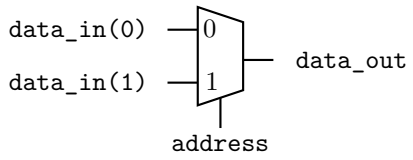
Digitaltekniska byggblock

Mattias Krysander

9 oktober 2018

Dokumentet listar användbara digitaltekniska typer av byggblock. För varje block visas symbol och vhdl-kod. Variabeldefintioner för `std_logic`-variabler har utelämnats. Blocken anpassas vid användning till uppgiftens specifika behov genom att t ex välja antalet bitar i en räknare, vilka funktioner räknaren ska ha etc.

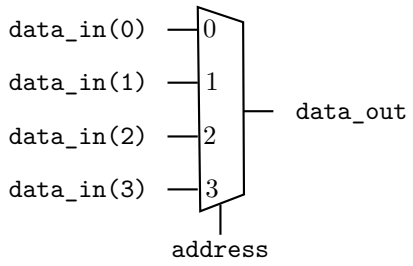
2-1 Multiplexer




```
signal data_in : std_logic_vector(1 downto 0);

data_out <= data_in(0) when address = '0'
         else data_in(1);
```

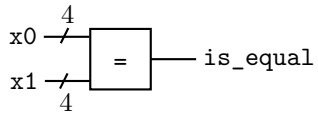
4-1 Multiplexer




```
signal data_in : std_logic_vector(3 downto 0);
signal address : unsigned(1 downto 0);

with address select
  data_out <= data_in(0) when "00",
              data_in(1) when "01",
              data_in(2) when "10",
              data_in(3) when "11",
              '-' when others;
```

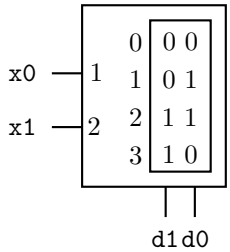
Komparator




```
signal x0 : unsigned(3 downto 0);
signal x1 : unsigned(3 downto 0);

is_equal <= '1' when x1 = x0
          else '0';
```

ROM



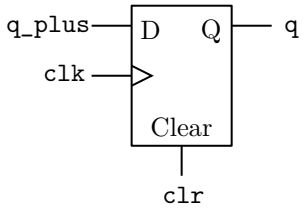

```
use ieee.numeric_std.all;

type ROM is array(0 to 3) of std_logic_vector(1 downto 0);
constant ROM_content : ROM := (0 => "00",
                               1 => "01",
                               2 => "11",
                               3 => "10");
signal data    : std_logic_vector(1 downto 0);
signal address : std_logic_vector(1 downto 0);

address <= x1 & x0;
data <= ROM_content(to_integer(unsigned(address)));
d0 <= data(0);
d1 <= data(1);
```

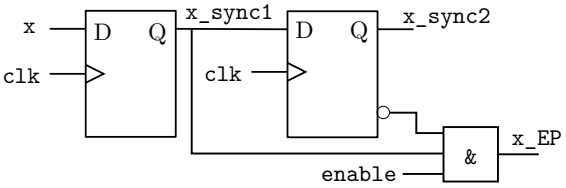
D-vippa

D-vippa med asynkron clear.



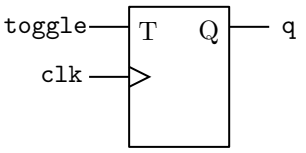

```
process(clk,clr)
begin
   if clr = '1' then
      q <= '0';
   elsif rising_edge (clk) then
      q <= q_plus;
   end if;
end process;
```

Synkronisering och enpulsning



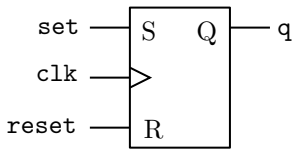

```
process(clk) begin
  if rising_edge(clk) then
    x_sync1 <= x;
    x_sync2 <= x_sync1;
  end if;
end process;
x_EP <= x_sync1 and (not x_sync2) and enable;
```

T-vippa



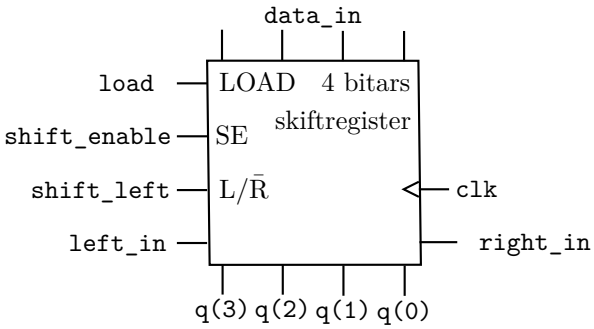

```
process(clk) begin
  if rising_edge(clk) then
    if (toggle = '1') then
      q <= not(q);
    end if;
  end if;
end process;
```

SR-vippa




```
process(clk) begin
  if rising_edge(clk) then
    if (set = '1') then
      q <= '1';
    elsif (reset = '1') then
      q <= '0';
    end if;
  end if;
end process;
```

Skiftregister

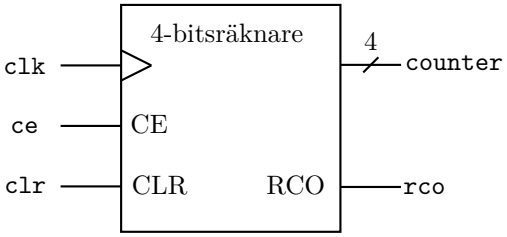



```
signal q : std_logic_vector(3 downto 0);
signal data_in : std_logic_vector(3 downto 0);

process(clk) begin
  if rising_edge(clk) then
    if (load = '1') then
      q <= data_in;
    elsif (shift_enable = '1') then
      if (shift_left = '1') then
        q <= q(2 downto 0) & right_in;
      else
        q <= left_in & q(3 downto 1);
      end if;
    end if;
  end if;
end process;
```

Räknare

Ett exempel på en 4-bitsräknare med asynkron clear, count enable och rippel carry out.




```
use ieee.numeric_std.all;

signal counter : unsigned(3 downto 0);

process(clk,clr) begin
   if clr = '1' then
      counter <= "0000";
   elsif rising_edge(clk) then
      if (ce = '1') then
         counter <= counter + 1;
      end if;
   end if;
end process;
rco <= '1' when ((ce = '1') and
                 (counter = 15))
       else '0';
```