**The D-Flip Flop:**
Inside a computer we want the output of gates to change only at specific times. We can add some circuitry to make sure that changes occur only when a *clock* changes (when the clock changes from 0 to 1).

**Example-1:** The VHDL Code for a D-Flip Flop. *(Behavioral Implementation)*

```vhdl
Entity D_FF is
Port(
    D : in bit;
    CLK : in bit;
    Q : out bit;
    Qbar: out bit);
End D_FF;

Architecture D_beh of D_FF is
Constant gate_delay: Time:=5 ns;
Begin
Process(D,CLK)
Begin
    If(CLK'EVENT and CLK='1')
    then
        Q<= D after gate_delay ;
        Qbar<= not D after gate_delay;
    End if;
End process;
End D_beh;
```

**Inputs** | **Outputs** | **Comments**
---|---|---
0 | 0 | 1 | RESET
1 | 1 | 0 | SET

---

<table>
<thead>
<tr>
<th>Signal</th>
<th></th>
<th>Ons</th>
</tr>
</thead>
<tbody>
<tr>
<td>% force CLK 1</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>% force D 1</td>
<td>CLK</td>
<td></td>
</tr>
<tr>
<td>% run 100 ns</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>100 ns</td>
<td>Qbar</td>
<td></td>
</tr>
</tbody>
</table>

Prepared By: Aws Yousif Al-Taie

Computer Science & Engineering Department
Registers:
The registers are referred to as one unit instead of addressing each storage element individually.

```plaintext
if(CLK'event and CLK='1')
This condition is only true when CLK = '1'
or when the control input is enabled
```
Example-2: The VHDL Code for an 8-bit Register.

```vhdl
ENTITY ENTITY_REG IS
PORT
  (D: in bit_vector(7 downto 0);
Q: out bit_vector(7 downto 0);
Clk: in bit;
Clear: in bit);
END ENTITY;

ARCHITECTURE BEHAVIOR_REG OF ENTITY_REG IS
BEGIN
PROCESS (Clk,D,Clear)
BEGIN
IF (Clear='1') THEN
Q<="00000000";
ELSIF (Clk'EVENT AND Clk='1') THEN
Q<=D;
END IF;
END PROCESS;
END ARCHITECTURE;
```

Note: n bit register is constructed with n flip-flops, is capable of storing n bits

<table>
<thead>
<tr>
<th>Signal</th>
<th>10ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>9</td>
</tr>
<tr>
<td>Q</td>
<td>9</td>
</tr>
<tr>
<td>Clk</td>
<td></td>
</tr>
<tr>
<td>Clear</td>
<td></td>
</tr>
</tbody>
</table>

Shift Registers (Shift Right):
A basic four-bit shift register can be constructed using four D flip-flops, as shown below. The operation of the circuit is as follows. The register is first cleared, forcing all four outputs to zero. The input data is then applied sequentially to the D input of the first flip-flop on the left (FF0). During each clock pulse, one bit is transmitted from left to right. Assume a data word to be 1001. The least significant bit of the data has to be shifted through the register from FF0 to FF3.
**Example-3:** VHDL Code for a 4-bit shift right register

```vhdl
entity shift_reg is
port (
  CLK, CLEAR, SI : in bit;
  Q : out bit_vector(0 to 3);
  SO : out bit
);
end shift_reg;

architecture behavioral of shift_reg is
signal shift : bit_vector(0 to 3)="0000";
begin
process(CLK,CLEAR,SI)
begin
  if (CLEAR = '1') then
    shift <= "0000";
  elsif(CLK'event and (CLK = '1')) then
    shift <= SI & shift(0 to 2);
  end if;
  SO <= shift(3);
end process;
Q <= shift;
end behavioral;
```

**Sample Input & Output:**

| force CLK 1 | CLK  | '1' |
| force CLEAR 0| CLEAR | '0' |
| force SI 1 | SI | '1' |
| run 300 | Q | 1000 |
| 300 ns | SO | '0' |
|         | shift | 1000 |

**Example-4:** The VHDL code for an 8-bit register, which is constructed with 8 D Flip Flops, using the component method.

```vhdl
entity DFF is
port (
  CLK, D_bit: in bit;
  Q_bit: out bit
);
end entity;

architecture Function_DFF of DFF is
begin
process(CLK, D_bit)
begin
  if (CLK'event and CLK='1') then
    Q_bit<=D_bit;
  end if;
end process;
end architecture;
```
entity Register_DFF is
port
(CLK: in bit;
D: in bit_vector(7 downto 0);
Q: out bit_vector(7 downto 0));
end entity;
architecture Function_Register of Register_DFF is
component DFF is
port
(CLK, D_bit: in bit;
Q_bit: out bit);
end component;
begin
DFF7: DFF port map (CLK,D(7), Q(7));
DFF6: DFF port map (CLK,D(6), Q(6));
DFF5: DFF port map (CLK,D(5), Q(5));
DFF4: DFF port map (CLK,D(4), Q(4));
DFF3: DFF port map (CLK,D(3), Q(3));
DFF2: DFF port map (CLK,D(2), Q(2));
DFF1: DFF port map (CLK,D(1), Q(1));
DFF0: DFF port map (CLK,D(0), Q(0));
end architecture;

Important Remark:-
The order of the signals in the port map must be the same as the order of the signals in the port of the component declaration.

Lab Exercise-1: Modify the above VHDL code, to shift bits the opposite side (Shift left).

Lab Exercise-2: Write the VHDL code for the following circuit and then simulate it. Your VHDL code should perform the following operation: - add ax,bx . Where ax is 16-bit and bx is 16-bit