Digital Design Group Project EENG 28010

Finite State Machine

Dr. Faezeh Arab Hassani
Department of Electrical and Electronic Engineering

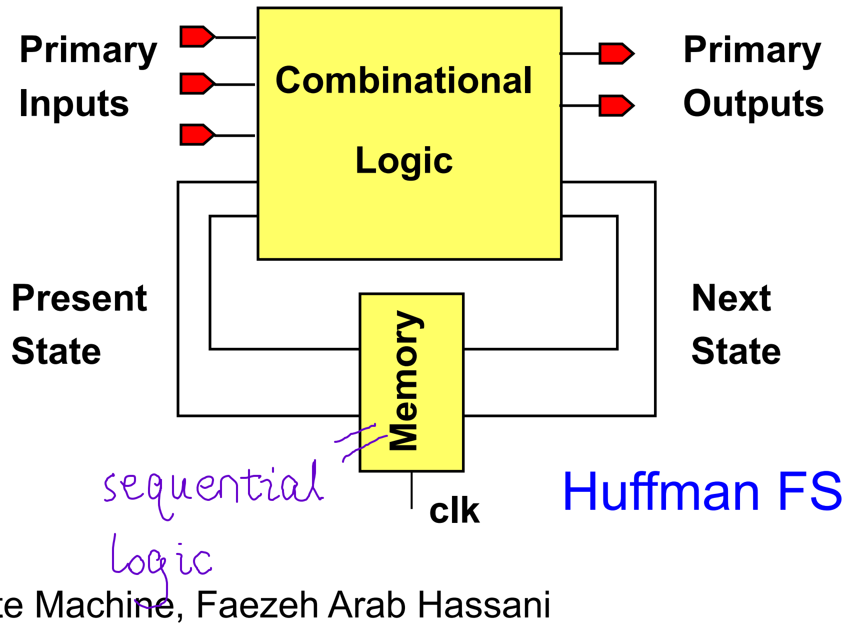
University of BRISTOL

Outline

- ❑ Mealy and Moore Machines
- ❑ Modelling a Mealy Machine
- ❑ Implementation of a Counter

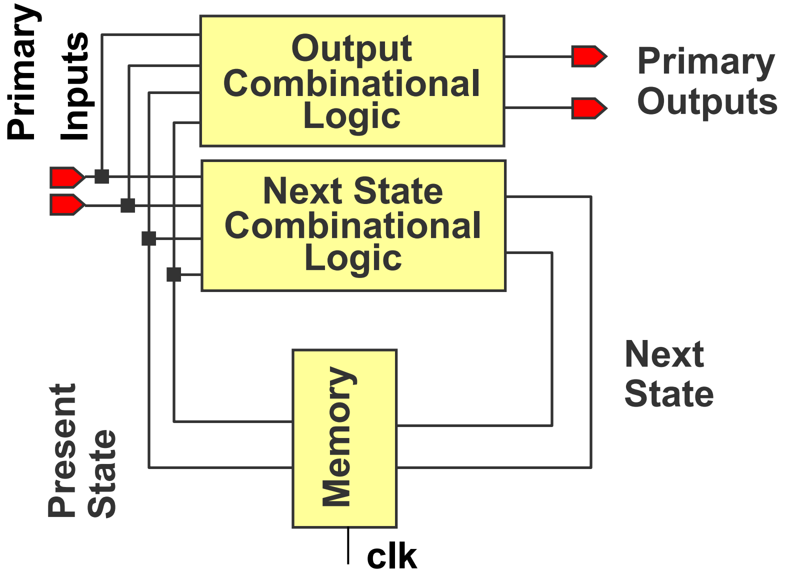
Finite State Machine (FSM) Synthesis

- Huffman model of Finite State Machines
  - FSMs can be modeled as a combinational part and a sequential part
  - Both Mealy and Moore type state machines can be described
  - Use a standard template to ensure that the synthesis tool recognizes a state machine

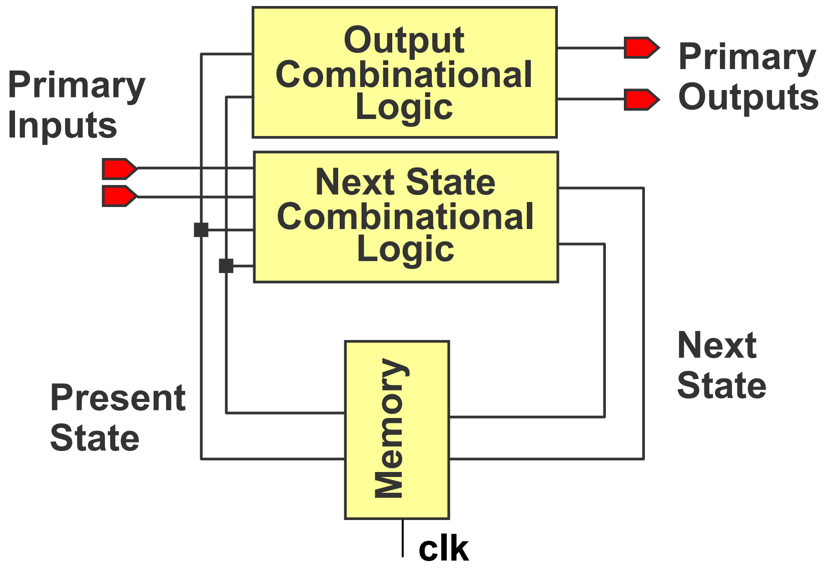


Huffman FSM Model

Mealy and Moore Machines



Mealy Machine



Moore Machine

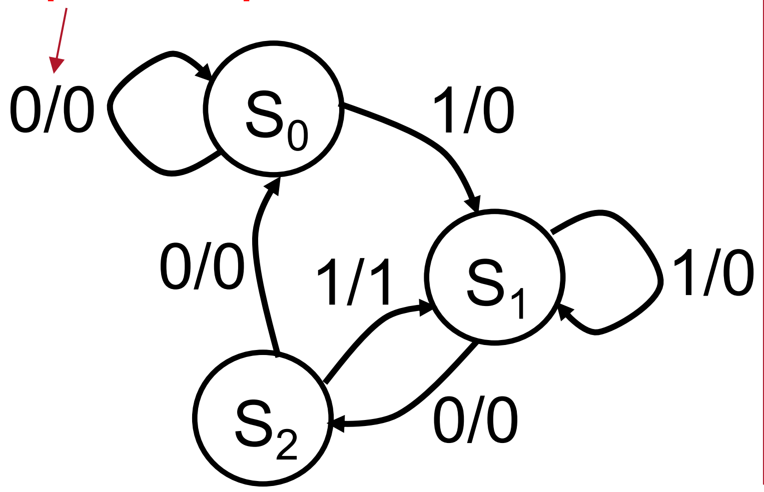
- Most straightforward way to synthesize a state machine is to use a process for each function
  - next state (combinational)
  - output (combinational)
  - Memory/state register (sequential)

Mealy Machine Design Example: Sequence Detector

- The circuit will examine a string of 0's and 1's applied to X input and generate an output Z = 1 when the input sequence ends in 1 0 1. The output Z = 1 coincides with the last 1 in 1 0 1. This circuit does not reset when a 1 output occurs. (X can only change between clock pulses.)
  - For example: X = 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0

    Z = 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0

input/output



State graph

We can implement this by 3 flip-flops for each of 3 states (i.e. one-hot approach). Or use enough flip-flops to have a combination of all states (i.e. encoded states).

2个FF 可表示4个state $2^2=4$ 00 01 10 11

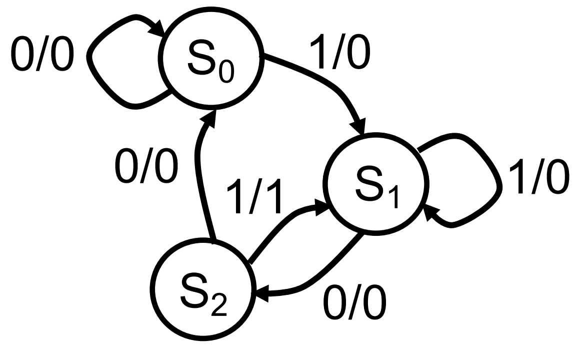
enough

Mealy Machine Design Example: Sequence Detector

$D_A = A^+$
$D_B = B^+$

Transition table

	AB	A^+B^+ X = 0	X = 1	Z X = 0	X = 1
S_0	00	00	01	0	0
S_1	01	10	01	0	0
S_2	10	00	01	0	1



AB \ X	0	1
00	0	0
01	1	0
11	x	x
10	0	0

only 1 number change

$A^+ = X'B$

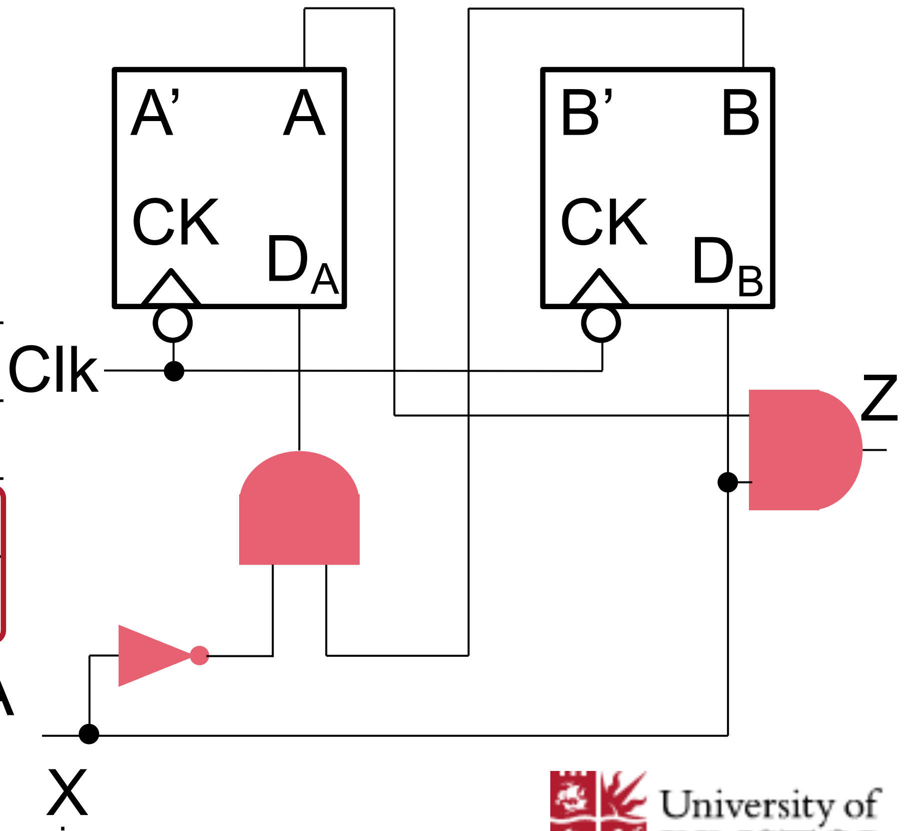
AB \ X	0	1
00	0	1
01	0	1
11	x	x
10	0	1

$B^+ = X$

AB \ X	0	1
00	0	0
01	0	0
11	x	x
10	0	1

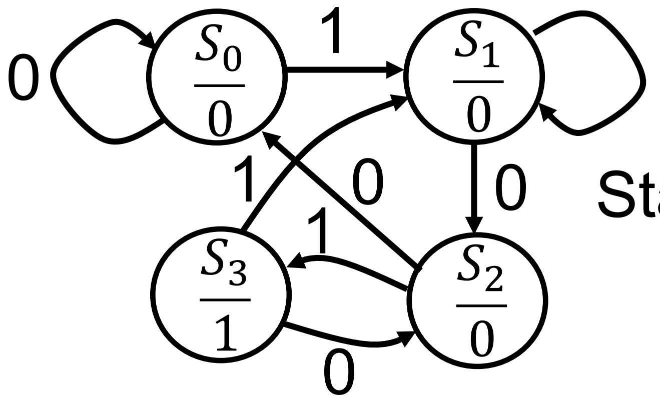
$Z = XA$

K-Maps



University of BRISTOL

Moore Machine Design Example: Sequence Detector



State graph

Transition table

	AB	A^+B^+ X = 0	A^+B^+ X = 1	Z
S_0	00	00	01	0
S_1	01	11	01	0
S_2	11	00	10	0
S_3	10	11	01	1

independent of X

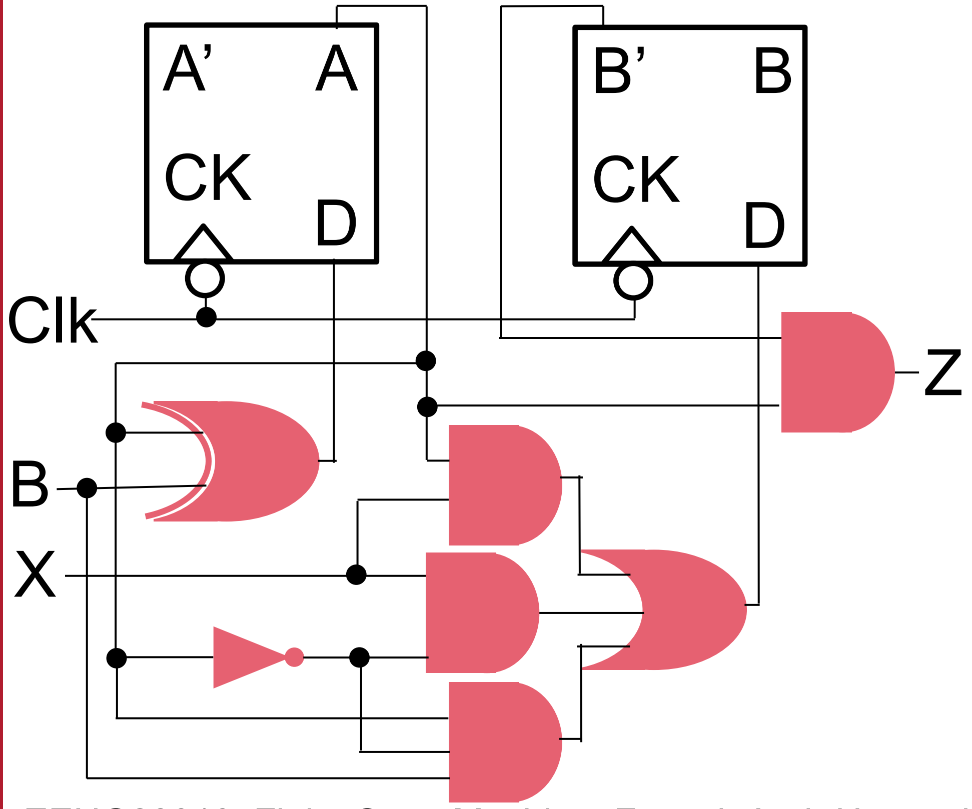
Z = AB'

AB \ X	0	1
00	0	0
01	1	1
11	0	0
10	1	1

AB \ X	0	1
00	0	1
01	0	1
11	1	0
10	0	1

$A^+ = A'B + AB'$
$= A \oplus B$

$B^+ = A'X$
$+ ABX' + AB'X$



Outline

- ❑ Mealy and Moore Machines
- ❑ Modelling a Mealy Machine
- ❑ Implementation of a Counter

Modelling a Mealy Machine

```
entity pattern_recog is
port (  X        : in     bit;
        CLK      : in     bit;
        RESET      : in     bit;
        Y          : out   bit );
end;
```

```
seq: process (clk, reset)
  begin
    if reset = '0' then
            CURRENT_STATE <= S0;
    elsif clk'event AND clk='1' then
            CURRENT_STATE <= NEXT_STATE;
    end if;
  end process; -- seq
```

```
combi_output: process(CURRENT_STATE, X)
  begin
            case CURRENT_STATE is
      when S0 =>
            Y <= 0;     ....
                        ....
```

```
combi_nextState: process(CURRENT_STATE, X)
  begin
    case CURRENT_STATE is
      when S0 =>
                        if X='0' then
                    NEXT_STATE <= S1;
                        else
                    NEXT_STATE <= S0;
                        end if;
      when S1 =>
                        if X='0' then
                    NEXT_STATE <= ?;
                        else
                        ....
      when Sn =>
                        if X='0' then
                    NEXT_STATE <= ?;
                        else
                    ~~Y <= '1';~~ -- typo in supplied code
                    NEXT_STATE <= ?;
                        end if;
    end case;
  end process; -- combi_nextState
```

University of BRISTOL

Outline

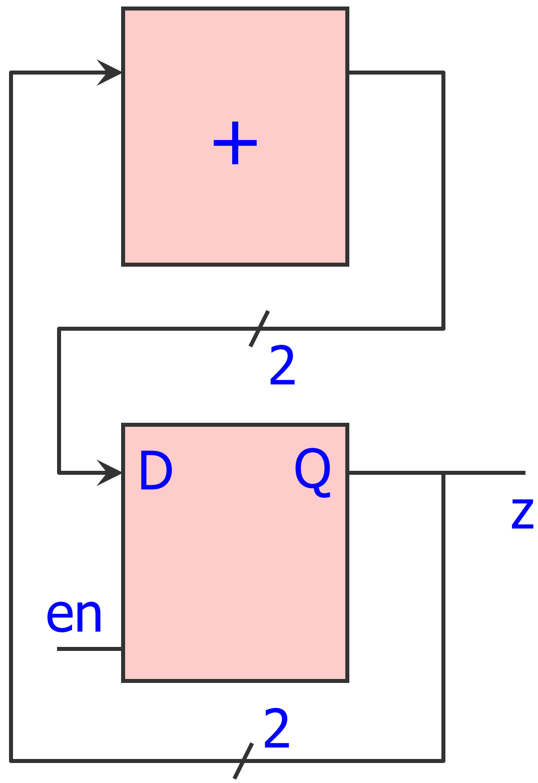
- ❑ Mealy and Moore Machines
- ❑ Modelling a Mealy Machine
- ❑ Implementation of a Counter

Wrong Implementation of a Counter

```
ENTITY incr IS
PORT (en: IN std_logic;
      z: out unsigned(0 to 1));
END incr;

ARCHITECTURE behav OF incr IS
BEGIN
      PROCESS(en)   X
          VARIABLE count: unsigned(0 to 1);
      BEGIN
          IF en = '1' THEN
              count := count +1;
          END IF;
          z <= count;
      END PROCESS;
END behav;
```

count infinitely without any aim

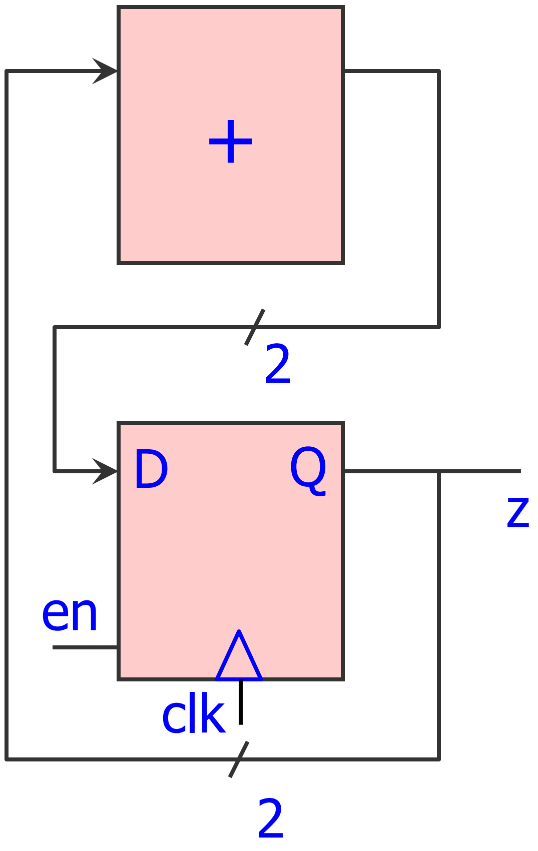


Do not use latches!

Correct Implementation of a Counter

```
ENTITY incr IS
PORT (en: IN std_logic;
      z: out unsigned(0 to 1));
END incr;

ARCHITECTURE behav OF incr IS
BEGIN
      PROCESS(clk)
            VARIABLE count: unsigned(0 to 1);
      BEGIN
            IF rising_edge(clk) THEN
                  IF en = '1' THEN    memory
                        count := count +1;
                  END IF;
            END IF;
            z <= count;
      END PROCESS;
END behav;
```



References

All of the material sourced from:

- Reference [3] in Assignment 1
- C. H. Roth and L. K. John (2017), “Digital Systems Design Using VHDL”, CENGAGE Learning.