# Códigos em VHDL -  FPGA

# COMANDO IF

**PORTA OR**

```
architecture orif of estrutura is

signal x: std_logic_vector(1 downto 0);
begin
x <= a & b;
process (a, b)
begin

if (x<="00") then
c<='0';
else
c<='1';
end if;
end process;
end orif;
```

# COMANDO CASE

**PORTA OR**

```
architecture caseor of estrutura is
signal x: std_logic_vector (1 downto 0);
begin
x<= a & b;

process (a, b)
begin
 case x is
          when "00" => c <='0';
          when "01" => c <='1';
          when "10" => c <='1';
          when "11" => c <='1';
end case;
end process;
end caseor;
```

# SOMADOR COMPLETO

**PROGRAMA SECUNDÁRIO**

```
entity programa_secundario is
port (
 a, b, cin: in std_logic;
 s, cout: out std_logic
);
end programa_secundario;

architecture soma of programa_secundario is
begin
s<= a xor b xor cin;
cout<= (a and cin) or (b and cin) or ( a and b);
end soma;
```

**PROGRAMA PRINCIPAL**

```
architecture hardware of teste0 is
signal c_aux:std_logic_vector(3 downto 0);
begin

sm0: entity work.programa_secundario
port map (a => ehkey(0), b => ehkey(1), cin=> ehkey(2),  s=> led(0), cout =>c_aux(0));
sm1: entity work.programa_secundario
port map (a => ehkey(3), b => ehkey(4), cin => c_aux(0), s=> led(1), cout =>c_aux(1));
sm2: entity work.programa_secundario
port map (a => ehkey(5), b => ehkey(6), cin => c_aux(1), s=> led(2), cout =>c_aux(2));
sm3: entity work.programa_secundario
port map (a => ehkey(7), b => ehkey(8), cin => c_aux(2), s=> led(3), cout =>c_aux(3));

end hardware;
```

# FIIP FLOP D

**PROGRAMA SECUNDÁRIO**

```
entity Flip_Flop_D is
	port
	(
	D  : in std_logic;
	En: in std_logic;
	Q  : out std_logic
	);
end Flip_Flop_D;
architecture behaviorOfTest of Flip_Flop_D is
--signal vtest: std_logic_vector(4 downto 0);
begin

process(En)
begin
if En='1' then
	if D='1' then
	Q<='1';
	else
	Q<='0';
	end if;
end if;
end process;

end behaviorOfTest;
```

**PROGRAMA PRINCIPAL**

```
entity teste0 is
	port
	(
	  --output ports
		Y : out std_logic;

		-- input ports
		D1,En1,D2,En2 : in std_logic
	);
end teste0;


architecture behaviorOfTest of teste0 is
signal A,B,C,D: std_logic;
begin

FLIP_FLOP1: entity work.Flip_Flop_D
port map(D => D1, En => En1, Q => A);

FLIP_FLOP2: entity work.Flip_Flop_D
```

```
port map(D => D2, En => En2, Q => B);

C <= A and B;
D <= (not A) and (not B);

Y <= not (C or D);

end behaviorOfTest;
```