Procesos para Syntehesis

A process represents a group of logic elements (gates and/or flip-flops) with signal inputs and signal outputs.

```
process(A, B, C)
  variable D: Std_Logic;
begin
  if A='1' then
     D := B;
  else
     D := B or C;
  end if;
  F <= D;
end process;
```

sensitivity list

A

B

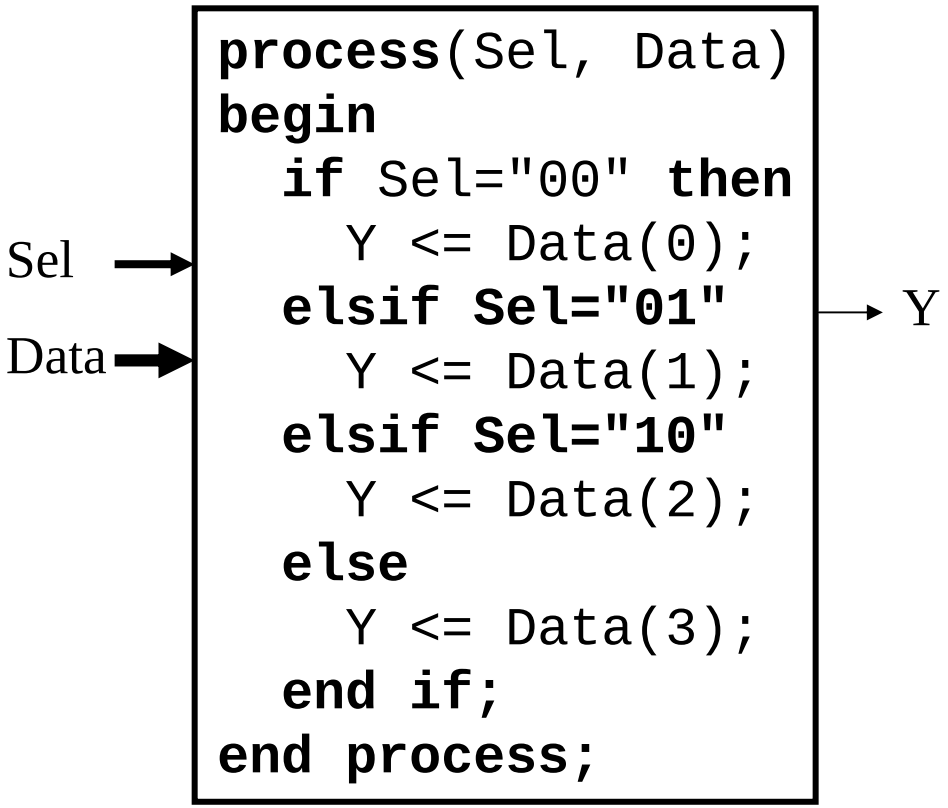
C

F

Inputs will appear in conditions or in right-hand side expressions and may be in the sensitivity list.

Outputs will be the target of signal assignments. (Internal declarations cannot be outputs).

Process Construct as Conditional CSA




```
process(Sel, Data)
begin
  if Sel="00" then
    Y <= Data(0);
  elsif Sel="01"
    Y <= Data(1);
  elsif Sel="10"
    Y <= Data(2);
  else
    Y <= Data(3);
  end if;
end process;
```

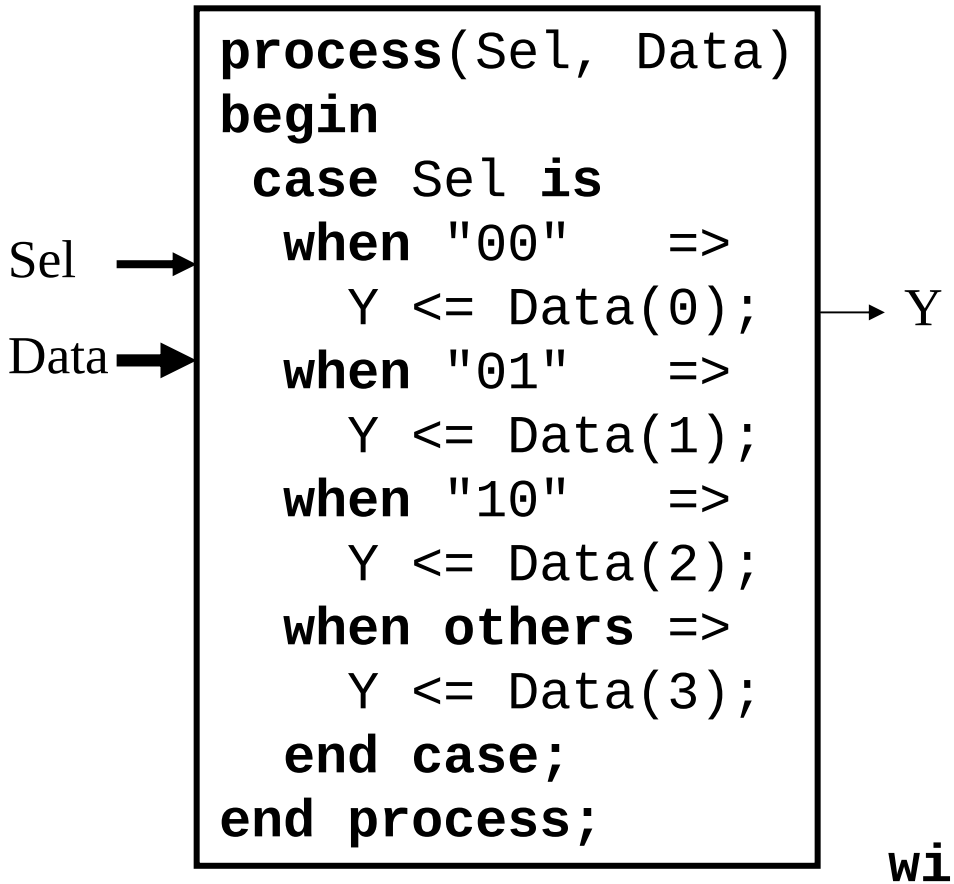
Asignación Concurrente de Señal (CSA)

Un proceso debe tener el mismo Que una CSA.

Preferido porque es mas corto y no hay que manejar Lista de sensibilidad.

```
Y <=Data(0) when Sel="00" else
    Data(1) when Sel="01" else
    Data(2) when Sel="10" else
    Data(3);
```

Process Construct as Selected CSA



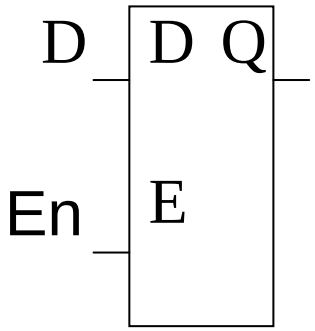

```
process(Sel, Data)
begin
 case Sel is
  when "00"   =>
    Y <= Data(0);
  when "01"   =>
    Y <= Data(1);
  when "10"   =>
    Y <= Data(2);
  when others =>
    Y <= Data(3);
  end case;
end process;
```

Un proceso debe tener el mismo Que una CSA.

Preferido porque es mas corto y no hay que manejar Lista de sensibilidad.

```
with Sel select
  Y <= Data(0) when "00",
       Data(1) when "01",
       Data(2) when "10",
       Data(3) when others;
```

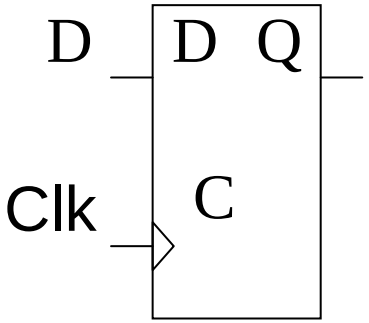
Elementos de almacenamiento



Latch

Nota: El "if" no puede ser puesto en la zona concurrente.

```
if En = '1' then
  Q <= D;
end if;
```

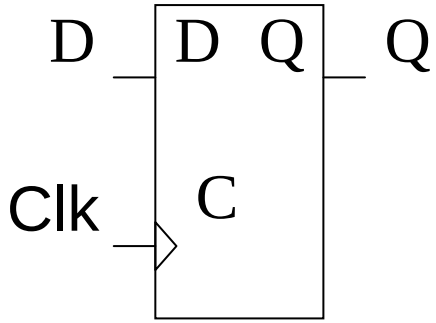


Flip-flop

```
if Clk'event and Clk = '1' then
  Q <= D;
end if;
```

```
rising_edge(Clk)
```

Flip-flops/Registros

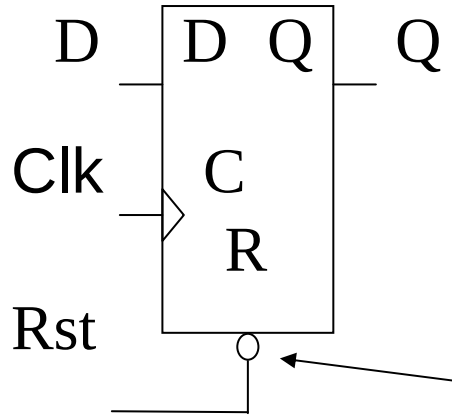


Flip-flops and registers must be represented as processes

```
process(Clk)
begin
  if rising_edge(Clk) then
    Q <= D;
  end if;
end process;
```

Este flip-flop tipo D no tiene inicialización.

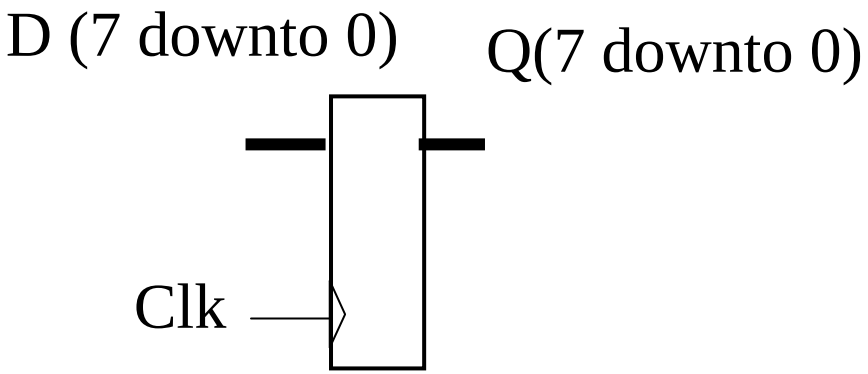
Schematic/VHDL Representations Flip-flops/Registros)

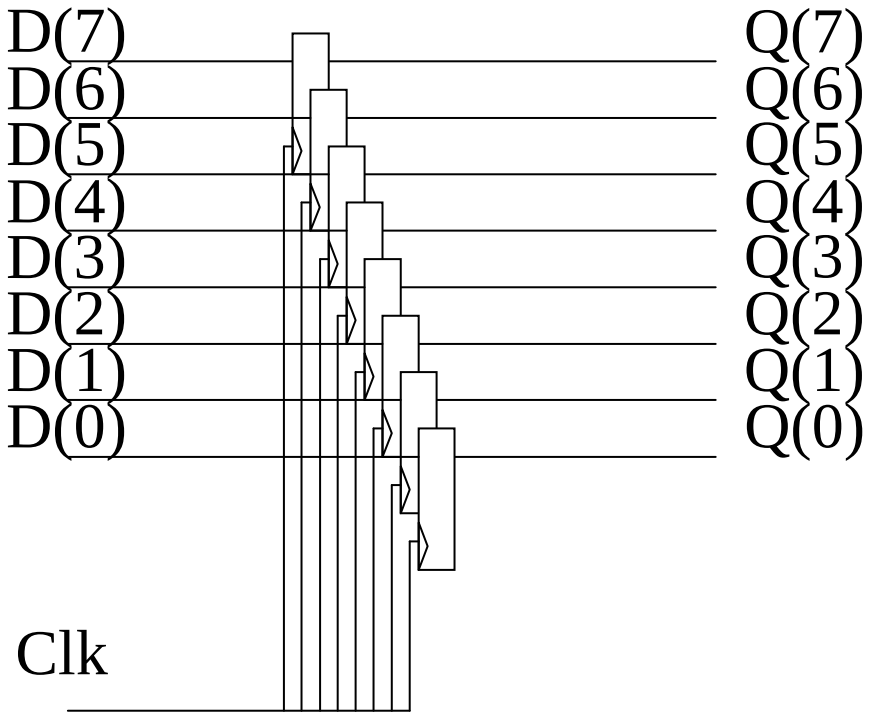


Active low

```
process(Rst, Clk)
begin
  if Rst = '0' then
    Q <= '0';
  elsif rising_edge(Clk) then
    Q <= D;
  end if;
end process;
```

Registro (8-bit)

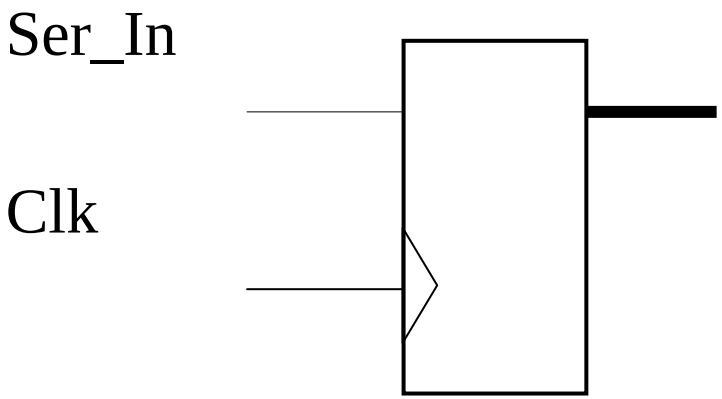





```
process(Clk)
begin
  if rising_edge(Clk) then
    Q <= D;
  end if;
end process;
```

Este registro no tiene inicialización.

Registro de desplazamiento (8-bit)



concatenacion

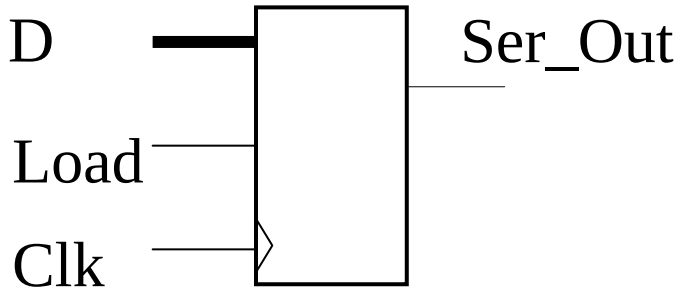
```
process(Clk)
begin
  if rising_edge(Clk) then
    Q <= Q(6 downto 0) & Ser_In;
  end if;
end process;
```

Este registro no tiene inicialización.

Entrada Paralela/salida Serie registro de desplazamiento (8-bit)

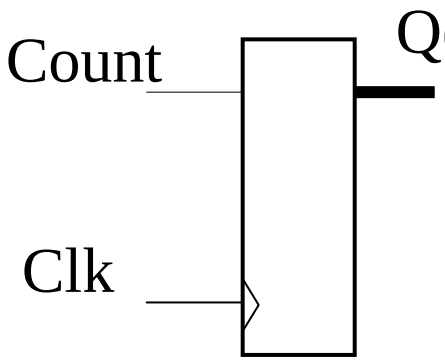
signal D:Std_Logic_Vector(7 downto 0);

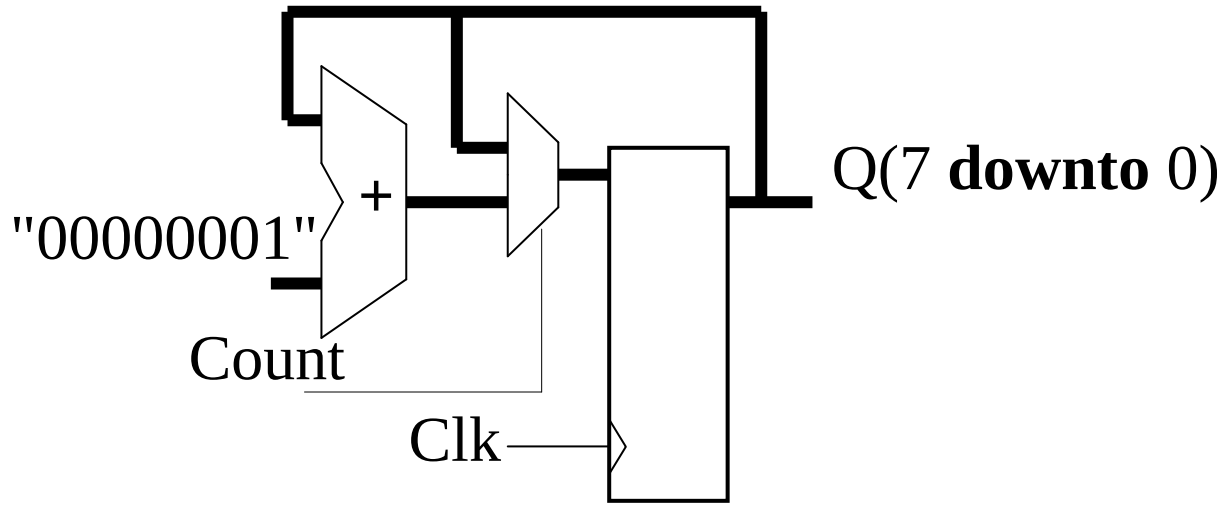
```
Ser_Out <= Q(7);

process(Clk)
begin
  if rising_edge(Clk) then
    if Load = '0' then
      Q <= D;
    else
      Q <= Q(6 downto 0) & '0';
    end if;
  end if;
end process;
```



concatenación

Contador Binario (8-bit)



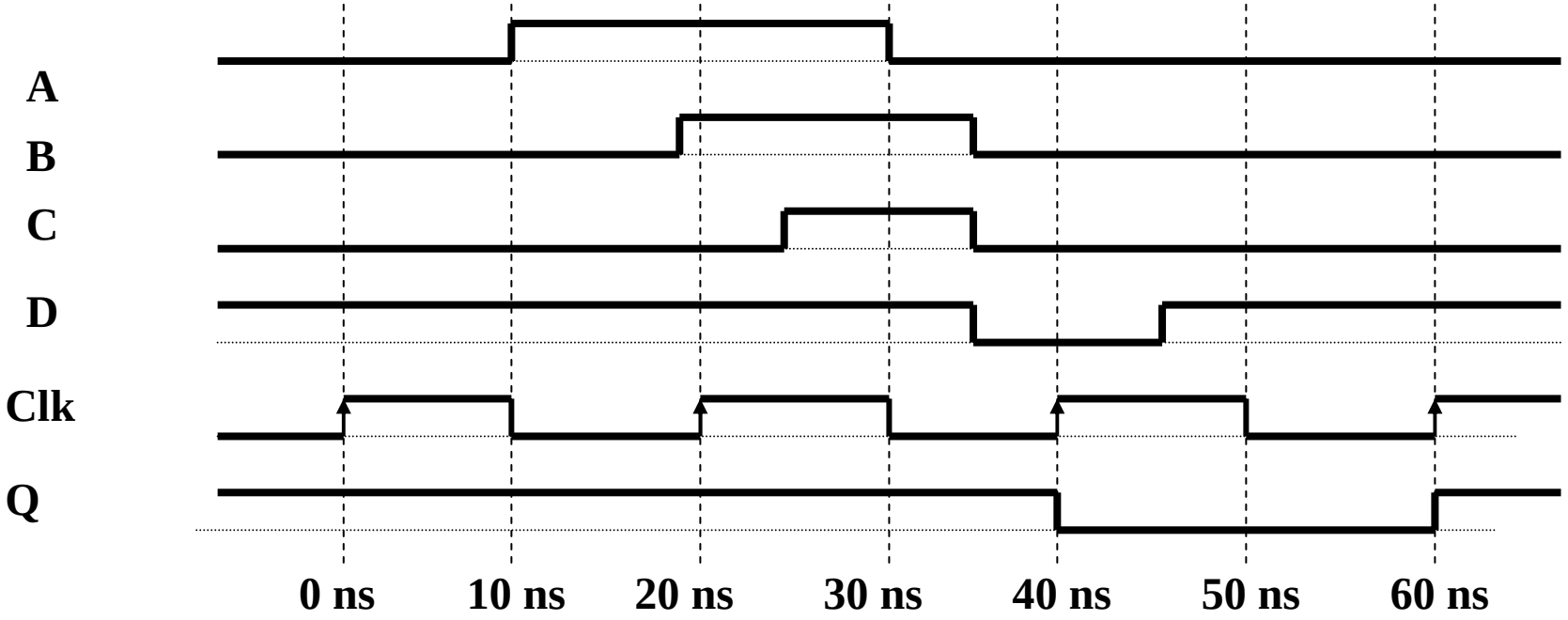


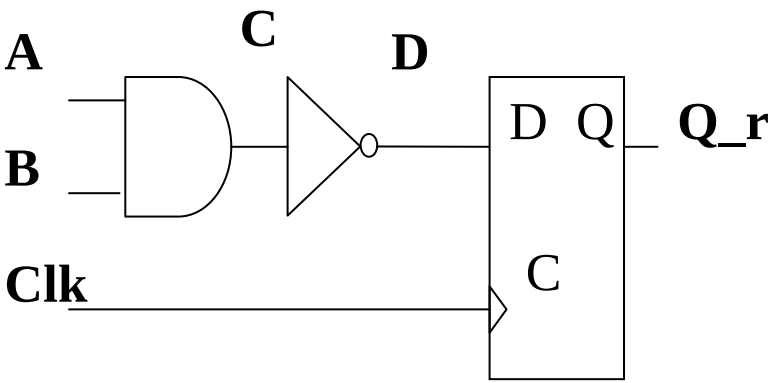

```
process(Clk)
begin
  if rising_edge(Clk) then
    if Count = '1' then
       Q <= Q + 1;
    end if;
  end if;
end process;
```

use IEEE.Std_Logic_unsigned.all;

Este registro no tiene inicialización.

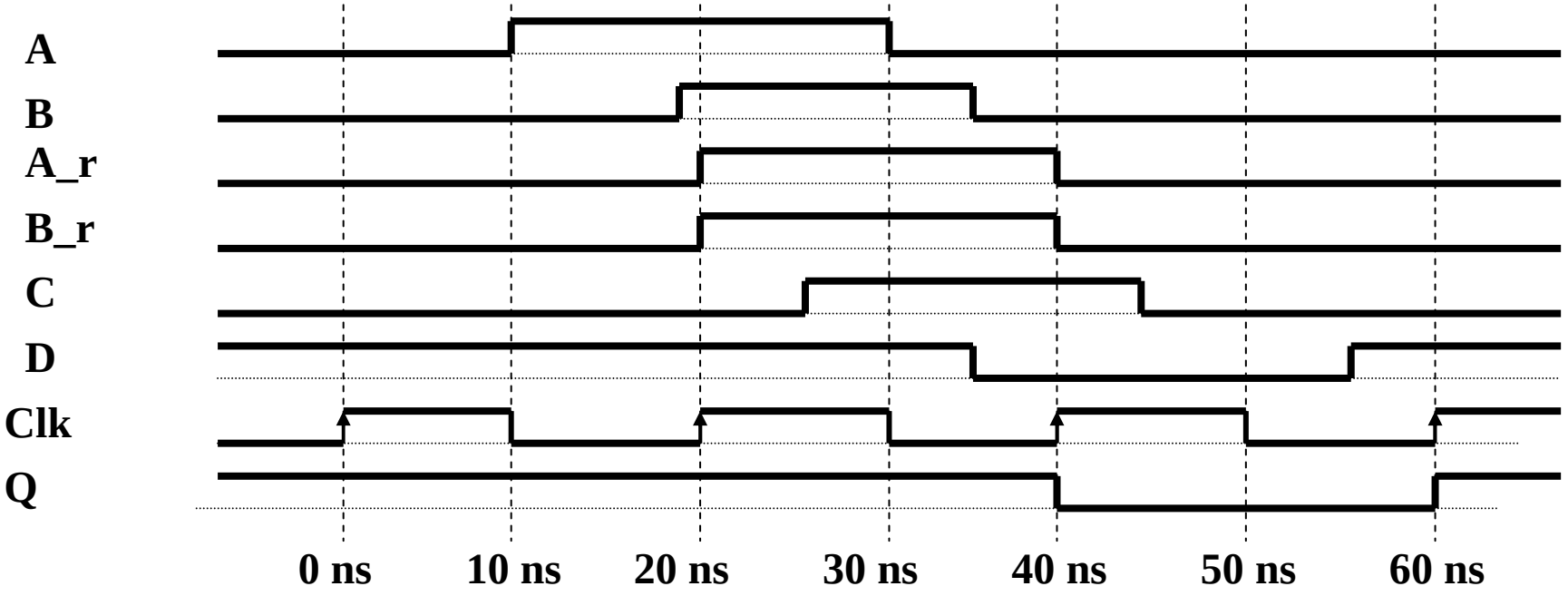
CSAs with Processes (one flip-flop)

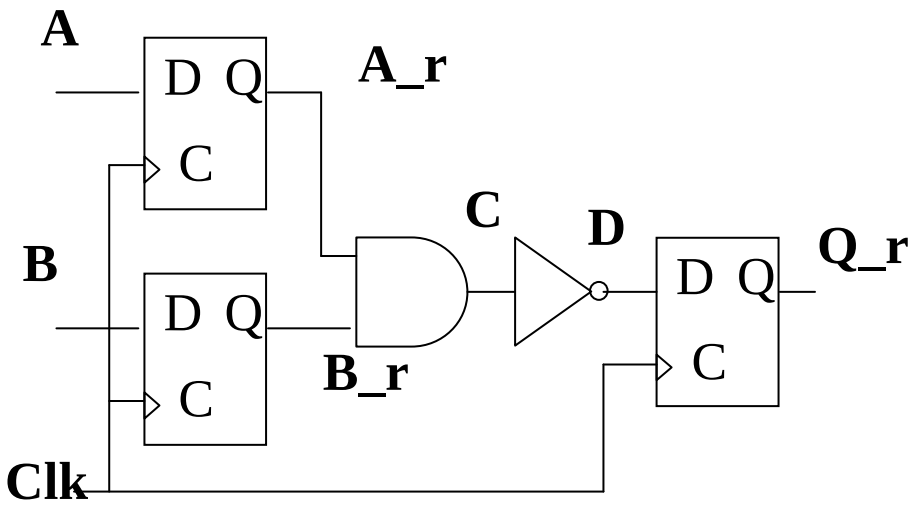





```
C <= A and B after 5 ns;

D <= not C after 10 ns;

process(Clk)
begin
  if rising_edge(Clk) then
    Q_r <= D;
  end if;
end process;
```

CSAs with Processes (multiple flip-flops)






```
C <= A_r and B_r after 5 ns;
D <= not C after 10 ns;
process(Clk)
begin
 if rising_edge(Clk) then
    Q_r <= D;
    A_r <= A;
    B_r <= B;
  end if;
end process;
```