LABORATÓRIO 2 – CIRCUITOS COMBINACIONAIS

EFRAIN MARCELO PULGAR PANTALEON[1]
FERNANDO LUCAS SOUSA SILVA[2]
MATHEUS GOMES DINIZ ANDRADE[3]
TEÓPHILO VITOR DE CARVALHO CLEMENTE[4]

Universidade Federal do Rio Grande do Norte, Departamento de Engenharia de Computação e Automação

1. INTRODUÇÃO

O presente relatório tem por objetivo analisar o funcionamento conjunto de portas lógicas, visto que, estas combinadas podem idealizar circuitos com diversas possibilidades, os ditos circuitos combinacionais. Assim, apresentaremos os resultados obtidos de acordo a cada item esclarecido no roteiro, de modo a mostrar os códigos usados para a emulação dos projetos e respectivos resultados adquiridos via o software Quartus.

2. METODOLOGIA

Para a realização deste laboratório foi necessário o conhecimento teórico a respeito do funcionamento das portas lógicas e quais suas finalidades práticas. Desse modo, para a experimentação prática foi utilizado o software Quartus II, ele possibilita a construção de diversos circuitos digitais a partir do uso de portas lógicas, flip-flops e outros componentes que podem ser simulados com seu uso, com isso foi possível a construção dos códigos em VHDL e posterior junção para que assim fosse possível a construção do circuito total como será mostrado nas seções a seguir.

3. RESULTADOS

Nesta seção serão apresentados os resultados obtidos nas simulações realizadas, como também os meios utilizados para as obter no software.

3.1. TABELA VERDADE

Baseada nas informações fornecidas pelo relatório foram observadas as combinações que deveriam ter obtidas nas saídas, a partir da quantidade de 1 e de 0, então se por exemplo se temos ABC todas em nível 1 a saída total deverá 3 em binário, se AB são 1 e C no 0 a saída deverá ser 2 em binário. A organização dela foi feita mediante A, B e C sendo as entradas S1 e S2 as saídas que

juntas representarão o número em binário como mostrado a seguir.

A	B	C	S1	S2	Número
1	1	1	1	1	3
1	1	0	1	0	2
1	0	0	0	1	1
0	0	0	0	0	0
1	0	1	1	0	2
0	1	0	0	1	1
0	0	1	0	1	1
0	1	1	1	0	2

Figura 1 – Tabelas verdade das saídas.

3.2. EQUAÇÕES DE CIRCUITO SIMPLIFICADAS

A seguir serão mostrados as equações dadas e a partir das propriedades e postulados foram simplificadas para ficar ao máximo reduzidas como veremos a seguir para S1, nela obtivemos uma combinação de AND's e OR's.

$S1 = A'\cdot B\cdot C + A\cdot B'\cdot C + A\cdot B\cdot C' + A\cdot B\cdot C$

$S1 = B\cdot C\cdot(A' + A) + A\cdot B'\cdot C + A\cdot B\cdot C'$

$S1 = B\cdot C + A\cdot B'\cdot C + A\cdot B\cdot C'$

$S1 = B\cdot(C + A\cdot C') + A\cdot B'\cdot C$

$S1 = B\cdot((C + A)\cdot(C + C')) + A\cdot B'\cdot C$

$S1 = B\cdot(C + A) + A\cdot B'\cdot C$

$S1 = B\cdot C + B\cdot A + A\cdot B'\cdot C$

$S1 = C\cdot(B + AB') + B\cdot A$

$S1 = C \cdot ((B + A) \cdot (B + B')) + B \cdot A$

$S1 = C \cdot (B + A) + B \cdot A$

$S1 = C \cdot B + C \cdot A + B \cdot A$

$S1 = A \cdot (B + C) + B \cdot C$

A seguir a simplificação para o S2 onde obtivemos combinação de portas XOR.

$S2 = A' \cdot B' \cdot C + A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B \cdot C$

$S2 = A' \cdot (B' \cdot C + B \cdot C') + A \cdot (B' \cdot C' + B \cdot C)$

$S2 = A' \cdot (B \text{ xor } C)' + A \cdot (B \text{ xor } C)$

$S2 = A \text{ xor } B \text{ xor } C$

3.3. REPRESENTAÇÃO EM PORTAS LÓGICAS

Para este tópico foi pedido a representação dos circuitos acima simplificados em portas lógicas e assim temos a noção de como serão as estruturas de S1 e S2. Para a realização desta parte foram criadas as portas lógicas e assim foi possível a montagem do circuito.

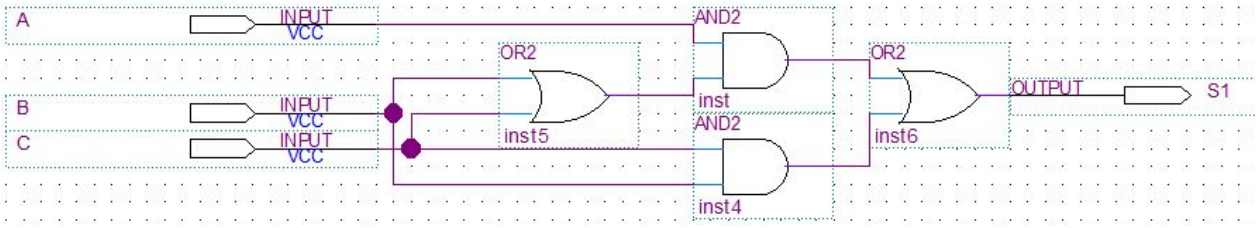

Figura 2 – Portas lógicas que compõem S1.

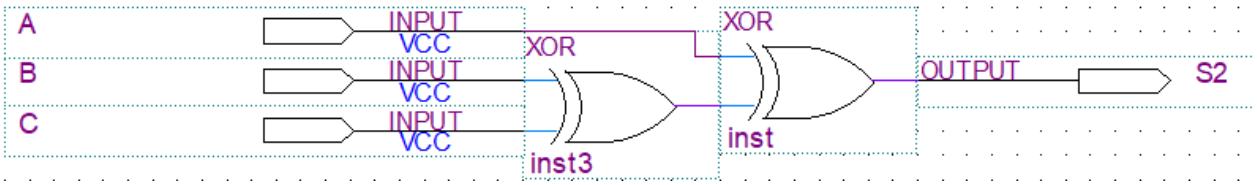

Figura 3 – Portas lógicas que compõem S2.

3.4. CIRCUITO EM VHDL E SIMULAÇÃO

No item 4 foi pedida a simulação do circuito total em VHDL, para isso foi necessário criar em projetos separados os respectivos componentes que precisaríamos, foram as portas AND, OR e XOR, de modo que também em cada respectivo código foi realizada a declaração de cada componente. Posteriormente, no código principal foi feita a junção dos componentes e juntamente com a instanciação para que assim fosse possível a compilação e obtenção dos resultados.

```
1  entity and_gate is
2  port(
3      input_and_1, input_and_2: in bit;
4      output_and: out bit
5  );
6  end and_gate;
7
8  architecture behav of and_gate is
9  begin
10
11 output_and <= input_and_1 and input_and_2;
12
13 end architecture;
```

Figura 4 – Código VHDL porta AND.

```
1  entity or_gate is
2    port(
3      input_or_1, input_or_2: in bit;
4      output_or: out bit
5    );
6  end or_gate;
7
8  architecture behav of or_gate is
9  begin
10
11   output_or <= input_or_1 or input_or_2;
12
13 end architecture;
```

Figura 5 – Código VHDL porta OR.

```
1  entity xor_gate is
2  port(
3      input_xor_1, input_xor_2: in bit;
4      output_xor: out bit
5  );
6  end xor_gate;
7
8  architecture behav of xor_gate is
9  begin
10
11 output_xor <= input_xor_1 xor input_xor_2;
12
13 end architecture;
```

Figura 6 – Código VHDL porta XOR.

A partir dos códigos acima vistos foi possível a criação de cada porta lógica, se atentando ao uso do PORT MAP em cada um, de modo que fosse possível no código principal que veremos logo abaixo o funcionamento de cada uma de forma combinada com as seguintes e assim sendo possível a obtenção dos resultados em binário que seriam a representação de 1, 2 ou 3.

```
1  entity circuito is
2  port(
3      a, b, c: in bit;
4      s1, s2: out bit
5  );
6  end circuito;
7
8  architecture behav of circuito is
9      --Sinais de S1
10     signal s_and_1: bit;
11     signal s_and_2: bit;
12     signal s_or: bit;
13
14     --Sinais de S2
15     signal s_xor: bit;
16
17     component and_gate is
18        port(
19           input_and_1, input_and_2: in bit;
20           output_and: out bit
21        );
22     end component and_gate;
23
24     component or_gate is
25        port(
26           input_or_1, input_or_2: in bit;
27           output_or: out bit
28        );
29     end component or_gate;
30
31     component xor_gate is
32        port(
33           input_xor_1, input_xor_2: in bit;
34           output_xor: out bit
35        );
36     end component xor_gate;
37
38 begin
39
40     -- S1
41     u1: or_gate port map(input_or_1 => b, input_or_2 => c, output_or => s_or);
42     u2: and_gate port map(input_and_1 => a, input_and_2 => s_or, output_and => s_and_1);
43     u3: and_gate port map(input_and_1 => b, input_and_2 => c, output_and => s_and_2);
44     u4: or_gate port map(input_or_1 => s_and_1, input_or_2 => s_and_2, output_or => s1);
45
46     --S2
47     u5: xor_gate port map(input_xor_1 => b, input_xor_2 => c, output_xor => s_xor);
48     u6: xor_gate port map(input_xor_1 => a, input_xor_2 => s_xor, output_xor => s2);
49
50 end behav;
```

Figura 7 – Código VHDL principal.

Com isso, realizamos a simulação do código com as 8 possibilidades de valores binários para A, B e C. É possível perceber que os resultados apresentados na figura 8, são compatíveis com os esperados e, assim, o código conta os bits.

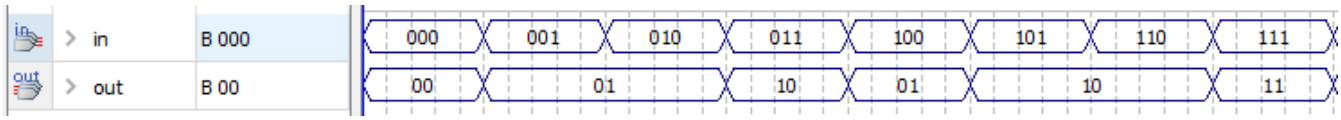

Figura 8 – Simulação do código VHDL principal.

4. CONCLUSÕES

A realização deste laboratório permitiu o melhor entendimento a respeito dos assuntos vistos nas aulas teóricas, colocando em prática a montagem de circuito com a combinação de portas lógicas simplificadas e análises do funcionamento do mesmo. Portanto, utilizamos da linguagem VHDL e simulações do ModelSim através do Quartus II para obtermos as nossas saídas teóricas desejadas.

5. REFERÊNCIAS

[1] QUARTUS II. Software de simulação.

[2] Vahid, Frank. Digital Design with RTL Design, VHDL, and Verilog Solution Manual. 2° Edição.2010.