

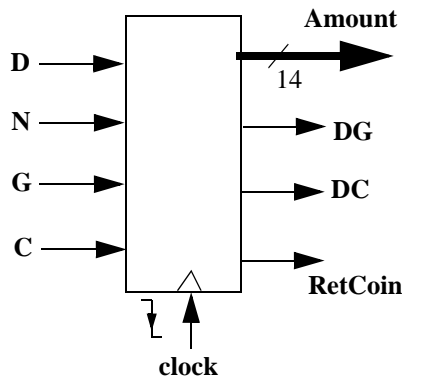
DIGITAL SYSTEM DESIGN EXAMPLE

A Vending Machine Controller

We will design a vending machine controller as a digital system by using the finite state machine (FSM) technique. The design is done through seven steps as discussed in class and in the lab.

1) The black-box view and the textual input/output relationship

A vending machine, by means of its controller, delivers gums and chips, both costing 35 cents. One can input only Dimes (10 cents) and Nickels (5 cents). There is **no** Coin Return button.



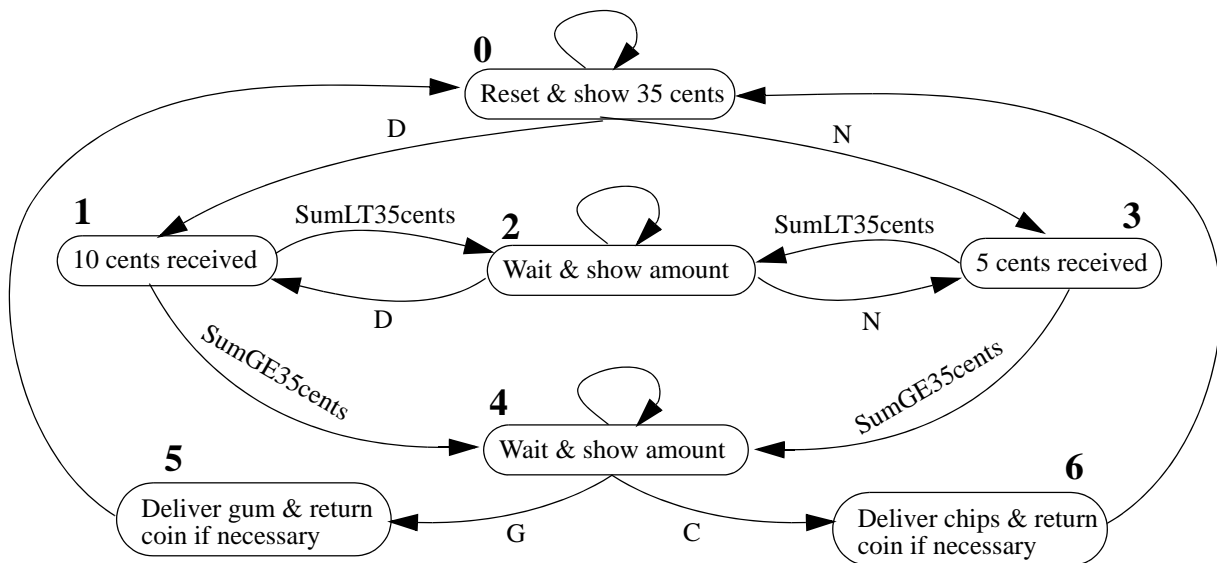
Inputs
D : Dime is input
N : Nickel is input
G : Gum is selected
C : Chips is selected

Outputs
Amount shown on two 7-segment display :
 1) Value of gum and chips (35 cents) or
 2) The coin input so far
DG : Deliver Gum
DC : Deliver chips
RetCoin : Return 5 cents

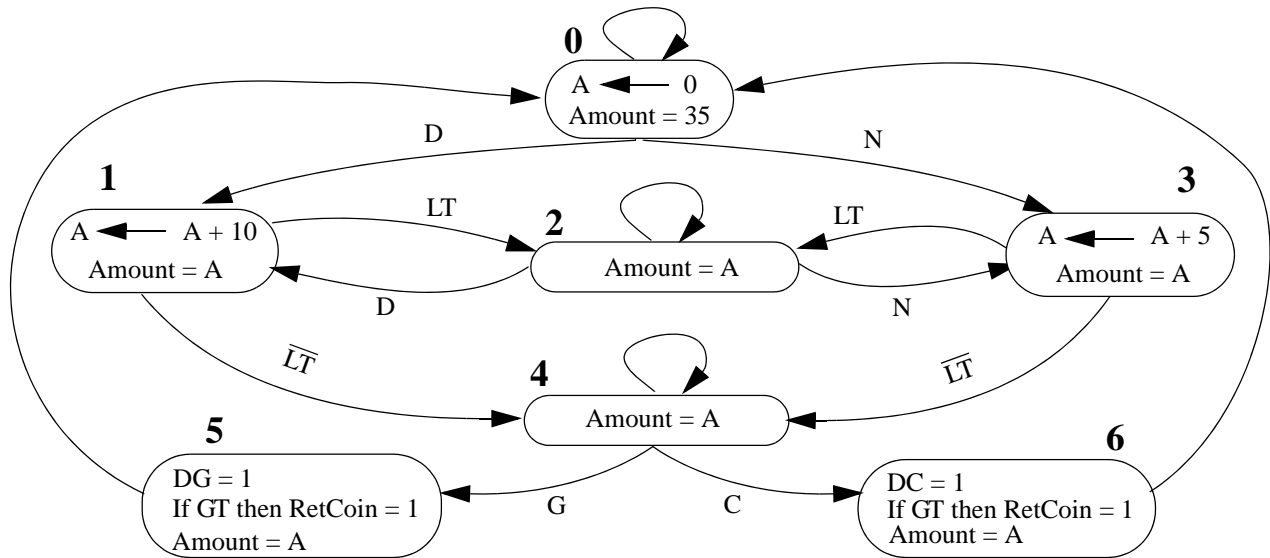
Textual Input/Output Relationship

After receiving the necessary amount (35 cents or 40 cents) and the selection is made, deliver gum or chips and if necessary return 5 cents

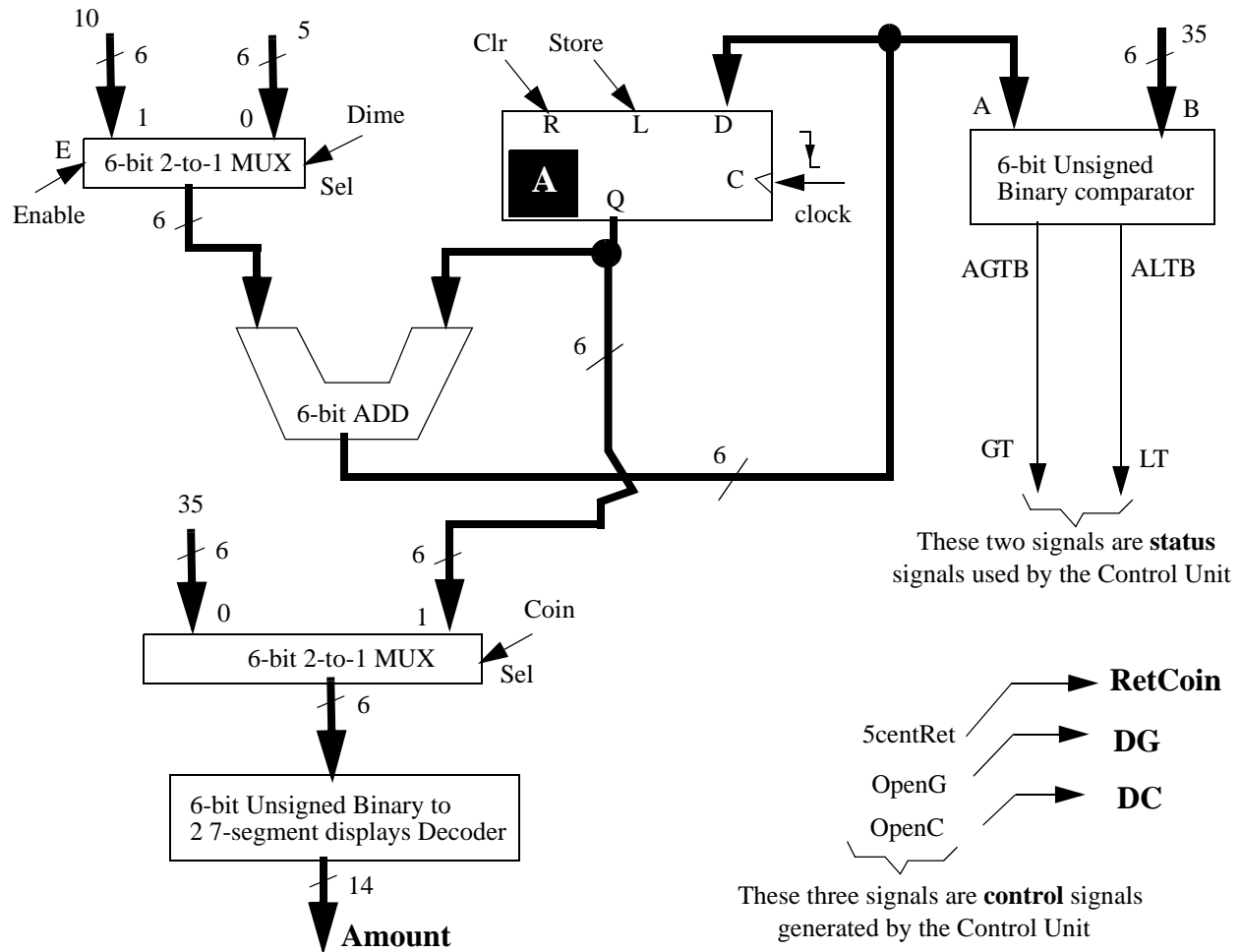
2) Obtain the operation diagram from the black-box view and the textual input/output relationship



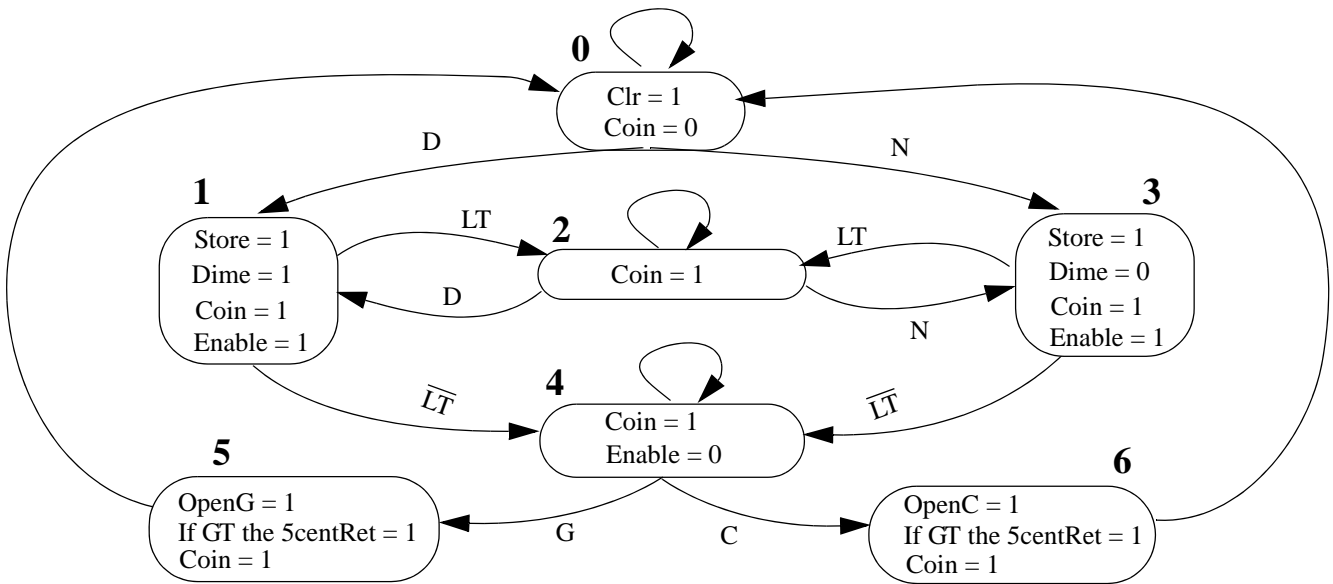
3) Obtain the high-level state diagram from the operation diagram



4) Obtain the datapath from the high-level state diagram

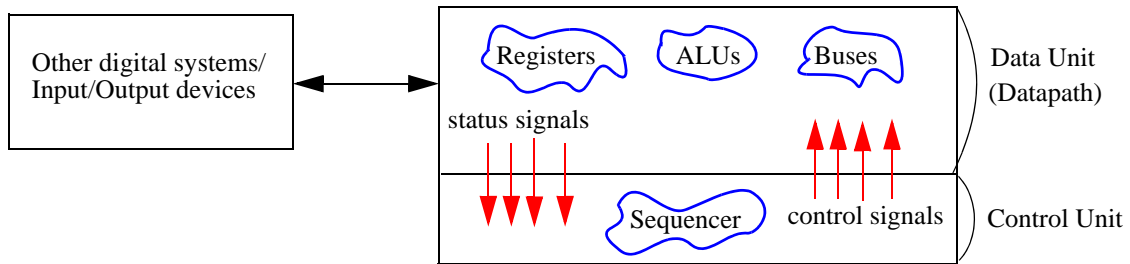


5) Obtain the low-level state diagram from the high-level state diagram and datapath

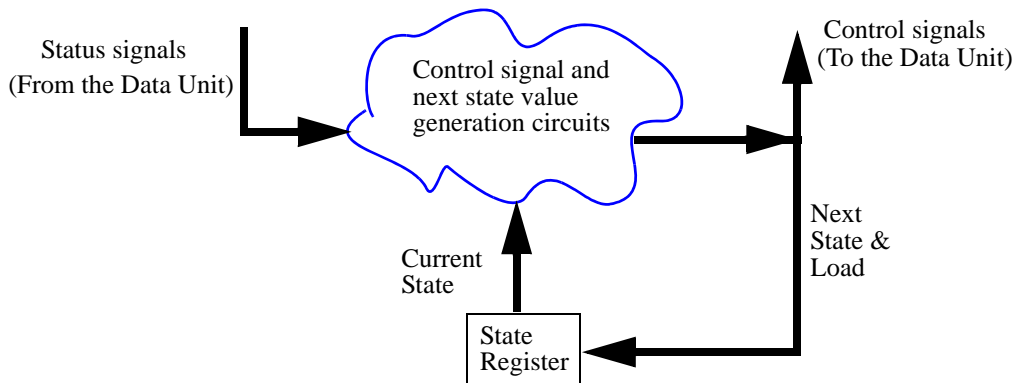


6) Decide about hardwiring versus microprogramming

The control unit is one of the two major parts of a digital system :

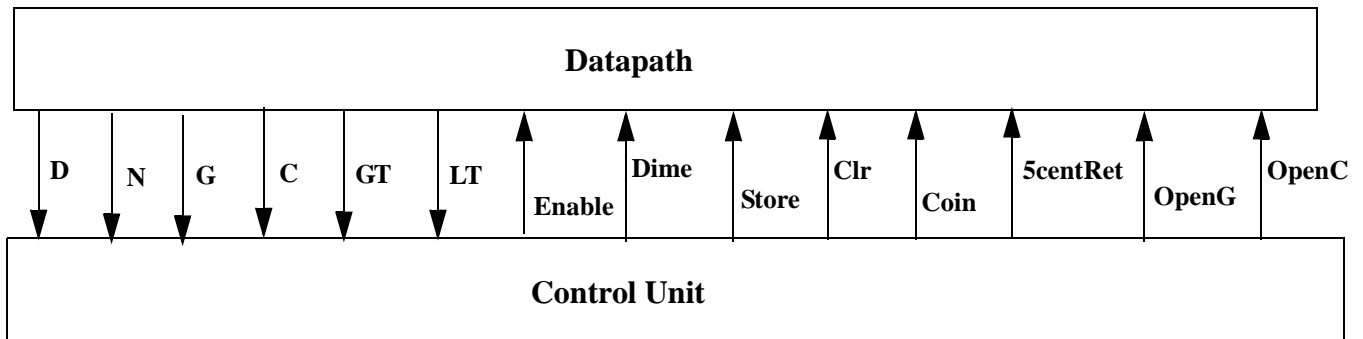


We will decide whether we will use hardwiring or microprogramming to implement the control unit !

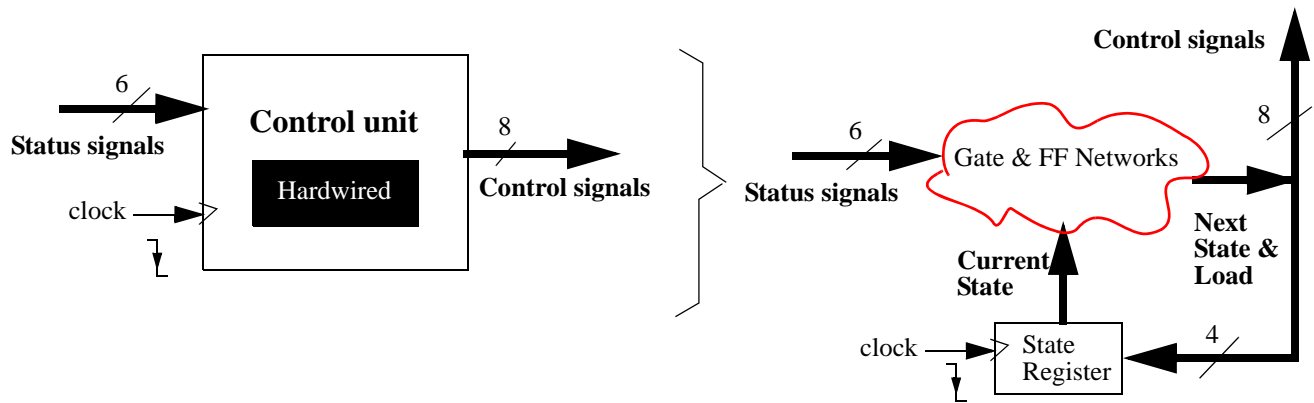


A **Load** signal needs to be generated to load the state register when needed, i.e. in a specific clock period.

We observe that the control unit has **six** status signals and **eight** control signals :

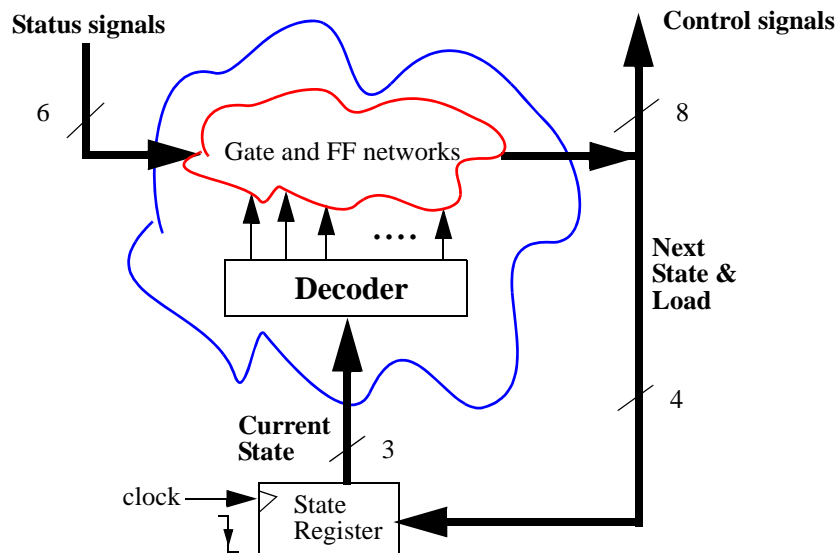


Since the vending machine controller is a simple digital system, we decide to use hardwiring.

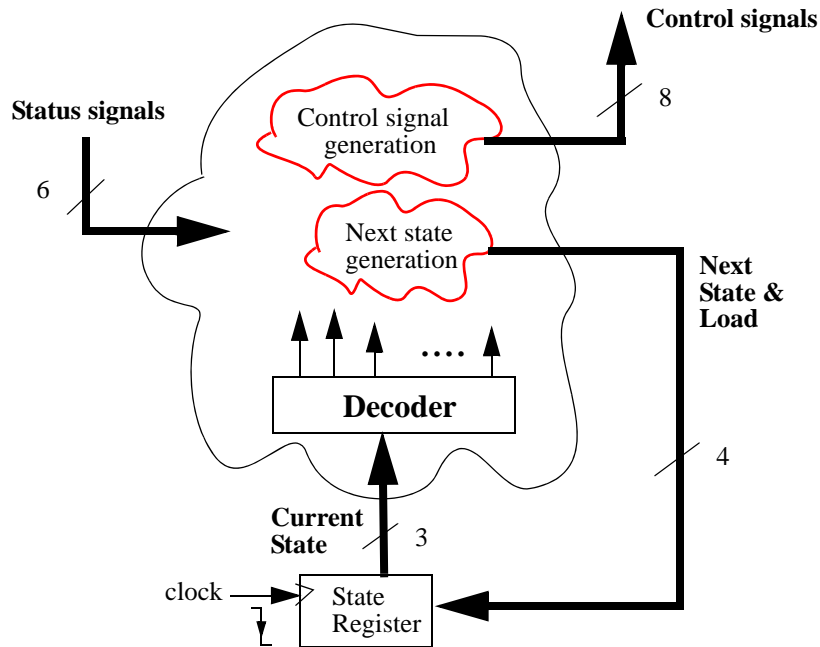


7) Implement the control unit based on the low-level state diagram and using hardwiring

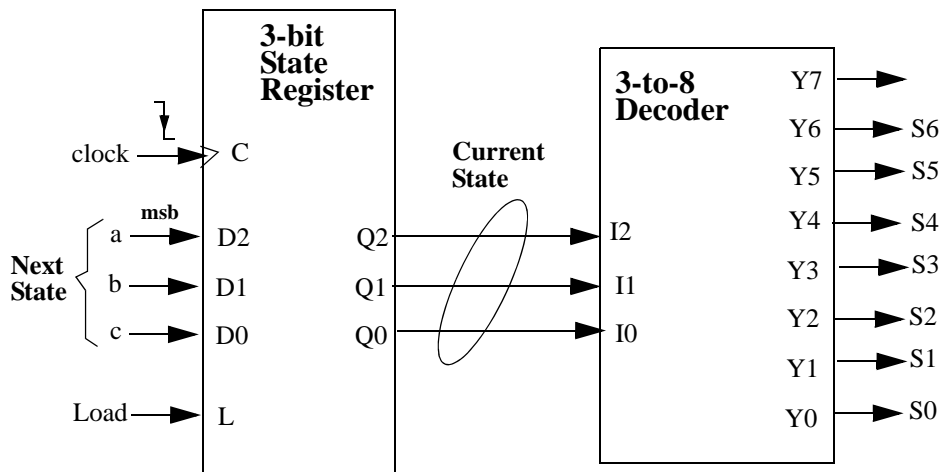
a) We separate the **decoder** from the other logic in the cloud :



b) We partition the cloud of gate and FF networks to two parts :



c) We have to decide if we want to use a counter or a register to keep track of the state. The decision will depend on the state transitions. We observe that we do not trace the states sequentially as 0, 1, 2, 3,... We trace them by skipping states often. Therefore, it will **not** help us if we use a counter. We will use a register. It will be a **3-bit register** since we have seven states. Since it is a 3-bit register, the decoder is a 3-to-8 decoder :



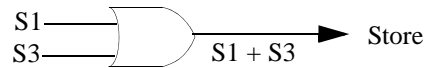
A **Load** input is needed to load a value to the state register when needed, i.e. in a specific clock period.

d) We will generate **control signals** for which we will use gate networks. In order to get the gate networks, we have to obtain expressions :

Control Signal Generation

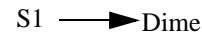
Store is 1 when it is state 1 or 3

$$\text{Store} = S1 + S3$$



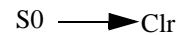
Dime is 1 when it is state 1

$$\text{Dime} = S1$$



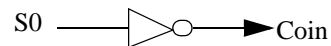
Clr is 1 when it is state 0

$$\text{Clr} = S0$$



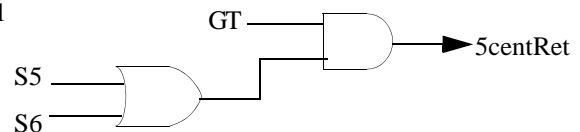
Coin is 1 when it is state 1 or 2 or 3 or 4

$$\begin{aligned} \text{Coin} &= \overline{S1 + S2 + S3 + S4 + S5 + S6} \\ &= \overline{S0} \end{aligned}$$



5centRet is 1 when it is state 5 and Gt is 1 or state 6 and GT is 1

$$\text{5centRet} = (S5 + S6)GT$$



OpenG is 1 when it is state 5

$$\text{OpenG} = S5$$



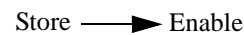
OpenC is 1 when it is state 6

$$\text{OpenC} = S6$$



Enable is 1 when it is state 1 or 3, same as Store !

$$\text{Enable} = \text{Store}$$

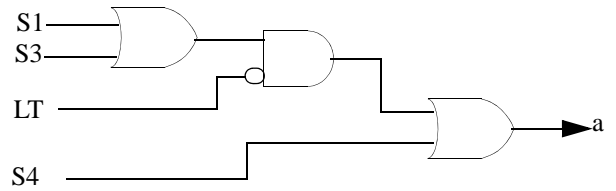


e) Finally, we will generate the **next state signals** for which we will use gate networks. In order to get the gate networks, we have to obtain expressions :

Next State Generation

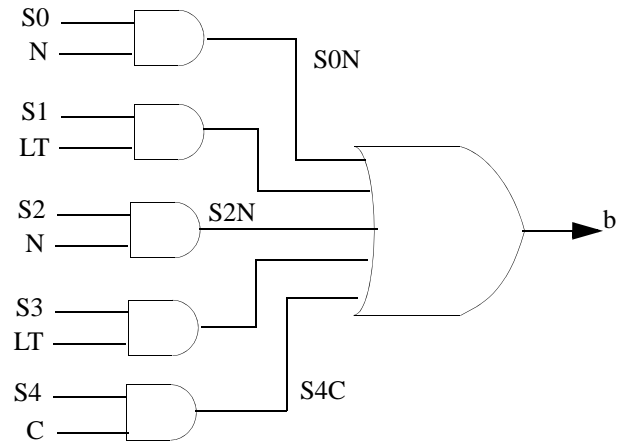
a is 1 when it is state 1 and LT is 0 or state 3 and LT is 0 or state 4

$$a = (S1 + S3)\overline{LT} + S4$$



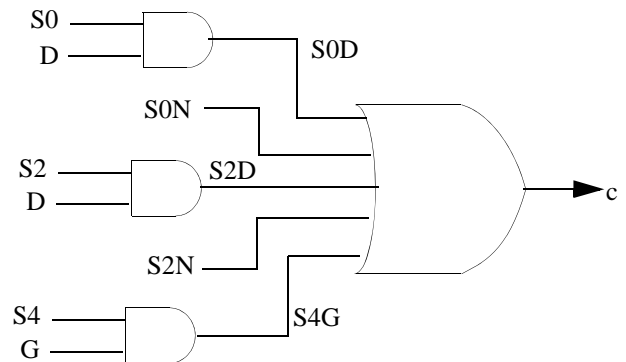
b is 1 when it is state 0 and N is 1 or state 1 and LT is 1 or state 2 and N is 1 or state 3 and LT is 1 or state 4 and C is 1

$$b = S0N + S1LT + S2N + S3LT + S4C$$



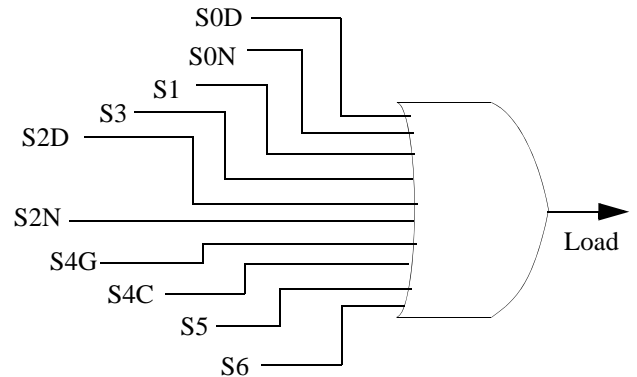
c is 1 when it is state 0 and D is 1 or state 0 and N is 1 or state 2 and D is 1 or state 2 and N is 1 or state 4 and G is 1

$$c = S0D + S0N + S2D + S2N + S4G$$



Load is 1 when it is state 0 and D is 1 or State 0 and N is 1 or state 1 or state 3 or state 2 and D is 1 or state 2 and N is 1 or state 4 and G is 1 or state 4 and C is 1 or state 5 or state 6

$$\text{Load} = S0D + S0N + S1 + S3 + S2D + S2N + S4G + S4C + S5 + S6$$



The full design is as follows :

