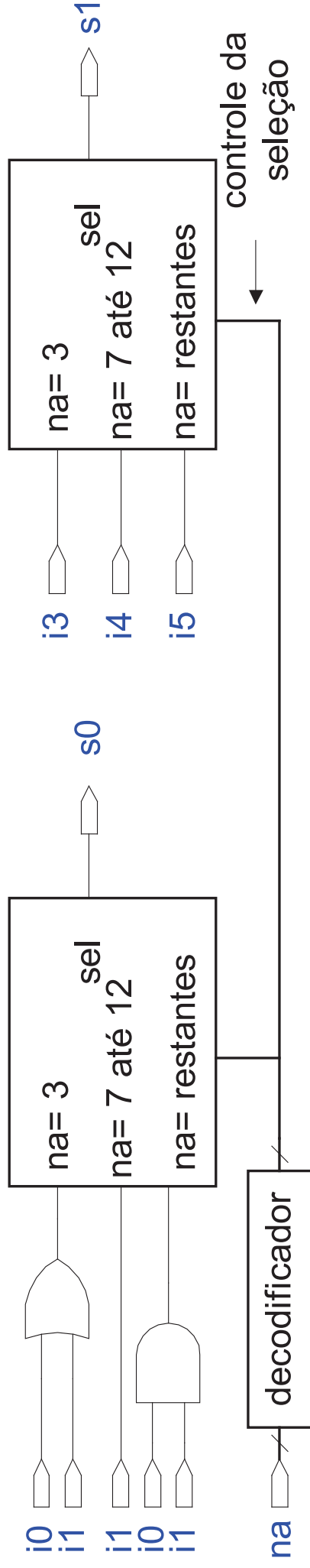


Construção CASE WHEN

- Exemplo:

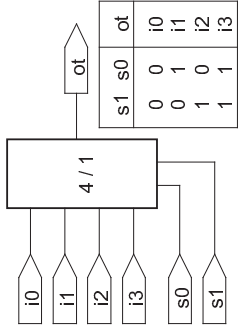
```
CASE na IS
  WHEN 3      => s0 <= i0 OR i1;   s1 <= i3;
  WHEN 7 TO 12 => s0 <= i1;       s1 <= i4;
  WHEN OTHERS => s0 <= i0 AND i1; s1 <= i5;
END CASE;
```

- Circuito equivalente:



Exemplo: circuito de seleção – CASE WHEN

- Comando sequencial
- Nota: s1 s2 são agrupados no sinal sel



```
1 ENTITY mux_5a IS
2   PORT (i0, i1, i2, i3      : IN  BIT;
3         s1, s0              : IN  BIT;
4         ot                  : OUT BIT);
5 END mux_5a;
6
7 ARCHITECTURE teste OF mux_5a IS
8   SIGNAL sel: BIT_VECTOR(1 DOWNTO 0);
9 BEGIN
10  sel <= s1 & s0;
11  abc: PROCESS (i0, i1, i2, i3, sel) --sinal "sel" inserido na lista
12  BEGIN
13    CASE sel IS
14      WHEN "00" => ot <= i0;
15      WHEN "01" => ot <= i1;
16      WHEN "10" => ot <= i2;
17      WHEN OTHERS => ot <= i3;
18    END CASE;
19    END PROCESS abc;
20 END teste;
```



Comando **WAIT**

- **WAIT**: suspende a execução de um processo
- Opções: 3 formas ou uma combinação delas
- **WAIT ON**: equivalente a uma lista de sensibilidade
processo suspenso até a mudança de valor de sinais relacionados
- **WAIT UNTIL**: processo suspenso; aguarda condição booleana
- **WAIT FOR**: processo suspenso por período de tempo

```
WAIT ON lista_de_sensibilidade;
```

```
WAIT UNTIL condicao_booleana;
```

```
WAIT FOR expressao_de_tempo;
```

```
WAIT ON lista_sensibilidade UNTIL condicao_booleana FOR expressao_de_tempo;
```

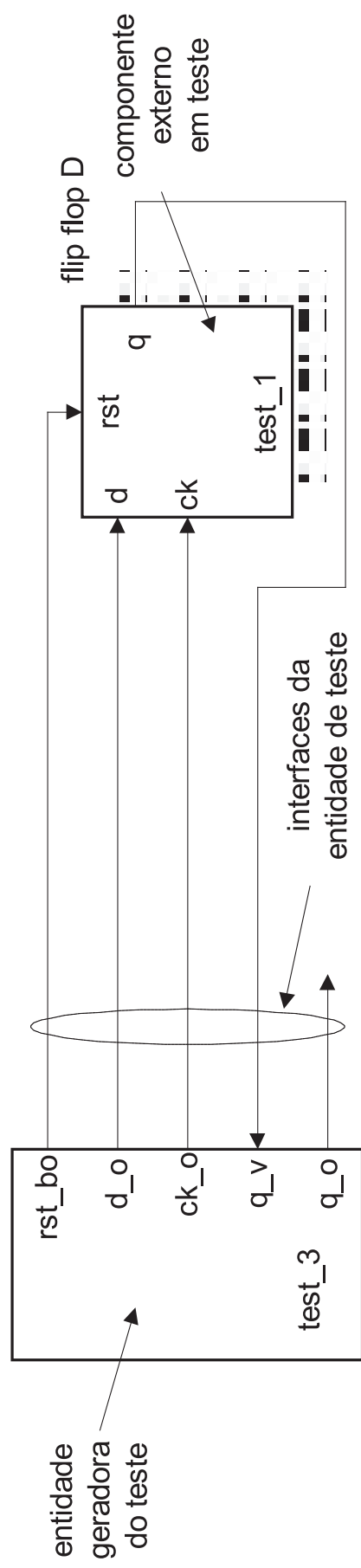
Comando **WAIT** – exemplos

- **1º caso:** processo suspenso
 - aguarda mudança de valor em **senal_a** ou **senal_b**
- **2º caso:** processo suspenso
 - aguarda a condição **senal_c =1** verdadeira
- **3º caso:** processo suspenso
 - aguarda um período de tempo de **100 ns**
- **4º caso:** processo suspenso
 - aguarda a condição **senal_d =0** por um período de, no máximo, **50 ns**

```
WAIT ON senal_a, senal_b;  
WAIT UNTIL senal_c = '1';  
WAIT FOR 100 ns;  
WAIT UNTIL senal_d = '0' FOR 50 ns;  
-- 1o caso  
-- 2o caso  
-- 3o caso  
-- 4o caso
```

Comando **WAIT** – observações

- **Não é permitido:** comando **WAIT** e lista de sensibilidade
- **Possível:** mais de um comando **WAIT** em um processo
- **Síntese:**
 - **WAIT ON** (equivalente a uma lista de sensibilidade)
 - **WAIT UNTIL** (inferência de registradores – capítulo 6)
- **Aplicações das outras formas:**
 - geração de estímulos de teste

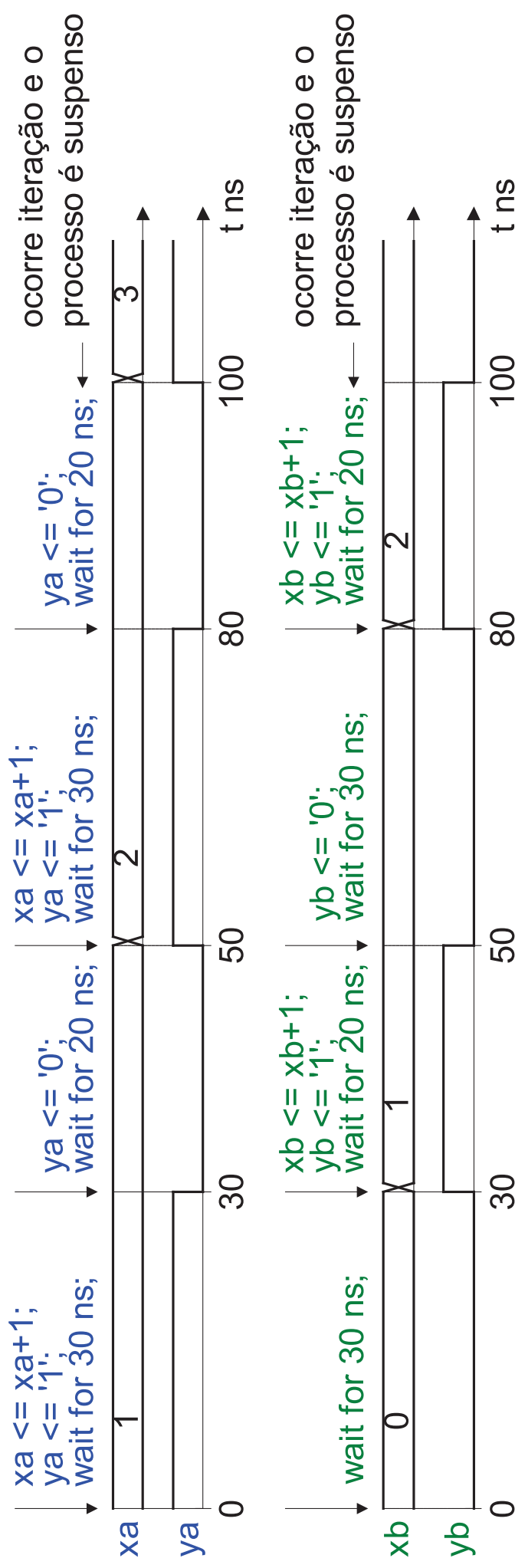


Comando **WAIT** – exemplo de operação

Note: não existe lista de sensibilidade; processo executado continuamente

Lembre: a iteração dos comandos concorrentes ocorre na suspensão do processo

8	abc: PROCESS	16	cde: PROCESS
9	BEGIN	17	BEGIN
10	xa <= xa +1;	18	WAIT FOR 30 ns;
11	ya <= '1';	19	xb <= xb +1;
12	WAIT FOR 30 ns;	20	yb <= '1';
13	ya <= '0';	21	WAIT FOR 20 ns;
14	WAIT FOR 20 ns;	22	yb <= '0';
15	END PROCESS abc;	23	END PROCESS cde;



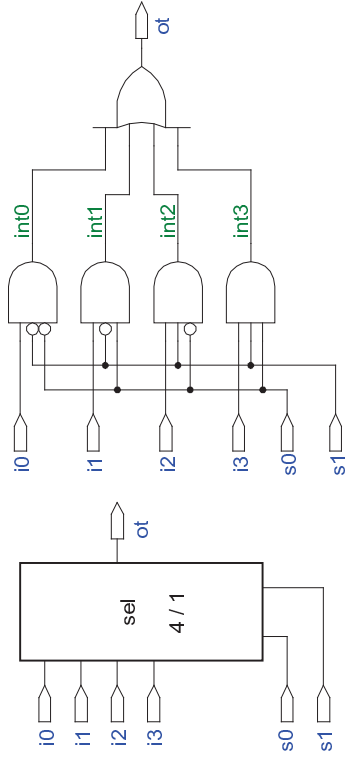
Exemplo: circuito de seleção – WAIT

- Processo sem lista de sensibilidade
- Execução do processo suspenso na linha 12

- sinais: int0 int1 int2 int3

incluídos na lista de sensibilidade

1	ENTITY mux_p1 IS
2	PORT (i0, i1, i2, i3 : IN BIT; -- entradas
3	s0, s1 : IN BIT; -- selecao
4	ot : OUT BIT); -- saida
5	END mux_p1;
6	
7	ARCHITECTURE teste OF mux_p1 IS
8	SIGNAL int0, int1, int2, int3 : BIT;
9	BEGIN
10	PROCESS
11	BEGIN
12	ot <= int0 OR int1 OR int2 OR int3;
13	int0 <= i0 AND NOT s1 AND NOT s0;
14	int1 <= i1 AND NOT s1 AND s0;
15	int2 <= i2 AND s1 AND NOT s0;
16	int3 <= i3 AND s1 AND s0;
17	WAIT ON i0, i1, i2, i3, s0, s1, int0, int1, int2, int3;
18	END PROCESS;
19	END teste;



Cuidados na descrição: comparações **WHEN ELSE** e **IF ELSE**

- Construção **WHEN ELSE**:

- concorrente
- 1ª condição válida: determina a expressão transferida

- Construção **IF ELSE**:

- sequencial
- 1ª condição válida: determina um conjunto de comandos a ser executado

```
signal_destino <= expressao_a WHEN opcao_1 ELSE  
                expressao_b WHEN opcao_2 ELSE  
                expressao_c;
```

```
IF condicao_1 THEN  
    comando_a;  
    comando_b;  
ELSIF condicao_2 THEN  
    comando_c;  
ELSE  
    comando_d;  
    comando_e;  
END IF;
```



Cuidados na descrição: comparações **WITH SELECT** e **CASE WHEN**

- Construção **WITH SELECT**:

- concorrente
- todas as condições devem ser relacionadas
 - cada condição determina uma expressão transferida

- Construção **CASE WHEN**:

- sequencial
- todas as condições devem ser relacionadas
 - cada condição determina um conjunto de comandos a ser executado

```
WITH expressao_escolha SELECT  
sinal_destino <= expressao_a WHEN opcao_1,  
expressao_b WHEN opcao_2,  
expressao_c WHEN OTHERS;
```

```
CASE expressao IS  
WHEN condicao_1 => comando_a;  
WHEN OTHERS    => comando_e; comando_f;  
END CASE;
```



Cuidados na descrição: **IF ELSE** e **CASE WHEN** aninhadas

- **IF ELSE** e **CASE WHEN** são comandos sequenciais:

- é possível aninhar vários comandos

- **Este estilo pode ser evitado**

Objetivo: melhorar a clareza do código

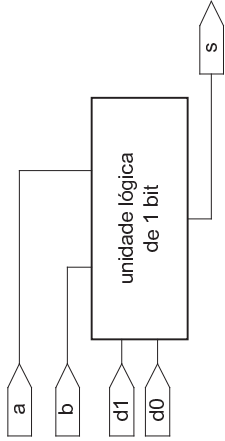
- **Exemplo:**

- descrição para uma Unidade Lógica Aritmética de 1 bit

- operações: complemento de **a**
 - “ou exclusivo” entre **a** e **b**
 - “e” entre **a** e **b**
 - “ou” entre **a** e **b**

• Descrições – IF ELSE com e sem aninhamento

- sinais que definem a decisão d0 e d1
- a concatenação dos sinais d0 e d1 evita o aninhamento
- código mais claro e conciso

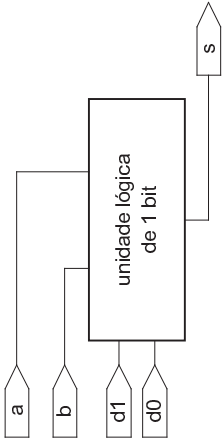


<pre> 1 ENTITY t1_if1 IS 2 PORT (d1,d0 : IN BIT; 3 a, b : IN BIT; 4 s : OUT BIT); 5 END t1_if1; 6 7 ARCHITECTURE teste OF t1_if1 IS 8 BEGIN 9 PROCESS (d1,d0,a,b) 10 BEGIN 11 IF d1 ='0' THEN 12 IF d0 ='0' THEN s <= NOT a; 13 ELSE s <= a XOR b; 14 END IF; 15 ELSE 16 IF d0 ='0' THEN s <= a AND b; 17 ELSE s <= a OR b; 18 END IF; 19 END IF; 20 END PROCESS; 21 END teste;</pre>	<pre> ENTITY t1_if2 IS PORT (d1,d0 : IN BIT; a, b : IN BIT; s : OUT BIT); END t1_if2; ARCHITECTURE teste OF t1_if2 IS SIGNAL d : BIT_VECTOR(1 DOWNTO 0); BEGIN d <= d1 & d0; PROCESS (d,a,b) BEGIN IF d ="00" THEN s <= NOT a; ELSEIF d ="01" THEN s <= a XOR b; ELSEIF d ="10" THEN s <= a AND b; ELSE d ="11" THEN s <= a OR b; END IF; END PROCESS; END teste;</pre>
--	---



• Descrições – CASE WHEN com e sem aninhamento

- a concatenação dos sinais d1 d0 evita o aninhamento
- novamente: código mais claro



<pre>1 ENTITY t1_cas1 IS 2 PORT (d1,d0 : IN BIT; 3 a, b : IN BIT; 4 s : OUT BIT); 5 END t1_cas1; 6 7 ARCHITECTURE teste OF t1_cas1 IS 8 BEGIN 9 PROCESS (d1,d0,a,b) 10 BEGIN 11 CASE d1 IS 12 WHEN '0' => 13 CASE d0 IS 14 WHEN '0' => s <= NOT a; 15 WHEN '1' => s <= a XOR b; 16 END CASE; 17 WHEN '1' => 18 CASE d0 IS 19 WHEN '0' => s <= a AND b; 20 WHEN '1' => s <= a OR b; 21 END CASE; 22 END CASE; 23 END PROCESS; 24 END teste;</pre>	<pre>ENTITY t1_cas2 IS PORT (d1,d0 : IN BIT; a, b : IN BIT; s : OUT BIT); END t1_cas2; ARCHITECTURE teste OF t1_cas2 IS SIGNAL d : BIT_VECTOR(1 DOWNTO 0); BEGIN d <= d1 & d0; PROCESS (d,a,b) BEGIN CASE d IS WHEN "00" => s <= NOT a; WHEN "01" => s <= a XOR b; WHEN "10" => s <= a AND b; WHEN "11" => s <= a OR b; END CASE; END PROCESS; END teste;</pre>
--	---



Cuidados na descrição: emprego de **IF ELSE** e **CASE WHEN**

- Construção **IF ELSE**:

- é possível uma nova expressão de escolha a cada estágio
- excesso de liberdade pode levar a erros:
 - exemplo: nem todas são condições cobertas
 - criação de elementos de memória desnecessários

- Construção **CASE WHEN**:

- expressão de escolha única
- condições de escolha não cobertas:
 - mensagem de erro na etapa de compilação

- Sempre que possível:

- empregar a construção **CASE WHEN**

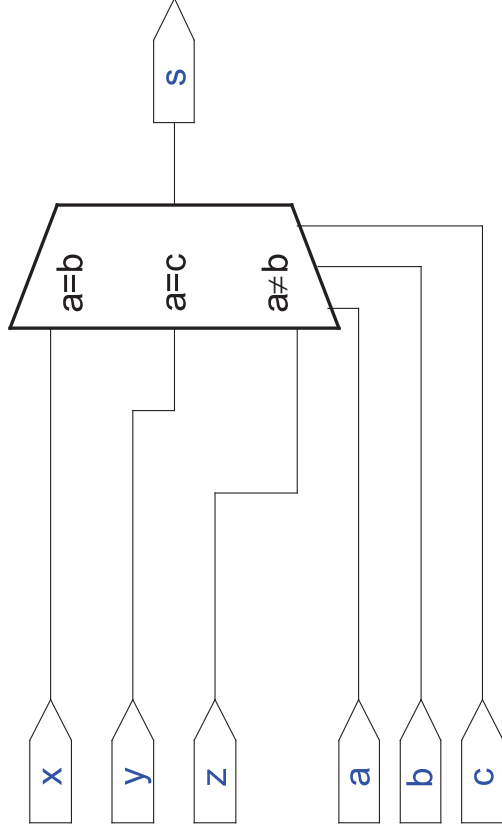
Cuidados na descrição: emprego de **IF ELSE** e **CASE WHEN**

- Especificação do problema

se $a = b$ então saída $s = x$

se $a = c$ então saída $s = y$

se $b \neq c$ então saída $s = z$



- Possível descrição empregando **IF ELSE**

1	ENTITY dont_if3 IS
2	PORT (x,y,z : IN BIT;
3	a,b,c : IN BIT;
4	s : OUT BIT);
5	END dont_if3;
6	
7	ARCHITECTURE teste OF dont_if3 IS
8	BEGIN
9	PROCESS (a,b,c,x,y,z)
10	BEGIN
11	IF a=b THEN s <= x;
12	ELSIF a=c THEN s <= y;
13	ELSIF b/=c THEN s <= z;
14	END IF;
15	END PROCESS;
16	END teste;



A descrição segue a especificação do problema?

- **Analisando a especificação do problema:** levantar uma tabela das condições
- **Lembrando: IF ELSE** – a sequência define a prioridade na escolha
 - linha 11: verificadas as condições 0, 1, 6 e 7 (maior prioridade)
 - linha 12: verificadas as condições 2 e 5
 - linha 13: deveria verificar 3 e 4 (nunca ocorre esta opção)

Condições possíveis levantadas

	a	b	c	a=b	a=c	b≠c
0	0	0	0	V	V	F
1	0	0	1	V	F	V
2	0	1	0	F	V	V
3	0	1	1	F	F	F
4	1	0	0	F	F	F
5	1	0	1	F	V	V
6	1	1	0	V	F	V
7	1	1	1	V	V	F

V = verdadeiro
F = falso

1	ENTITY dont_if3 IS
2	PORT (x,y,z : IN BIT;
3	a,b,c : IN BIT;
4	s : OUT BIT);
5	END dont_if3;
6	
7	ARCHITECTURE teste OF dont_if3 IS
8	BEGIN
9	PROCESS (a,b,c,x,y,z)
10	BEGIN
11	IF a=b THEN s <= x;
12	ELSIF a=c THEN s <= y;
13	ELSIF b/=c THEN s <= z;
14	END IF;
15	END PROCESS;
16	END teste;



A descrição segue a especificação do problema?

- Não
- Devido à flexibilidade da construção **IF ELSE**
 - problemas da especificação foram encobertos
- Opção pela construção **CASE WHEN**:
 - falha na especificação seria detectada
 - é necessário verificar todas as opções
- Neste caso, o problema pode ser facilmente detectado
- Descrições com muitas decisões podem encobrir problemas