Uso de COMPONENTES

O componente é uma estrutura que referencia diretamente uma entidade e possibilita a instanciação e replicação da mesma sem a necessidade de descrevê-la novamente. A forma geral para **declarar** um componente é:

COMPONENT nome [**IS**]
[**GENERIC** (lista de parâmetros);]
[**PORT** (lista de portas);]
END COMPONENT [nome];

Ex.: **COMPONENT** inv
PORT (e: **IN** bit; s: **OUT** bit);
END COMPONENT;

A forma geral para **instanciar** um componente é:
ref_id: [**COMPONENT**] nome_do_componente | **ENTITY** nome_da_entidade
[(nome_da_arquitetura)] | **CONFIGURATION** nome_da_configuração
[**GENERIC MAP** (parametros)] [**PORT MAP** (lista de portas)];

Onde, ref_id é uma referência ao componente que está sendo instanciado, possibilitando a réplica do mesmo componente, bastando mudar o identificador de referência.

Ex.: u1:inv **PORT MAP** (e=>, s => s1);
u2:inv **PORT MAP** (e=>, s => s1);

Exemplo de aplicação:

Comparador de 1 bit. Vamos

```
1.  ENTITY comp IS
2.  PORT(           e1,e2:IN BIT;
3.                  sh,sl,si:OUT BIT);
4.  END comp;
5.
6.  ARCHITECTURE comp_fluxo_dados OF comp IS
7.  BEGIN
8.      sh <= '1' when e1>e2 else '0';
9.      sl <= '1' when e1<e2 else '0';
10.     si <= '1' when e1=e2 else '0';
11. END comp_fluxo_dados;
```

Usando o componente, comparando 5 bits:

```
1.  ENTITY compComComponente IS
2.  PORT (          ent1,ent2: IN BIT_VECTOR (5 DOWNTO 0); -- entradas do somador
3.                  saiH,saiL,saiI: OUT BIT_VECTOR (5 DOWNTO 0)); -- vai um
4.  END compComComponente;
5.
6.  ARCHITECTURE estrutural OF compComComponente IS
7.  COMPONENT comp
8.  PORT(   e1,e2:IN BIT;
9.          sh,sl,si:OUT BIT);
10. END COMPONENT;
11.
12. BEGIN
13.     x0: comp PORT MAP (ent1(0), ent2(0), saiH(0), saiL(0), saiI(0));
14.     x1: comp PORT MAP (ent1(1), ent2(1), saiH(1), saiL(1), saiI(1));
15.     x2: comp PORT MAP (ent1(2), ent2(2), saiH(2), saiL(2), saiI(2));
16.     x3: comp PORT MAP (ent1(3), ent2(3), saiH(3), saiL(3), saiI(3));
17.     x5: comp PORT MAP (ent1(4), ent2(4), saiH(4), saiL(4), saiI(4));
18. END estrutural;
```

Obs. para usar o componente, é necessário salvar o código do componente, no mesmo projeto.

ELEMENTOS GENÉRICOS.
A instrução **GENERIC** fornecem um meio de levar informações com valores constantes para entidades de projeto e blocos, podendo perpassar por todos os componentes do código.

Ex.:

```
ENTITY registrador IS
  GENERIC (tamanho: integer);
  PORT (          Clk: IN bit;
                Enable: IN bit;
                   Din: IN bit_vector (tamanho-1 DOWNTO 0);
                   Dout: OUT bit_vector (tamanho-1 DOWNTO 0));
END registrador;
```

FOR GENERATE
Geração automática de circuitos regulares, circuitos que seguem uma lei de formação.
- exemplo:
    - unidades lógicas aritméticas
    - somadores
    - multiplicadores

Montagem:

```
nome: FOR indentificador IN valor_inicial TO valor_final GENERATE
        -- comando concorrente
END GENERATE nome;
```

Exemplo de aplicação:
Comparador de 1 bit. Vamos

```
1.  ENTITY comp IS
2.  PORT(             e1,e2:IN BIT;
3.                    sh,sl,si:OUT BIT);
4.  END comp;
5.
6.  ARCHITECTURE comp_fluxo_dados OF comp IS
7.  BEGIN
8.      sh <= '1' when e1>e2 else '0';
9.      sl <= '1' when e1<e2 else '0';
10.     si <= '1' when e1=e2 else '0';
11. END comp_fluxo_dados;
```

Usando o componente, comparando 5 bits:

```
1.  ENTITY comparadorComComponenteFor IS
2.      GENERIC(    nComp:INTEGER :=6); --declarando um item com definição de valor generico.
3.      PORT (          ent1,ent2: IN BIT_VECTOR (nComp-1 DOWNTO 0); -- entradas do somador
4.                    saiH,saiL,saiI: OUT BIT_VECTOR (nComp-1 DOWNTO 0)); -- vai um
5.  END comparadorComComponenteFor;
6.
7.  ARCHITECTURE estrutural OF comparadorComComponenteFor IS
8.
9.  COMPONENT comp
10. port(   e1,e2:in bit;
11.         sh,sl,si:out bit);
12. END COMPONENT;
13.
14. BEGIN
15.   aut: FOR i IN 0 TO nComp-1 GENERATE
16.             x: comp PORT MAP (ent1(i), ent2(i), saiH(i), saiL(i), saiI(i));
17. END GENERATE aut;
18.
19. END estrutural;
```