

# CS/CoE 535

## Acceleration of Networking Algorithms in Reconfigurable Hardware

### Lecture 4

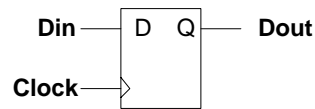
Washington University  
Fall 2001

<http://www.arl.wustl.edu/~lockwood/class/cs535/>

Copyright 2001, John W Lockwood  
Lockwood@arl.wustl.edu

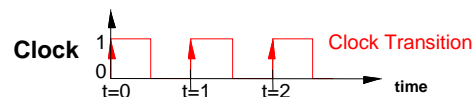
## Synchronous Storage Elements

- Values change at times governed by clock



- Clock

- Input to circuit

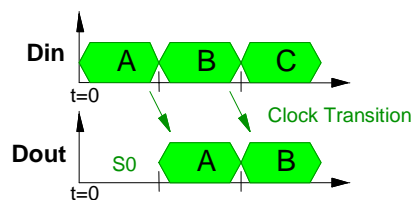


- Clock Event

- Example: Rising edge

- Flip/Flop

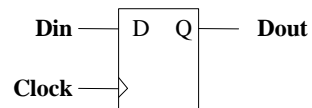
- Transfers Value From  $D_{in}$  to  $D_{out}$  on Clock event



## Edge Triggered Flop

```
entity flop is
  port (clk      : in std_logic;
        Din      : in std_logic;
        Dout     : out std_logic);
end flop;

architecture behavioral of flop is
begin
  flop:process(clk)
  begin
    if (clk='1' and clk'event) then
      Dout <= Din;
    end if;
  end process flop;
end behavioral;
```



## Components

```
architecture structural of my_module is

  component flop32
    port (clk      : in  std_logic;
          Dout     : in  std_logic_vector(31 downto 0);
          Din      : out std_logic_vector(31 downto 0));
  end component;

  ... other components ...

  DataReg: flop32
    port map (clk      => clk,
              Din      => d_mod_in,
              Dout     => d_mod_out);

  ... other connections ...

end structural;
```

## Finite State Machine (FSM)

- Terminology
  - Time:  $t$
  - State:  $S$ 
    - State variables:  $S = \{ S_1, S_2 \dots S_k \}$
    - Current state:  $S(t)$
    - Next state:  $S(t+1)$
  - State transition function  $\delta()$ 
    - Assigns next state
- State Transition
  - $S(t+1) = \delta( X , S(t) )$

## State Declaration

```
type states is (init, pad, dout);
```



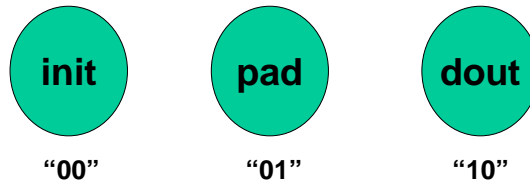
```
signal state, next_state : states;
```

## State Encoding

- Given state machine with K states
- Choose encoding to maintain state
  - Binary Encoding
    - Number of Flops:  $O(\log_2 k)$
    - Minimizes number of Flops
  - One-Hot Encoding
    - Number of Flops:  $O(k)$
    - Reduces the next-state logic
    - Uses fewer levels of logic cells
    - Enables high-speed state machines
  - Automatic Encoding
    - Feature of many synthesis tools

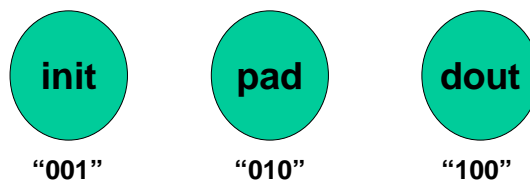
## State Declaration

- Binary Encoding



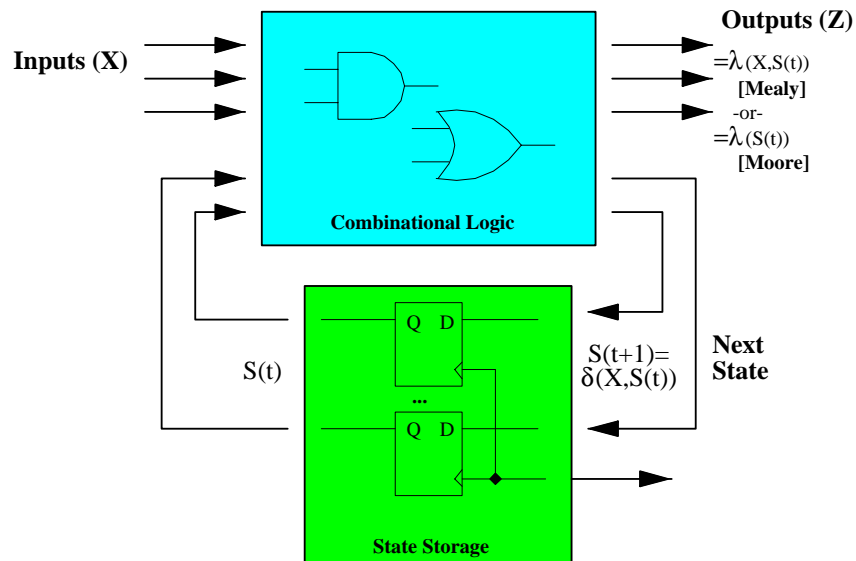
- Number of Flops = 2

- One-Hot Encoding



- Number of Flops = 3

## Implementation of the FSM

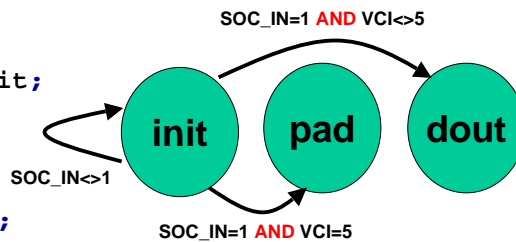


## State Transition

```

state_trans:process(state, soc_in, cntr, data_in)
begin
  case state is
    when init =>
      if (soc_in='1') then
        if (data_in(19 downto 4)=x"0005") then
          nxt_state <= pad;
        else
          nxt_state <= dout;
        end if;
      else
        nxt_state <= init;
      end if;
    ...
  end case;
end process state_trans;

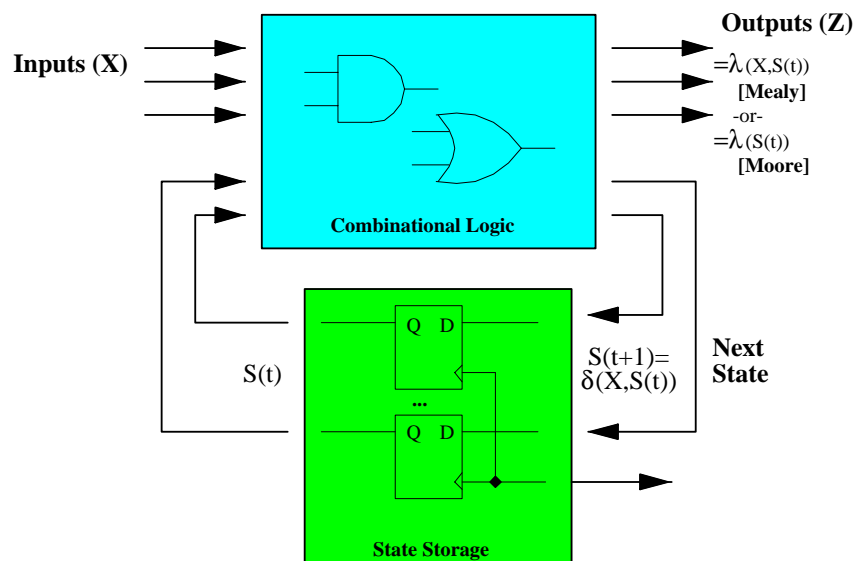
```



## State Assignment Transition

```
clkd: process(clk)
begin
  if (clk'event and clk='1') then
    if (reset='1') then
      state <= init;
    else
      state <= next_state;
    end if;
  end if;
end process clkd;
```

## Operation of a Counter



## Counter Entity

```
entity counter_4_bit is
  port (clk      : in std_logic;
        reset    : in std_logic;
        cnt_en   : in std_logic;
        load_cntr : in std_logic;
        init_val  : in std_logic_vector(3 downto 0);
        output    : out std_logic_vector(3 downto 0));
end counter_4_bit;

architecture behavioral of counter_4_bit is
  signal cnt : unsigned(3 downto 0);
begin
  ... See next page ...
  output <= std_logic_vector(cnt);
end behavioral;
```

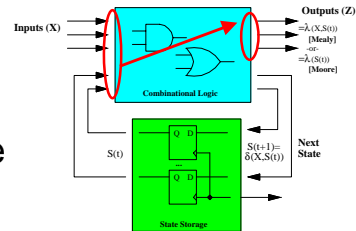
## Count Process

```
count: process(clk)
begin
  if (clk'event and clk='1') then
    if (reset='0') then
      cnt <= (others => '0');
    else
      if (load_cntr='1') then
        cnt <= unsigned(init_val);
      elsif (cnt_en='1') then
        cnt <= cnt + 1;
      else
        cnt <= cnt;
      end if;
    end if;
  end if;
end process count;
```

## Mealy and Moore FSMs

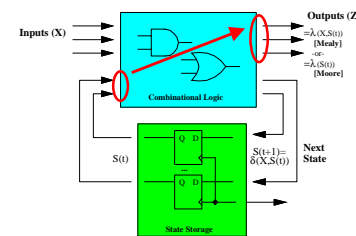
- Mealy Machine

- Output ( $Z$ ) =  $\lambda(X, S(t))$
- Output is a combinatorial function of input and state



- Moore Machine

- Output ( $Z$ ) =  $\lambda(S(t))$
- Output is a combinatorial function only of state
- Consider effect of input transition



## FSM Outputs

```
cnt_en <= '1' when (state=dout) else '0';
    -- Moore Output
```

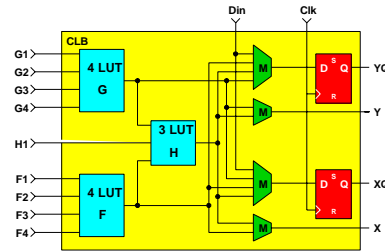
```
data_sel <= '0' when
  (((state=init) and (soc_in='1')) or
  ((state=dout) and (cntr="1101") and (soc_in='1')) or
  ((state=dout) and (cntr<"1101")))
  else '1';
    -- Mealy Output
```

## Implementation of FSM in FPGA

- LookUp Tables (LUTs)

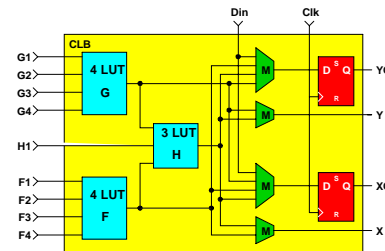
- LUTs generates next-state transition

- Function of Input and current state
- $Next\_state = \delta(X,S)$



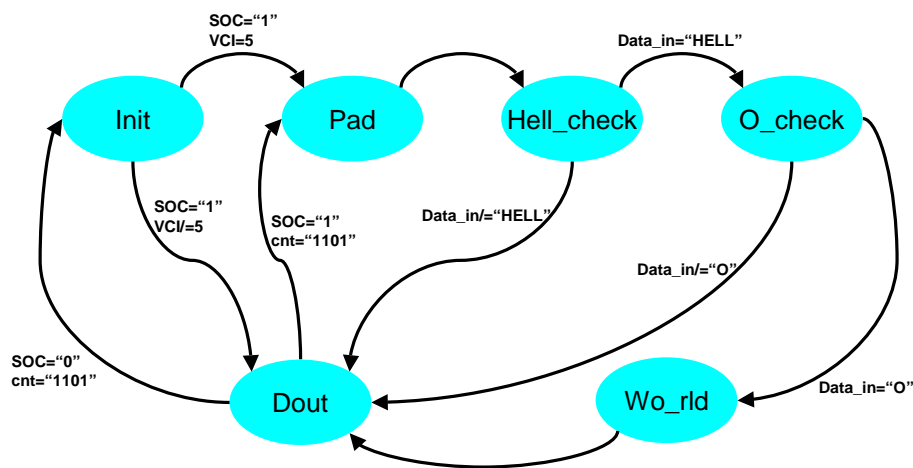
- LUTs generates outputs

- Output =  $\lambda(S)$  (Moore)
- Output =  $\lambda(X,S)$  (Mealy)

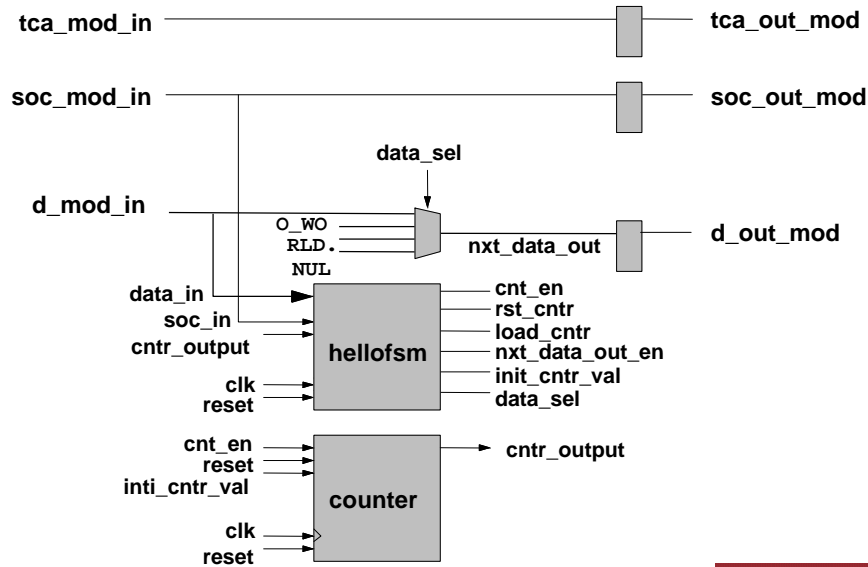


- Flops store current state

## Finite State Machine Bubble Diagram



# Hello, World Structure



# Timing Diagram for Hello World

