Lecture 17

- Logistics
  - HW5 due on Wednesday
  - HW6 will be out on Wednesday, due in one week
  - Lab6 this week
- Last lecture
  - Memory storage elements
  - State diagrams
- Today
  - Registers
  - Counters
  - Start of Finite State Machine (FSM)

The "WHY" slide

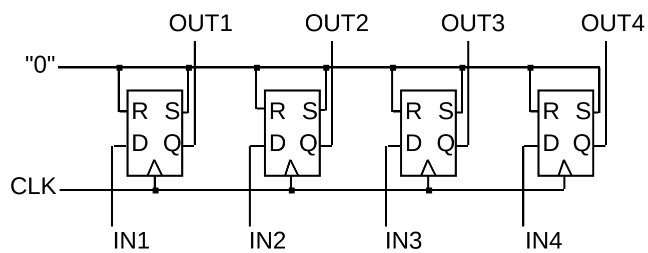
- Registers and Counters
  - Registers and counters are very simple yet powerful examples of how you can use the basic memory elements to conduct productive behavior. They are used everywhere in a computer.
- Finite State Machine
  - This is what we have been waiting for in this class. Using combinational and sequential logics, now you can design a lot of clever digital logic circuits for functional products. We will learn different steps you take to go from word problems to logic circuits in the next few lectures.

Registers

- Group of storage elements read/written as a unit.
  - Store related values (e.g. a binary word)
- Collection of flip-flops with common control
  - Share clock, reset, set lines
- Example:
  - Storage registers
  - Shift registers
  - Counters

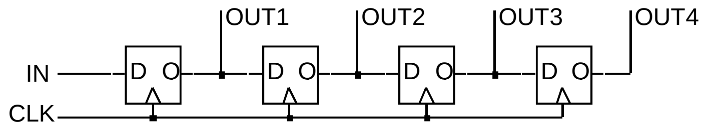
Storage registers

- Basic storage registers uses flip flops
- Example: 4 bit storage register



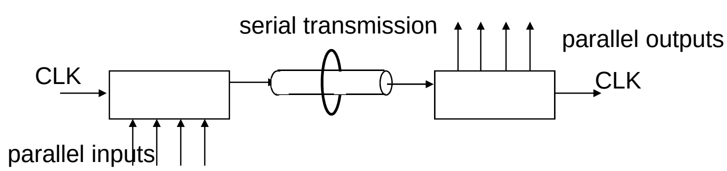
Shift registers

- Hold successively sampled input values
  - Delays values in time
  - Example: 4-bit shift register
    - Stores 4 input values in sequence

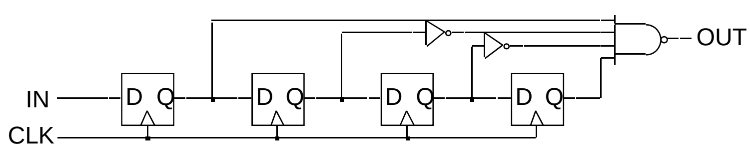


Shift-register applications

- Parallel-to-serial conversion for signal transmission

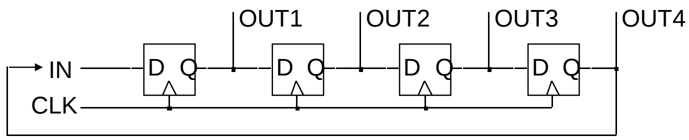


- Pattern recognition (circuit recognizes 1001)

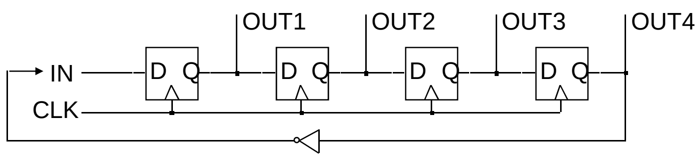


Counters

- Ring counter: Sequence is 1000, 0100, 0010, 0001
  - Assuming one of these patterns is the starting state

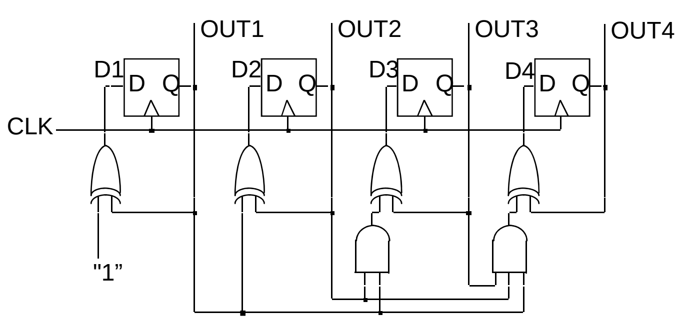


- Johnson counter: Sequence is 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000



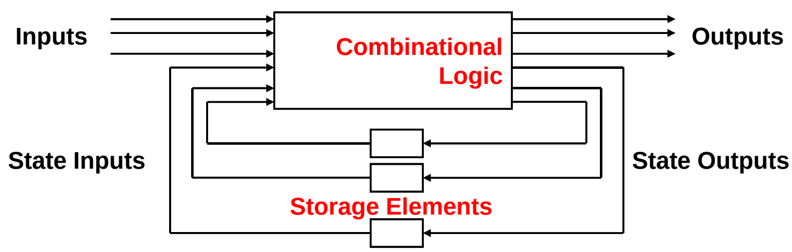
A binary counter

- Has logic between flip-flops



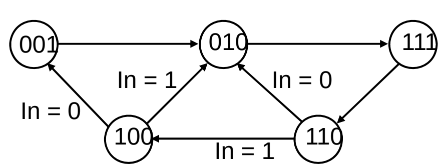
“States” for finite state machines are kept in the storage elements

- Combinational logic and storage elements
  - Localized feedback loops
  - Choice of storage elements alters the logic



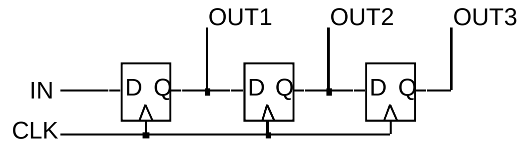
Finite-state machines (FSMs)

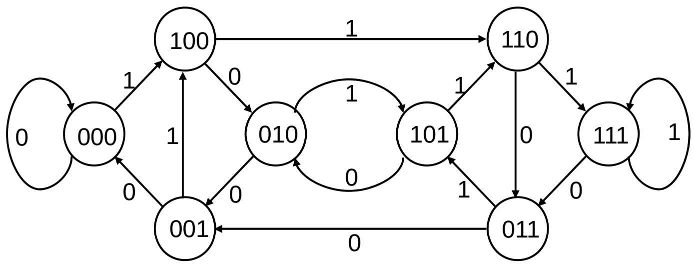
- States: Possible storage-element values
- Transitions: Changes in state
  - Clock synchronizes the state changes
- Sequential logic
  - Sequences through a series of states
  - Based on inputs and present state



Drawing state diagrams

- Show input values on transition arcs
- Show output values in state nodes





Counters revisited

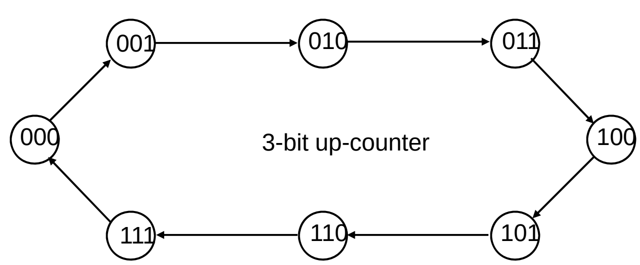
- Great simple examples of state machines
  - Output is the counter's state
- Next state is well defined
  - Does not depend on input (no inputs)

FSM design procedure (using counters)

1. Draw a state diagram
2. Draw a state-transition table
3. Encode the next-state functions
   - Minimize the logic using k-maps
4. Implement the design

We will use a '3-bit up counter' as an example

1. Draw a state diagram



2. Draw a state-transition table

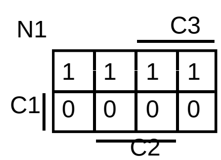
- Like a truth-table
  - State encoding is easy for counters → Use count value

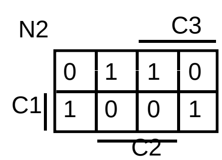
current state		next state	
0	000	001	1
1	001	010	2
2	010	011	3
3	011	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	000	0

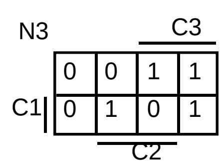
3. Encode the next state functions

- Assume D flip-flops as state elements

C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0







N1 := C1'

N2 := C1C2' + C1'C2
:= C1 xor C2

N3 := C1C2C3' + C1'C3 + C2'C3
:= C1C2C3' + (C1' + C2')C3
:= (C1C2) xor C3

4. Implement the design

- 3 flip-flops hold state
  - Counter is synchronously clocked
- Minimized logic computes next state