

Forelesning 05

Regular Sequential Circuits

Hieu Nguyen

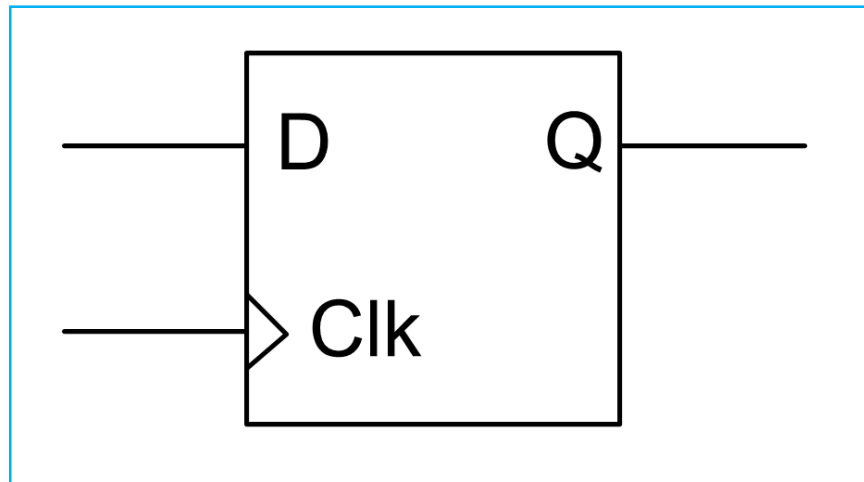
Innledning

- En sekvensielle krets er en krets med **minne** som danner **intern tilstand** for kretsen
- Forskjell fra kombinasjonskretser: utgangssignaler er en funksjon av både inngangssignaler og interne tilstand
- **Synkron** metode er mest brukt i praksis i design av sekvensielle kretser
 - Alle lagringselementer er kontrollert (synkronisert) av et **globalt** klokkesignal, og data blir prøvd/samlet og lagret på enten **stigningsflanke** eller **fallflanke** av klokkesignalet.
 - Skille mellom lagringskomponenter fra kretsen, og derfor blir utviklingsprosess **sterkt forenklet**

D-FF og Register

- D-FF er en av mest grunnleggende lagringskomponenter i sekvensielle kretser
- Symbolet og funksjonstabell av en positive flanke-triggeret D-FF

Symbol



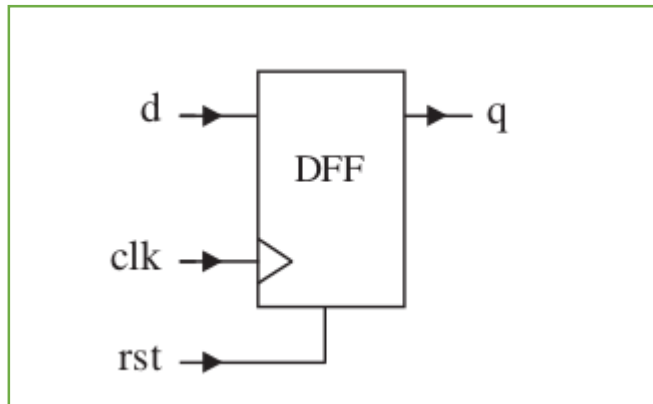
Sannhetstabell

CLK	Q*
0	Q
1	Q
	D

D-FF med Asynkron Nullstilling/Reset

- Vi bruker nullstillingssignal til å sette D-FF til '0'
- Operasjon av nullstillingssignalet er avhengig av klokkesignalet

Symbol

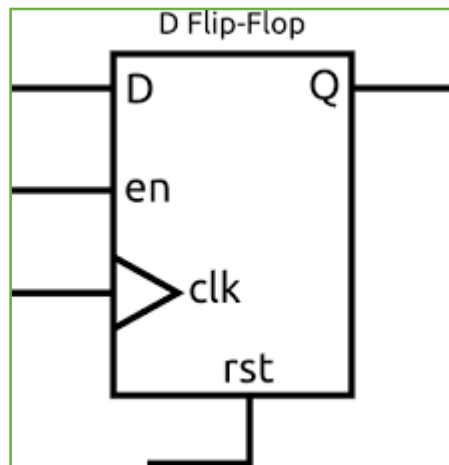


Sannhetstabell

Rst/Nullstilling	Clk	Q*
1	-	0
0	0	Q
0	1	Q
0	↑	D

D-FF med Synkron 'Enable'

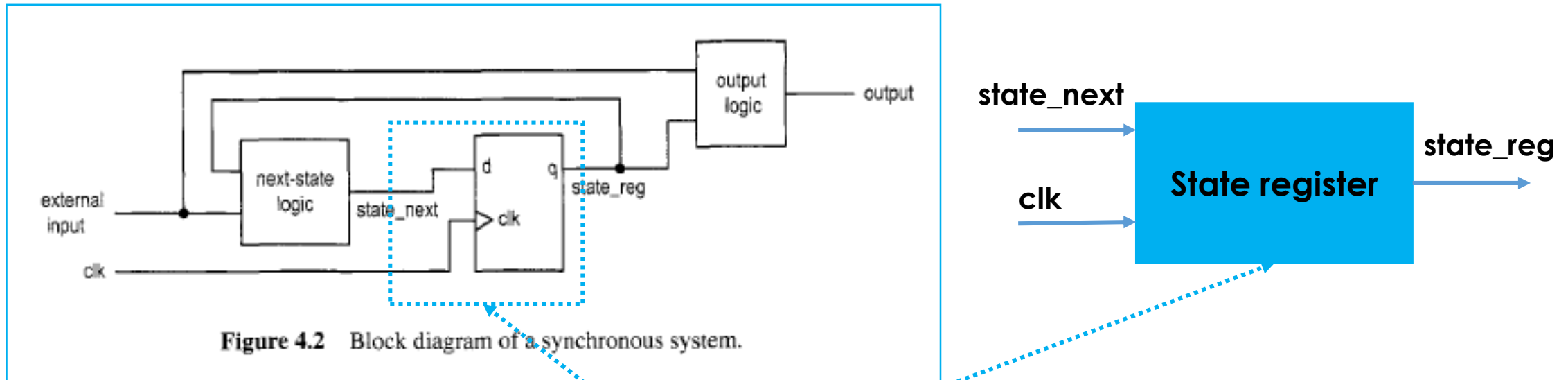
Symbol



Sannhetstabell

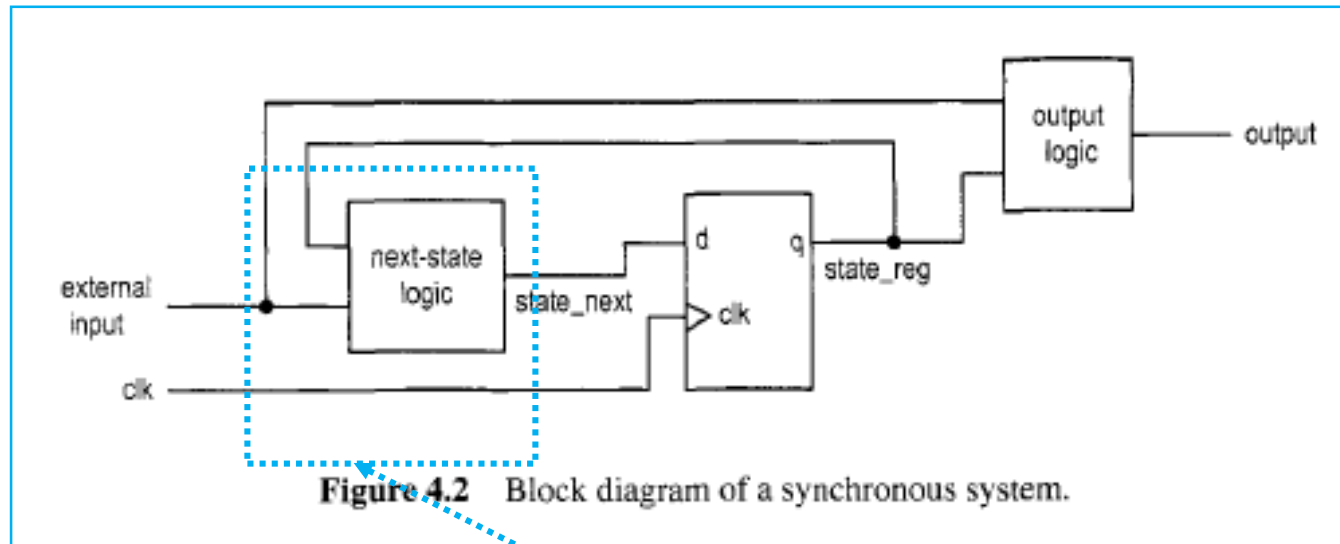
Rst	Clk	En	Q*
1	-	-	0
0	0	-	Q
0	1	-	Q
0		0	Q
0		1	D

Synkroner Systemer (1)

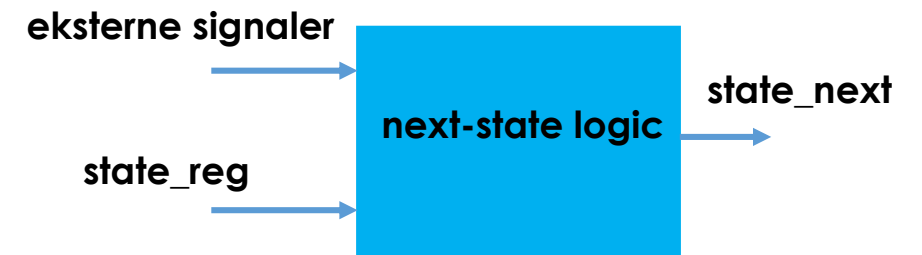


State register/ Tilstandsregister: En samling av D-FF-er som blir kontrollert av et felles klokkesignal

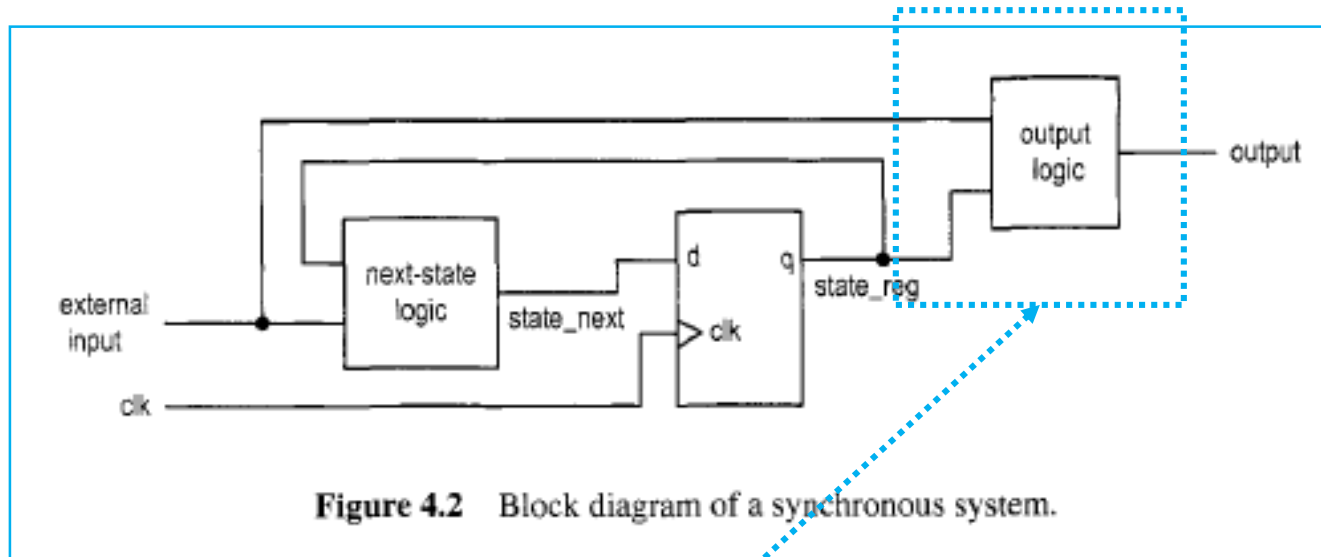
Synkroner Systemer (2)



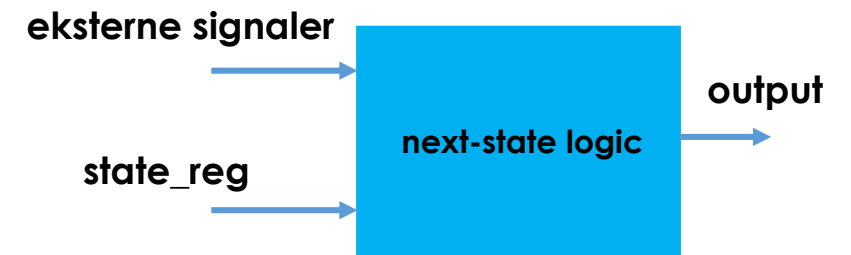
Next-state logikk: er kombinasjonskrets som har eksterne signaler og intern tilstand som inngangssignaler.



Synkroner Systemer (3)



Outputlogikk: er kombinasjonskrets som genererer utgangssignaler



Synkrone Systemer (4)

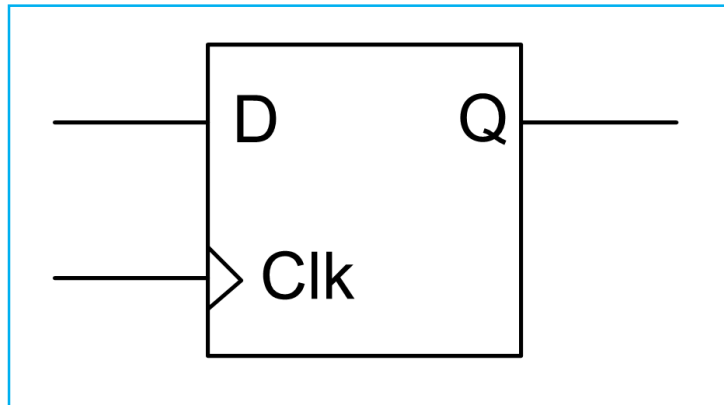
State register/ Tilstandsregister: En samling av D-FF-er som blir kontrollert av et felles klokkesignal

Next-state logikk: er kombinasjonskrets som har eksterne signaler og intern tilstand som inngangssignaler.

Output-logikk: er kombinasjonskrets som genererer utgangssignaler

VHDL-koder for sekvensielle kretser

D-FF



CLK	Q*
0	Q
1	Q
↑	D

```
library ieee;
use ieee.std_logic_1164.all;

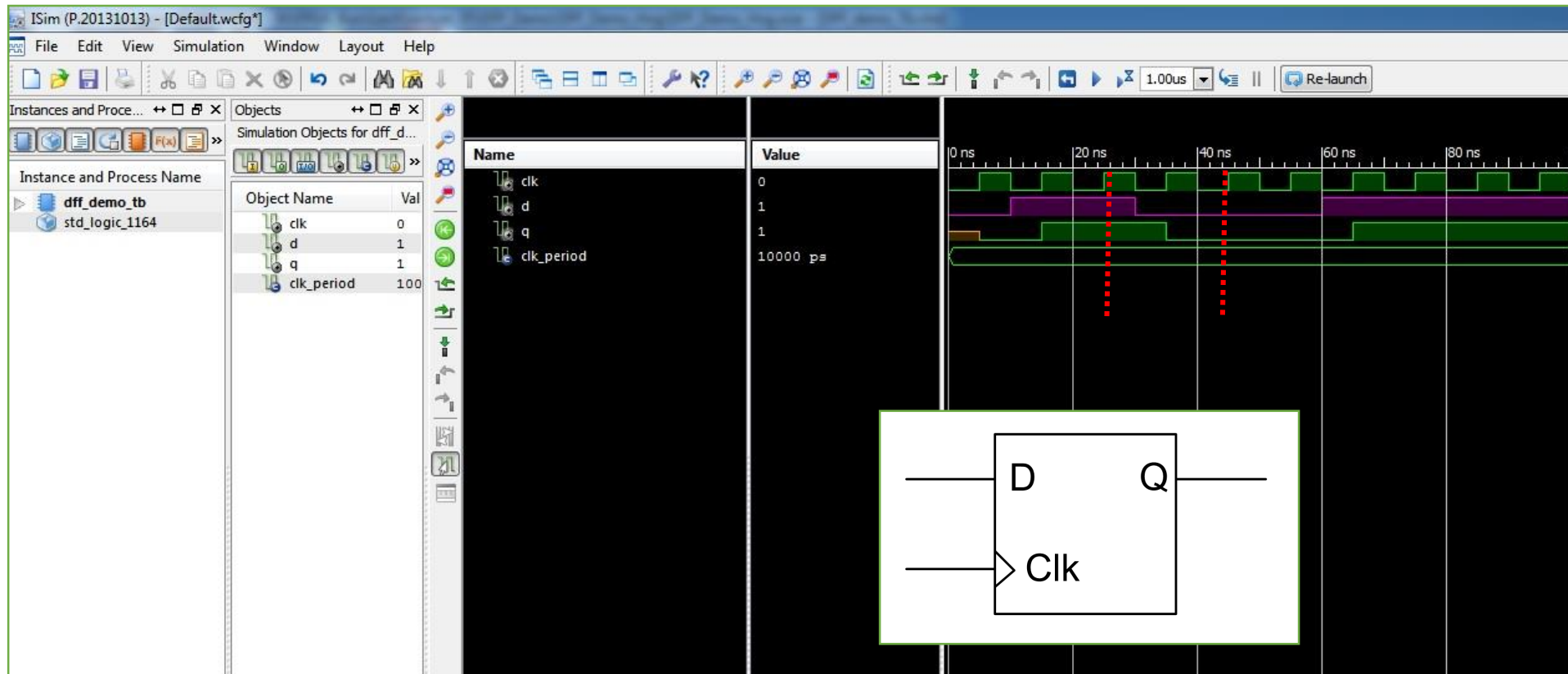
entity d_ff is
    port(
        clk: in std_logic;
        d: in std_logic;
        q: out std_logic
    );
end d_ff;

architecture arch of d_ff is
begin
    process(clk)
    begin
        if (clk'event and clk='1') then
            q <= d;
        end if;
    end process;
end arch;
```

VHDL-koder for sekvensielle kretser

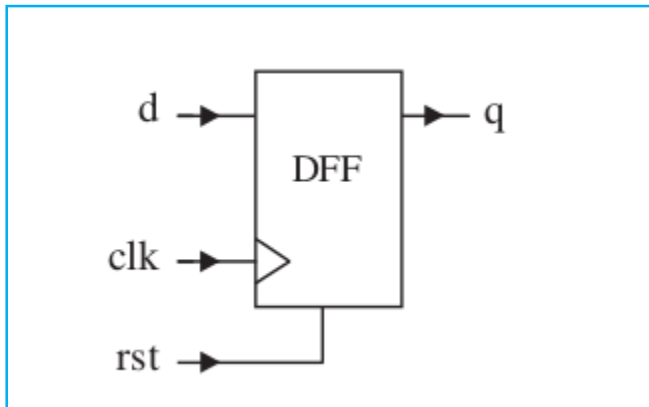
D-FF


- Simuleringskurveform



VHDL-koder for sekvensielle kretser

D-FF med nullstillingssignalet



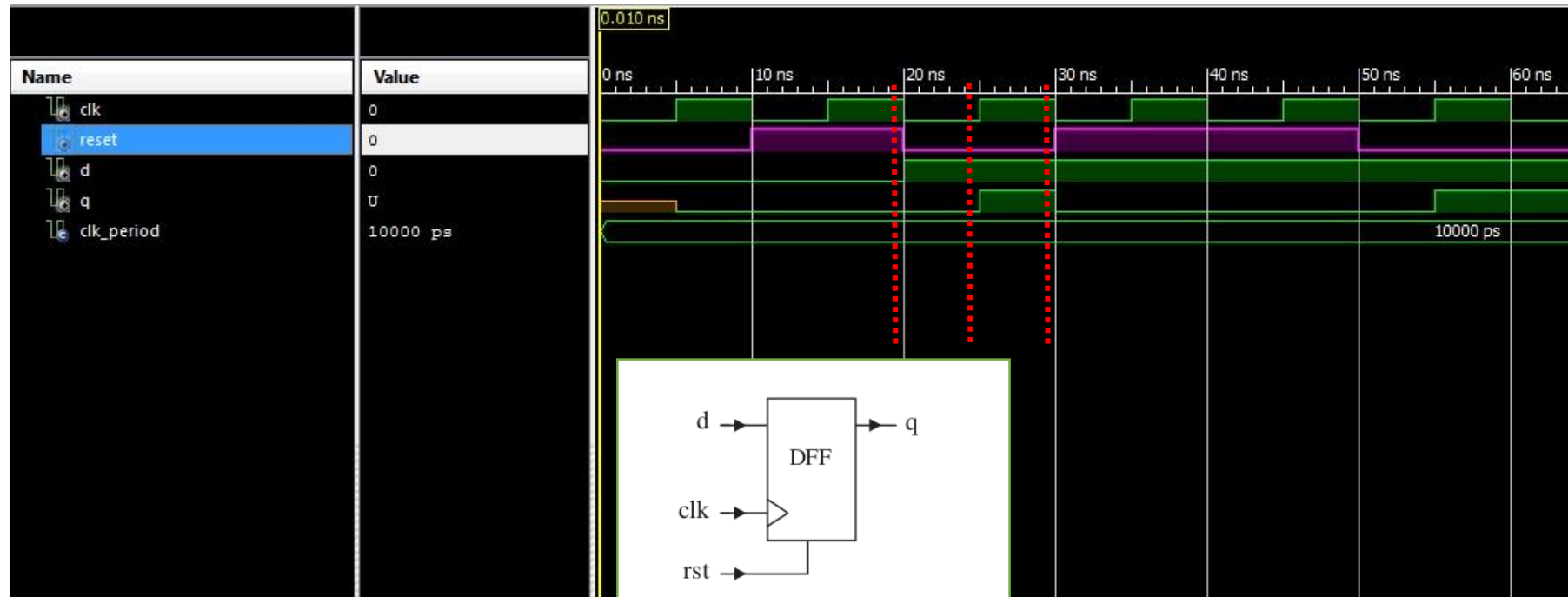
Rst/Nullstilling	Clk	Q*
1	-	0
0	0	Q
0	1	Q
0		D

```
2 library ieee;
3 use ieee.std_logic_1164.all;
4
5 entity d_ff_reset is
6     port(
7         clk, reset: in std_logic;
8         d: in std_logic;
9         q: out std_logic
10    );
11 end d_ff_reset;
12
13 architecture arch of d_ff_reset is
14 begin
15     process(clk, reset)
16     begin
17         if (reset='1') then
18             q <= '0';
19         elsif (clk'event and clk='1') then
20             q <= d;
21         end if;
22     end process;
23 end arch;
```

VHDL-koder for sekvensielle kretser

D-FF med nullstillingssignalet

- Simuleringskurveform



VHDL-koder for sekvensielle kretser

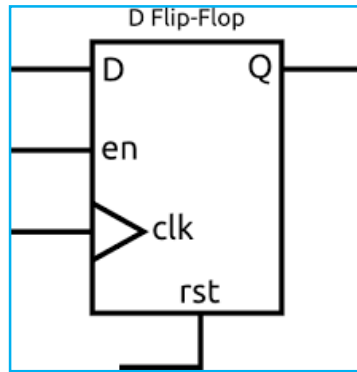
- D-FF vs. D-FF med nullstillingssignalet



```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity d_ff is
5     port(
6         clk: in std_logic;
7         d: in std_logic;
8         q: out std_logic
9     );
10 end d_ff;
11
12 architecture arch of d_ff is
13 begin
14     process(clk)
15     begin
16         if (clk'event and clk='1') then
17             q <= d;
18         end if;
19     end process;
20 end arch;
```

```
2 library ieee;
3 use ieee.std_logic_1164.all;
4
5 entity d_ff_reset is
6     port(
7         clk, reset: in std_logic;
8         d: in std_logic;
9         q: out std_logic
10    );
11 end d_ff_reset;
12
13 architecture arch of d_ff_reset is
14 begin
15     process(clk, reset)
16     begin
17         if (reset='1') then
18             q <= '0';
19         elsif (clk'event and clk='1') then
20             q <= d;
21         end if;
22     end process;
23 end arch;
```

VHDL-koder for sekvensielle kretser

D-FF med enable-signal



Rst	Clk	En	Q*
1	-	-	0
0	0	-	q
0	1	-	q
0		0	Q
0		1	D

```
library ieee;
use ieee.std_logic_1164.all;

entity d_ff_en is
    port(
        clk, reset: in std_logic;
        en: in std_logic;
        d: in std_logic;
        q: out std_logic
    );
end d_ff_en;

architecture arch of d_ff_en is
begin
    process(clk, reset)
    begin
        if (reset='1') then
            q <='0';
        elsif (clk'event and clk='1') then
            if (en='1') then
                q <= d;
            end if;
        end if;
    end process;
end arch;
```

TestBench For Sekvensielle Kretser (1)

- Library & Entity

```
28  LIBRARY ieee;  
29  USE ieee.std_logic_1164.ALL;  
30  
31  -- Uncomment the following library declaration if using  
32  -- arithmetic functions with Signed or Unsigned values  
33  --USE ieee.numeric_std.ALL;  
34  
35  ENTITY dff_en_tb IS  
36  END dff_en_tb;
```


TestBench For Sekvensielle Kretser (2)

- Architecture | Component Declaration

```
38 ARCHITECTURE behavior OF dff_en_tb IS
39
40     -- Component Declaration for the Unit Under Test (UUT)
41
42     COMPONENT d_ff_en
43     PORT(
44         clk      : IN  std_logic;
45         reset    : IN  std_logic;
46         en       : IN  std_logic;
47         d        : IN  std_logic;
48         q        : OUT std_logic
49     );
50     END COMPONENT;
```

TestBench For Sekvensielle Kretser (3)

- Architecture | Testbench Signal Declaration

```
52
53  --Inputs
54  signal clk    : std_logic := '0';
55  signal reset  : std_logic := '0';
56  signal en     : std_logic := '0';
57  signal d      : std_logic := '0';
58
59  --Outputs
60  signal q : std_logic;
61
62  -- Clock period definitions
63  constant clk_period : time := 10 ns;
64
```

TestBench For Sekvensielle Kretser (4)

- Architecture | UUT or Component Instantiation

```
65 BEGIN
66
67     -- Instantiate the Unit Under Test (UUT)
68     uut: d_ff_en PORT MAP (
69         clk    => clk,
70         reset  => reset,
71         en     => en,
72         d      => d,
73         q      => q
74     );
75
```

TestBench For Sekvensielle Kretser (5)

- Architecture | Clock Signal Generation

```
76      -- Clock process definitions
77      clk_process :process
78      begin
79          clk <= '0';
80          wait for clk_period/2;
81          clk <= '1';
82          wait for clk_period/2;
83      end process;
84
```

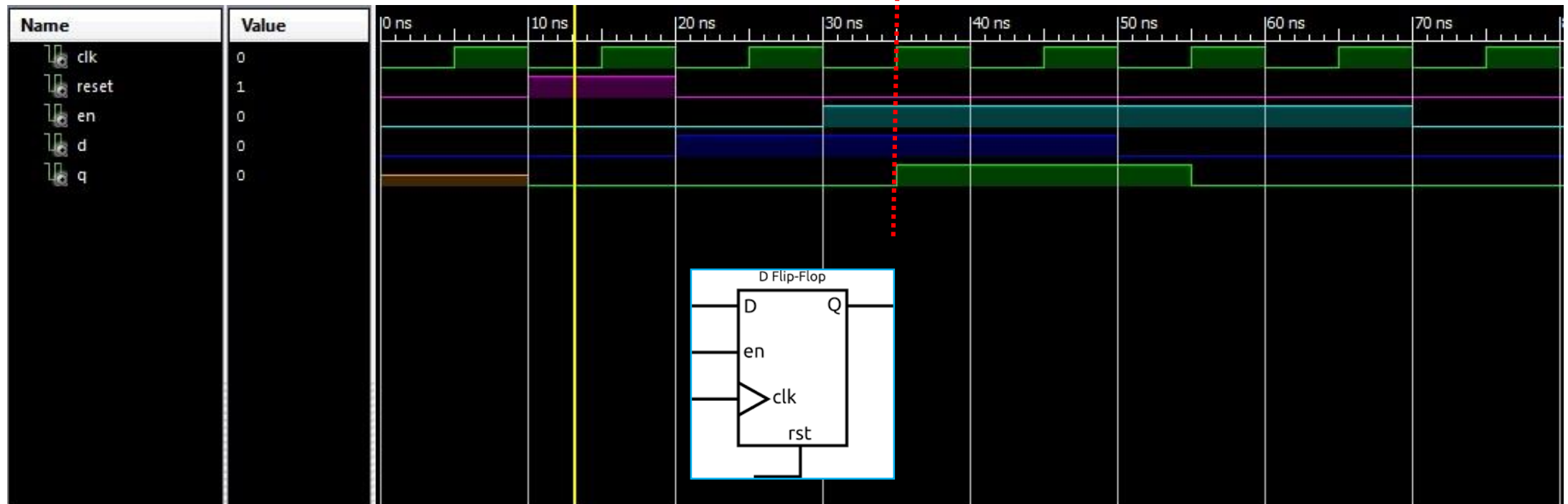
TestBench For Sekvensielle Kretser (6)

- Architecture | Generation of Other Signals – Reset, Enable, Input data

```
86      -- Stimulus process
87      stim_proc: process
88      begin
89          -- hold reset state for 100 ns.
90          wait for 10 ns;
91          reset    <= '1';
92          en       <= '0';
93          wait for clk_period*1;
94          d        <= '1';
95          reset    <= '0';
96          wait until falling_edge(clk);
97          wait until falling_edge(clk);
98          en <= '1';
99          wait until falling_edge(clk);
100         wait until falling_edge(clk);
101         d <= '0';
102         wait until falling_edge(clk);
103         wait until falling_edge(clk);
104         en <= '0';
105         wait;
106     end process;
```

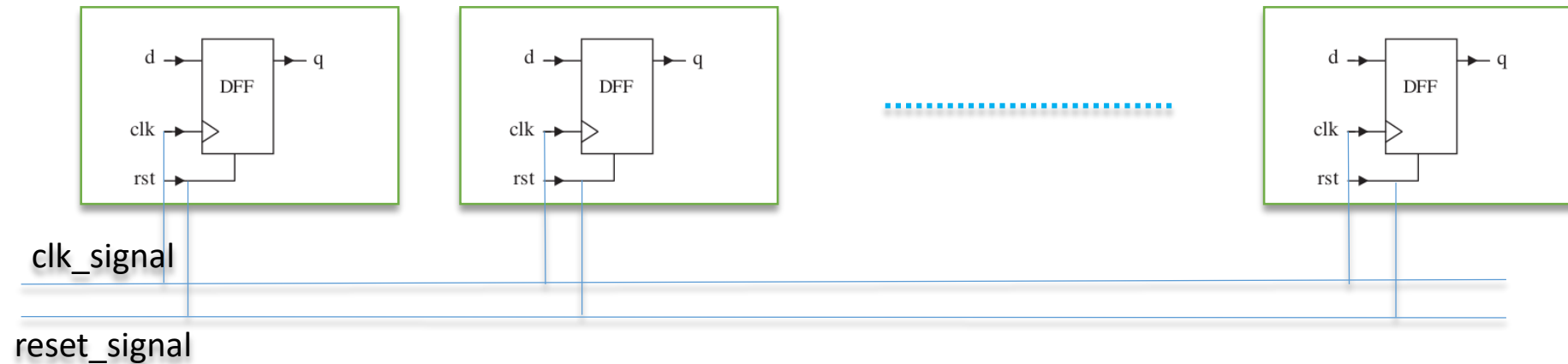
VHDL-koder for sekvensielle kretser: D-FF med enable-signal

- Simuleringskurveform



Register

- Register er en samling av D-FF-er som blir kontrollert av et felles klokkesignal og nullstillingssignal



Register

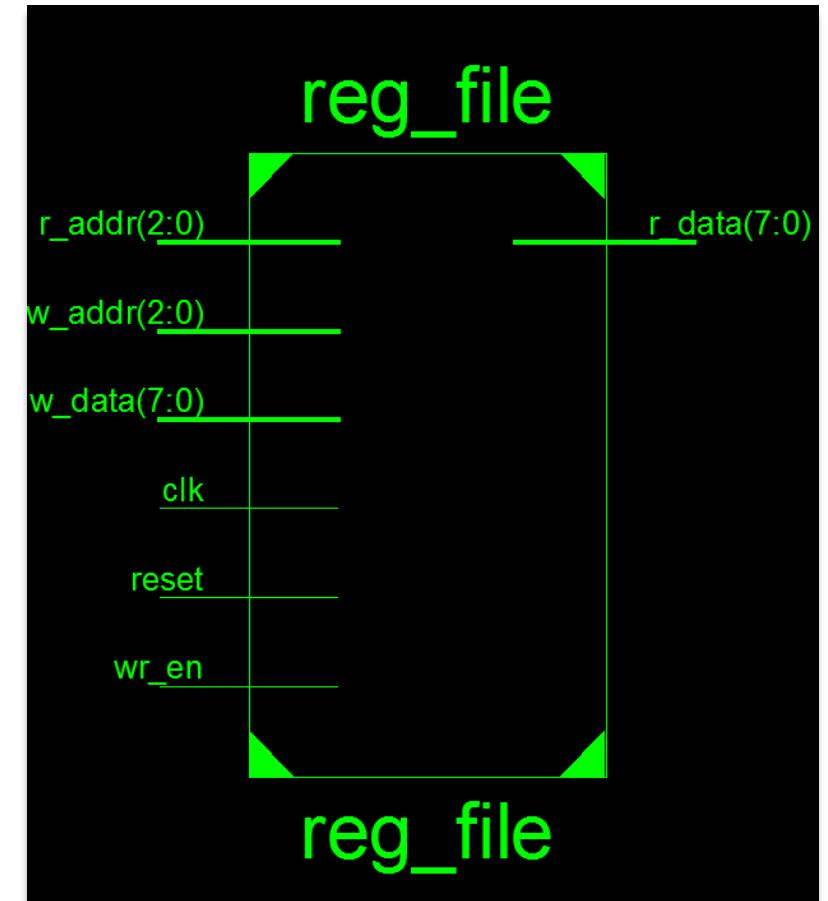
- I likhet med D-FF, kan register ha valgfri asynkront nullstillingssignal og synkront enable-signal
- VHDL-koden er identisk til koden til D-FF, unntatt at vi bruker datatabell av datatype (std_logic_vector)
- Eksempel

```
library ieee;
use ieee.std_logic_1164.all;
entity reg_reset is
    port (
        clk, reset: in std_logic;
        d: in std_logic_vector(7 downto 0);
        q: out std_logic_vector(7 downto 0)
    );
end reg_reset;

architecture arch of reg_reset is
begin
    process(clk, reset)
    begin
        if (reset='1') then
            q <= (others=>'0');
        elsif (clk'event and clk='1') then
            q <= d;
        end if;
    end process;
end arch;
```


Register-fil/ Register file

- En registerfil er en samling av registrer med en inngangssport og en eller flere utgangssporter
- write_address signal (w_addr): spesifiserer hvor data blir lagret
- Read_address signal (r_addr): spesifiserer hvor data blir avlest
- Register/Registerfil blir brukt som rask og midlertidig lagring



Register-fil/ Register file

- Registerfil som består 4 registrer
hver har 8 D-FF

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_file is
  generic(
    B: integer:=8; -- number of bits
    W: integer:=2  -- number of address bits
  );
  port(
    clk, reset: in std_logic;
    wr_en: in std_logic;
    w_addr, r_addr: in std_logic_vector (W-1 downto 0);
    w_data: in std_logic_vector (B-1 downto 0);
    r_data: out std_logic_vector (B-1 downto 0)
  );
end reg_file;

architecture arch of reg_file is
  -----define the register file-----
  type reg_file_type is array (2**W-1 downto 0) of
    std_logic_vector(B-1 downto 0);
  signal array_reg: reg_file_type;
  -----
begin
  process(clk,reset)
  begin
    if (reset='1') then
      array_reg <= (others=>(others=>'0'));
    elsif (clk'event and clk='1') then
      if wr_en='1' then
        array_reg(to_integer(unsigned(w_addr))) <= w_data;
      end if;
    end if;
  end process;
  -- read port
  r_data <= array_reg(to_integer(unsigned(r_addr)));
end arch;
```