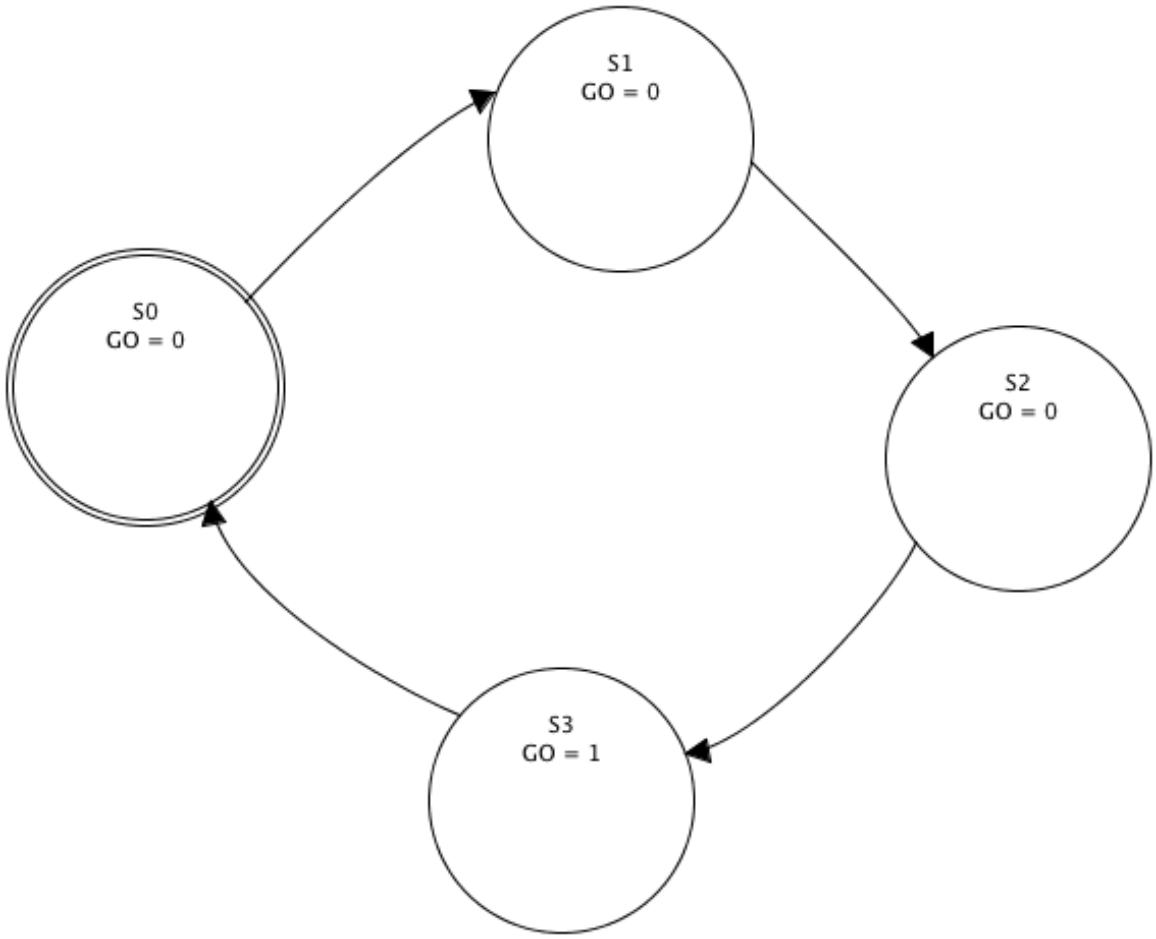


PAPER AND PENCIL DESIGN AND IMPLEMENTATION STEPS

1. **Create** state diagram.
2. **Assign** binary numbers to each state name. This will be the **stored machine state**.
3. **Create** truth tables for combinational logic blocks in the machine model: next state, output
4. **Minimize** the next state and output equations using K-maps or Boolean Algebra techniques.
5. **Draw** the circuit schematic on paper or in a CAD tool.
6. **Verify** the solution using either paper simulation or CAD tool simulation.
7. **Build** the physical circuit.
8. **Test** the physical circuit to ensure correct construction.

STEP 1: CREATE STATE DIAGRAM

Let's design and build a simple Moore machine that generates a logic 1 on an output called GO every four clock periods. These type of circuits are often used to control valves or solenoids in production plants, and a slightly more complex but similar circuit is used to time spark plug timing in modern automobiles. Because GO = 1 every four clock periods, a simple four-state state machine can be used. The machine is drawn as a state diagram where each state is identified by a name with the output values for that state shown below the state name. The clock edge controls memory sampling and thus changes memory state. These clock edges are shown on the diagram as arrows. In this machine, there are no inputs besides the clock. The **reset state** is shown with a double-ring.



STEP 2: ASSIGN BINARY NUMBERS TO EACH STATE NAME

Four states require two state memory bits because $\log_2 4 = 2$. Thus four binary numbers — one for each state — will be stored through time as the machine progresses through its sequence. Let's assign binary number 00 to S0, binary number 01 to S1, binary number 10 to S2, and binary number 11 to S3. This is called **standard binary encoding** because the state names suggest the encoded stored value. **Remember** that stored values are the Q outputs of flip flops.

STATE NAME	Q1	Q0
S0	0	0
S1	0	1
S2	1	0
S3	1	1

STEPS 3 AND 4: CREATE TRUTH TABLES AND EQUATIONS

Let's do the output combinational logic block first. **Remember** that the output logic depends only on the current stored state: GO(Q).

	INPUTS		OUTPUT
STATE NAME	Q1	Q0	GO
S0	0	0	0
S1	0	1	0
S2	1	0	0
S3	1	1	1

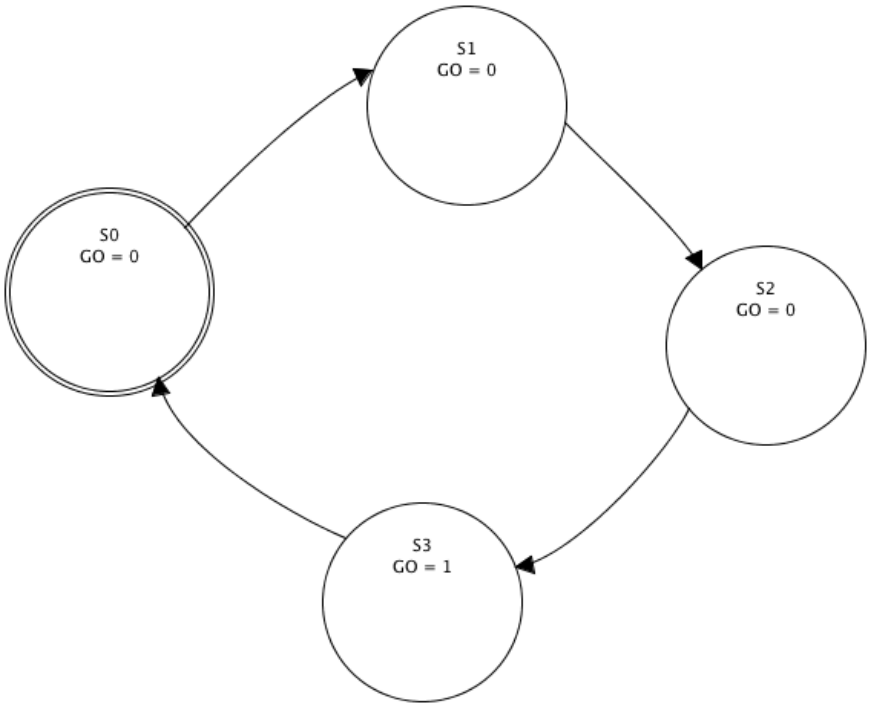
The solution to the equation doesn't require Boolean algebra or K-maps because there is only one minterm:

$GO(Q) = \sum m(3)$

$GO(Q) = Q1 \cdot Q0$

Now repeat the process for the other combinational logic block — the block creating the next state equations D1 and D0 — one data input for each memory flip flop. This block is always a little more complicated because it requires any additional machine inputs and it also requires us to keep **current state stored in the memory (Q)** separate in our minds from **next state to be transitioned to and stored (D).**

Let's redraw the state machine diagram.



Also **remember** that the data inputs are functions of Q and I: D(Q,I).
In this machine there no other inputs.

	INPUTS		OUTPUTS	
STATE TRANSITION	**Q1**	**Q0**	**D1**	**D0**
S0 —> S1	0	0	0	1
S1 —> S2	0	1	1	0
S2 —> S3	1	0	1	1
S3 —> S0	1	1	0	0

The truth table shows the **current state feedback signal inputs** Q1 and Q0.
The truth table shows the **next state outputs** D1 and D0.
The standard binary encodings are complete for current and next state values.

The K-maps for D1 and D0 and minimized equations are:

Q1 \ Q0	0	1
0	0	1
1	1	0

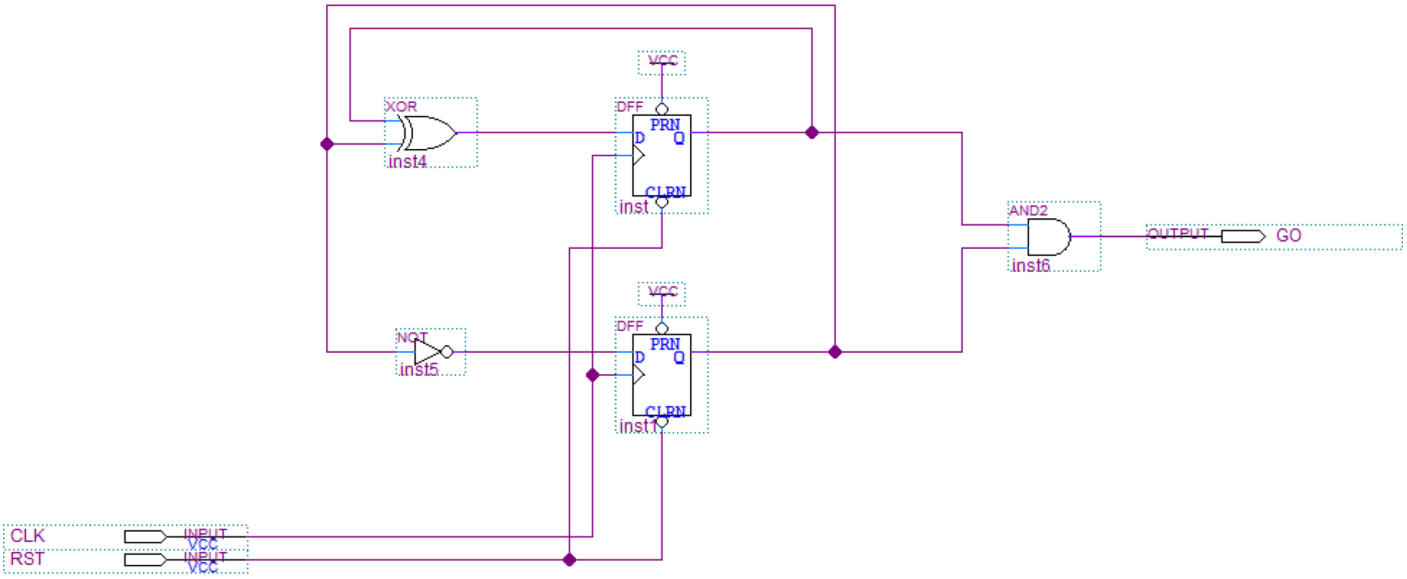
D1 = Q1 xor Q0

Q1 \ Q0	0	1
0	1	0
1	1	0

D0 = Q0_

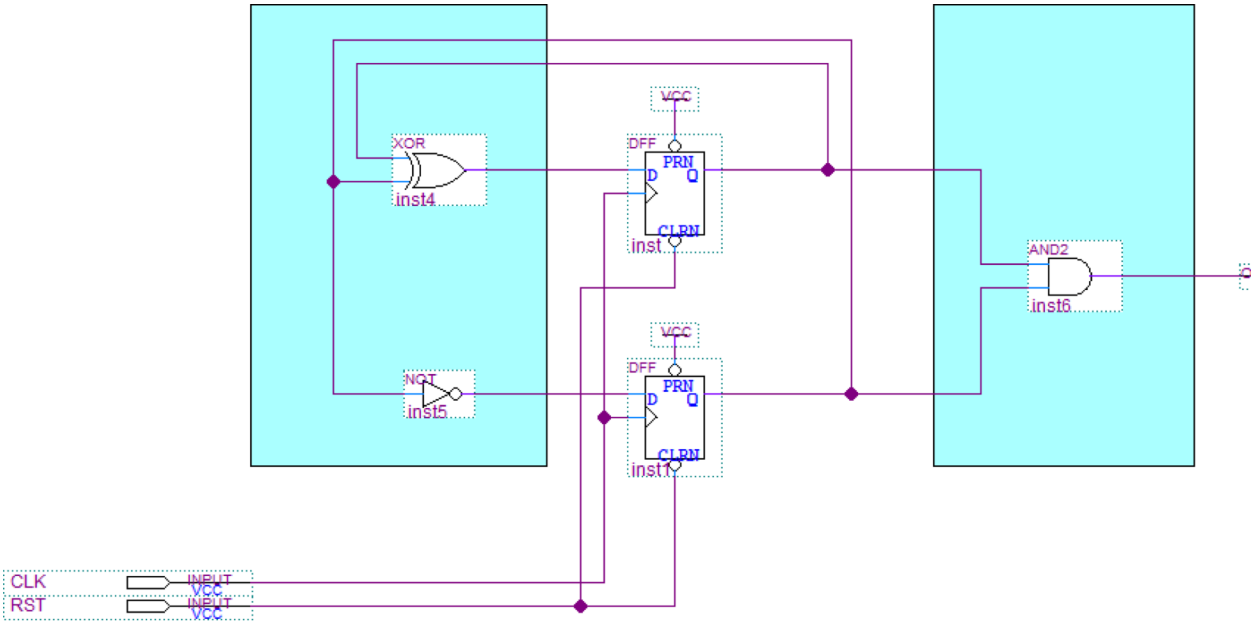
STEP 5: **DRAW THE CIRCUIT ON PAPER OR IN A CAD TOOL**

Let's use Quartus as the CAD tool for schematic and simulation. The Quartus schematic is shown below. **Note** that the D flip flop element has both preset and reset signals. Preset is rarely used and most often the preset signals are simply tied to the non-active level to ensure that preset never occurs. In this case, bubble symbology shows preset active with logic-0 so the presets are tied to the non-active voltage VCC. Standard practice also has the flip flops organized with Q1 at the top of the diagram followed by Q0 below it.

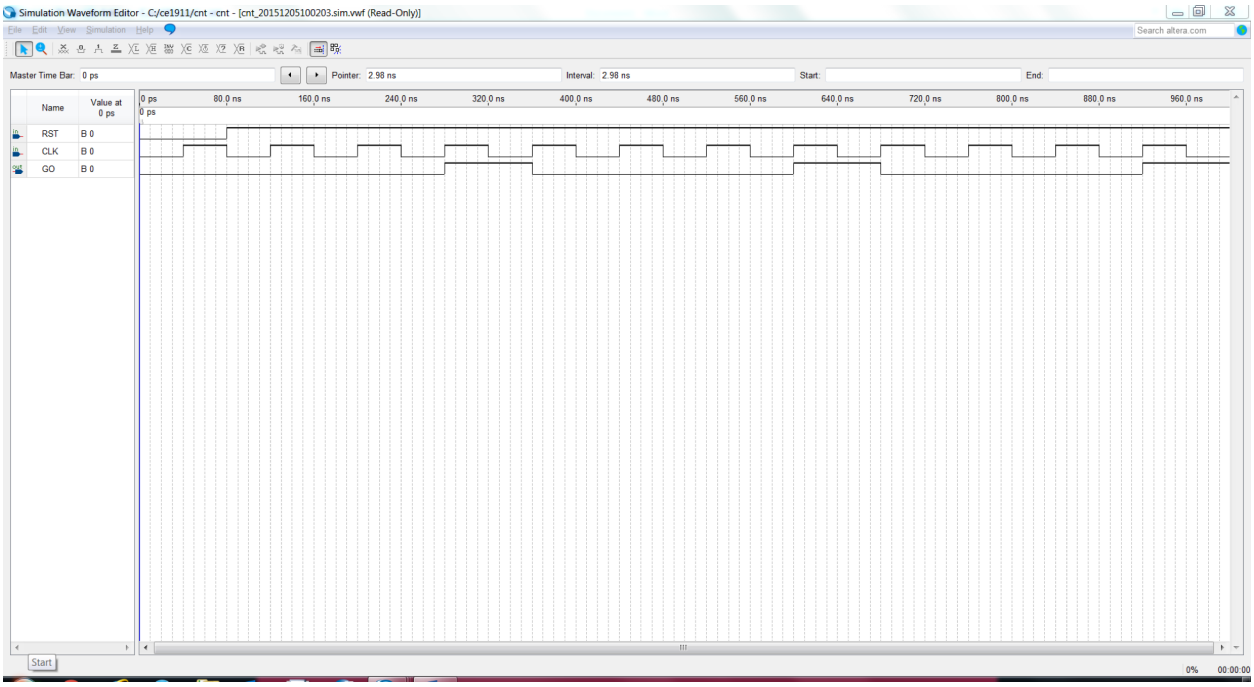


The Moore Machine Model is nicely implemented. The next state combinational logic is in place on the left hand side of the diagram. The state memory is in the middle. The output combinational logic is on the right hand side of the diagram. The common clock and reset signals are attached.

Here is the circuit again with the combinational logic blocks highlighted to make them more visible.

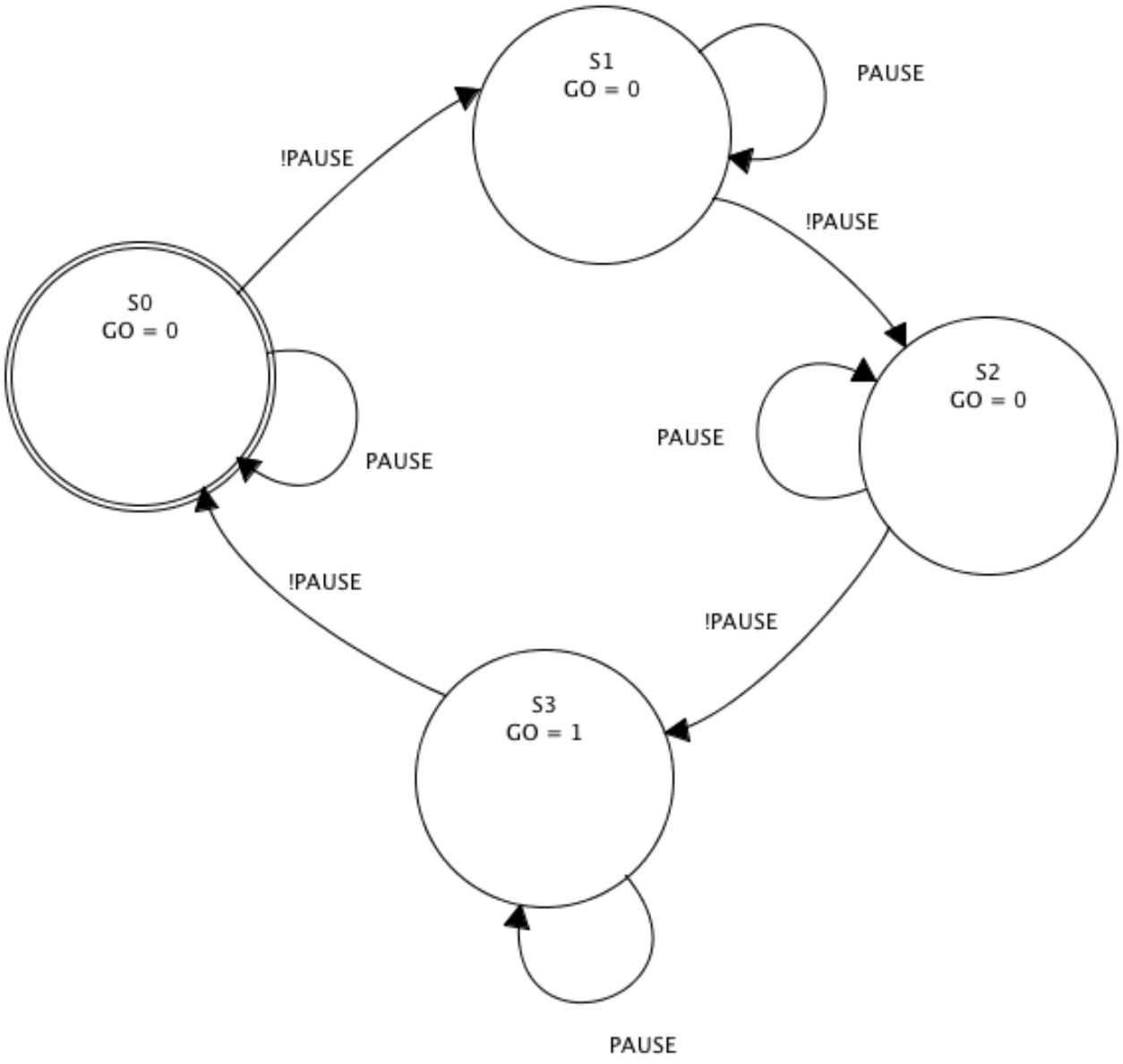


Simulation in Quartus follows standard practice. The inputs RST and CLK are added along with the output GO. For simple machines that have no additional inputs, I usually run simulation for two to three complete machine cycles so that the periodic sequential behavior can be visualized and verified — in this case, a pulse should occur for every clock period when the machine is in state S3. **Note** how simulation is started by **resetting** the machine into state S0 = 00. I typically bring reset back high in the middle of the first clock period**.** This allows the machine to make its first transition on the next rising edge.



This completes the first machine design and simulation. Now, let's modify the machine so that we can pause the sequence on command. Let's add an input called PAUSE and as long as PAUSE = 1 the machine repeats the current state as the next state.

STEP 1: CREATE STATE DIAGRAM



The state diagram shows the new control input on each arrow. The control input determines which arrow gets followed. Each state must have $\mathbf{2^n}$ arrows leaving it where n is the number of inputs. Our previous circuit had no control input and thus $\mathbf{2^0 = 1}$ arrow leaving each state (representing the clock edge transition). Now, the circuit has one control arrow and $\mathbf{2^1 = 2}$ arrows must leave each state: one clock transition arrow for each possible input combination.

STEP 2: ASSIGN BINARY NUMBERS TO EACH STATE NAME

Let's use standard binary encoding again.

STATE NAME	Q1	Q0
S0	0	0
S1	0	1
S2	1	0
S3	1	1

STEPS 3 AND 4: CREATE TRUTH TABLES AND EQUATIONS

The output only depends on current state. Thus the addition of an input does not change the output equation for GO. There is no need to repeat the truth table.

GO(Q) = Q1·Q0

But, the addition of an input **does change the combinational logic block creating the next state voltages**. Now, each D equation is a function of current state Q and the input PAUSE.

D(Q,PAUSE)

Let's take a look at the new truth table. **Remember** that in the state diagram PAUSE means PAUSE = 1 and !PAUSE means PAUSE = 0.

	INPUTS			OUTPUTS	
STATE TRANSITION	Q1	Q0	PAUSE	D1	D0
S0 —> S1	0	0	0	0	1
S0 —> S0	0	0	1	0	0
S1 —> S2	0	1	0	1	0
S1 —> S1	0	1	1	0	1
S2 —> S3	1	0	0	1	1
S2 —> S2	1	0	1	1	0
S3 —> S0	1	1	0	0	0
S3 —> S3	1	1	1	1	1

This table is more complex of course because now each state has two transitions leaving it. The inputs to the combinational logic block are arranged with Q1 and Q0 as most significant bits.

This keeps the states together in the table as the input changes in the least significant bit of the input number line.

The K-maps and equations are:

Q0 PAUSE

Q1 \	00	01	11	10
0	0	0	0	1
1	1	1	1	0

Q0 PAUSE

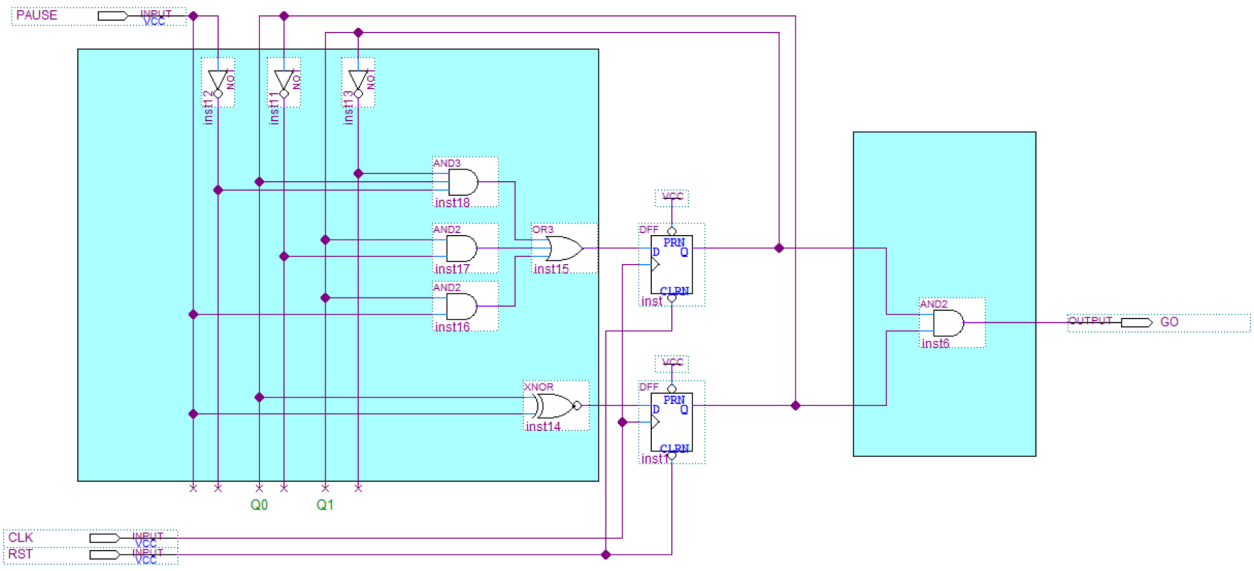
Q1 \	00	01	11	10
0	1	0	1	0
1	1	0	1	0

$$D1 = \overline{Q1} \bullet Q0 \bullet \overline{PAUSE} + Q1 \bullet \overline{Q0} + Q1 \bullet PAUSE$$

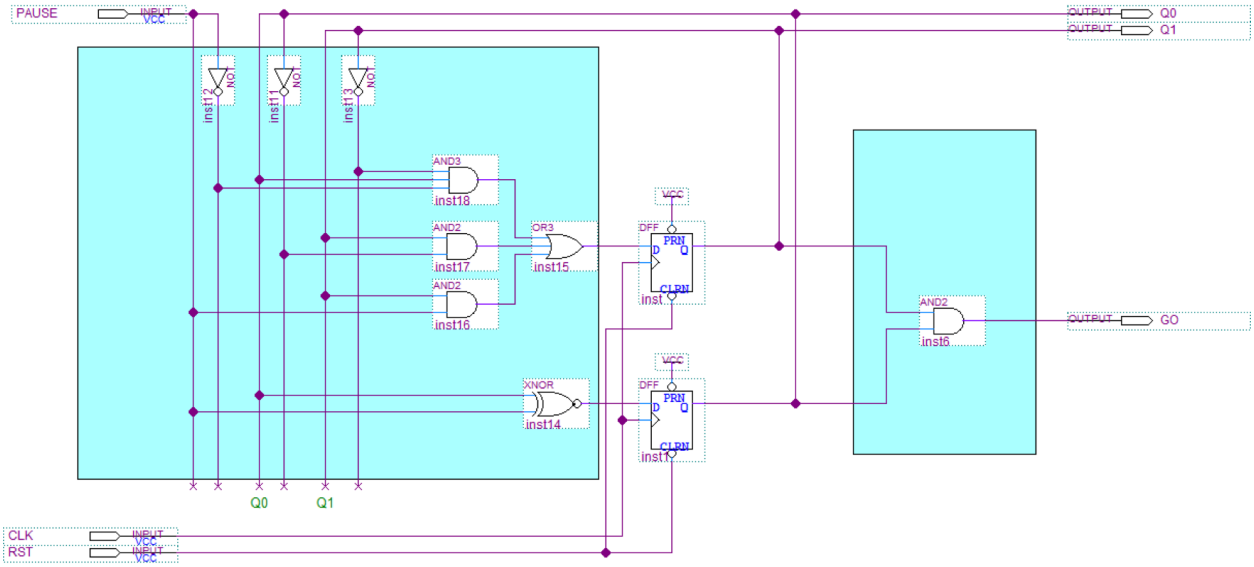
$$D0 = \overline{Q0} \bullet \overline{PAUSE} + Q0 \bullet PAUSE = Q0 \; xnor \; PAUSE$$

STEP 5: **DRAW THE CIRCUIT ON PAPER OR IN A CAD TOOL**

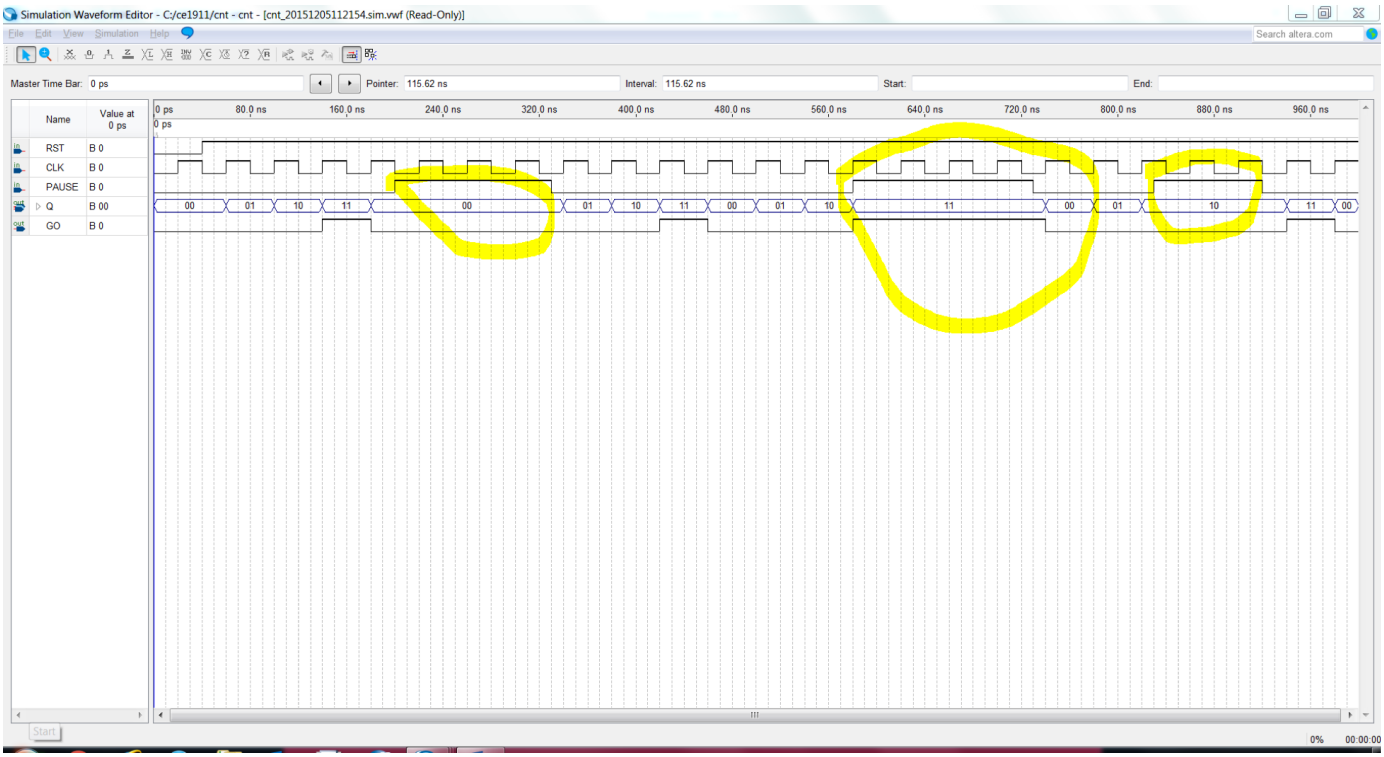
Let's use Quartus as the CAD tool for schematic and simulation. The Quartus schematic is shown below. **Note** that the D flip flop element has both preset and reset signals. Preset is rarely used and most often the preset signals are simply tied to the non-active level to ensure that preset never occurs. In this case, bubble symbology shows preset active with logic-0 so the presets are tied to the non-active voltage VCC. Standard practice also has the flip flops organized with Q1 at the top of the diagram followed by Q0 below it.



Simulation of the circuit is a little more complex. It would be good to see the state voltages in order to verify the “hold” behavior. Thus, I’ve added two additional machine outputs and tapped the state voltages.



Now, the simulation waveforms include the memory control signals RST and CLK, as well as the transition control signal PAUSE. Note the ordering: RST, CLK, PAUSE, Q, GO. This is very standard for state machine timing diagrams and the expectation is that you will follow it in your work. I’ve highlighted the pause segments and indeed the machine holds in the state that is paused.



HOMEWORK

Now, complete this homework as a Microsoft Word document and email to me in PDF format by the end of the next academic day. **Use** the Print—> CutePDF Printer option to convert Microsoft Word to PDF.

1. **Redesign** the machine so that the output GO=1 in states S0 and S1 and not in states S2 and S3. Also, modify the machine so that the pause behavior occurs only in states S2 and S3 and never occurs in states S0 and S1. **You do not have to draw the new state diagram**. Instead, begin by creating the truth tables for the output logic GO(Q) and the next state combinational logic equations D(Q,PAUSE).
2. **Draw** and **simulate** your solution in Quartus. **Ensure** that your simulation tests both pause and non-pause periods of time. Your simulation should also show pause only occurring in states S2 and S3 and not occurring in states S0 and S1. **Use** multiple simulations if that makes showing your verification easier.
3. **Note** that your Microsoft Word document should include truth tables (use insert table), K-map tables and the derived equations (try to show K-map groups by shading table cells) for D1, D0, and GO. The document should also include the Quartus schematic and simulation waveform diagrams. **Review** the “Printing Simulation Diagram” information on the course website under the Design Software link if needed.