Digital Systems - 6

Pipelines. Sequential Datapath and Control Section. State Machines

Pere Palà Schönwälder

iTIC http://itic.cat

March 2023

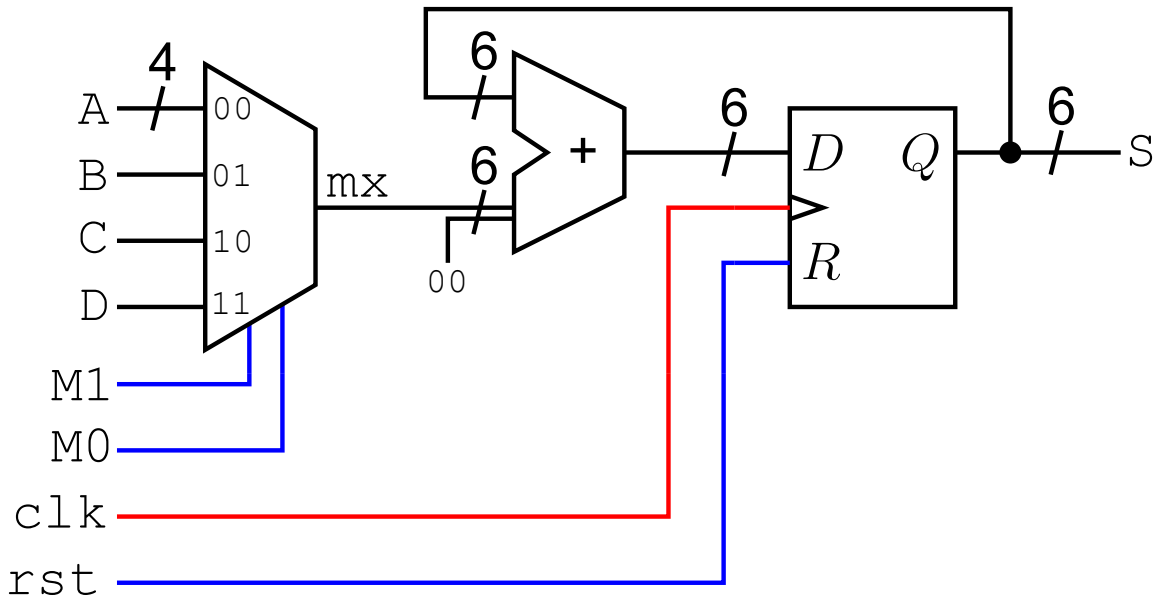
Problem: Long Combinatorial Paths

- Long and/or complex combinatorial paths
- Limited resources

- Pipeline
- Introduces delays

Sequential Datapath and Control Section

- Objective: To add 4 numbers of 4 bits
- Restriction: Only one adder.
- Datapath
- Control Section



Datapath

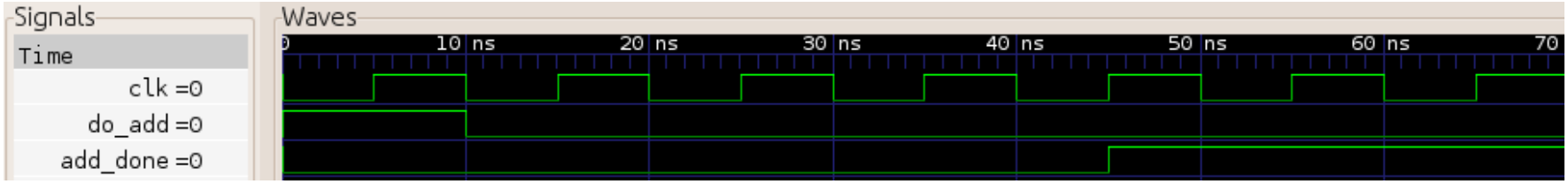
```
signal A, B, C, D, MuxOut     : std_logic_vector(3 downto 0);
signal M                      : std_logic_vector(1 downto 0);
signal addA, addB, addOut, Q  : std_logic_vector(5 downto 0);
signal clk, rst               : std_logic;
signal do_add, add_done       : std_logic;
begin
  with M select
    MuxOut <= A when "00", B when "01",
              C when "10", D when "11", "----" when others;
  addA   <= Q;
  addB   <= "00" & mx;
  addOut <= std_logic_vector(unsigned(addA)+unsigned(addB));

  process(clk)
  begin
    if rising_edge(clk) then
      if rst='1' then
        Q <= (others => '0');
      else
        Q <= addOut;
      end if;
    end if;
  end process;
end;
```

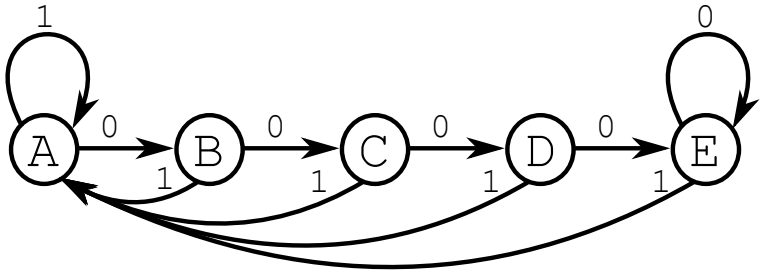
Control Section

Control Signals

- Trigger the process
- Indicate completion



State Transition Diagram



Depending on the signal `do_add`

State Transitions

Present State	Next State	Actions
A	A, if `do_add=1`	`M <= "00"`
	B, if `do_add=0`	`add_done <= '0'`
B	A, if `do_add=1`	`M <= "01"`
	C, if `do_add=0`	`add_done <= '0'`
C	A, if `do_add=1`	`M <= "10"`
	D, if `do_add=0`	`add_done <= '0'`
D	A, if `do_add=1`	`M <= "11"`
	E, if `do_add=0`	`add_done <= '0'`
E	A, if `do_add=1`	`M <= "--"`
	E, if `do_add=0`	`add_done <= '1'`

State Machine Coding

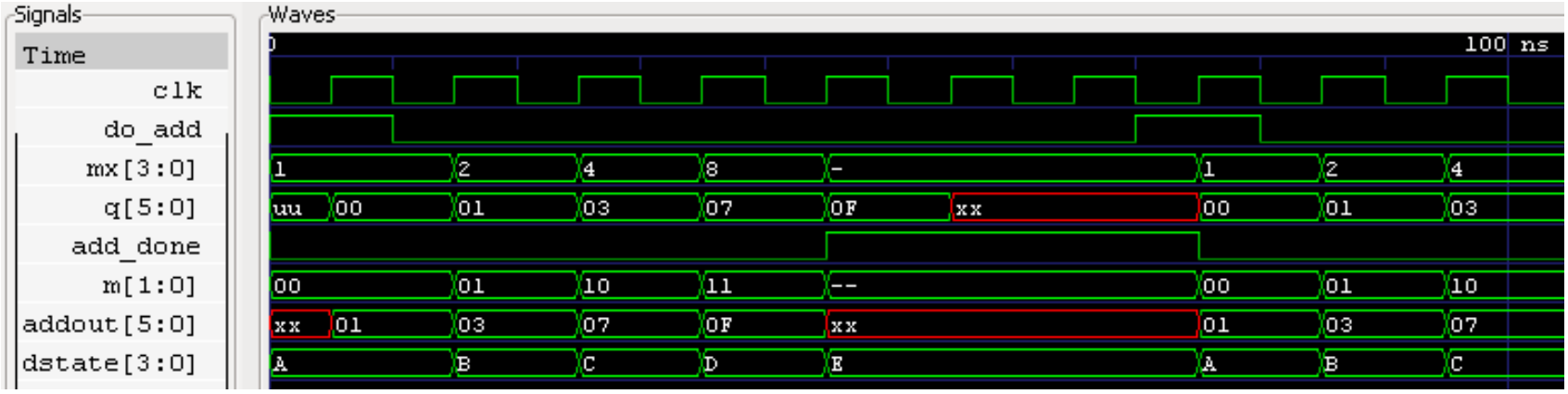
```
type t_state is (sA,sB,sC,sD,sE);
signal pr_state,nx_state : t_state;
```

```
  process(clk)
  begin
    if rising_edge(clk) then
       pr_state <= nx_state;
    end if;
  end process;
```

```
  rst<=do_add;
```

```
  process(pr_state,do_add) -- In VHDL 2008 process(all) allowed
  begin
    case pr_state is
    when sA =>  --actions corresponding to state A
      M         <= "00";     --1: Outputs. Caution!!!
      add_done <= '0';       --Do not forget any output!
      if do_add='1' then
        nx_state <= sA;      --2: Compute next state
      else                   --depending on present state
        nx_state <= sB;      --and the input(s)
      end if;
    when sB =>  --actions corresponding to state B
  ...
```

Simulation Results



- Showing 1+2+4+8 = x0F = 001111
- `add_done` signals when the result is available.
- Has to be read on the next clock edge!
- Additional signal `dstate` created to debug the state.