State Machines

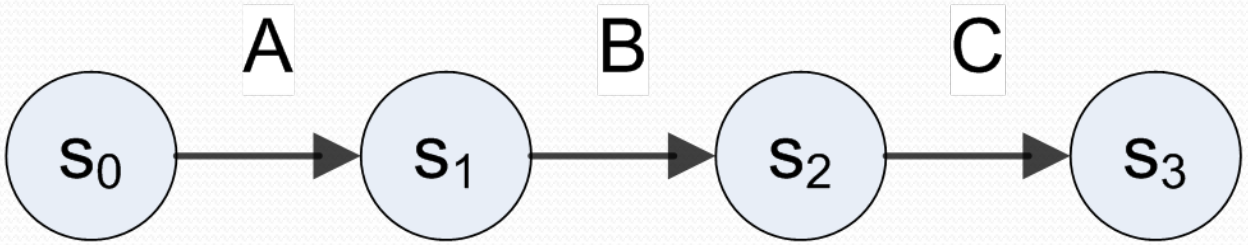
Chia-Tien Dan Lo
Computer Science and Software Engineering
Southern Polytechnic State University

A System Modeling Technique

- State machines provide a way to describe a system's behavior.
- States and their transitions wrt events.
- An important and frequently used technique in CS and other fields
  - Language
  - Compiler
  - Computer Architecture
  - Operating Systems
  - Software Engineering
  - Computation Theory
  - Control Systems
  - …

State Transition System

- An abstract machine to study computation
  - A set of states
  - Transitions between states
- Labeled transitions
  - Inputs
  - Conditions
  - Actions
- A state machine that recognizes “ABC”



Formally

- A state transition system is a triple

(Q, Σ, δ) where

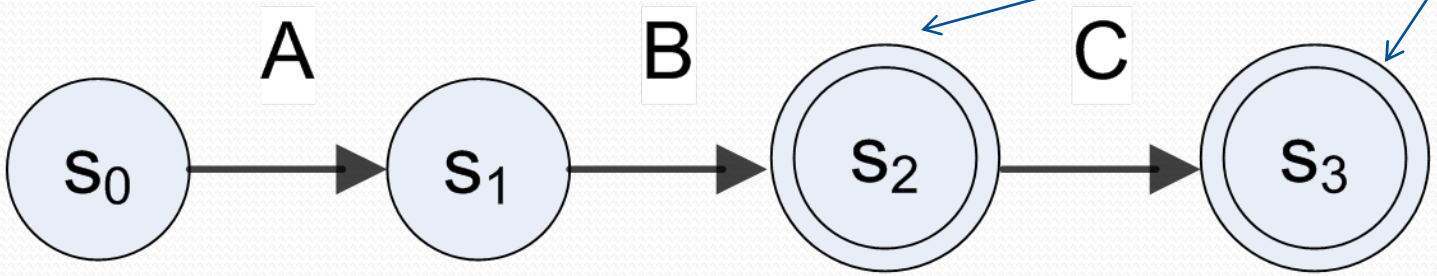
Q : a set of states

Σ : a set of symbols, called alphabet

$\delta : Q \times \Sigma \rightarrow Q$

Finite Automata

- State machines with finite number of states are called finite state machines.
- There are 4 states in the previous example.
- What if the system is in S3, …
- We need to output something at S3.
  - It is called finite state or accepting state.
- More then one finite states is possible!



Deterministic Finite Automata

- Give an input, the DFA will transfer to "another" state.
  - Don't have choice!
- Formally,

$A = (Q, \Sigma, \delta, q_0, \mathrm{F})$ where

a. Q is a finite set of states,

b. Σ is a finite set of labels or symbols,

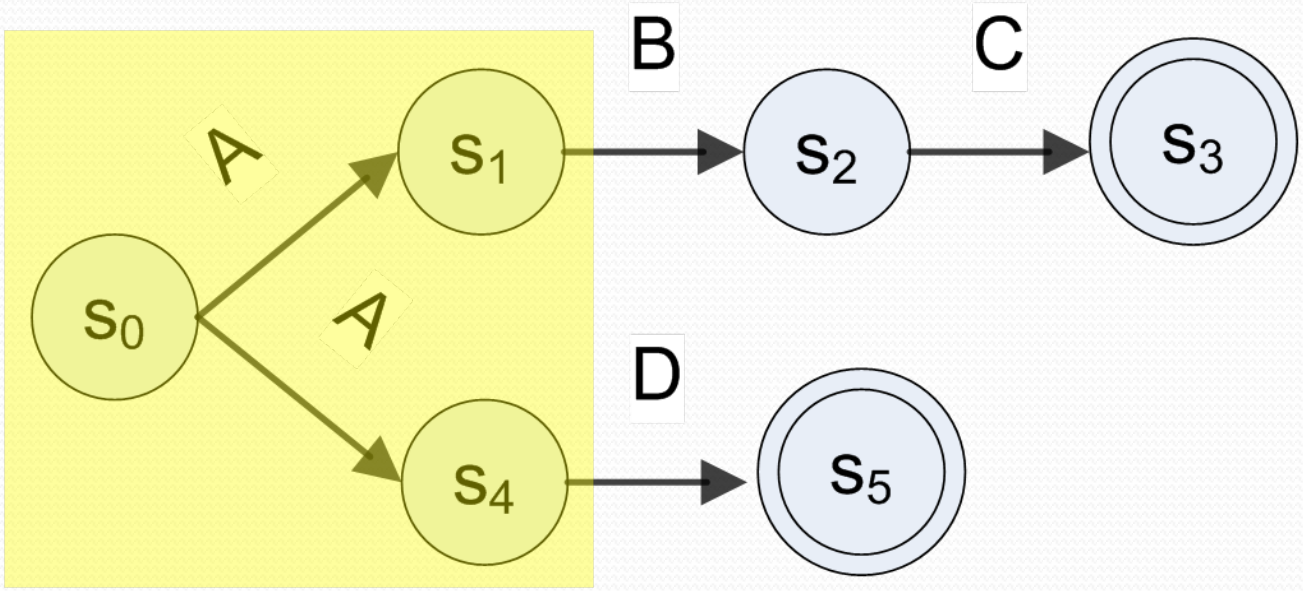
c. δ is a transition function, $\delta: Q \times \Sigma \longrightarrow Q$,

d. $q_0 \in Q$ is an initial state, and

e. $F \subseteq Q$ is a set of finite or accepting states.

Nondeterministic Finite Automata

- Give an input, the NFA will transfer to a set of states.
  - Have choices!
- E.g., recognize “ABC” and “AD”

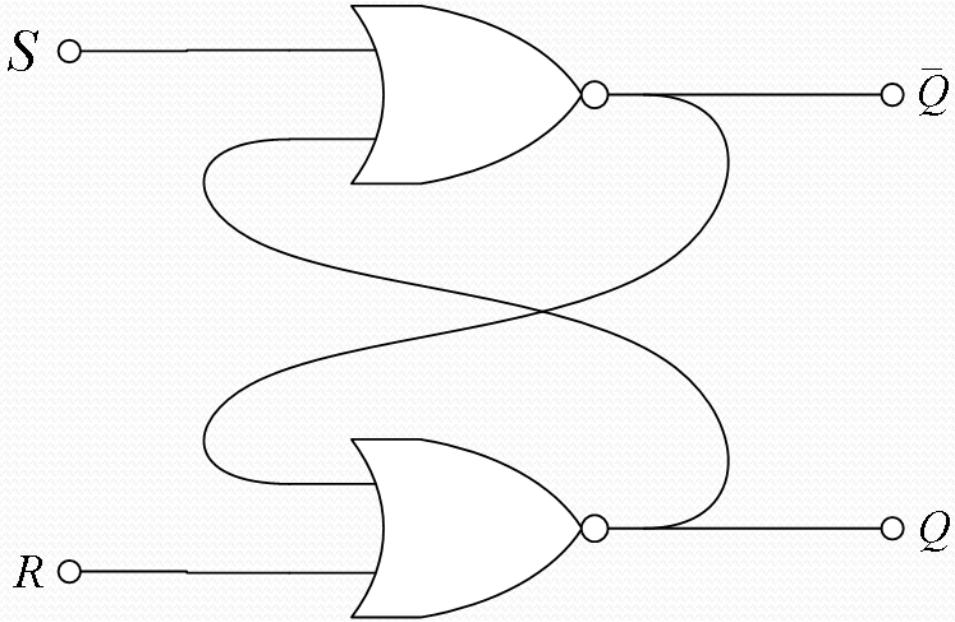


Latch

- Hardware implementation for FAs requires state transition logic, and a state variable tracking current states.
- State variable is a register which is based on latches.
- A latch is a FA with two stable states, able to keep one bit information.
- No clock involved (flip-flops involve clocks)

SR NOR Latch

- Outputs depend on both inputs and its state



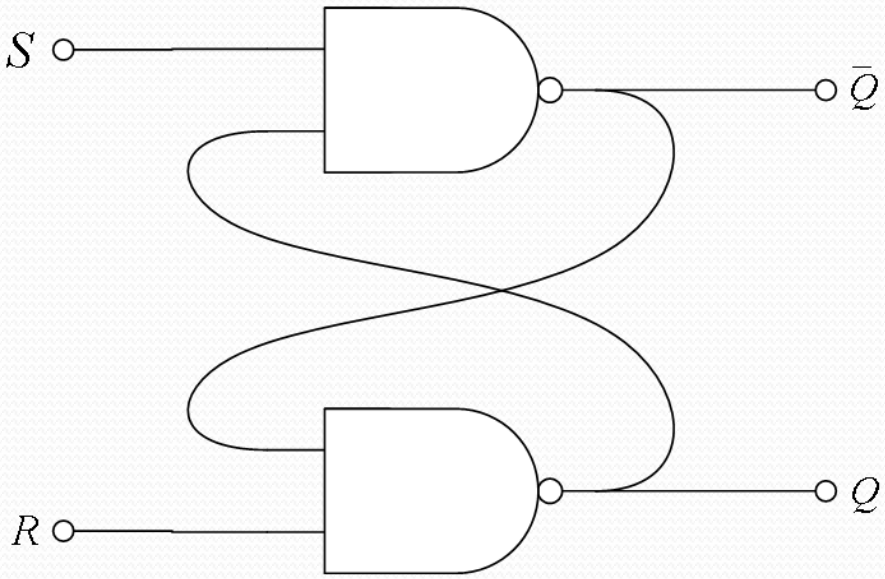
Next value

S	R	Q'
0	0	Q
0	1	0
1	0	1
1	1	Restricted

Both outputs are o

SR NAND Latch

- NAND gates may be cheaper in some cases

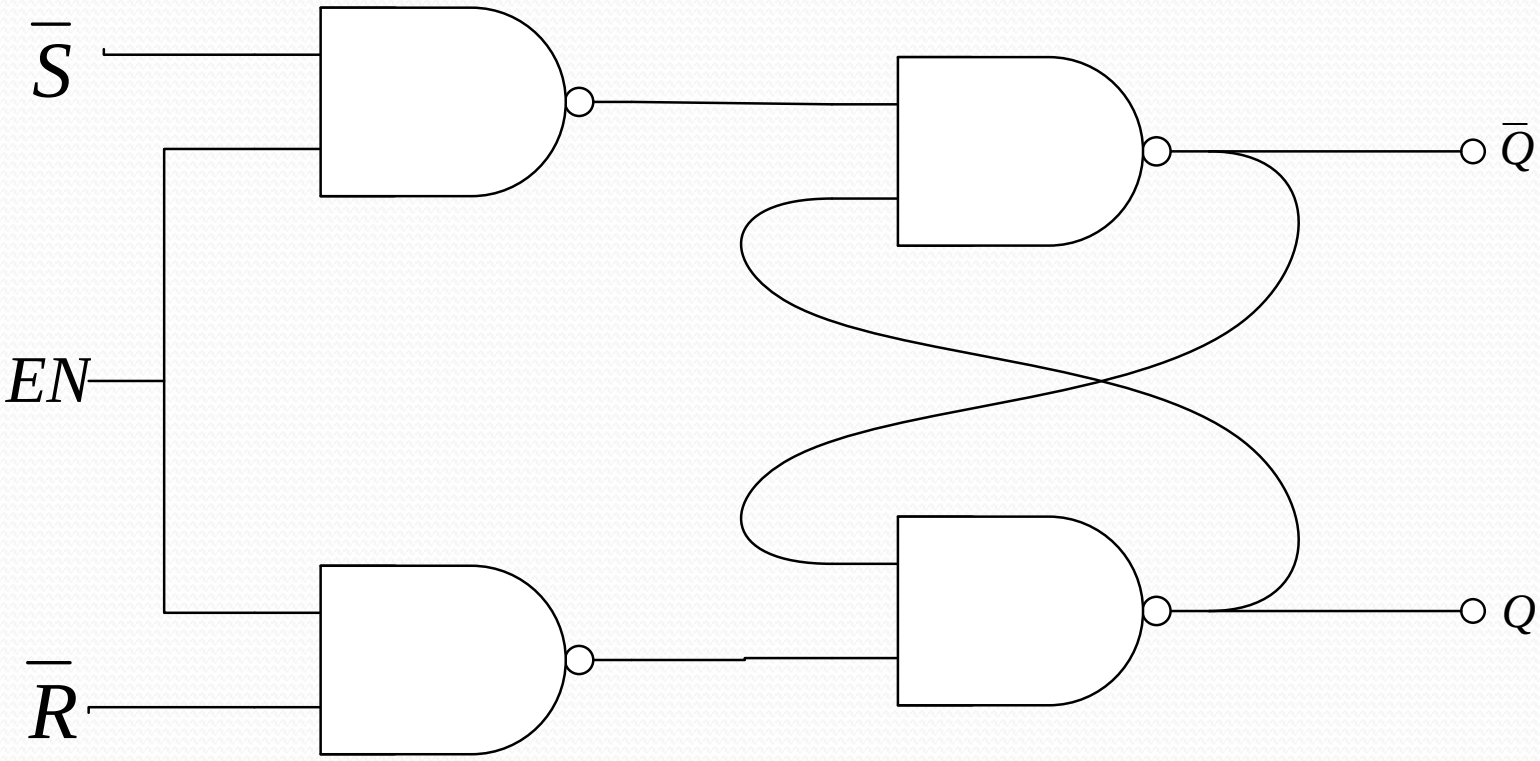


S	R	Q'
0	0	Restricted
0	1	0
1	0	1
1	1	Q

Do you see differences between this one the NOR one?

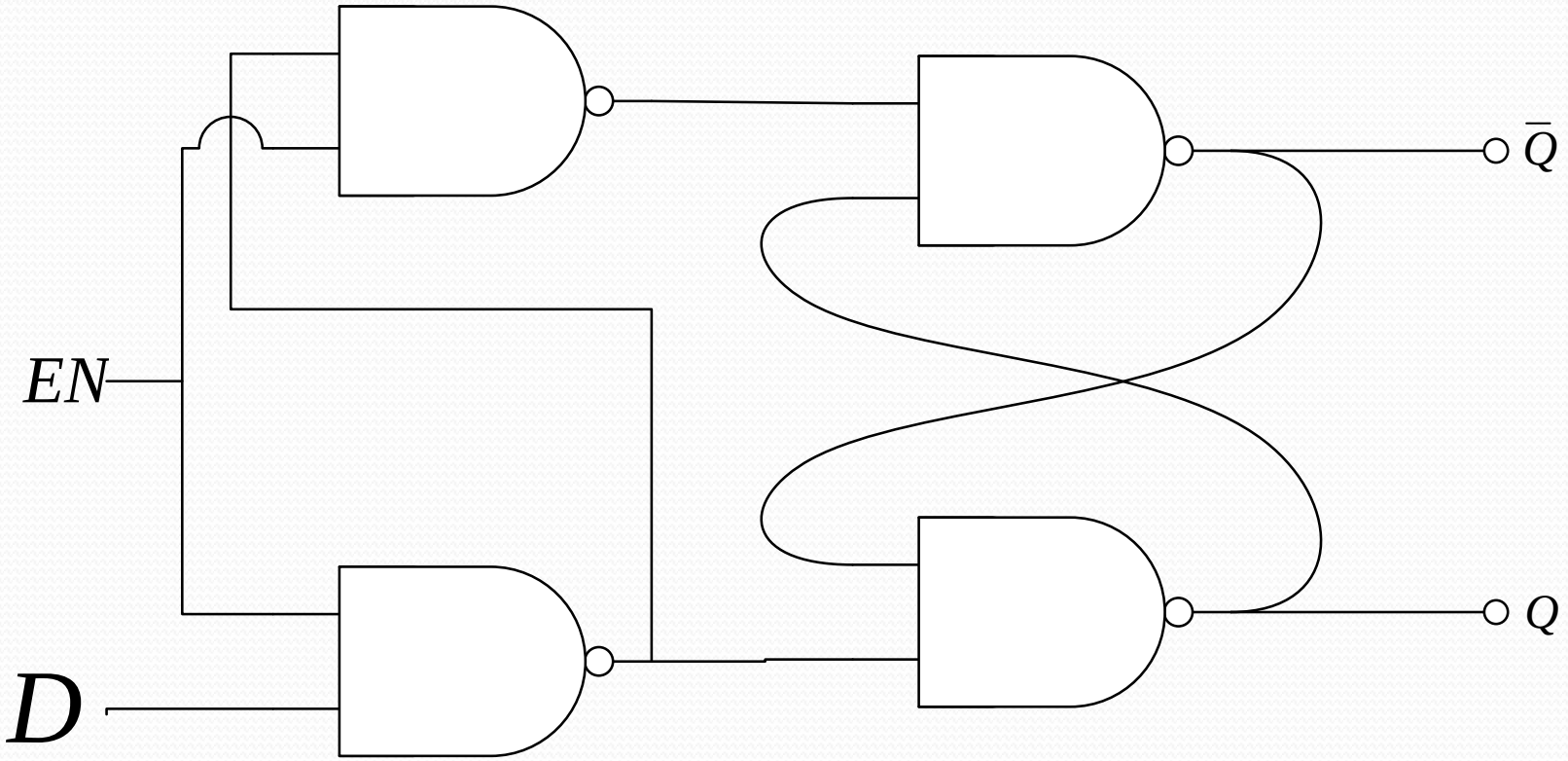
Gated SR NAND Latch

- Add enable to SR NAND Latch
- When enable is true, it works as normal.
- When enable is false, it keeps its state.



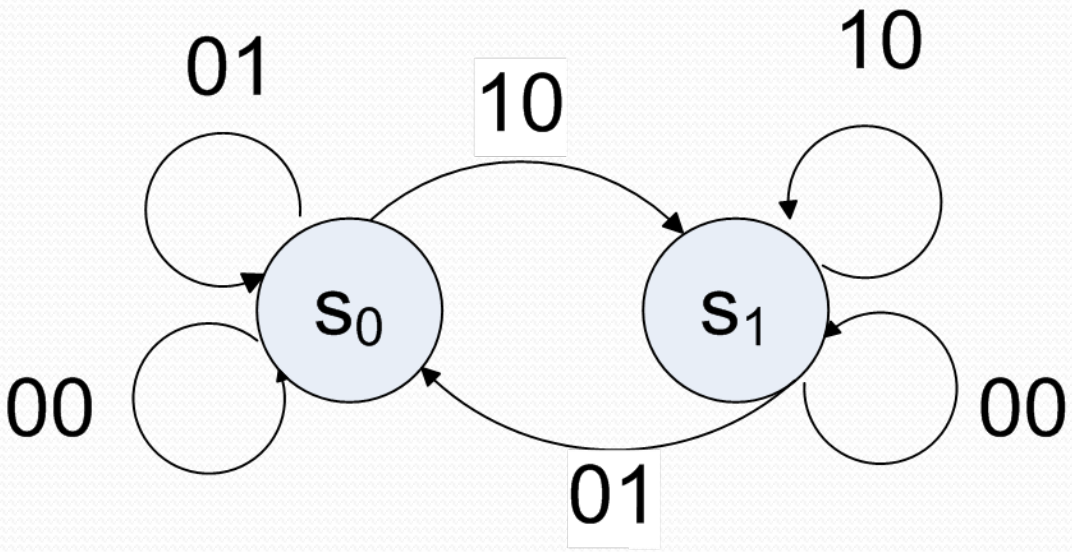
D-Type SR NAND Latch

- Improve the gated version to avoid the restricted input



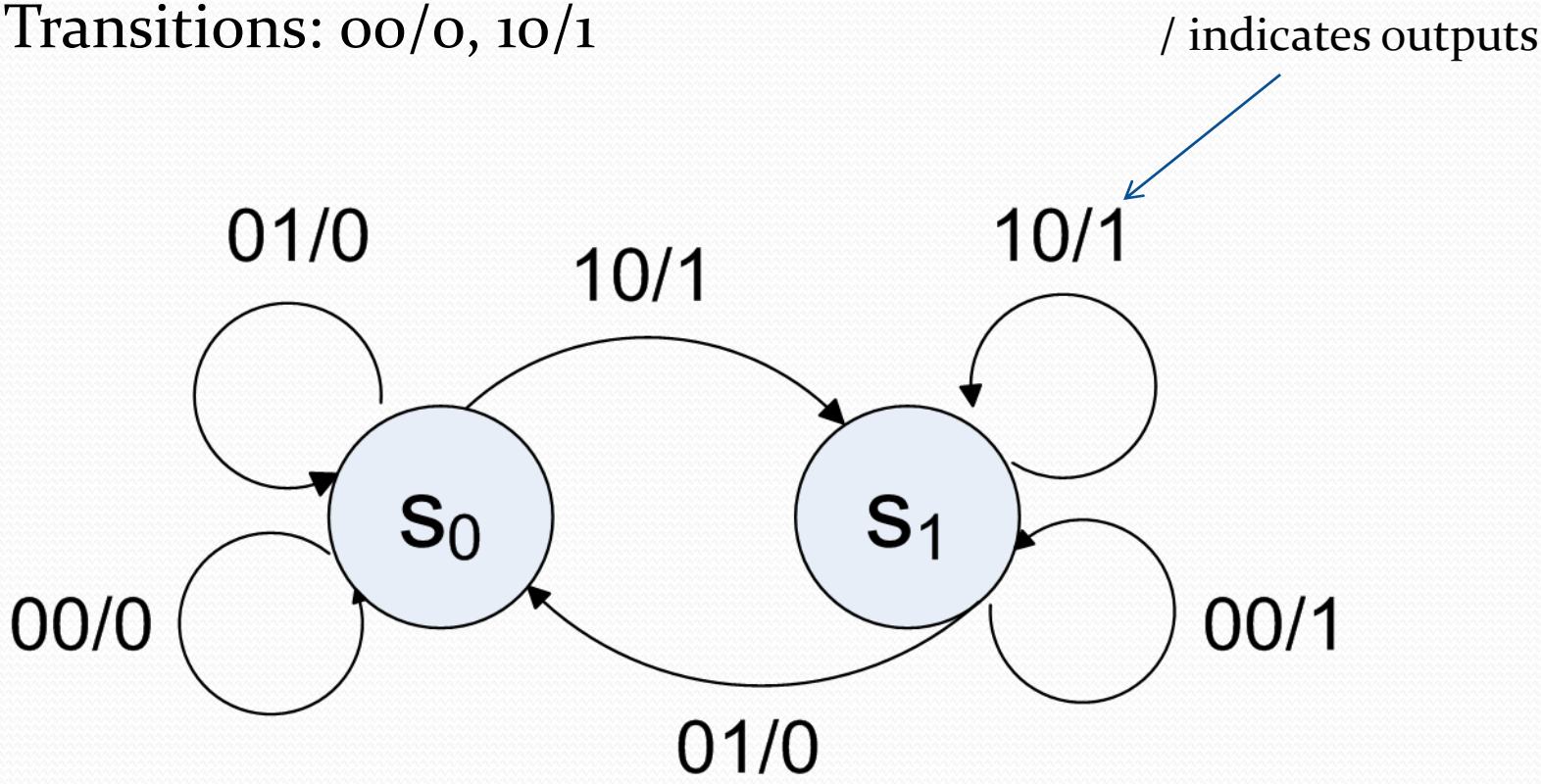
SR NOR Latch as a FSM

- State transition diagram
  - Nodes as states
  - Arcs as transitions
  - Labels as input "SR" values



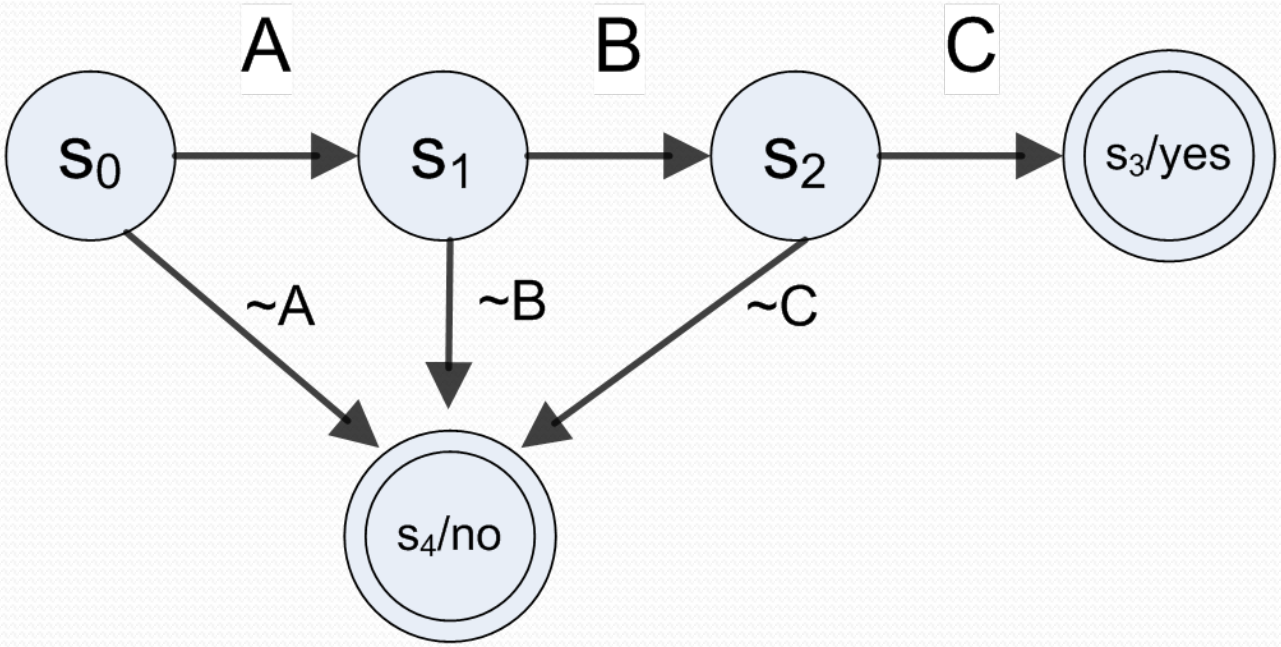
Annotated State Transition Diagram

- Outputs may be associated with a state or a transition
  - States: S0/0, S1/1
  - Transitions: 00/0, 10/1



Acceptor/Recognizer

- Given an input, an FA moves from the initial state to a final state, in which it outputs something like “yes” or “1.”
- An acceptor/Recognizer that recognizes “ABC”



Transducer

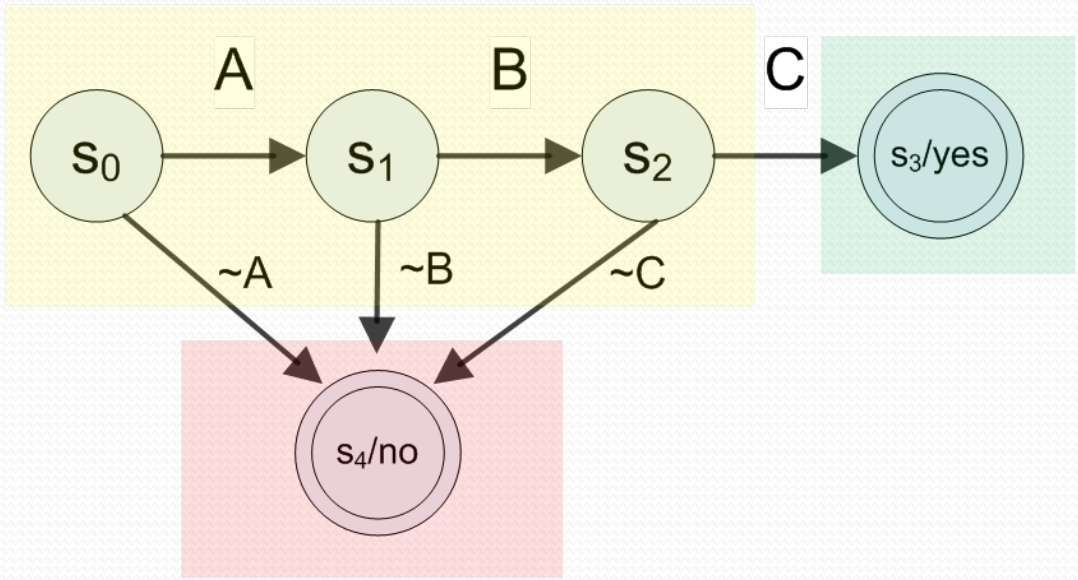
- FSMs that produce outputs other than just 0 or 1
- The outputs are based on inputs and its current state
- The outputs are used to control other components in the system
- A good example is a control unit in a CPU.

Types of Transducers

- Moore machines
  - Outputs depend on states
  - Outputs associated with states in state transition diagrams
- Mealy machines
  - Outputs depend on both inputs and states
  - Outputs associated with transitions in state transition diagrams

Moore Machine

- State decoder
  - 5 states need a 3-bit register
  - 3 outputs:
    - Computation (00)
    - Failure (01)
    - Success (10)



State Decoder

- R: state register, C: output

$$C_1 = \overline{R_2} R_1 R_0$$

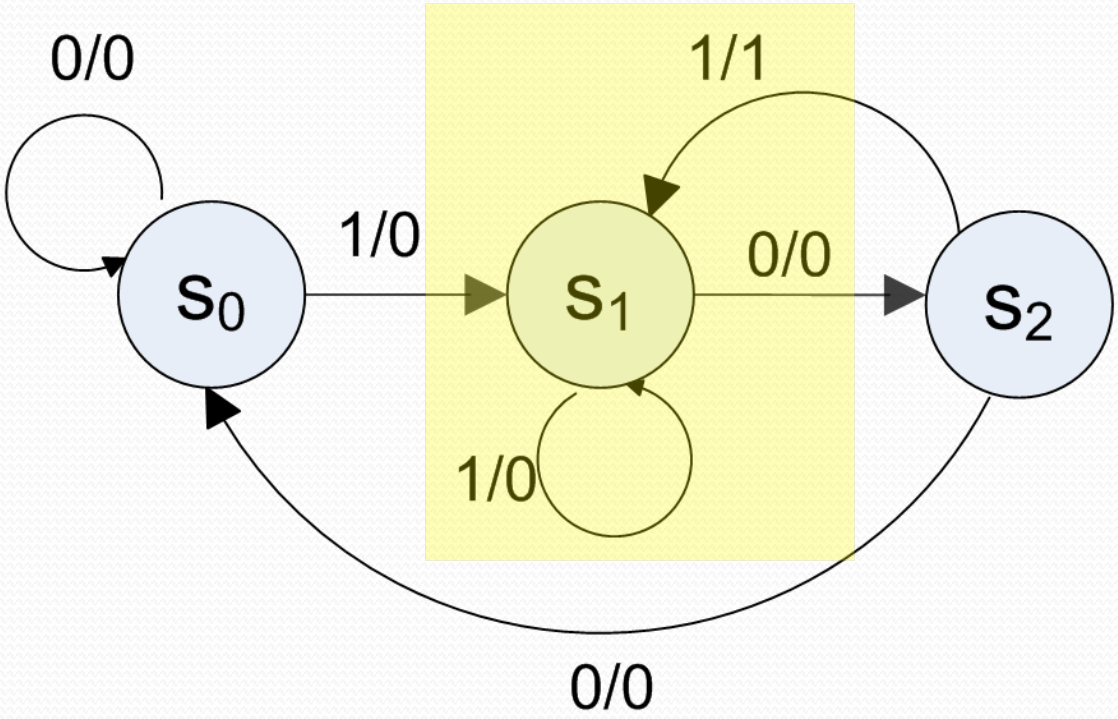
$$C_0 = R_2 R_1 R_0$$

R_2	R_1	R_0	C_1	C_0
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1

Outputs depend only on the states

Mealy Machines

- Outputs depend on both inputs and states
- E.g., a string transcriber
  - Place an one on each occurrence of “101”
  - 10101 -> 00101



Drawbacks of FAs

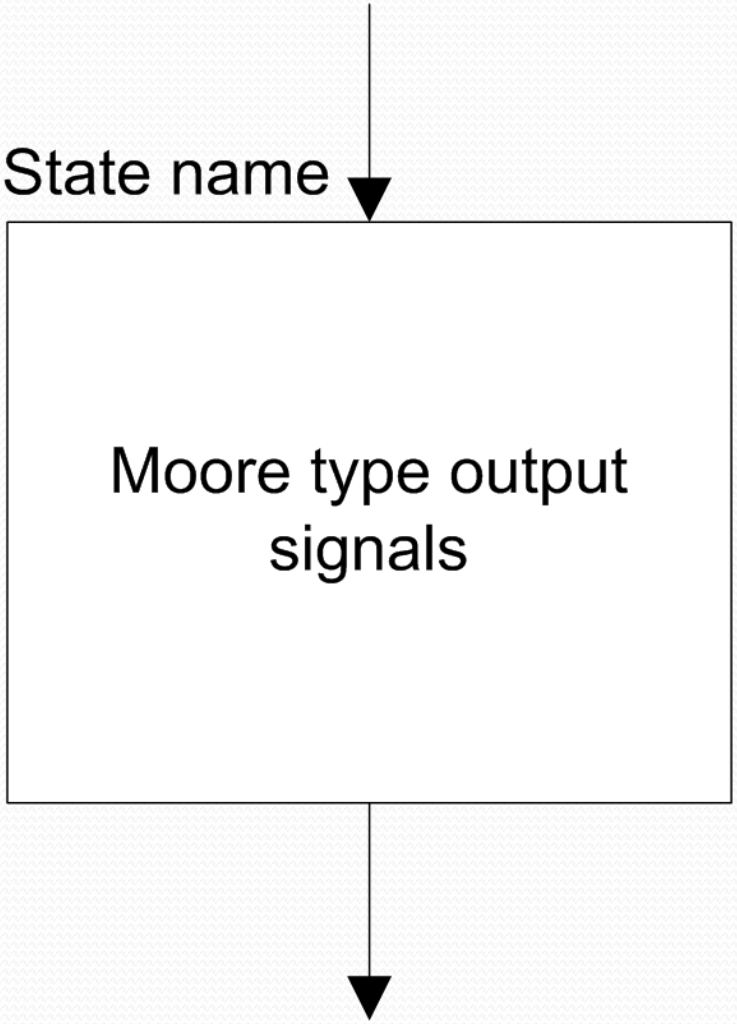
- No timing information
- No algorithms for outputs

Algorithmic State Machines

- ASM charts are used to describe ASMs
- ASM chart
  - State box
  - Decision box
  - Conditional output box
  - Timing
    - On each rising edge of a clock, transition from one state to another occurs.

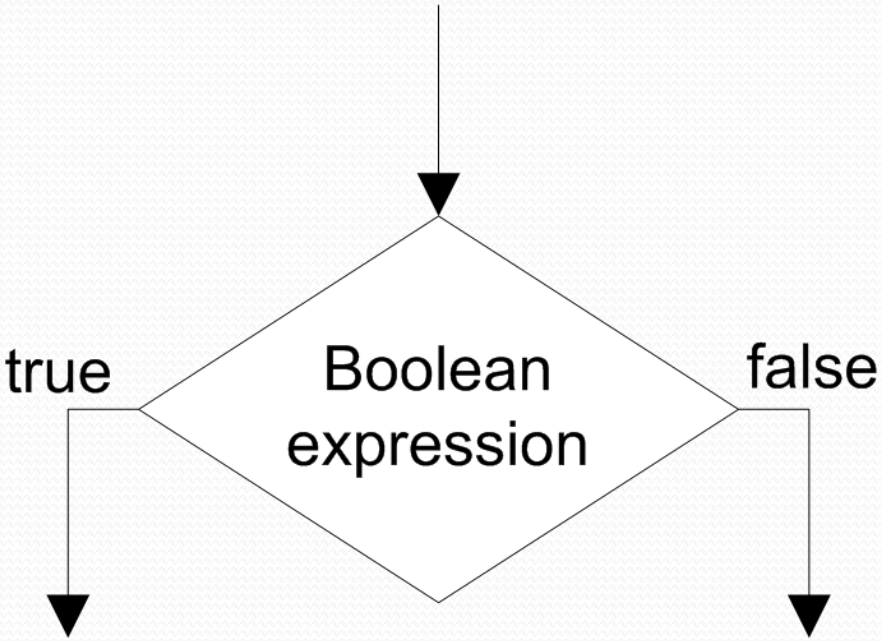
State Box

- Rectangle
- Contain one state
- List Moore type outputs



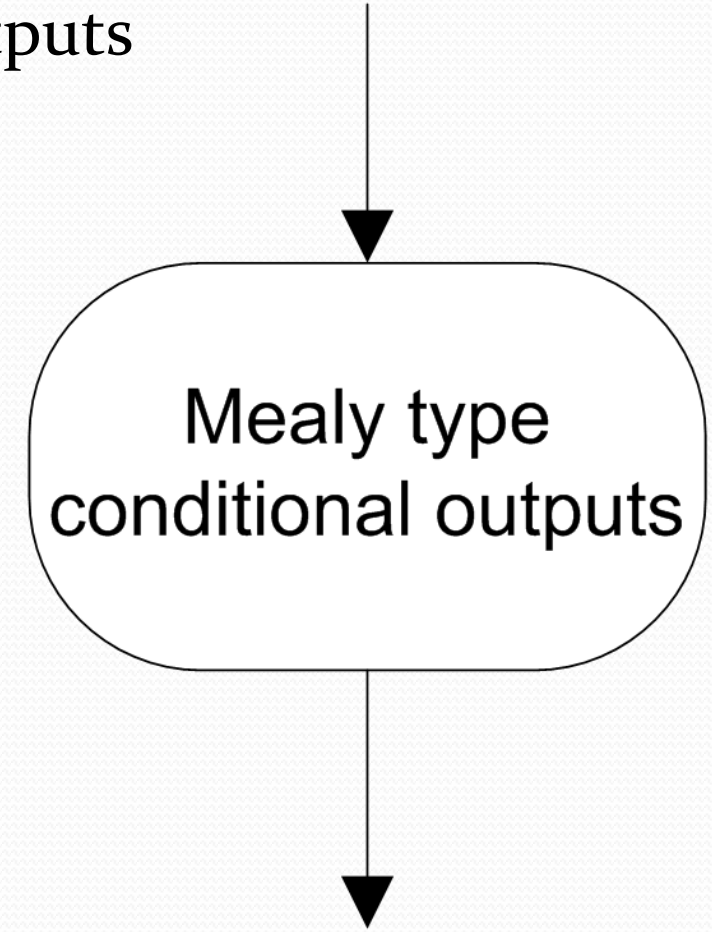
Decision Box

- Diamond
- Transfer between state and state, state and conditional box
- Boolean expressions for conditions



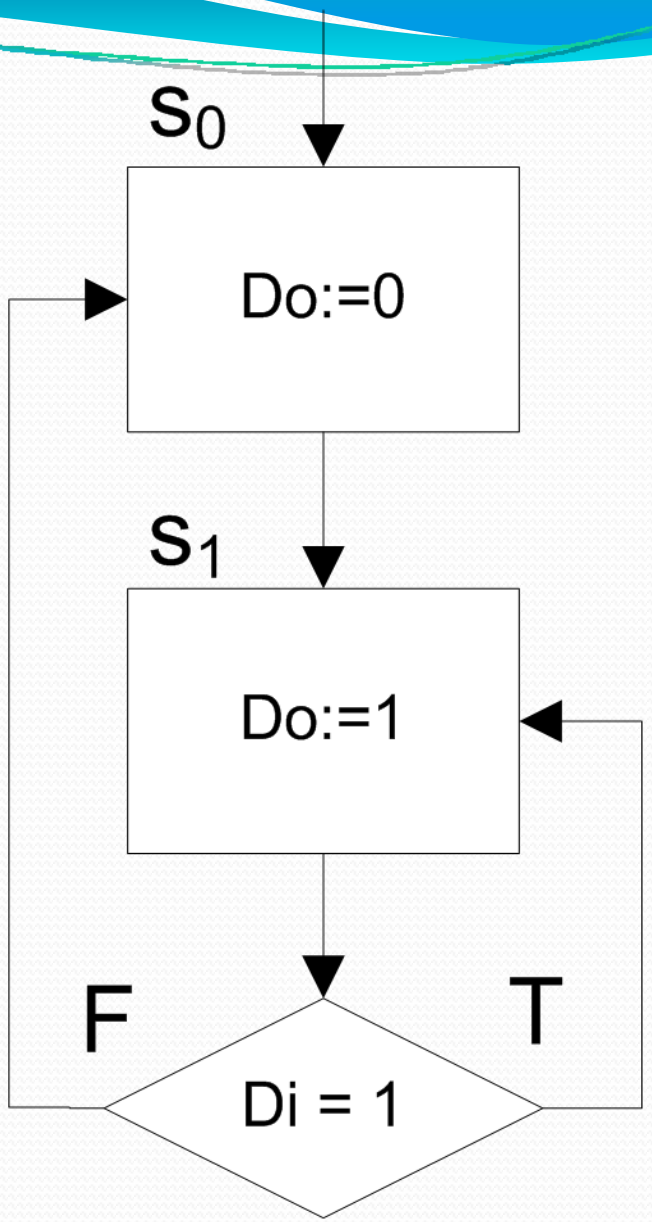
Conditional Output Box

- Oval
- List Mealy type outputs



An ASM Chart

- := assignment
- = comparison



Larger Design Concerns

- Small design can be handled well using
  - Block diagrams
  - Truth tables
  - Boolean expressions
  - K-Maps
  - Optimized logic
  - Schematic design
- Larger design
  - It is hard to deal with in gate levels.
  - A higher level of abstraction is needed.

Hardware Description Languages

- A high level abstraction to describe hardware circuit
- Two popular HDLs:
  - Very high speed integrated circuit hardware description language (VHDL)
  - Verilog
- Similar to programming languages like Java or C, but only syntax-wise
- Semantically, HDLs are like, perhaps, concurrent languages such as Ada.

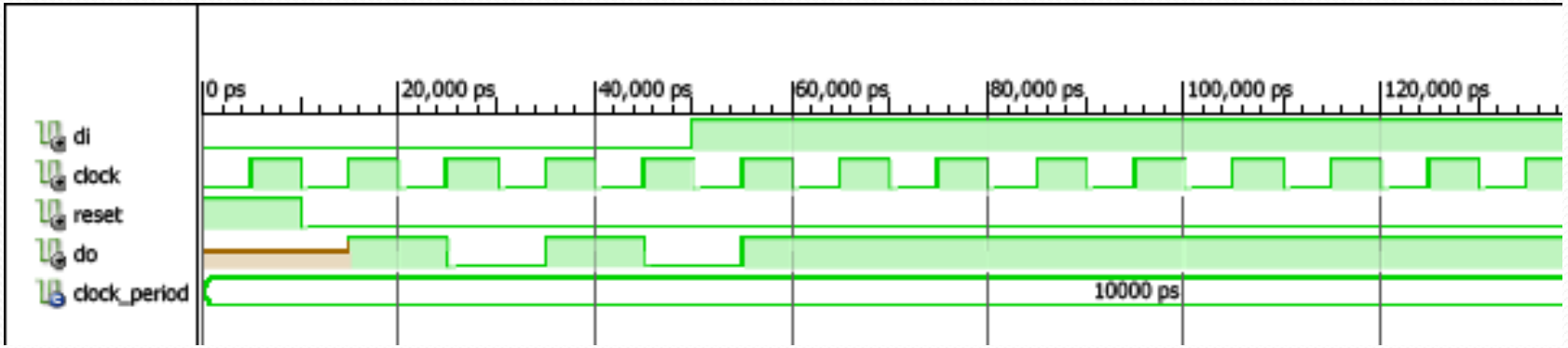
VHDL

- Describe circuits in several levels
  - Behavior, RTL, Gates
- Signal assignments
  - Variables, wire, etc.
- Control statements
  - If-then-else
  - Case
- Modularization
  - Component instantiation
  - Replications (for generate)
- Parameterization
  - Generic design
- Simulations

VHDL Implementation

```
asm: process(reset, clock, Di)
begin
    if reset='1' then
        c_state <= s0;
    elsif clock='1' and clock'event then
        case c_state is
            when s0 =>
                c_state <= s1;
                Do <= '1';
            when s1 =>
                if Di='1' then
                    c_state <= s1;
                    Do <= '1';
                else
                    c_state <= s0;
                    Do <= '0';
                end if;
        end case;
    end if;
end process;
```

Waveforms

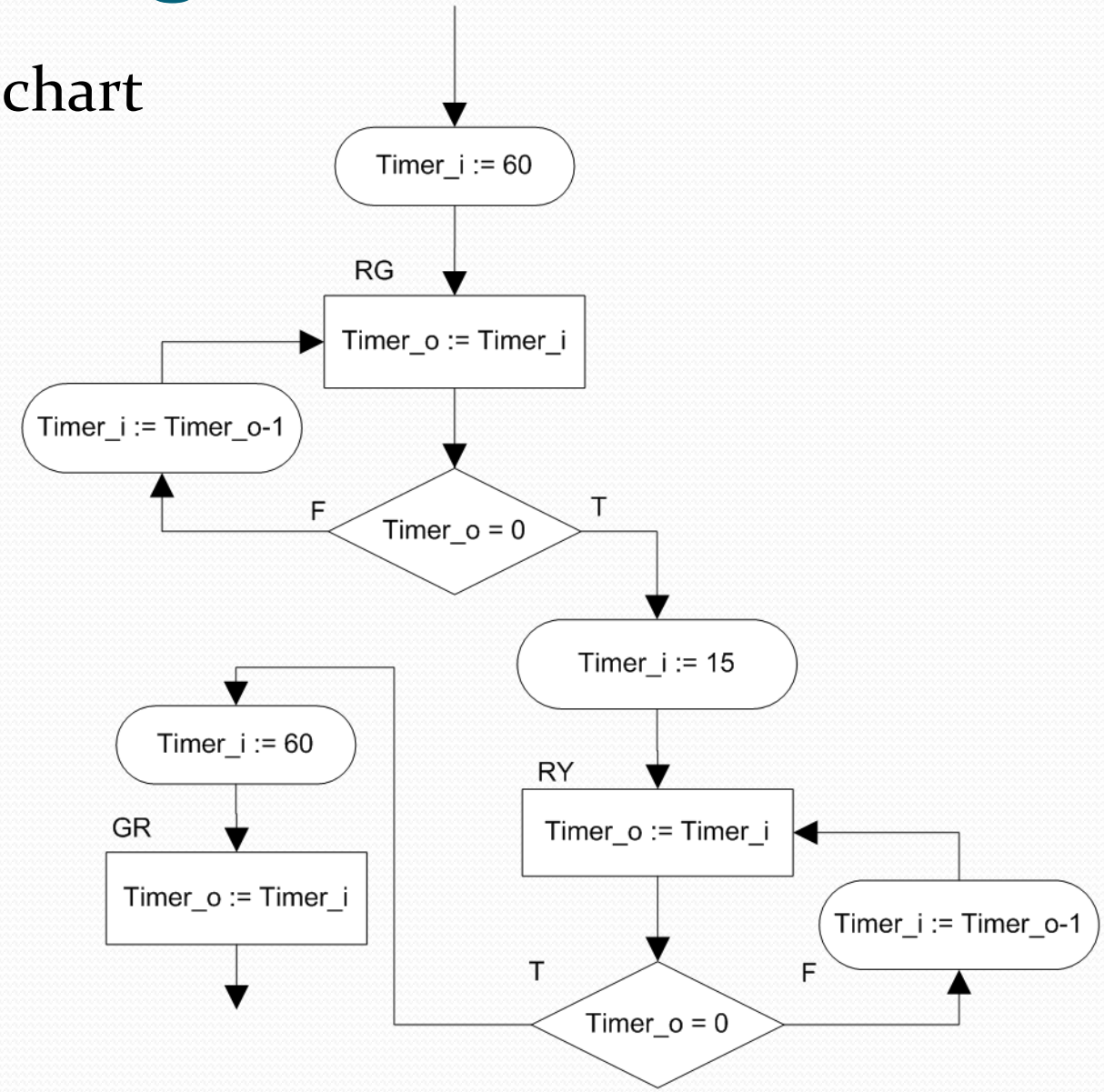

0 ps
20,000 ps
40,000 ps
60,000 ps
80,000 ps
100,000 ps
120,000 ps
di
clock
reset
do
clock_period
10000 ps

Timing Modeling

- Add timing information to an ASM
- E.g., traffic light control
  - State names
    - two letters from RGY
    - First letter fro vertical road, second one for horizontal road
  - Green delay time: 60
  - Yellow delay time: 15

Traffic Light Control

- Partial chart



Questions