N26F300
VLSI SYSTEM DESIGN
(GRADUATE LEVEL)

FSM and Controller

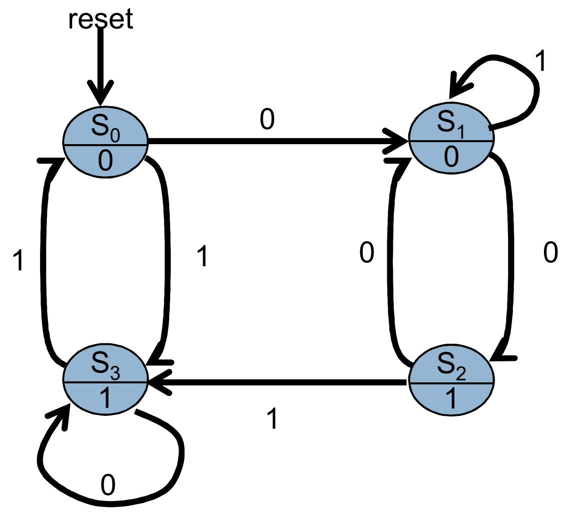
Outline

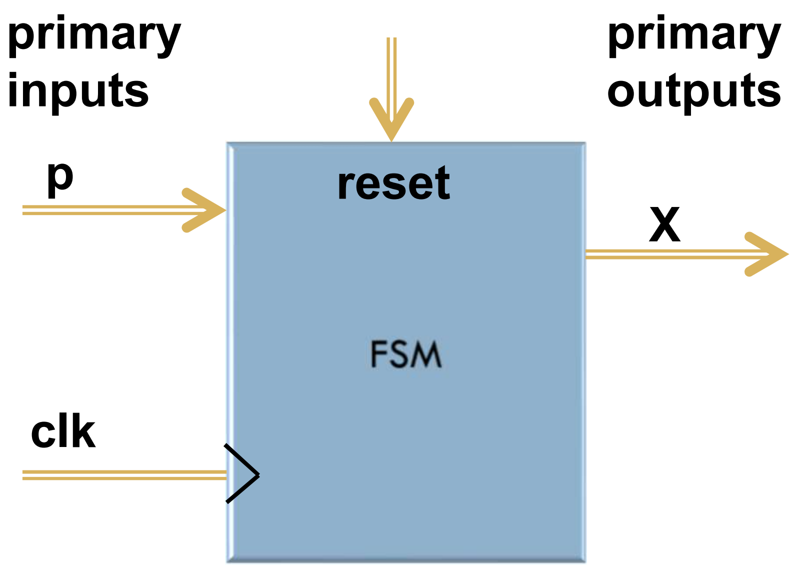
- Moore & Mealy Revisited
- Examples of FSM
- Control external hardware
  - Controller for timer, ADC and memory access
  - WatchDog Timer
  - DMAC

[Material adapted from “FSM based Digital Design Using Verilog HDL”by Minns and Elliott]

Finite State Machine (FSM)

- A digital sequential block controlled by one or more inputs with predefined finite states. The machine can move from one state to another state.





Synchronous FSM

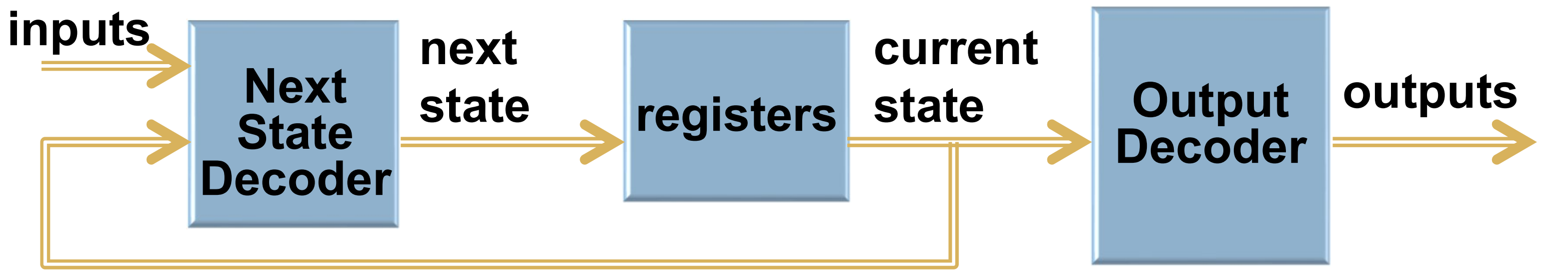
- An FSM that can only change states only if a clock pulse occurs.
- States can be identified by using a number of flip-flops inside the FSM block.

$$\#\text{ of states} = 2^{\text{Number of FFs}}$$

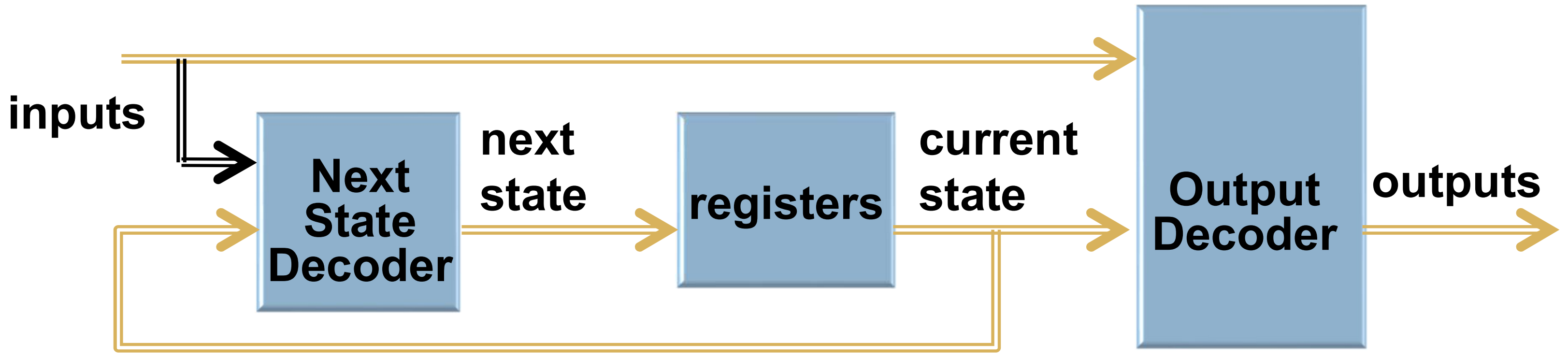
$$\Rightarrow \quad \#\text{ of FFs} = \log_2(\#\text{ of states})$$

Moore and Mealy Machines

- Moore model



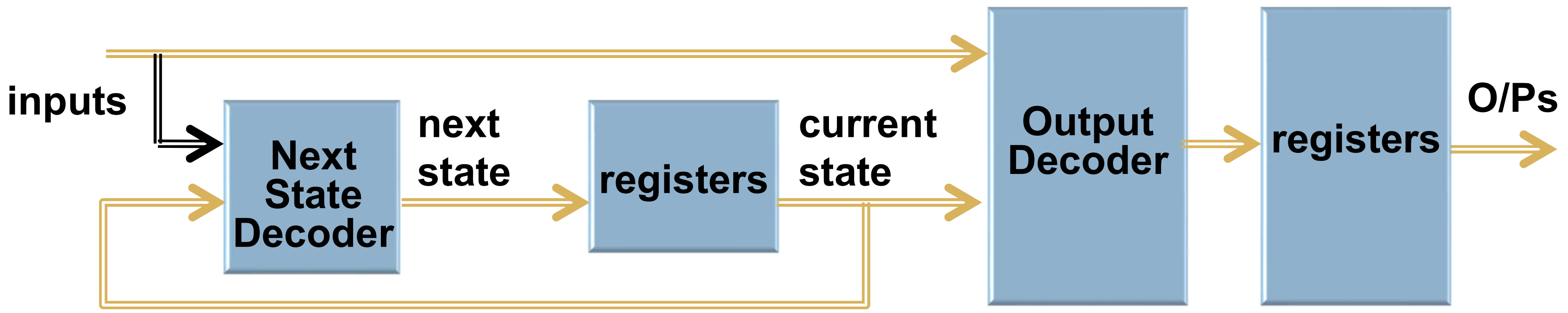
- Mealy model



Mealy vs. Moore models

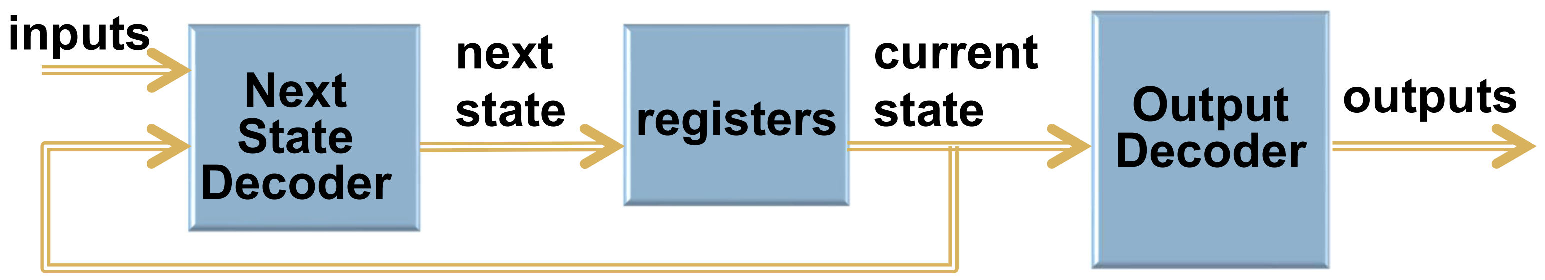
- Moore is safer to use
  - Outputs change at clock edge
  - Not like Mealy, output follow input in asynchronous way
-
  - Output = function of inputs and the present state
  - Not like Moore, output depends only on the present state
-
  - Complete in the same cycle
  - Not like Moore, more logic may be needed to decode state into outputs

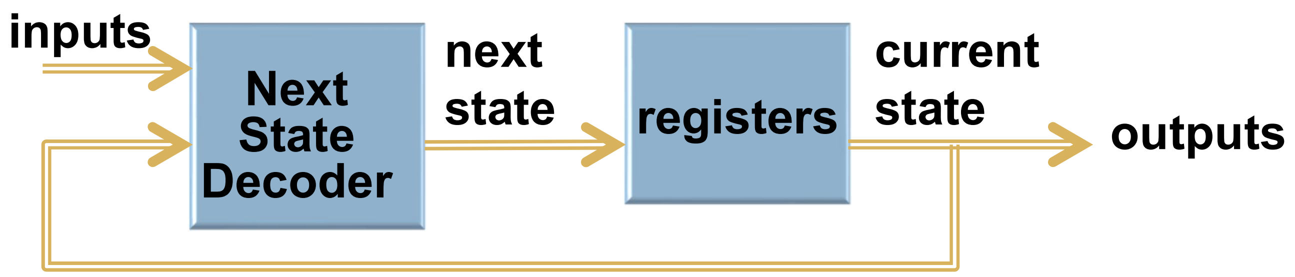
Synchronous Mealy Model



- Synchronous Mealy machines avoid the potential glitches and change of outputs asynchronously

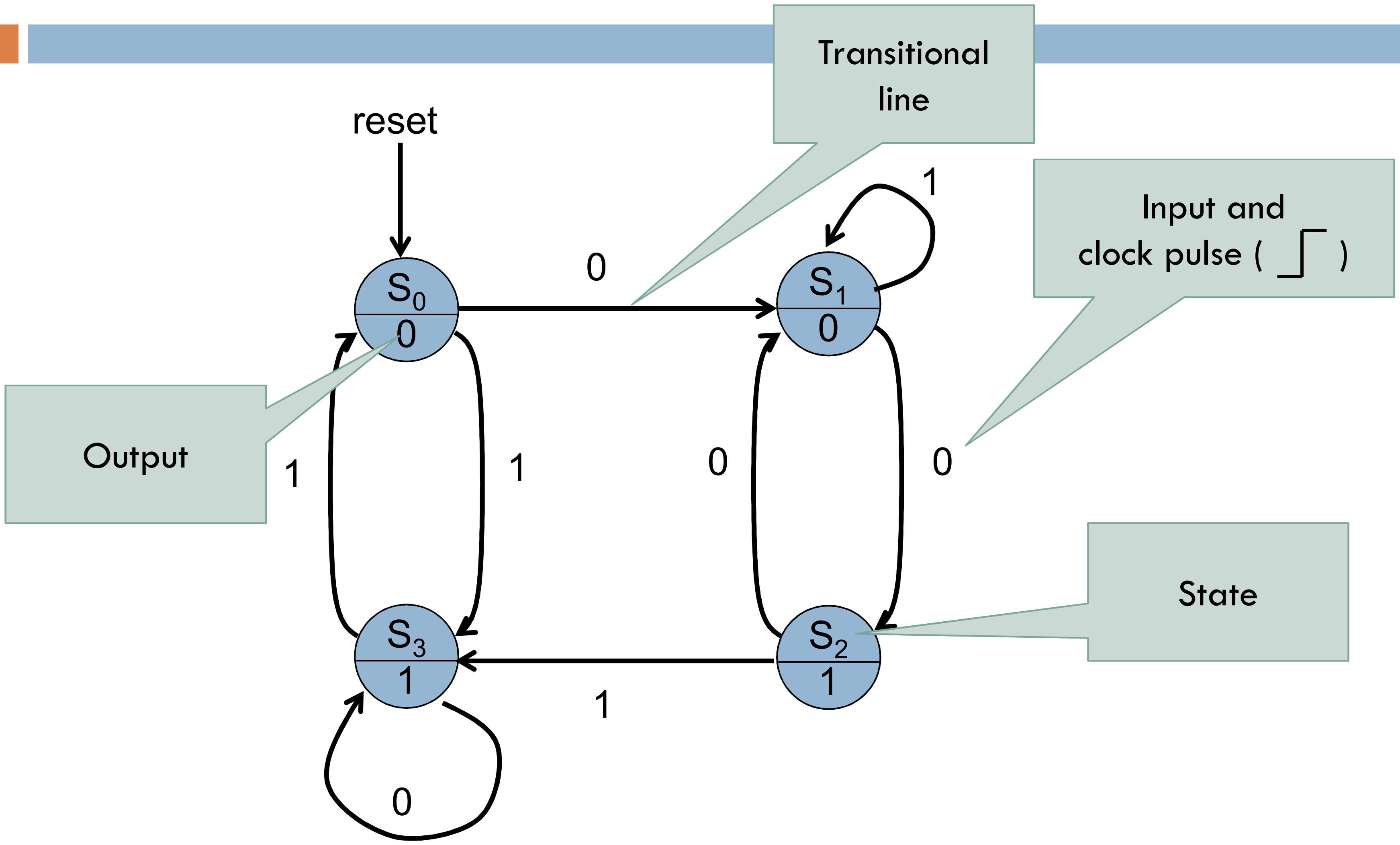
Moore Machines





One of basic forms of many asynchronous counters

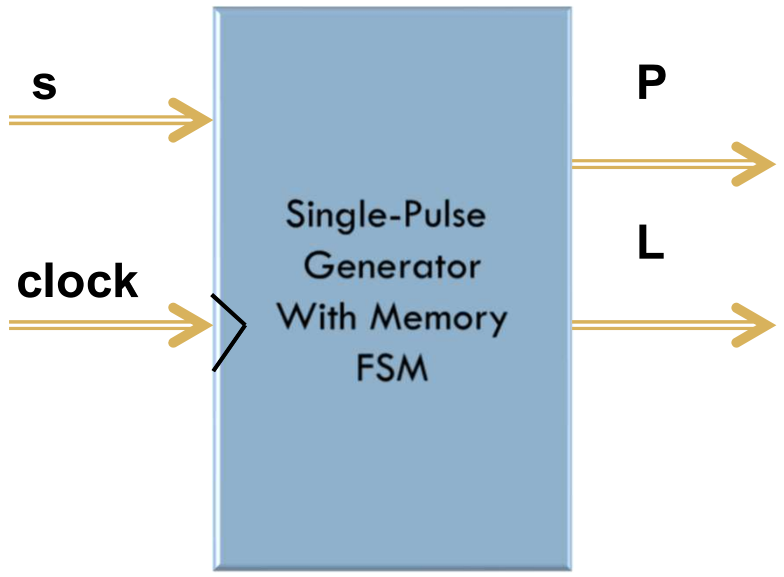
State Transition Graph (STG)



EXM1: A Single-Pulse Generator Circuit with Memory (SPGM)

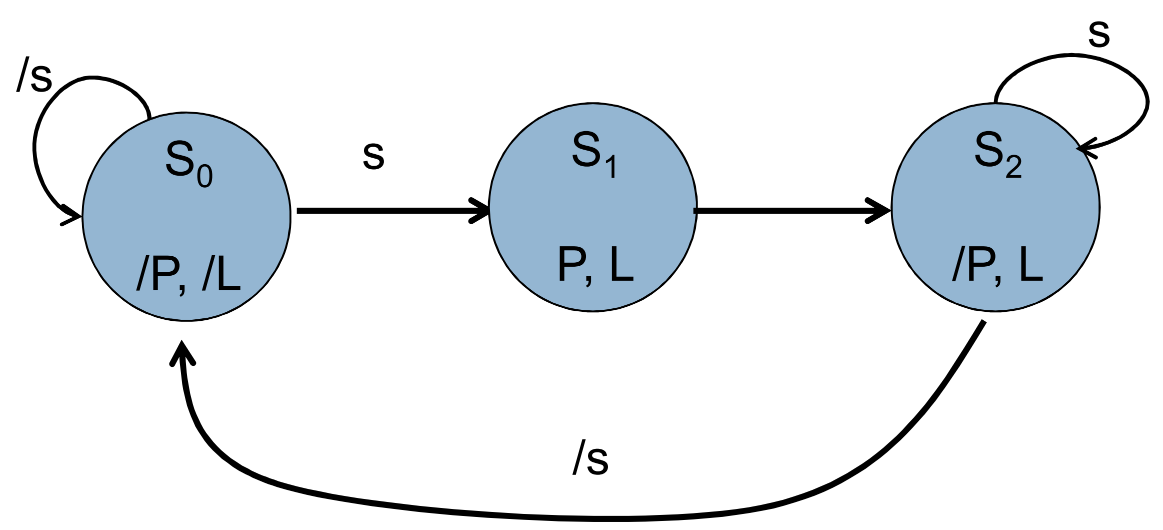
- Problem:
  - When input s = 1,
    - produce a single output pulse at the output P
    - Set output L to 1
  - When input s = 0,
    - Clear output L to zero

  - L: memory indicator



STG for SPGM (SPGM-1)

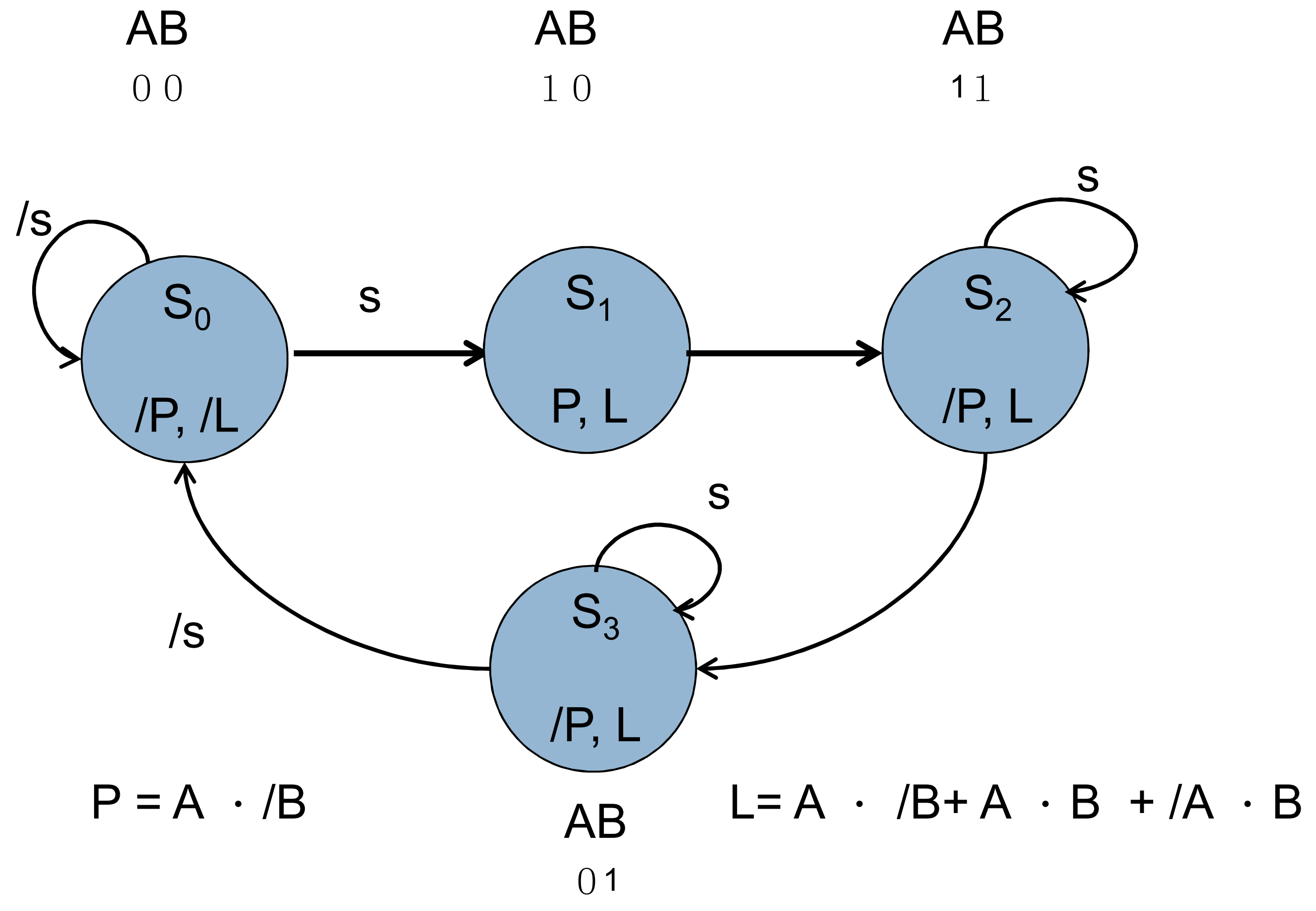
State encoding	AB	AB	AB
	0 0	1 0	0 1



$P = A \cdot /B$

$L = A \cdot /B + /A \cdot B$

STG for SPGM complying with Unit Distance Pattern (SPGM-2)

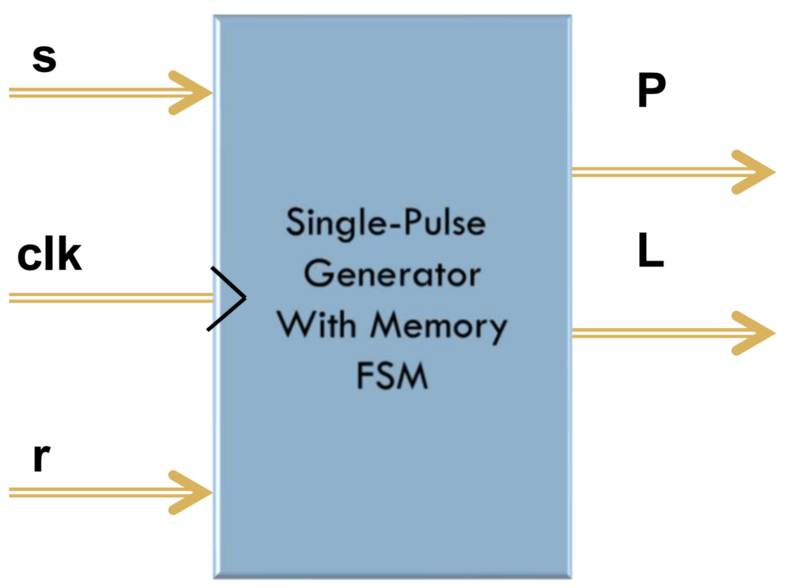


P = A · /B

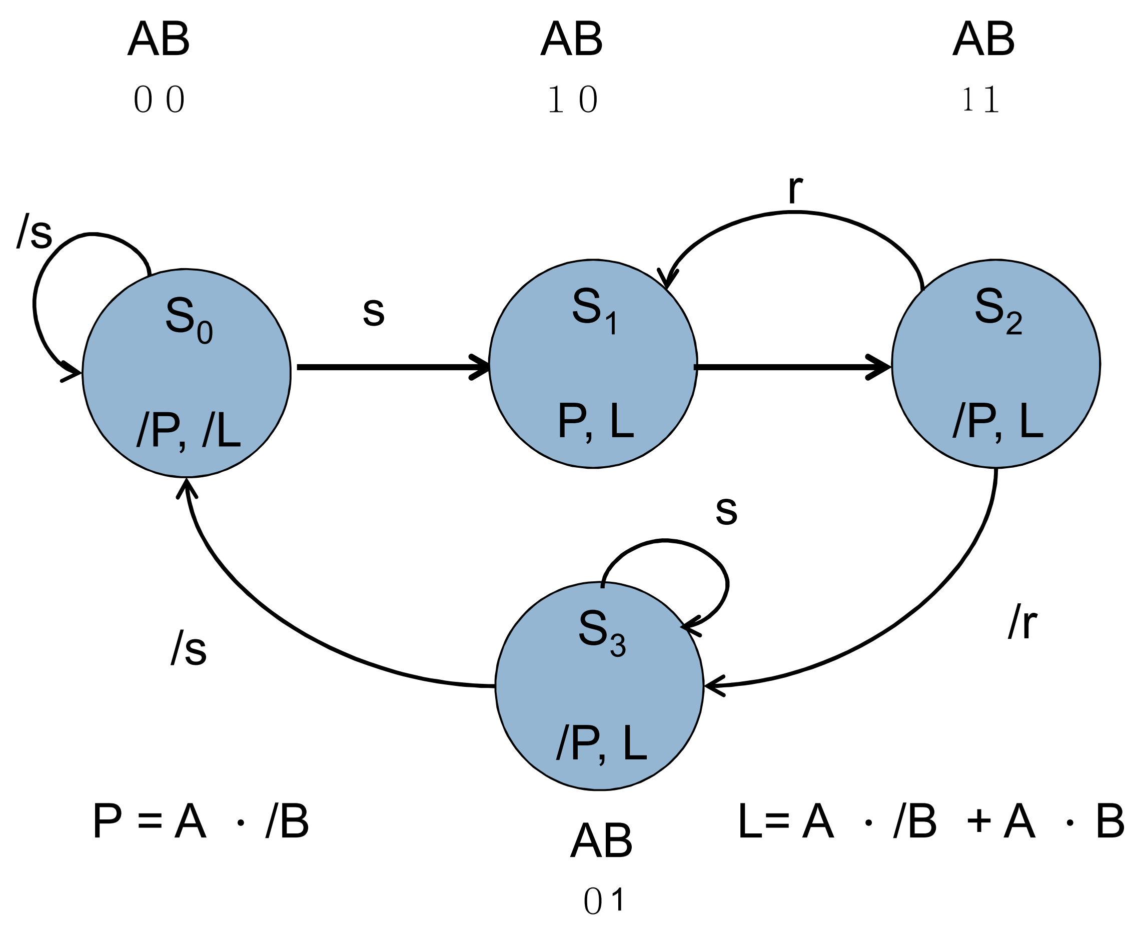
L= A · /B+ A · B + /A · B

EXM2: A Single-Pulse Generator Circuit with Memory (SPGM-3)

- Problem:
  - When input s = 1,
    - produce a single output pulse at the output P
    - Set output L to 1
  - When input s = 0,
    - Clear output L to zero
  - When input r =1,
    - Let P = clk
  - When input r =0,
    - Resume its single pulse



STG for SPGM complying with Unit Distance Pattern (SPGM-3MR)



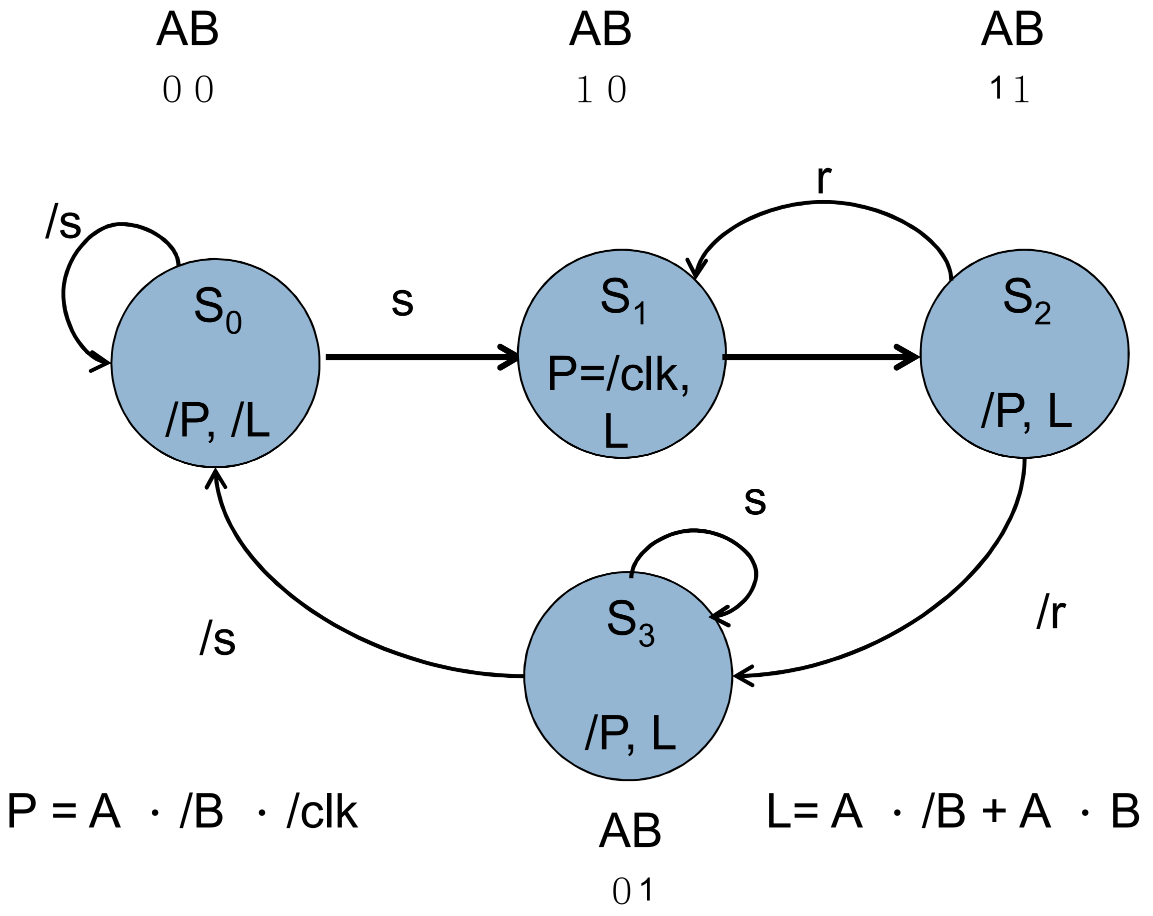
P = A · /B

L= A · /B + A · B + /A · B

Moore to Mealy

- Make the output P depend on the state1 as well as clk
- That is to say a direct control path from the input to the output

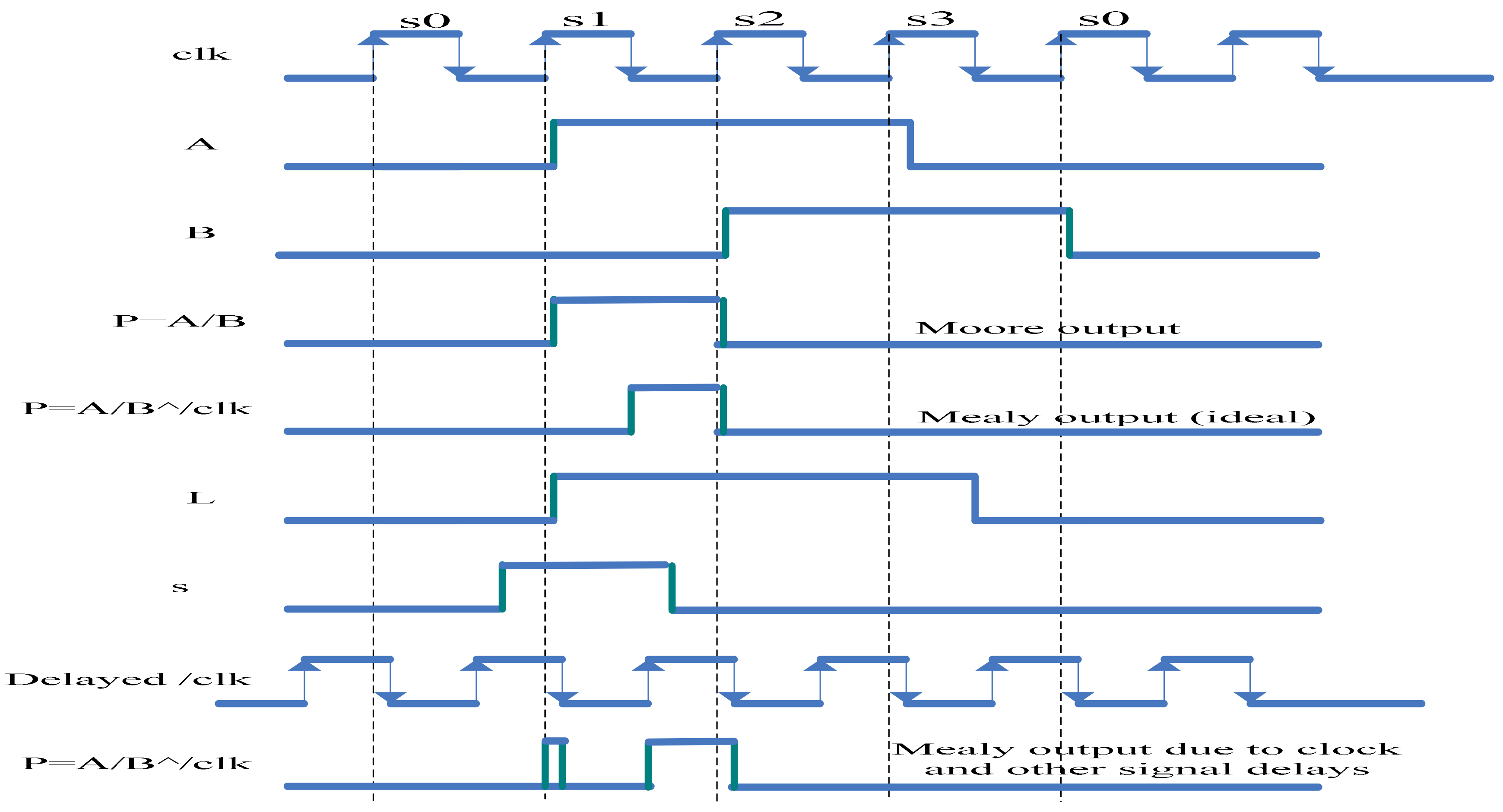
SPGM Using Mealy Model (SPGM-3ML)



P = A · /B · /clk

L= A · /B + A · B + /A · B

Waveform of Moore and Meanly

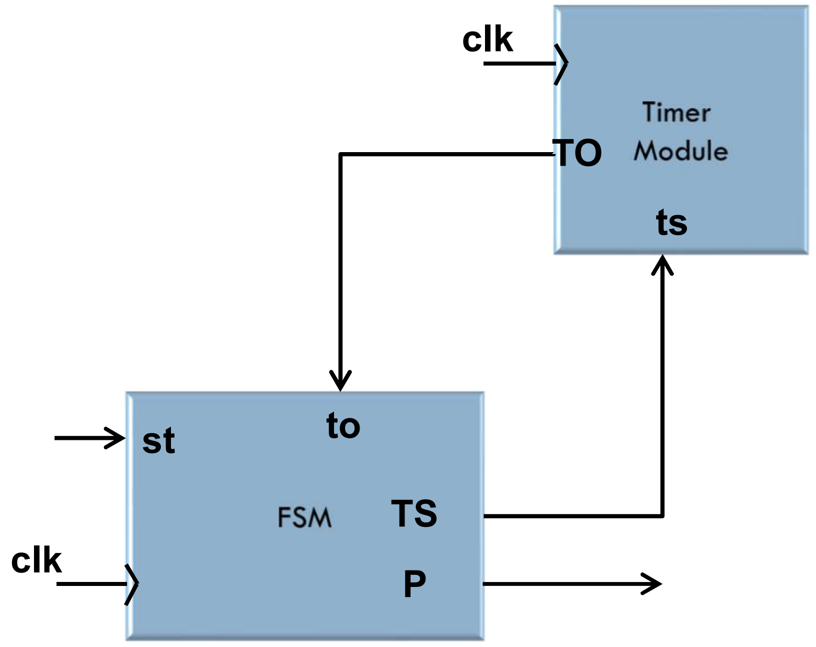


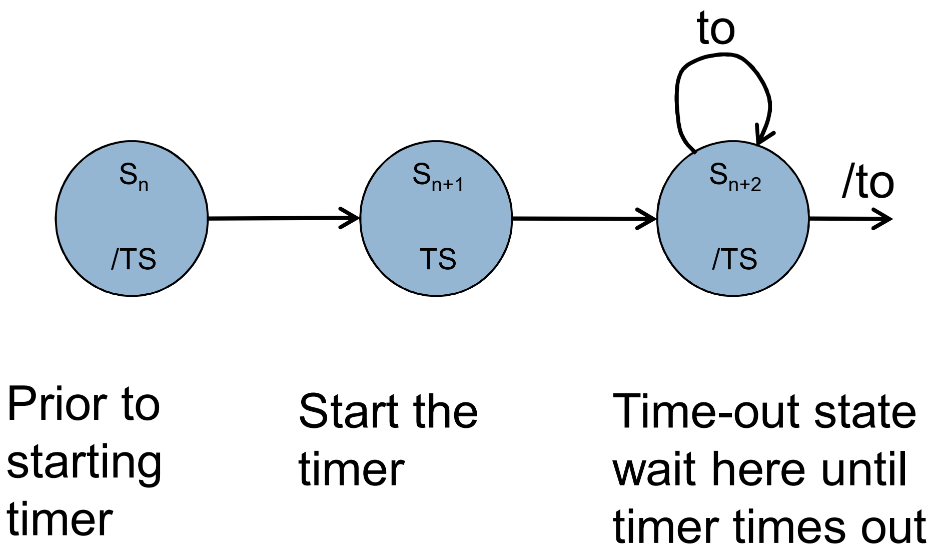
Implementation of Wait

- How to let a FSM to wait in a state for predefined period?

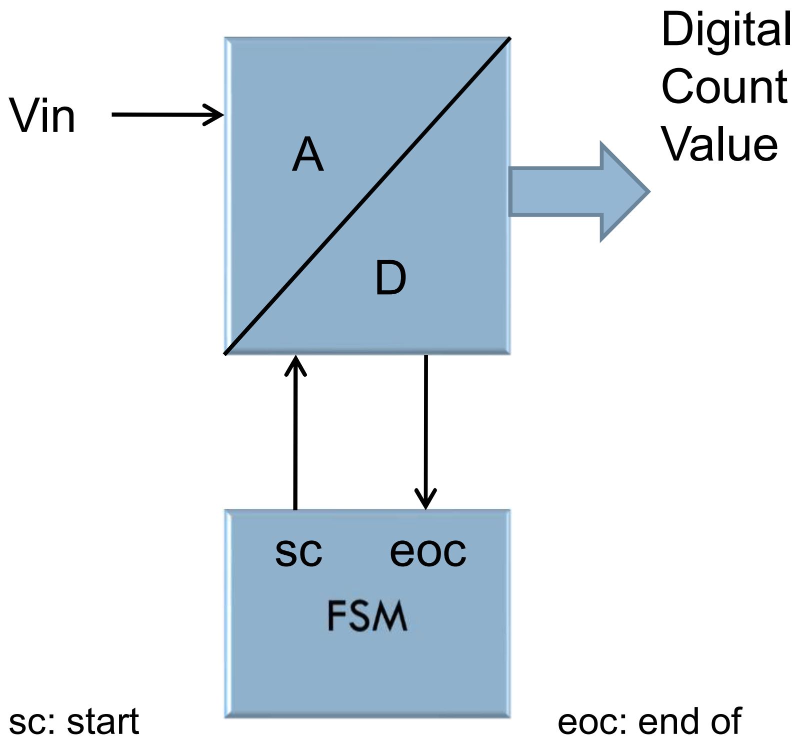
Timer Unit and FSM

- State sequence to control the timing module



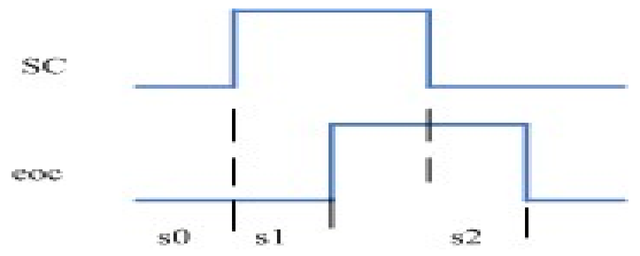


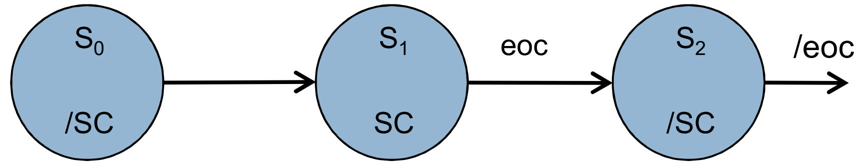
Controlling an Analog-to-Digital Conversion (ADC)



sc: start conversion

eoc: end of conversion

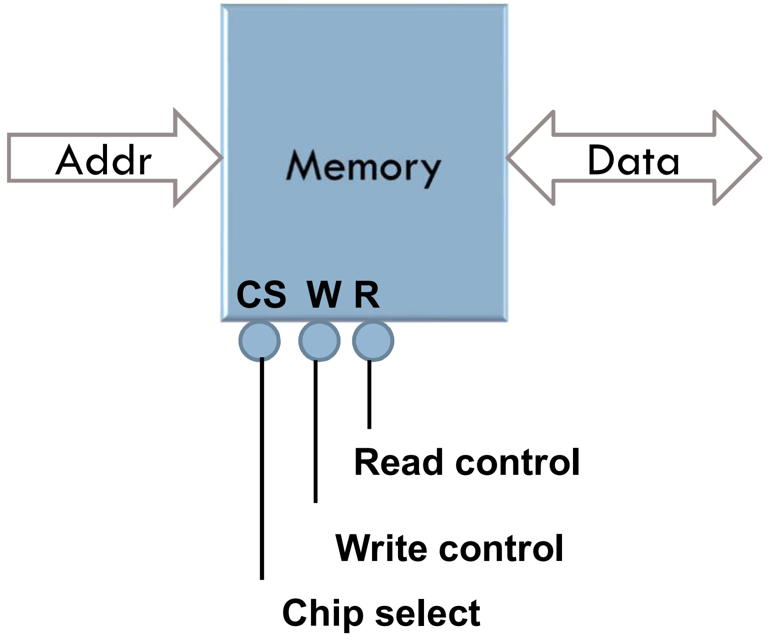




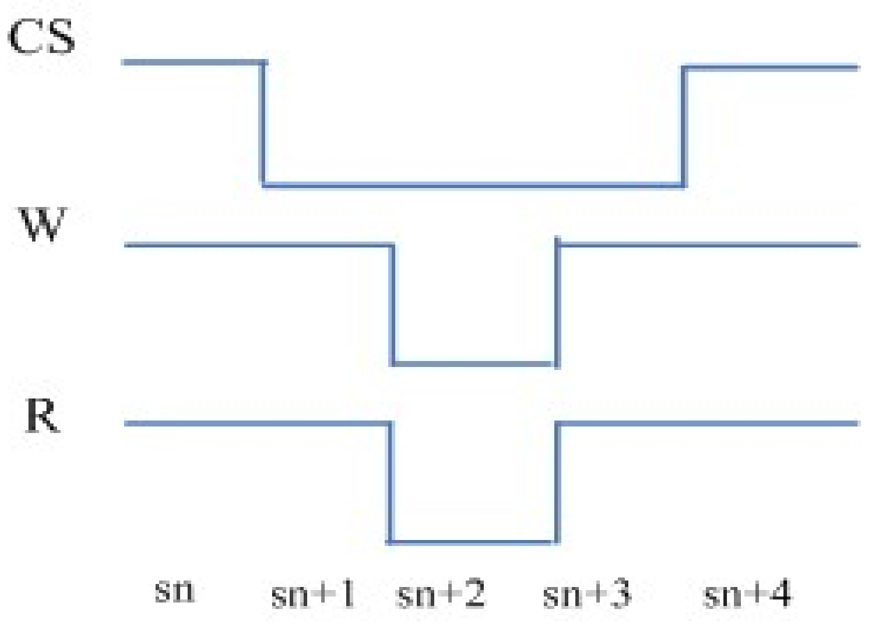
Generic Memory Device

I/O

R

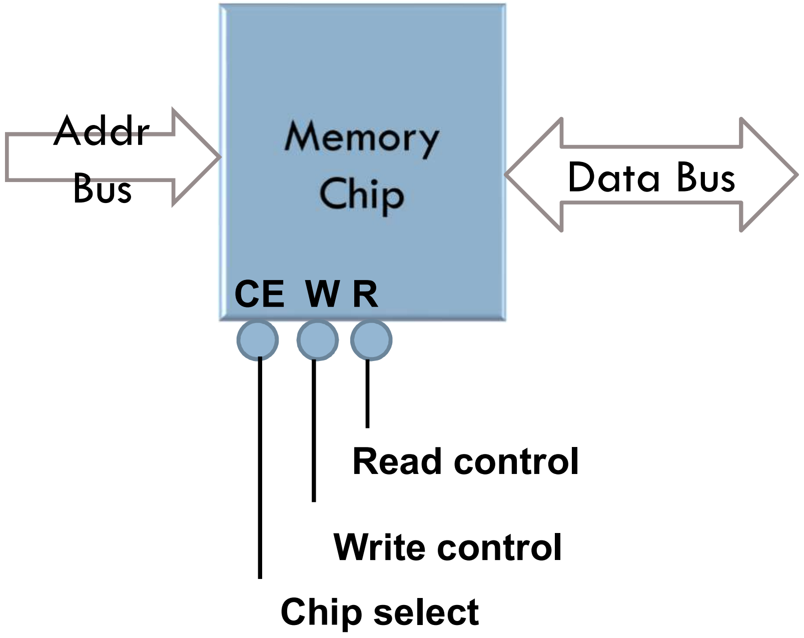


Memory Timing

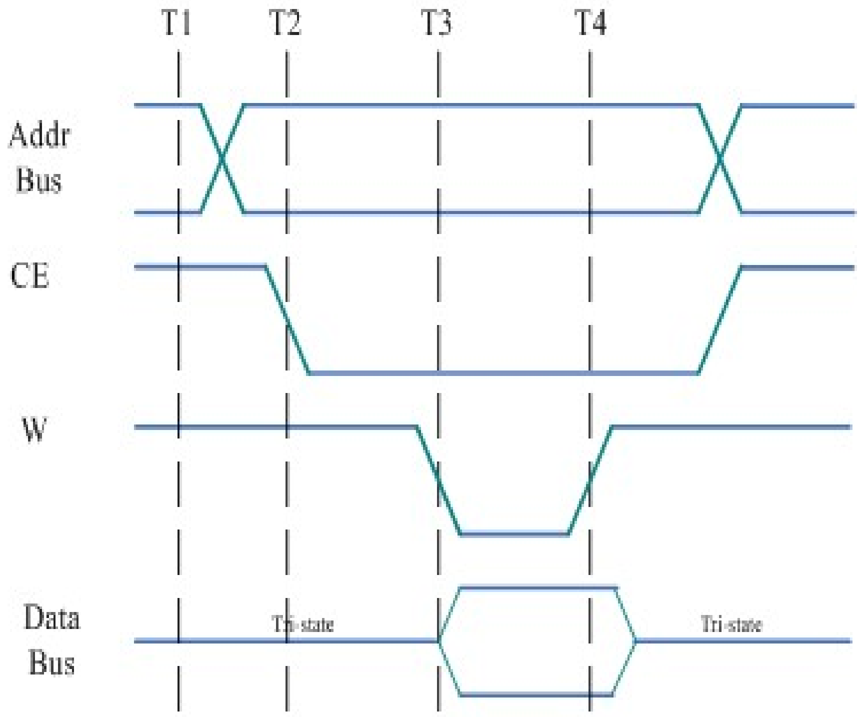


Timing Waveform of a Memory Device

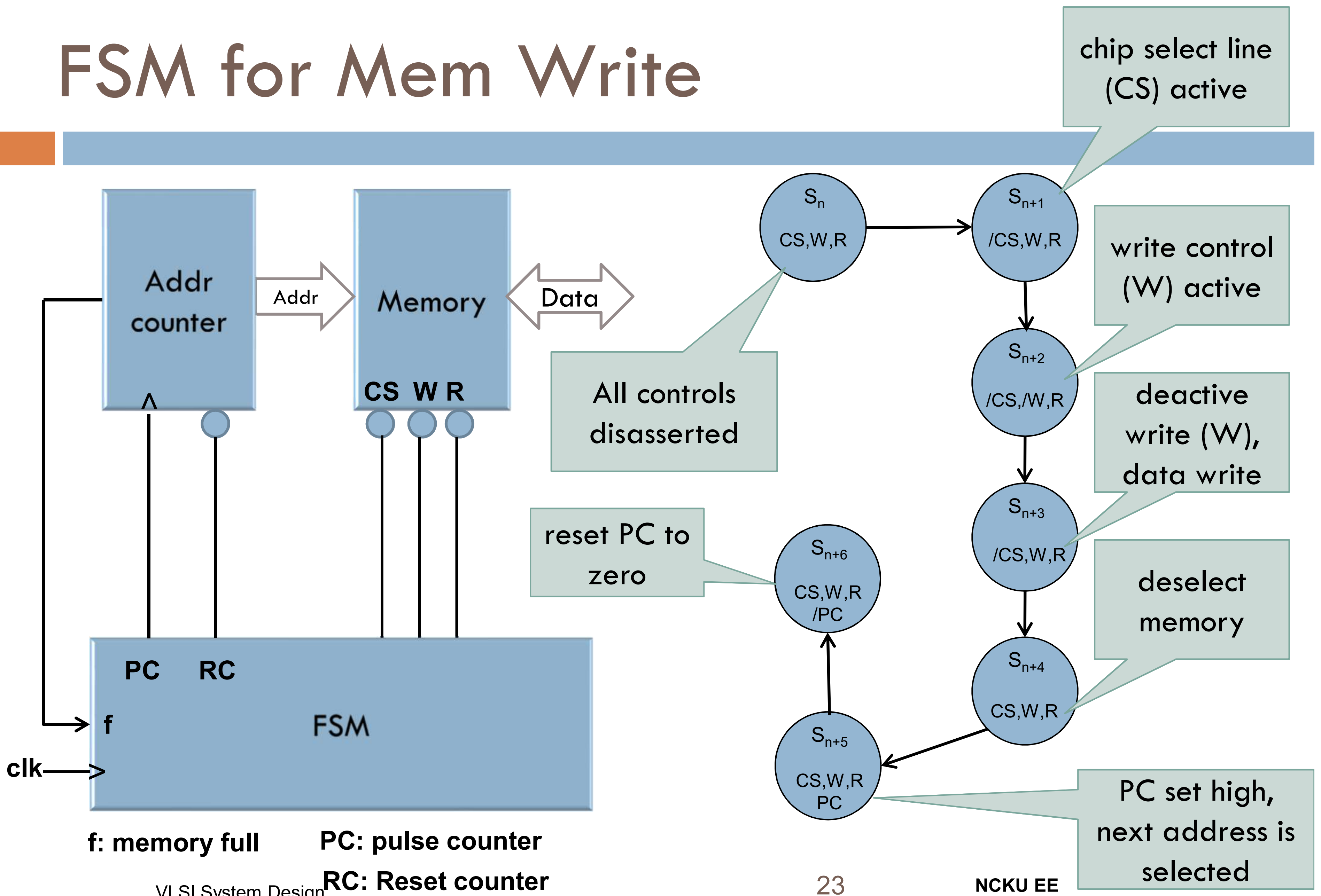
I/O



Memory Timing



FSM for Mem Write

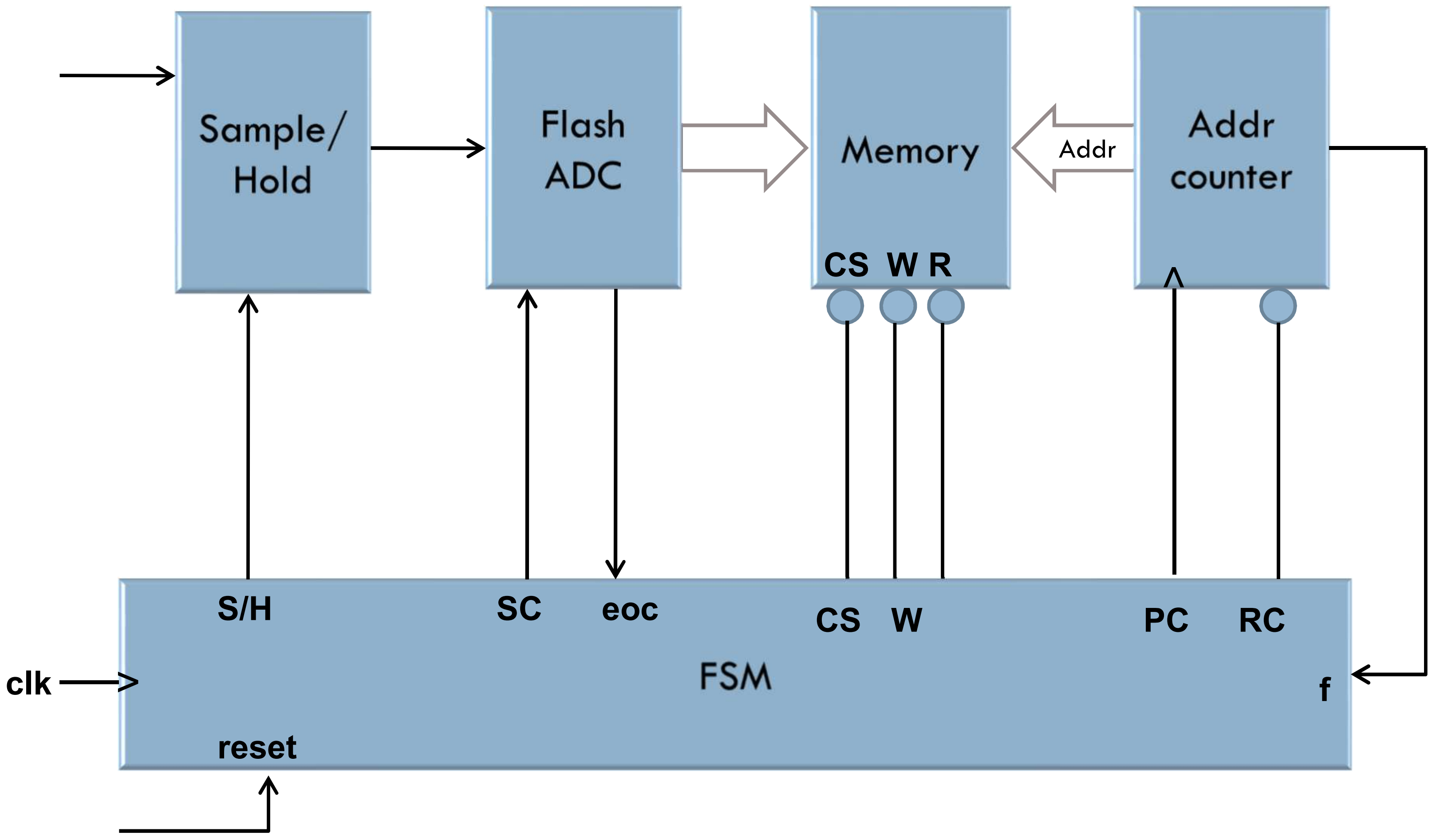


f: memory full

PC: pulse counter

RC: Reset counter

Small Data Acquisition System (DAS)



Exercise!

How CPU interact with I/O devices (or IPs)?

- CPU-to-device communication
  - Dedicate I/O other than memory ➔ Port-mapped I/O
    - Isolate from memory space
    - Require supports from a special class of CPU instructions
    - Dedicated bus
  - Share I/O bus with memory ➔ Memory-mapped I/O
    - Share the same memory space
    - Require less support from ISA, regular memory instructions are sufficient
    - Less internal logic, therefore, cheaper, faster, easier to build

Port-mapped I/O (Isolated I/O)

- A special class of CPU instructions designed specifically for performing I/O
  - Example: Intel 80x86 and IBM 370 computer systems
  - Limited support for addressing modes
    - Typically only for simple load-and-store operations between CPU registers and I/O ports
    - To add a constant to a port-mapped device registers
      - Read the port to a CPU register
      - Add the constant the CPU register
      - Write the results back to the port
  - Waning in popularity

Memory-Mapped I/O (MMIO)

- Parts of address space are assigned to I/O devices
- The memory and registers of the I/O devices are associated with address values.
- When an address is accessed by the CPU, it may go to a physical RAM or instead refer to the I/O devices
- Use fewer instructions and run faster than port I/O
- Most common one

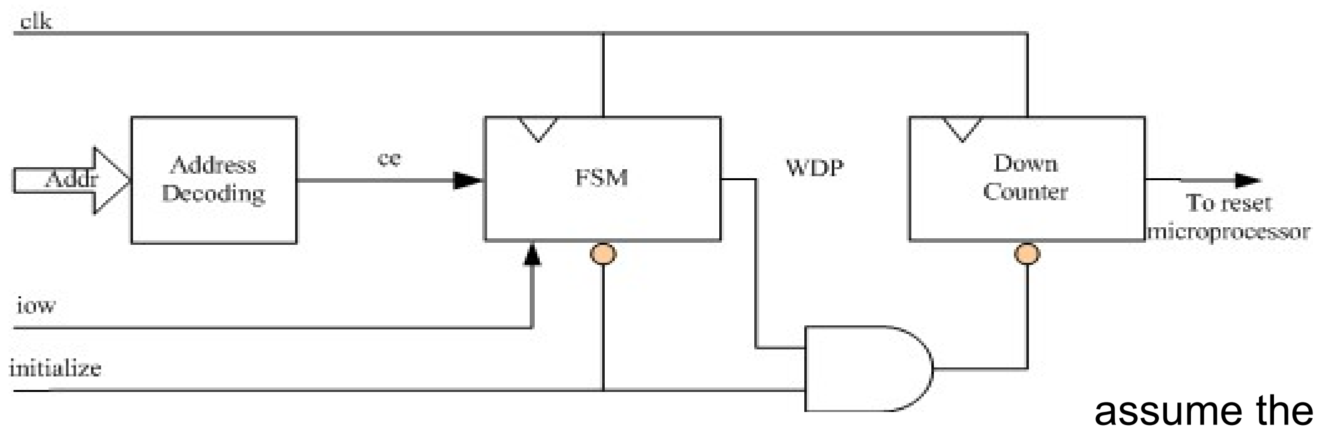
A Sample System Memory Map

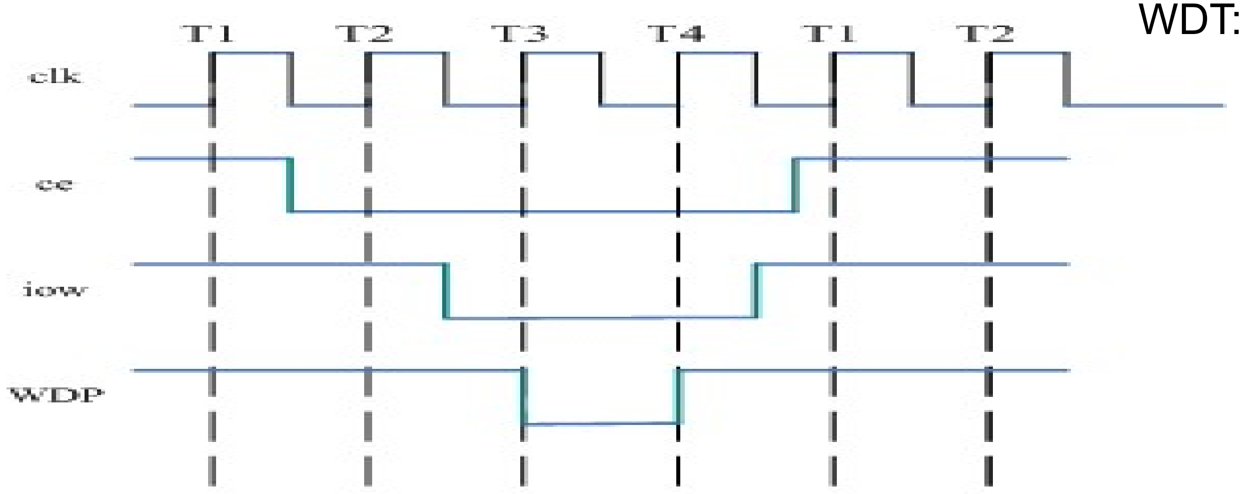
Address range Hex	Size	Device
0000-7FFF	32 KB	RAM
8000-80FF	256B	General-Purpose I/O
9000-90FF	256B	Sound controller
A000-A7FF	32 KB	Video controller/text-mapped display RAM
C000-FFFF	16 KB	ROM

Clocked Watchdog Timer (WDT)

- Addressable device that can be written to on a regular basis
- Regularly reinitialize to a known count value
- If the µcontroller does not write to the WDT between counting-down period and WDT reach zero, the µcontroller will be reset.

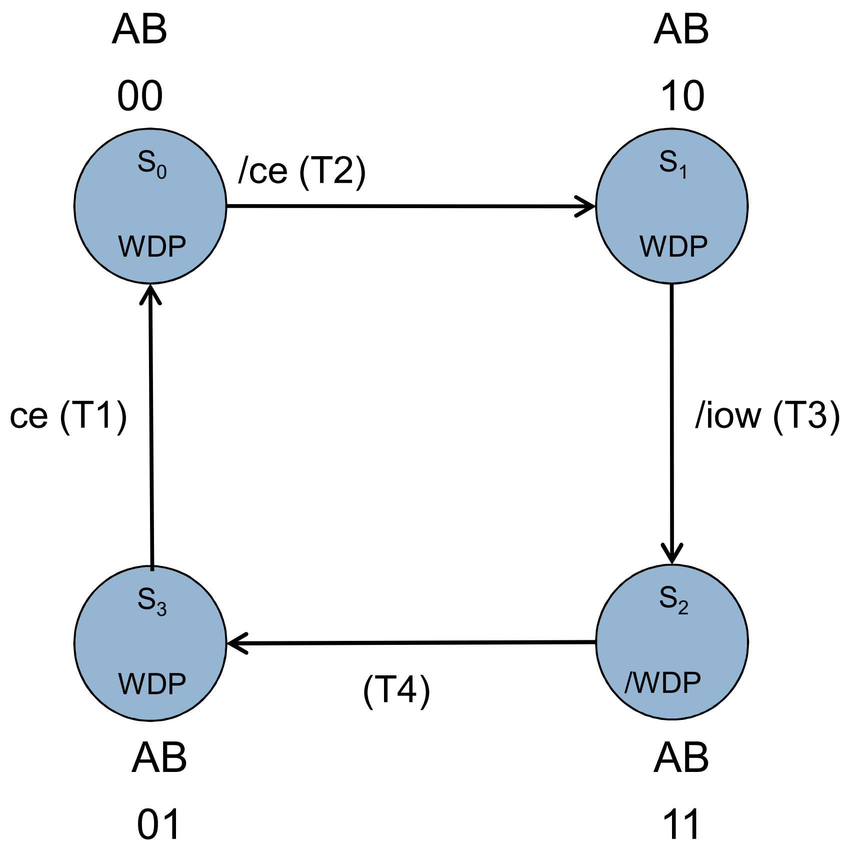
Block Diagram for a WDT





assume the address of WDT: 300h

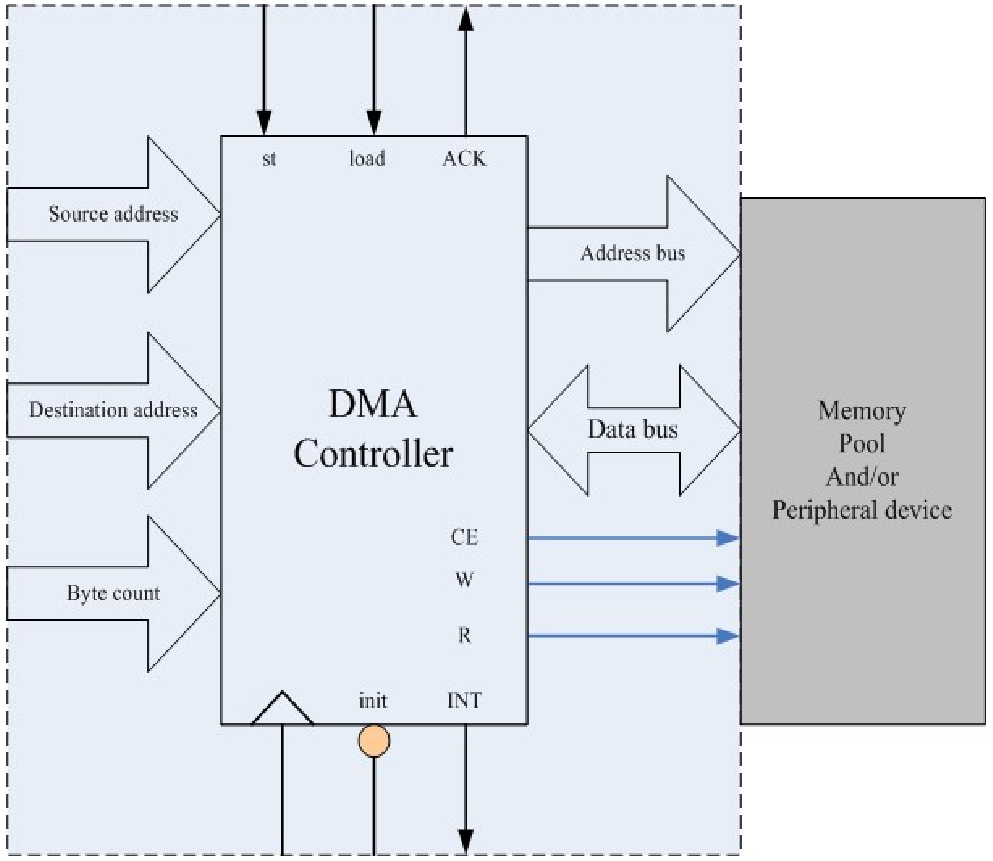
State Diagram for the WDT



One Hot Technique

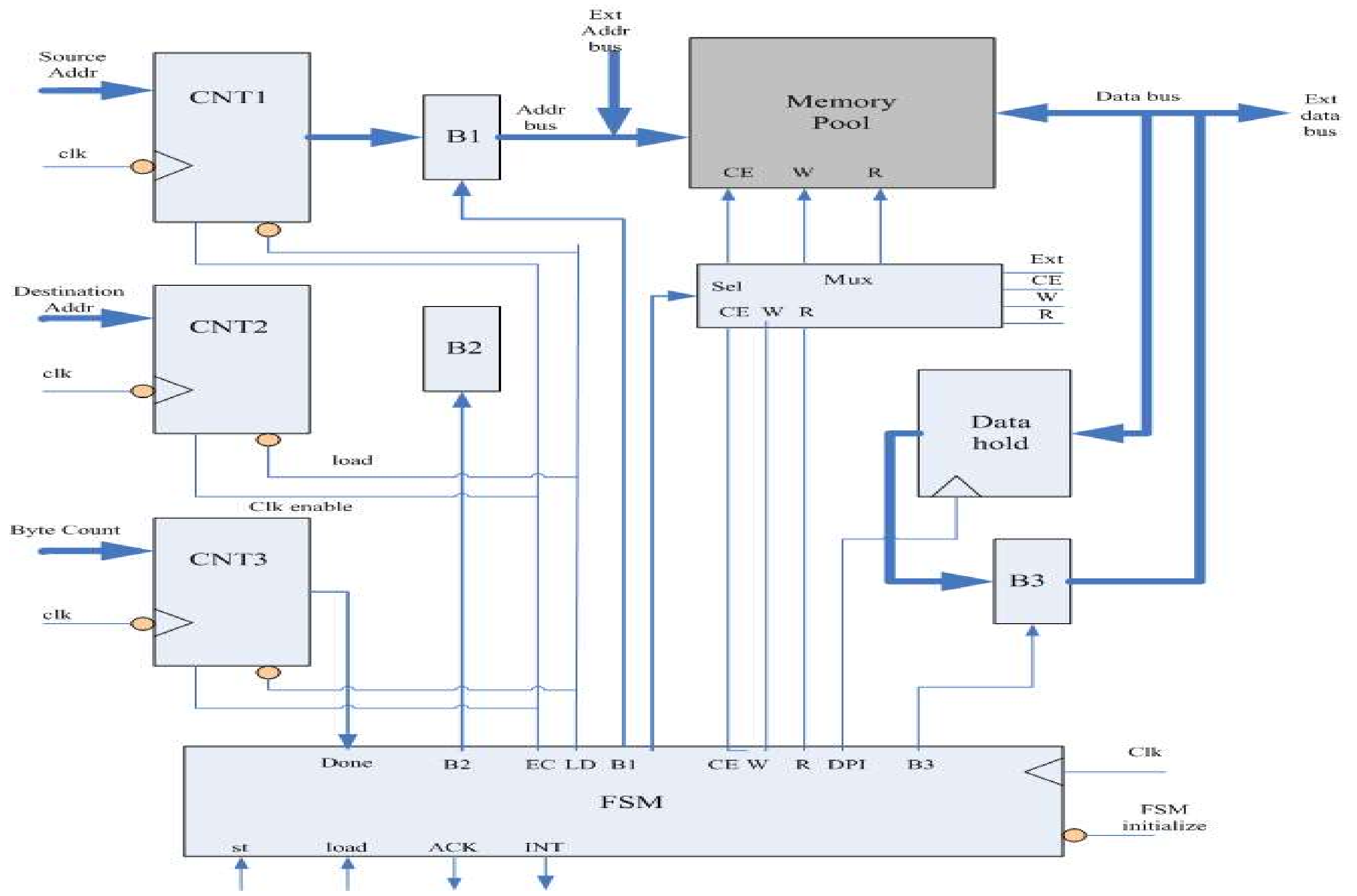
- Assign a flip-flop for each state
- Disadvantage: wasteful if the number of states is large
- Advantage:
  - in theory avoid the generation of output glitches
  - require fewer logic levels
- Often use in FPGAs since its architecture consists of many cells that can be programmed to be FFs

DMA Controller (DMAC)



- The DMAC can share the loading of the microprocessor
- Allow data to be move from one part of the memory system to another or to a peripheral device

Possible Detailed Block Diagram



General Steps

- Start (st) DMA
- Accept source, destination, and words/byte to be transferred
- Interrupt the microprocessor to let it know it is to take over the memory/peripheral
- Microprocessor isolates itself from these devices and send the load signal to DMA

Transactions

1. Select the source address and read its contents into a buffer
2. Select the destination address and deposit the buffer content into this address
3. Decrement the byte counter and advance the source & destination counter
4. Repeat 1 to 3 until all data transactions are complete (i.e., byte counter $\rightarrow$ 0)

Steps for FSM

1. Wait for the start signal st
2. Provide an interrupt to the μP to get it isolate itself from the memory
3. Wait for a load signal from the μP; when obtained, loading the source, destination, and byte count into the relevant counters
4. Source memory needs to be selected and data read from the memory into data holding registers

Steps for FSM

5. Source address needs to be isolated from the memory and the destination memory selected
6. Data in the holding register needs to be transferred into the output buffer B3 and store into the memory destination address

Steps for FSM

7. Decrement byte counter and checked if all bytes of data have been transferred
8. If there are more bytes to transfer, repeat 1 to 7 again. This is to continue until all bytes are transferred

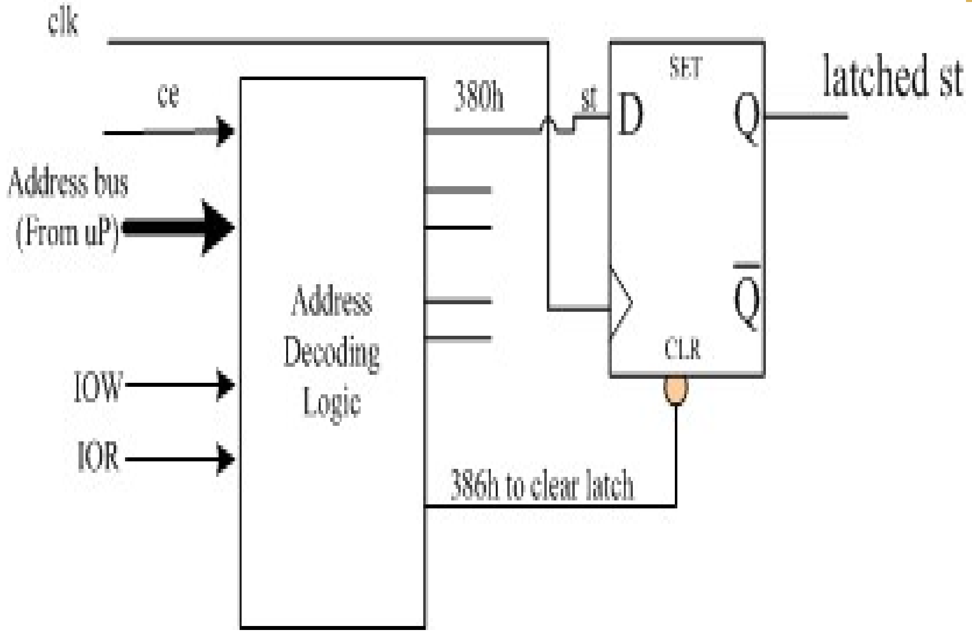
Control DMAC from μP

- Previous DMAC starts with a signal, st. Useful in avoiding address decoding logic
- A more appropriate way – via the memory (or I/O) map of the μP.

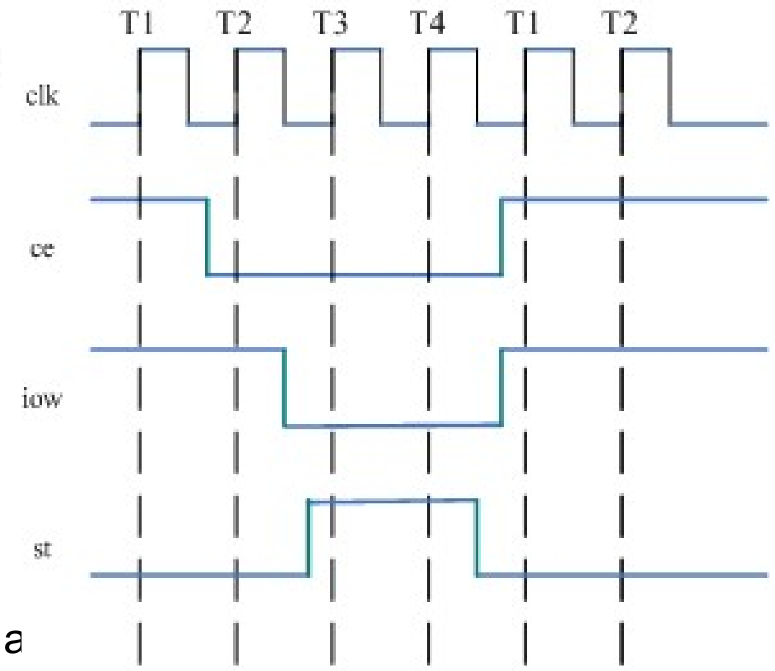
- In the following example, assume the spare address for DMAC is 380h

St from µP for the FSM

Block Diagram

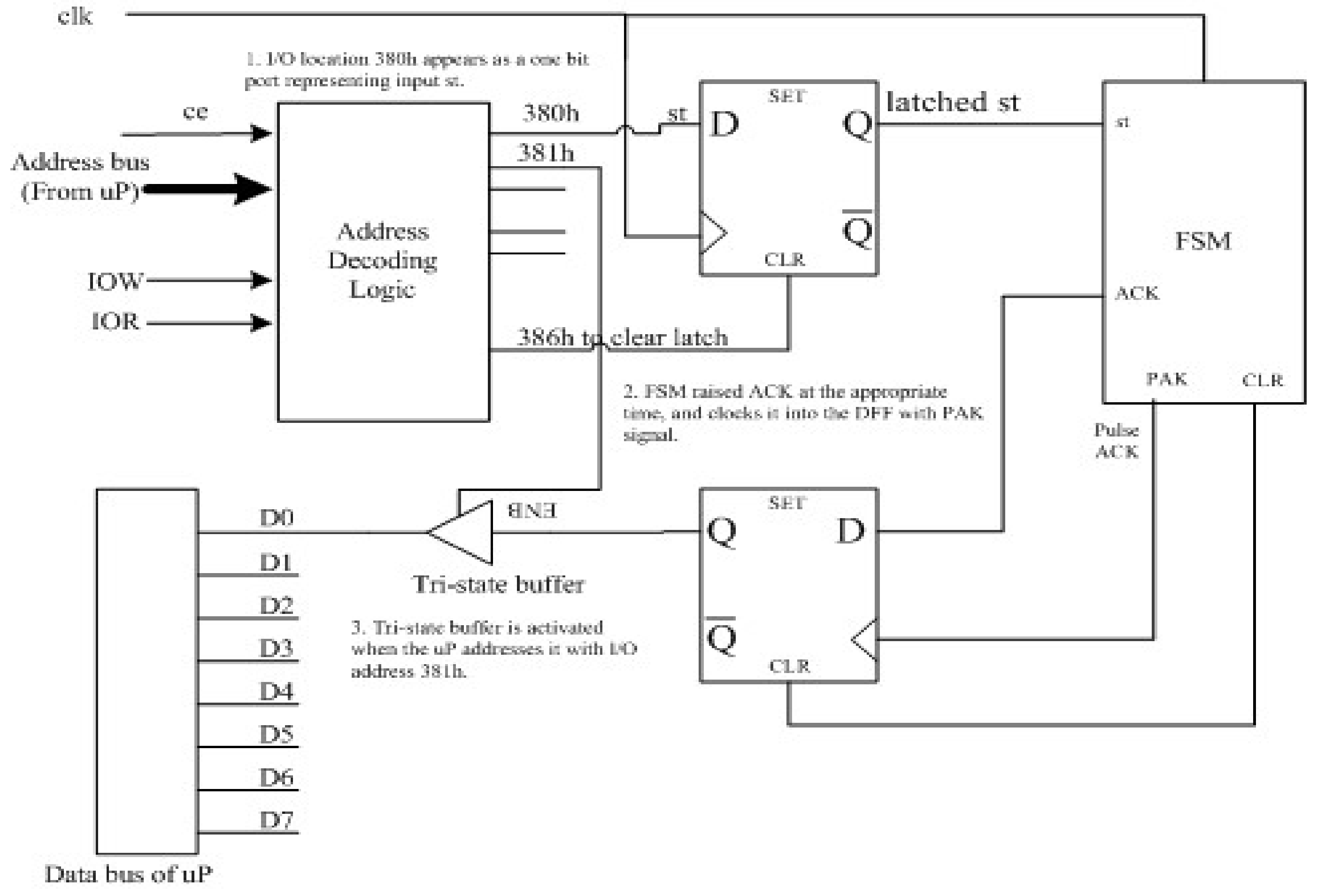


Timing waveform



I/O location 380h appears as a o/p port representing the input st. This needs to be latched into a 1-bit buffer so it can be ready by the FSM.

Whole Block Diagram



1. I/O location 380h appears as a one bit port representing input st.

2. FSM raised ACK at the appropriate time, and clocks it into the DFF with PAK signal.

3. Tri-state buffer is activated when the uP addresses it with I/O address 381h.