

Relatório: Atividade 2

Filipe Augusto Parreira Almeida

RA: 2320622

12 de abril de 2024

Conteúdo

1 Resumo	2
2 Introdução	3
2.1 Dispositivos Lógicos Programáveis - PLDs	3
2.1.1 SPLDs	3
2.1.2 CPLDs	5
2.1.3 FPGAs	6
2.2 Placa de Desenvolvimento Utilizada	8
3 Implementação	10
3.1 Parte 1	10
3.2 Parte 2	12
4 Conclusão	16
4.1 Parte 1	16
4.2 Parte 2	17
Referências	19

1 Resumo

O presente relatório aborda o conceito técnico e prático de **PLDs**, mais especificamente o **FPGA DE10-Lite**, portanto, para melhor entendimento do mesmo, é necessário que se apresente o conceito de **PLD**, que consequentemente engloba os **SPLDs**, **CPLDs** e os **FPGAs**.

2 Introdução

2.1 Dispositivos Lógicos Programáveis - PLDs

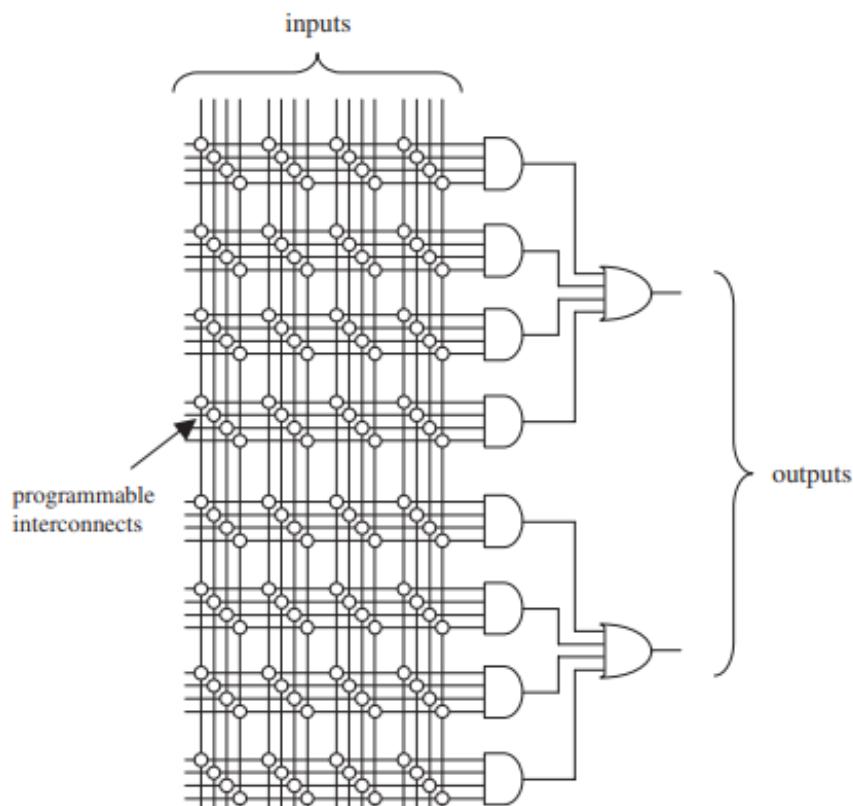
PLDs são dispositivos que possuem hardware fixo, onde a programabilidade se estende ao **nível de hardware**, ou seja, é um dispositivo de **uso geral** cujo hardware pode ser programado para **usos específicos** (PEDRONI, 2010). O conjunto de PLDs é composto pelos **SPLDs** (dispositivos lógicos programáveis simples), **CPLDs** (dispositivos lógicos programáveis complexos) e os **FPGAs** (sigla em inglês, field programmable gate arrays).

2.1.1 SPLDs

Os **SPLDs** tem um subconjunto que é composto pelos seguintes dispositivos:

- **PAL** (programmable array logic), lançado por volta da década de 1970, pela Monolithic Memories (PEDRONI, 2010). Sua arquitetura básica é formada por entradas, saídas, interconexões programáveis e portas lógicas **AND** e **OR**. Esta construção é baseada na ideia de que é possível escrever qualquer função lógica utilizando **soma** e **produto**. Sua arquitetura básica é representada pelo seguinte esquema:

Figura 1: Arquitetura Básica PAL.

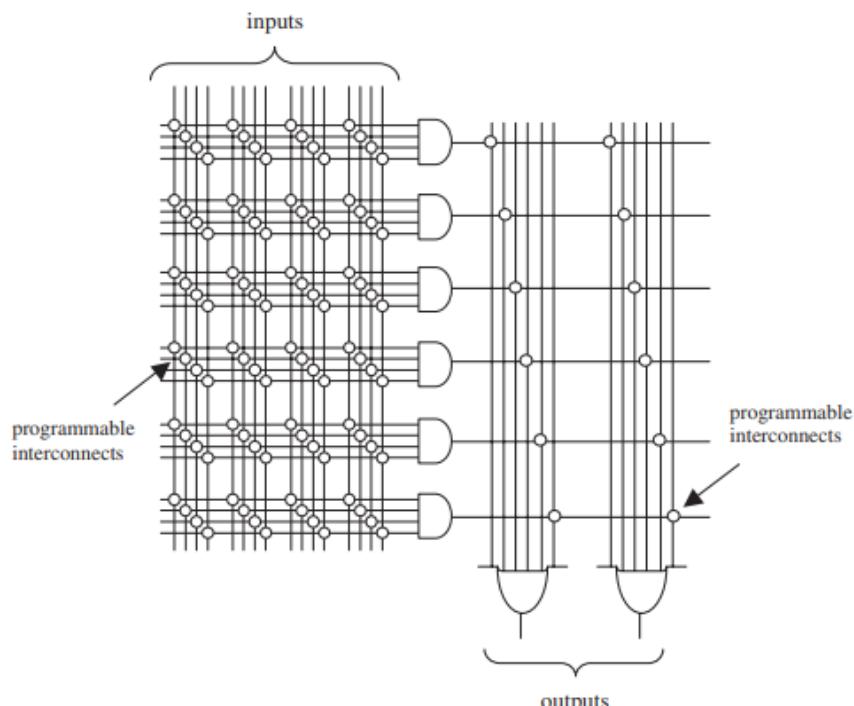


Fonte: (PEDRONI, 2010)

- **PLA** (programmable logic array), lançado pela Signetics basicamente no mesmo período que os **PAL** (PEDRONI, 2010), os **PLAs** tem uma **arquitetura básica muito semelhante** à do PAL, mas com uma estrutura mais compacta e eficiente.

Ihante aos PAL, porém acrescentando a opção de se programar também as saídas AND. Sua arquitetura básica é representada pelo seguinte esquema:

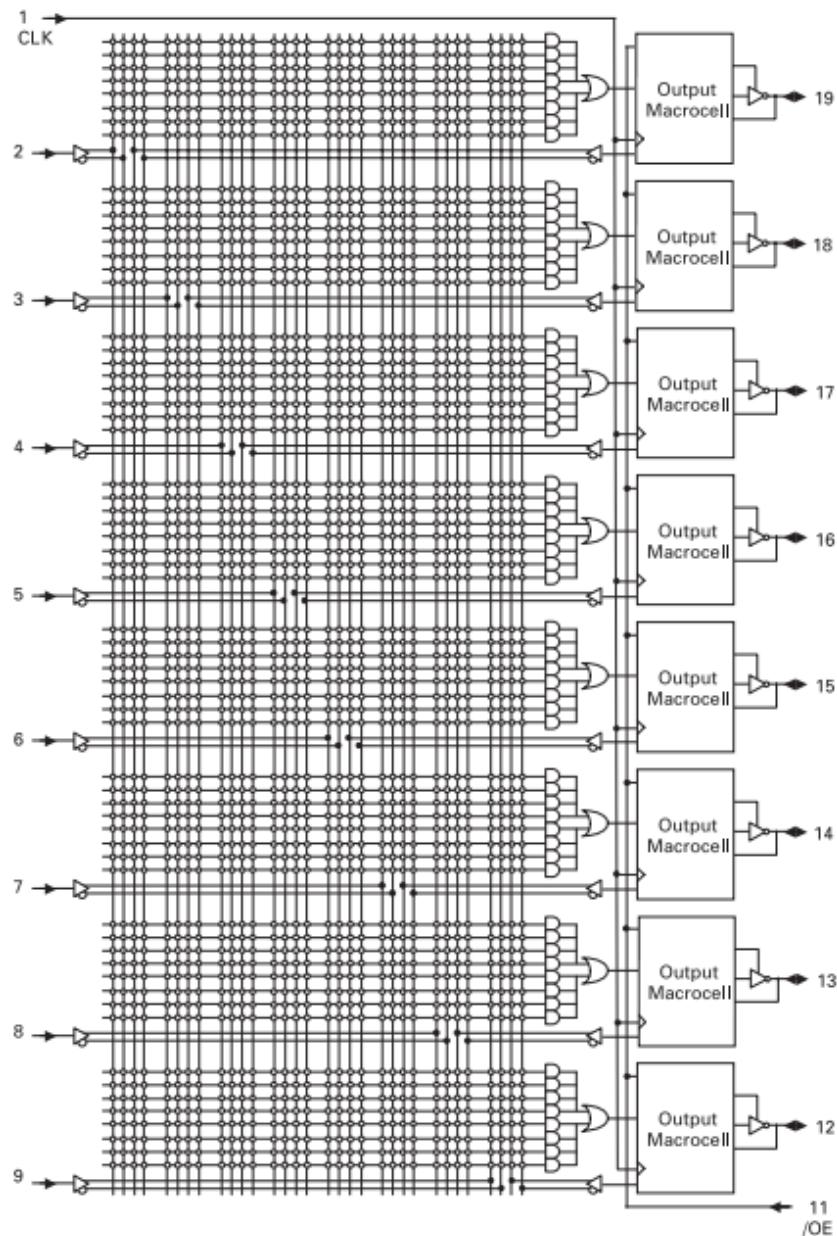
Figura 2: Arquitetura Básica PLA.



Fonte: (PEDRONI, 2010)

- **GAL** (generic array logic), foi lançado pela Lattice no início da década de 1980 (PEDRONI, 2010), ele possui diversas melhorias importantes comparado com os primeiros dispositivos **PAL**. Dentre elas foi construída uma célula de saída mais sofisticada, um **flip-flop**, porta **XOR**, cinco **multiplexadores**, memórias **EEPROM**, sinal de **retorno** (feedback) da macrocélula para o arranjo programável, e uma **assinatura eletrônica** para identificação e proteção do projeto (PEDRONI, 2010). Abaixo, a representação esquemática do chip **GAL 16V8**:

Figura 3: Arquitetura Básica GAL.

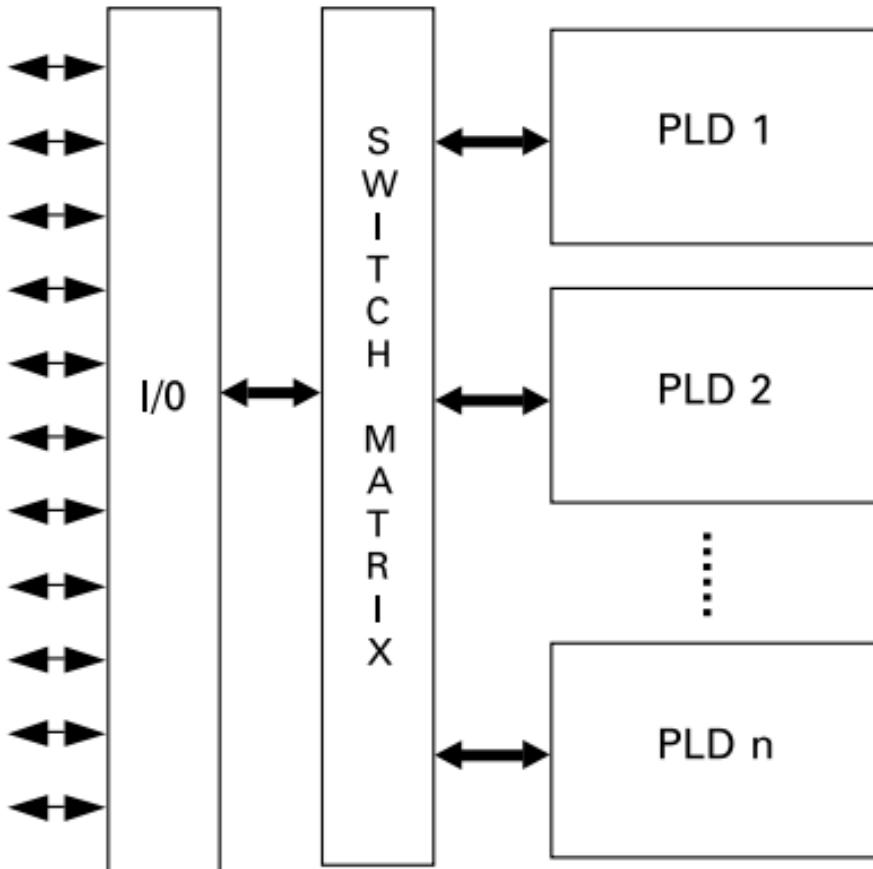


Fonte: (PEDRONI, 2010)

2.1.2 CPLDs

Posteriormente os **SPLDs** foram substituídos pelos **CPLDs**, ou **PLDs Complexos**. Que consiste basicamente na integração de diversos SPLDs em um mesmo *chip*, com a implementação também de *drivers* I/O atualizados, suporte **JTAG** e uma gama de portas de I/O disponíveis para o usuário. Como exemplos de **CPLDs** temos a série **XC9500** da Xilinx e a série **MAX3000** da Altera (PEDRONI, 2010). A arquitetura básica de um **CPLD** é demonstrada pelo seguinte diagrama:

Figura 4: Arquitetura Básica CPLD.

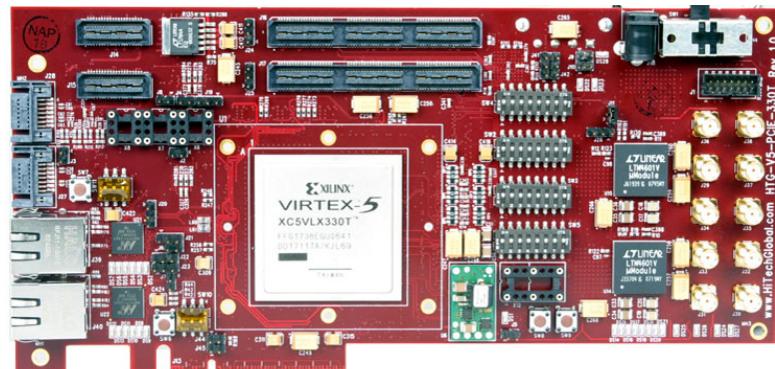


Fonte: (PEDRONI, 2010)

2.1.3 FPGAs

Em meados da década de 1980 os **FPGAs** foram lançados pela Xilinx. A arquitetura deles é formada por uma **matriz de blocos lógicos** ou configuráveis, blocos de memória **RAM** (inicialmente formados por memórias **SRAM** de uso geral), blocos **DSP** (utilizado para processamento de sinais digitais), e circuitos **PLL** (para gerenciamento de clock). Abaixo, se encontram as fotos de dois FPGAs, o **Virtex 5** da Xilinx, e o **Stratix III** da Altera, e também uma tabela de comparação entre eles:

Figura 5: FPGA Virtex 5 - Xilinx



Fonte: (HITECH GLOBAL, s.d.)

Figura 6: FPGA Stratix III - Altera



Fonte: (FPGAKEY, s.d.)

Figura 7: Stratix III x Virtex 5

Construção:	Xilinx Virtex 5 (LX series)	Altera Stratix III (L series)
Tecnologia	CMOS 65nm (SRAM)	CMOS 65nm (SRAM)
Voltagem do core	1V	0.9V or 1.1V
Número de CLBs (Virtex)	2,400 – 25,920	-----
Número de LABs (Stratix)	-----	1,900 – 13,520
Número de Slices (Virtex)	2 por CLB = 4,800 – 51,840	-----
Número de ALMs (Stratix)	-----	10 por LAB = 19,000 – 135,200
Número de flip-flops	4 por Slice = 19,200 – 207,360	2 por ALM = 38,000 – 270,400
Frequência máxima	550MHz	600MHz
Consumo de potência estática	Comparable to Stratix III	0.7W ⁽¹⁾ – 3.8W ⁽²⁾
Memória SRAM:		
Memória SRAM dos blocos para usuário	1.15 Mb – 10.4 Mb	1.8M – 16.2M
Memória SRAM distribuída (das Slices ou ALMs)	0.32 Mb – 3.4 Mb	0.6M – 4.2M
Total SRAM	1.47M – 13.8M	2.4M – 20.4M
Frequência máxima SRAM	550MHz	600MHz
DSP:		
Número de multiplicadores 18x25 (Virtex) 18x18 Or 36x36 (Stratix)	32 – 192 -----	----- 216 – 768 or 54 – 192
Frequência máxima DSP	550MHz	550MHz
Gerenciamento de clock:		
Pinos de clock; Número de networks de clock	20; 32 GCLKs + 96 RCLKs	32; 16 GCLKs + 88 RCLKs
PLLs	2 – 6	4 – 12
DLLs	12 (deskew do clock)	4 (interface com memória)
I/O:		
Número de pinos disponíveis	400 – 1,200	288 – 1,104
Padrões de I/O suportados	Todos da Seção 10.9 e outros	Todos da Seção 10.9 e outros
OCT (on-chip termination) com calibração automática	Sim	Sim

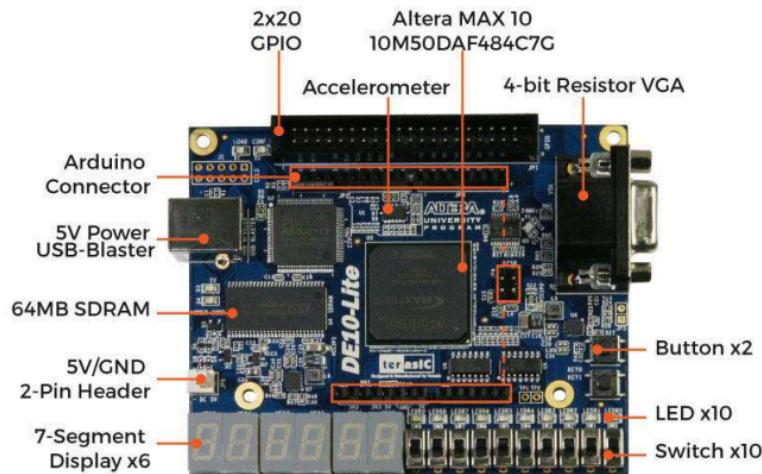
(1) Menor chip, com todos LABs em uso em modo low-power, sem SRAMs ou DSPs em uso.
(2) Maior chip, com todos os LABs, SRAMs e DSPs em uso, com 25% dos LABs em modo de alta freqüência.

Fonte: (PEDRONI, 2010)

2.2 Placa de Desenvolvimento Utilizada

A placa de desenvolvimento utilizada para implementação da síntese dos códigos abordados neste relatório é a **DE10-Lite**, fabricada pela **Terasic**, ela utiliza o **FPGA MAX 10 10M50DAF484C7G**, da Altera; 64MB de SDRAM, 6 displays de 7 segmentos, 10 LEDs e 10 switches e entre outras especificações que são apresentadas pela imagem abaixo:

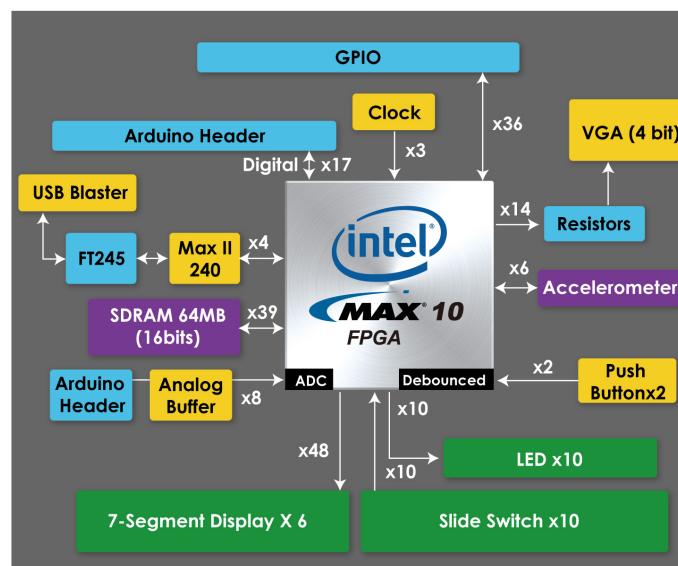
Figura 8: Componentes DE10-Lite



Fonte: (TERASIC TECHNOLOGIES, 2020)

Também é possível realizar essa representação via diagrama de blocos da seguinte forma:

Figura 9: Diagrama de Blocos DE10-Lite



Fonte: (TECHNOLOGIES, T., s.d.)

3 Implementação

A implementação do que é pedido para ser apresentado no relatório pode ser dividido em duas partes: a implementação do código utilizado na **atividade anterior** (implementação das portas lógicas), e o desenvolvimento de um **novo algoritmo**, a partir de uma certa ideia:

3.1 Parte 1

O requisito para esta parte é a implementação do código da **Atividade 1** na placa de desenvolvimento. O código em si é basicamente a representação das portas lógicas para duas entradas. Segue o código em **VHDL**:

```
1 -- ENTIDADE
2 ENTITY Atividade1 IS PORT (
3     a, b: IN STD_LOGIC;
4     outNOTa, outNOTb, outOR, outAND,
5         outNAND, outNOR, outXNOR, ourXOR: OUT STD_LOGIC
6 );
7 END Atividade1;
8
9 -- Declara o de Bibliotecas/Pacotes
10 LIBRARY ieee;
11 USE ieee.std_logic_1164.all;
12
13 -- ARQUITETURA
14 ARCHITECTURE arq_AT1 OF Atividade1 IS
15 BEGIN
16     outNOTa <= NOT a;
17     outNOTb <= NOT b;
18     outOR <= a OR b;
19     outAND <= a AND b;
20     outNAND <= a NAND b;
21     outNOR <= a NOR b;
22     outXNOR <= a XNOR b;
23     outXOR <= a XOR b;
24 END arq_AT1;
```

Código 1: Código VHDL Atividade 1

Para implementação na placa de desenvolvimento, foi utilizado, para visualização das **saídas**, os **LEDs** presentes na placa, e utiliza-se dois **switches** para as **entradas A e B**. O software utilizado para esta implementação foi o **Quartus II**, e a partir do **Pin Planner** presente no mesmo, foi realizada a configuração dos pinos que representam os LEDs e os switches para as entradas e as saídas.

Por meio do manual de usuário da placa de desenvolvimento é possível identificar quais pinos correspondem aos **LEDs** e quais correspondem aos **switches**. Todas essas informações são apresentadas nas seguintes tabelas presentes no manual da placa:

Figura 10: Tabela de pinos switches.

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

Fonte: (TERASIC TECHNOLOGIES, 2020)

Figura 11: Tabela de pinos LEDs.

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTL

Fonte: (TERASIC TECHNOLOGIES, 2020)

Sendo assim, no *Quartus II* foi realizado as seguintes configurações para as entradas e saídas:

Figura 12: Tabela Pin Planner Quartus II Atividade 1.

a	Input	PIN_C10
b	Input	PIN_C11
zAND	Output	PIN_A8
zNAND	Output	PIN_A9
zNOR	Output	PIN_A10
zNOTA	Output	PIN_B10
zNOTB	Output	PIN_D13
zOR	Output	PIN_C13
zXNOR	Output	PIN_E14
zXOR	Output	PIN_D14

Fonte: Autoria Própria

Ou seja, a **entrada A** corresponde ao **primeiro switch** da direita para a esquerda, e a **entrada B** corresponde ao **segundo**; e as **saídas** estão em ordem da direita para a esquerda começando a partir do **AND**.

3.2 Parte 2

Para a segunda parte, temos o enunciado que diz que se deve criado um código em **VHDL** que implemente a lógica para um **circuito de alarme**, onde ele possui as seguintes entradas:

- Porta do Carro (P=0 representa a porta fechada e P=1 representa a porta aberta)
- Motor (M=0 representa motor desligado e M=1 representa motor ligado)
- Freio de mão (F=0 representa o freio de mão desativado e F=1 representa o freio ativado)
- Cinto de Segurança (C=0 representa o motorista sem cinto e C=1 representa o motorista com cinto)

E o alarme deve ser acionado nas seguintes condições:

- Porta do motorista está aberta e o motor está ligado;
- Porta está fechada, motor ligado e o motorista não estiver usando o cinto de segurança.
- Porta está fechada, motor ligado e freio de mão ativado;

Dado o problema, é perceptível que pode ser melhor interpretado com a criação de uma **tabela-verdade**, que possibilita uma melhor visualização de como implementar esta lógica de forma mais eficiente. Sendo assim foi criado uma tabela-verdade que relaciona todas as **possibilidades de entrada** e todas as **saídas possíveis**:

Tabela 1: Tabela-Verdade - Atividade 2

P	M	F	C	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Definido uma tabela-verdade é possível que se crie um **mapa de karnaugh** a partir da mesma. Permitindo uma **melhor otimização** para a implementação da lógica em **VHDL**. Segue o mapa de karnaugh criado com base na Tabela 1:

Figura 13: Mapa de Karnaugh

		FC				
		00	01	11	10	
PM		00	0	0	0	0
		01	1	0	1	1
		11	1	1	1	1
		10	0	0	0	0

Com base na Figura 13 é possível que se realize a melhor otimização possível da expressão lógica, com o isolamento das saídas 1's. Obtém-se então os seguintes agrupamentos:

Figura 14: Mapa de Karnaugh Agrupado

		FC			
		00	01	11	10
PM	00	0	0	0	0
	01	1	0	1	1
	11	1	1	1	1
	10	0	0	0	0

Logo, é possível gerar a expressão lógica com base na ideia de **soma de produtos** por meio do **mapa de karnaugh**, saindo de uma expressão totalmente **mal otimizada** (Equação 1) e chegando em uma expressão **otimizada** (Equação 2):

$$Y = \overline{PM}\overline{FC} + PM\overline{FC} + PM\overline{FC} + PMFC + PM\overline{C} + \overline{PM}FC + \overline{PM}\overline{C} \quad (1)$$

$$Y = M\overline{C} + PM + FM \quad (2)$$

Chegando em uma expressão otimizada, basta transcrevê-la em **VHDL**, onde definimos as **entradas**, e a **saída** do circuito; a saída **recebe** a expressão lógica otimizada. O código **VHDL** é demonstrado abaixo:

```

1 -- Declara o de Bibliotecas e Pacotes
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4
5 -- Entidade
6 ENTITY atividade_2 IS PORT(
7   -- Entradas
8   p, m, f, c : IN STD_LOGIC;
9   -- Saída
10  sirene : OUT STD_LOGIC
11 );
12 END atividade_2;
13
14 -- Arquitetura
15 ARCHITECTURE main OF atividade_2 IS
16
17 BEGIN
18   -- Saída recebe expressão lógica otimizada
19   sirene <= (m AND (NOT c)) OR (p AND m) OR (f AND m);
20
21 END main;
```

Código 2: Código VHDL Atividade 2

Analizando o código, nota-se que a entidade criada recebe o nome de “*atividade_2*” e ela está definindo **p**, **m**, **f**, **c** como portas do tipo **STD_LOGIC**, que no escopo deste relatório recebe valores **0 ou 1**; as portas de entrada em questão representam **a porta**, **o motor**, **o farol** e **o cinto de segurança**, respectivamente; e há somente **uma saída** que no caso é a sirene. A arquitetura da entidade, definida como **main**, contém somente a transcrição da expressão lógica e sua atribuição para a saída.

Para a implementação na placa de desenvolvimento, foi realizado as seguintes atribuições de pinos:

Figura 15: Tabela Pin Planner Quartus II Atividade 2.

in c	Input	PIN_C12
in f	Input	PIN_D12
in m	Input	PIN_C11
in p	Input	PIN_C10
out sirene	Output	PIN_B11

Fonte: Autoria Própria

Onde as entradas seguem um padrão semelhante ao da parte 1, sendo da **direita para a esquerda** de acordo com a ordem da **tabela-verdade**; a saída foi atribuída ao último LED da **direita para a esquerda**, mais precisamente o **LEDR9**.

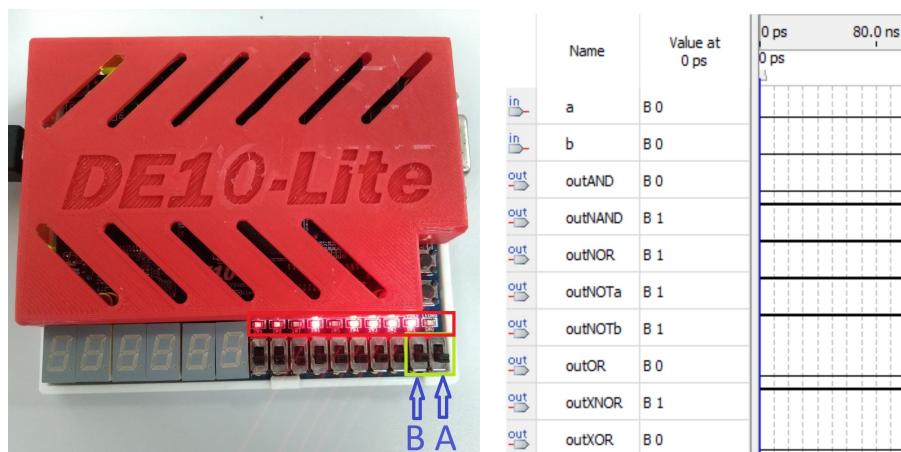
4 Conclusão

Já realizada a implementação de ambas as partes nas placas de desenvolvimento, obteve-se os seguintes resultados:

4.1 Parte 1

Segue fotos tiradas da placa de desenvolvimento executando o algoritmo implementado para **diferentes entradas** onde é possível que se visualize também as **saídas para as respectivas entradas**, junto a elas uma comparação da **simulação de forma de onda** via *software* para as entradas apresentadas:

Figura 16: Comparativo simulação e implementação - Entrada 00 - Parte 1



Fonte: Autoria Própria

Figura 17: Comparativo simulação e implementação - Entrada 10 - Parte 1



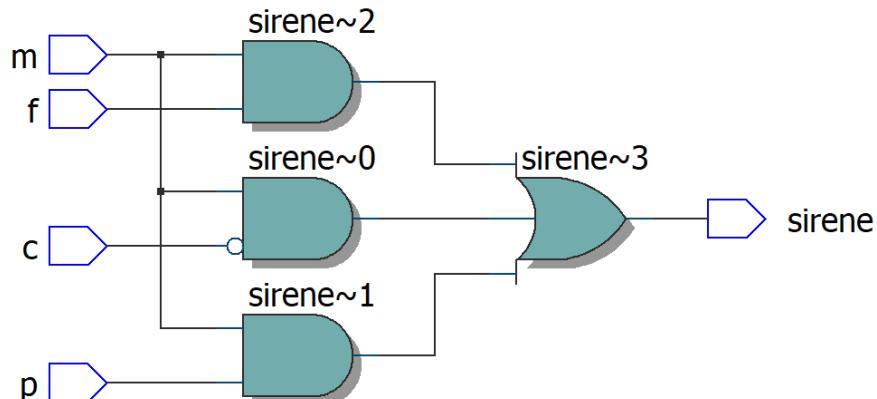
Fonte: Autoria Própria

Há de se perceber que nesta parte de implementação o resultado foi o **esperado**, quando comparamos a implementação real na placa de desenvolvimento e a simulação via *software*.

4.2 Parte 2

Realizada as modificações da proposta inicial, de forma que otimize o máximo possível, obteve-se o seguinte diagrama de circuitos:

Figura 18: Diagrama RTL - Parte 2



Fonte: Autoria Própria

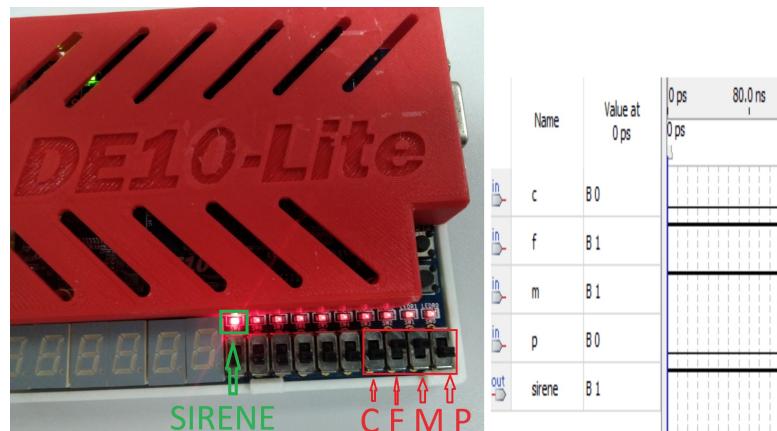
E como na parte 1, foi realizada a implementação real na placa de desenvolvimento, portanto segue as imagens com da placa com o código VHDL implementado. Foi registrado o resultado para entradas distintas, para que seja possível a visualização do funcionamento, e junto a esses registros, a simulação via software, para comparação.

Figura 19: Comparativo simulação e implementação - Entrada 0000 - Parte 2



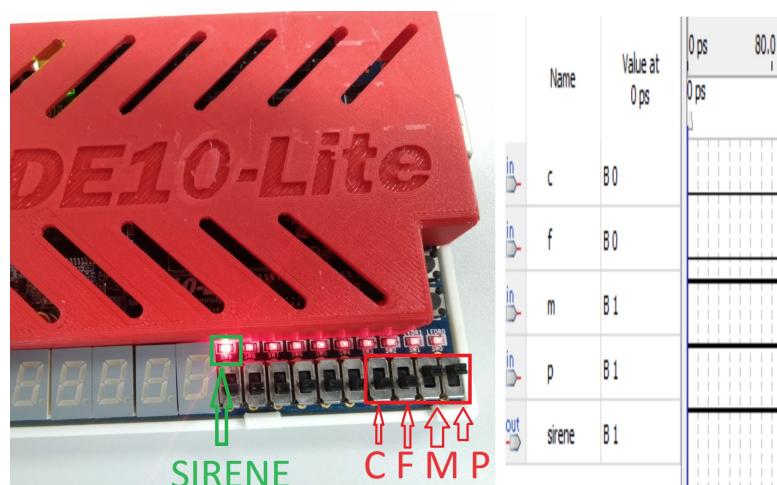
Fonte: Autoria Própria

Figura 20: Comparativo simulação e implementação - Entrada 0110 - Parte 2



Fonte: Autoria Própria

Figura 21: Comparativo simulação e implementação - Entrada 1100 - Parte 2



Fonte: Autoria Própria

Analisando as imagens percebe-se que a implementação foi realizada com **sucesso**, validando a parte teórica de otimização de expressões lógicas via mapa de karnaugh.

Referências

FPGAKEY. Altera Stratix III Enhanced FPGA - FPGA Familis - FPGAkey.
Disponível em: <<https://www.fpgakey.com/altera-family/stratix-iii-enhanced>>.

HITECH GLOBAL. Virtex-5 LX330T FX200T SX240T PCI Express SATA board.
Disponível em: <<https://www.hitechglobal.com/boards/pciexpresslx330t.htm>>.

PEDRONI, Volnei A. **Eletrônica digital moderna e VHDL**. Rio de Janeiro, RJ: Elsevier, 2010. P. 619. ISBN 9788535234657.

TECHNOLOGIES, T. **Terasic - All FPGA Boards - MAX 10 - DE10-Lite Board**.
Disponível em: <<https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=218&No=1021&PartNo=2#heading>>.

TERASIC TECHNOLOGIES. **Manual da Placa de Desenvolvimento DE10-Lite**. [S.l.], 2020. Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=China&No=1021&FID=a13a2782811152b477e60203d34b1baa.