

# SISTEMA DIDÁCTICO DE BAIXO CUSTO COM FPGA DE ALTA DENSIDADE

André Filipe da Silva Oliveira



Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2012



Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: André Filipe da Silva Oliveira, Nº 1101281, 1101281@isep.ipp.pt

Orientação científica: José Vieira do Santos, jvs@isep.ipp.pt



Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

12 de Dezembro de 2012





## *Agradecimentos*

Várias pessoas e instituições contribuíram directa ou indirectamente para a efectivação deste trabalho e gostaria de aqui lhes exprimir os meus agradecimentos.

Em relação às instituições, gostaria de agradecer ao Instituto Superior de Engenharia do Porto, nas pessoas do Eng. José Vieira dos Santos, que no papel de orientador, me impulsionou para a realização deste trabalho, me deu constante apoio e orientação, que foram indispensáveis para a conclusão do mesmo.

Em relação aos agradecimentos pessoais, não poderia deixar de começar por agradecer ao Eng. Mário Felgueiras pela assistência incansável e rapidez de resposta aos pedidos que lhe foram feitos e aos meus amigos, pela constante confiança e encorajamento demonstrados.

Por último e principalmente, gostaria de agradecer à minha família, em especial aos meus irmãos que não deixaram de me motivar e que suportaram as minhas alterações de personalidade criadas pelas dificuldades que foram surgindo durante este trabalho.



## *Resumo*

O uso das *Field-Programmable Gate Array* tem crescido de forma exponencial. Com isto dito, é importante que os engenheiros electrotécnicos estejam familiarizados com este tipo de tecnologia.

Foi com o intuito de passar estas valências para os alunos do ISEP, que surgiu a ideia de criar um sistema didáctico, que permitisse ao alunos aprender a trabalhar com estes dispositivos.

O seguinte trabalho iniciou-se com base num estudo das características destes dispositivos e das suas potencialidades, seguido de uma avaliação do que o mercado tem para oferecer.

Posteriormente, com base em toda a informação reunida, foi definida a arquitectura do sistema, que levou selecção de dispositivos a incluir no mesmo, e culminando na concepção do esquema eléctrico do sistema e da placa de circuito impresso correspondente ao protótipo do mesmo.

As principais directivas para este projecto foram o uso de uma FPGA de alta densidade e a concepção da ferramenta com o custo de projecto o mais reduzido possível.

## *Palavras-Chave*

FPGA, sistema didáctico, FPGA de alta densidade, custo de projecto o mais reduzido possível.









## *Abstract*

The use of Field-Programmable Gate array has grown exponentially. With that said, it is important to electrical engineers to be familiar with this technology.

It was with this tough in mind, that the idea of develop an educational system for the ISEP students came to place, so they could learn how to operate with this devices.

The following work was initiated based on a study of the characteristics and capabilities of these devices, followed bay an assessment of what the market as to ofer.

Subsquently,the system architecture was set, based on all the information gathered, leading to the selections of the devices to be included on the system , culminating on the design of the electrical schematic of the system and the corresponding printed circuit board prototype.

The main directives for this project were a use of a high density FPGA and keeping the design costs as low as possible.

## *Keywords*

FPGA, educational system, high density FPGA, design costs as low as possible.



# Índice

<b>1. INTRODUÇÃO .....</b>	<b>7</b>
1.1. CONTEXTUALIZAÇÃO .....	7
1.2. OBJECTIVOS .....	8
1.3. CALENDARIZAÇÃO .....	9
1.4. ESTRUTURA DO RELATÓRIO .....	9
<b>2. ESTADO DA ARTE.....</b>	<b>13</b>
2.1. PROGRAMMABLE READ ONLY MEMORY (PROM) .....	14
2.2. PROGRAMMABLE LOGIC ARRAY (PLA).....	16
2.3. PROGRAMMABLE ARRAY LOGIC (PAL).....	18
2.4. GENERIC ARRAY LOGIC (GAL).....	19
2.5. COMPLEX PROGRAMMABLE LOGIC DEVICE (CPLD) .....	20
2.5.1. Matriz de Interligações de um CPLD .....	21
2.5.2. Blocos de Lógica de um CPLD.....	22
2.6. FIELD-PROGRAMMABLE GATE ARRAY(FPGA).....	23
2.6.1. Tecnologias de Programação .....	26
2.6.2. Blocos de I/O .....	37
2.6.3. Arquitecturas de FPGA's.....	39
2.6.4. Modos de Configuração de uma FPGA.....	59
2.6.5. Linguagens de Programação de FPGA.....	69
2.6.6. Kits de Desenvolvimento Baseados em FPGA.....	76
<b>3. PROJECTO DO SISTEMA DIDÁCTICO .....</b>	<b>88</b>
3.1. REQUISITOS DE HARDWARE DO SISTEMA DIDÁCTICO.....	88
3.2. ARQUITECTURA DO SISTEMA .....	90
3.2.1. FPGA .....	91
3.2.2. Microcontrolador.....	92
3.2.3. Conector VGA.....	93
3.2.4. Periféricos de I/O.....	94
<b>4. IMPLEMENTAÇÃO DO PROJECTO.....</b>	<b>96</b>
4.1. SELECÇÃO DE COMPONENTES .....	97
4.1.1. FPGA .....	97
4.1.2. Selecção da Memória de Inicialização da FPGA.....	102
4.1.3. Microcontrolador.....	103
4.1.4. Transceiver RS232 .....	105
4.1.5. LCD .....	105
4.1.6. DAC's de Interface com o Conector VGA .....	106

4.1.7.	<i>Slot Cartão SD</i> .....	107
4.1.8.	<i>Oscilador</i> .....	107
4.2.	CONCEPÇÃO DO PROTÓTIPO DO SISTEMA .....	108
4.2.1.	<i>Esquema de Ligações</i> .....	108
4.2.2.	<i>Placa de Circuito Impresso</i> .....	122
4.2.3.	<i>Programação dos Dispositivos do Sistema</i> .....	132
4.2.4.	<i>Estudo de Custos de Fabrico do Sistema Didático</i> .....	136
<b>5.</b>	<b>VALIDAÇÃO DO SISTEMA</b> .....	<b>140</b>
5.1.	SOCKET FPGA .....	140
5.2.	LISTA DE ENCARGOS .....	143
<b>6.</b>	<b>CONCLUSÕES</b> .....	<b>144</b>
<b>7.</b>	<b>REFERÊNCIAS BIBLIOGRÁFICAS</b> .....	<b>146</b>
<b>ANEXO A. SÍMBOLO DA FPGA EP3C5E144C7N</b> .....		<b>148</b>
<b>ANEXO B. ESQUEMA DE LIGAÇÕES DA PCB INICIAL</b> .....		<b>149</b>
<b>ANEXO C. LAYOUT DA PCB INICIAL</b> .....		<b>154</b>
<b>ANEXO D. ESQUEMA DE LIGAÇÕES DA PLACA INFERIOR</b> .....		<b>155</b>
<b>ANEXO E. ESQUEMA DE LIGAÇÕES DA PLACA SUPERIOR</b> .....		<b>159</b>
<b>ANEXO F. ESQUEMA DE LIGAÇÕES DO PROTÓTIPO</b> .....		<b>161</b>
<b>ANEXO G. LAYOUT DA PCB DO PROTÓTIPO</b> .....		<b>164</b>

## Índice de Figuras

Figura 1	Famílias de dispositivos PLD. [1] .....	14
Figura 2	Arquitectura de uma PROM de 4 entradas. [2] .....	15
Figura 3	Arquitectura interna da PLA. [2].....	17
Figura 4	Arquitectura interna da PAL. [2].....	18
Figura 5	Diagrama de blocos da GAL22V10. [2].....	19
Figura 6	Arquitectura de um CPLD. [1] .....	20
Figura 7	Estrutura de um bloco de lógica de um CPLD. [2] .....	22
Figura 8	Arquitectura de uma Macro célula. [2].....	23
Figura 9	Arquitectura básica de uma FPGA. [2] .....	24
Figura 10	Estrutura base de bloco de lógica programável. [1] .....	25
Figura 11	Configuração de uma LUT. [1] .....	26
Figura 12	Ligações <i>antifuse</i> não programadas.....	27
Figura 13	Implementação de uma função lógica através da programação dos <i>antifuse</i> . ....	27
Figura 14	Estados possíveis de <i>antifuse</i> . a) <i>antifuse</i> no estado não programado; b) <i>antifuse</i> após programação. [1] .....	28
Figura 15	Exemplo de um sistema de redundância tripla. ....	29
Figura 16	Célula programável baseada em SRAM. [3] .....	30
Figura 17	Transístor MOS Standard <i>Versus</i> Transístor EEPROM. a) Transístor MOS Standard. b) Transístor EEPROM. [1].....	32
Figura 18	Célula de memória baseada em transístor EEPROM. [1] .....	33
Figura 19	Célula e memória EEPROM. [1].....	35
Figura 20	Bancos de <i>general purpose</i> I/O de uma FPGA. [3].....	37
Figura 21	Bloco de lógica baseado em <i>multiplexers</i> . [3].....	40
Figura 22	Função lógica e respectiva tabela de verdade. [3] .....	41
Figura 23	LUT baseada em <i>transmission gates</i> . [3] .....	42
Figura 24	Células de configuração ligadas em cadeia. [3] .....	43
Figura 25	LUT representada como um elemento multifacetado. [3] .....	43
Figura 26	Vista simplificada de uma LC da Xilinx. [3] .....	44
Figura 27	<i>Slice</i> da Xilinx, constituída por duas LC. [3] .....	45
Figura 28	CLB que contem quatro <i>slices</i> . [3] .....	46
Figura 29	Arquitectura de uma FPGA com <i>block</i> RAM. [3].....	47
Figura 30	Vista aproximada dos blocos de multiplicação. [3].....	48
Figura 31	Funções que formam um bloco MAC. [3].....	49



Figura 32	<i>Chip com um microprocessor core embebido, fora do circuito principal da FPGA. [3] ..</i>	51
Figura 33	<i>Microprocessor cores embebidos no circuito principal da FPGA. [3].....</i>	52
Figura 34	<i>Clock tree. [3].....</i>	53
Figura 35	<i>Clock manager e daughter clocks gerados. [3] .....</i>	54
Figura 36	<i>Clock difuso criado pelo fenómeno de jitter. [3].....</i>	55
Figura 37	<i>Uso do clock manager para remover o jitter. [3] .....</i>	55
Figura 38	<i>Uso do clock manager para efectuar uma síntese de frequência. [3] .....</i>	56
Figura 39	<i>Uso do clock manager para criar daughter clocks desfasados. [3].....</i>	56
Figura 40	<i>Monitorização e correcção de um daughter clock. [3] .....</i>	57
Figura 41	<i>Uso de um bus para comunicação entre dispositivos. [3].....</i>	58
Figura 42	<i>Uso de transceivers de alta velocidade para comunicar entre dispositivos. [3].....</i>	59
Figura 43	<i>FPGA SRAM vista como um deslocador de registos longo. [3].....</i>	60
Figura 44	<i>Serial Load com a FPGA a funciona como mestre. [3].....</i>	62
Figura 45	<i>FPGA's conectadas em daisy-chain. [3] .....</i>	63
Figura 46	<i>Parallel Load com a FPGA a funcionar como mestre (técnica original). [3] .....</i>	63
Figura 47	<i>Parallel Load com a FPGA como mestre (técnica moderna). [3].....</i>	64
Figura 48	<i>Parallel Load com a FPGA como escravo. [3] .....</i>	65
Figura 49	<i>Evolução do encapsulamento dos dispositivos. [7] .....</i>	66
Figura 50	<i>Conceito básico subjacente à tecnologia BST. [8] .....</i>	67
Figura 51	<i>Registos Boundary Scan do JTAG. [3] .....</i>	68
Figura 52	<i>Diferentes níveis de abstracção. [3].....</i>	70
Figura 53	<i>Exemplo de código de operações RTL. [3] .....</i>	71
Figura 54	<i>Exemplo de código de descrição funcional de um circuito. [3] .....</i>	71
Figura 55	<i>Níveis de abstracção do Verilog. [3].....</i>	73
Figura 56	<i>Níveis de abstracção, Verilog Versus VHDL. [3] .....</i>	75
Figura 57	<i>Cyclone III Starter Development Kit da Altera. [10].....</i>	76
Figura 58	<i>Cyclone III Development Kit da Altera.[10].....</i>	77
Figura 59	<i>Kit EZ1CUSB da Easy FPGA. [11] .....</i>	78
Figura 60	<i>Kit EZ1KCUSB da Easy FPGA. [11].....</i>	79
Figura 61	<i>Kit DE0-Nano Board da Terasic. [12] .....</i>	80
Figura 62	<i>Kit DE2 Board da Terasic. [12].....</i>	81
Figura 63	<i>IGLOO Nano Starter Kit da Actel. [13].....</i>	83
Figura 64	<i>IGLOO Plus Starter Kit. [13] .....</i>	84
FIGURA 65	<i>Kit Avnet Spartan-6 LX9 MicroBoard da Xilinx. [14] .....</i>	85
Figura 66	<i>Spartan 3AN Evaluation Kit da Xilinx. [14] .....</i>	86
Figura 67	<i>Arquitectura do sistema didáctico. ....</i>	90
Figura 68	<i>Implementação do protocolo RS232. ....</i>	91
Figura 69	<i>Funções do microcontrolador. ....</i>	93

Figura 70	Interface da FPGA com o conector VGA.....	94
Figura 71	Periféricos de I/O.....	95
Figura 72	FPGA EP3C5E144C7N da Altera.[19] .....	98
Figura 73	LE da família <i>Cyclone</i> III. [15] .....	100
Figura 74	Estrutura de um LAB da <i>Cyclone</i> . [15] .....	102
Figura 75	Memoria Flash de inicialização de FPGA EPCS16SI8N. [19] .....	103
Figura 76	<i>Programmable Interface Controller</i> (PIC) 18F4550 da Microchip.[19].....	104
Figura 77	MAX232 com encapsulamento SOIC de 16 pinos.[19].....	105
Figura 78	LCD de 2 linhas e 16 caracteres. [19] .....	105
Figura 79	<i>Slot</i> cartão SD <i>standard</i> . [19] .....	107
Figura 80	Oscilador de 50MHz. [19].....	107
Figura 81	Bloco de alimentação da <i>board</i> . .....	110
Figura 82	Símbolo da FPGA EP3C5E144C7 da Altera. ....	111
Figura 83	Encapsulamento EQPF de 144 pinos da FPGA EC3P5E144C7. ....	111
Figura 84	Circuito de configuração da FPGA. ....	112
Figura 85	Esquema de ligações do PIC. ....	114
Figura 86	Esquema de ligações do conector RS232. ....	115
Figura 87	Saídas disponíveis para o utilizador. ....	115
Figura 88	Circuito do oscilador. ....	116
Figura 89	Pente macho que interliga o PIC com a placa superior, com porto de programação incluído.....	116
Figura 90	Pente macho q interliga a FPGA com os periféricos da placa superior.....	117
Figura 91	Esquema de ligação do LCD. ....	118
Figura 92	Esquema de ligações do conector VGA. ....	119
Figura 93	Esquema de ligações do LED's.....	120
Figura 94	Esquema de ligações dos botões de pressão.....	120
Figura 95	Esquema de ligações cartão SD.....	121
Figura 96	Mecanismo de selecção de <i>clock</i> .....	122
Figura 97	Aspecto final da placa inferior do sistema. ....	123
Figura 98	<i>Layout</i> da face de cima (TOP) da placa de circuito impresso inferior. ....	124
Figura 99	<i>Layout</i> da face de baixo (BOTTOM) da placa de circuito impresso inferior. ....	125
Figura 100	Componentes da placa de circuito impresso inferior.....	126
Figura 101	Aspecto final da placa superior. ....	128
Figura 102	<i>Layout</i> da face de cima (TOP) da placa de circuito impresso superior. ....	129
Figura 103	<i>Layout</i> da face de baixo (TOP) da placa de circuito impresso superior. ....	130
Figura 104	Componentes da placa de circuito impresso superior. ....	131
Figura 105	PICKIT III da Microchip.....	133
Figura 106	Ambiente de trabalho genérico do MPLAB. ....	133

Figura 107	Aspecto geral do compilador mikroC.....	134
Figura 108	USB-BLASTER da Altera. ....	135
Figura 109	Aspecto genérico do software Quartus II. ....	135
Figura 110	Socket criado para a FPGA. ....	141
Figura 111	FPGA soldada no socket. ....	141
Figura 112	Vista da face de cima (TOP) da placa do protótipo.....	142
Figura 113	Vista da face de baixo (BOTTOM) da placa do protótipo. ....	143

## *Índice de Tabelas*

Tabela 1	Calendarização da tese .....	9
Tabela 2	Quatro modos de configuração originais. [3] .....	61
Tabela 3	Tabela de configuração dos pinos MSEL da FPGA. [15] .....	113
Tabela 4	Lista de componentes da placa inferior. ....	127
Tabela 5	Lista de componentes da placa superior do sistema. ....	132
Tabela 6	Custo de aquisição dos componentes do sistema. [19] [20] .....	137
Tabela 7	Custo de aquisição de componentes para produção da placa base do sistema. [19] [20]	139
Tabela 8	Lista de encargos .....	143







## *Acrónimos*

PLD	–	<i>Programmable Logic Device</i>
PROM	–	<i>Programmable Read Only Memory</i>
CPLD	–	<i>Complex Programmable Logic Device</i>
SPLD	–	<i>Simple Programmable Logic Device</i>
PLA	–	<i>Programmable Logic Array</i>
PAL	–	<i>Programmable Array Logic</i>
GAL	–	<i>Generic Array Logic</i>
EEPROM	–	<i>Electrically-Eraseable Programmable Read Only Memory</i>
CMOS	–	<i>Complementary Metal-Oxide Semiconductor</i>
PIM	–	<i>Programmable Interconnect Matrix</i>
FPGA	–	<i>Fied-Programmable Gate Array</i>
ASIC	–	<i>Application-Specific Integrated Circuit</i>
ISP	–	<i>In-System Programmable</i>
LUT	–	<i>Lookup Table</i>
SRAM	–	<i>Static Ramdom Access Memory</i>
RAM	–	<i>Ramdom Access Memory</i>
DRAM	–	<i>Dynamic Ramdom Access Memory</i>
IP	–	<i>Intellectual Property</i>



JTAG	– <i>Joint Test Action Group</i>
MOS	– <i>Metal-Oxide Semiconductor</i>
EPROM	– <i>Eraseable Programmable Read Only Memory</i>
FFT	– <i>Fast Fourier Transform</i>
CLB	– <i>Configurable Logic Block</i>
LAB	– <i>Logic Array Block</i>
LC	– <i>Logic Cell</i>
LE	– <i>Logic Element</i>
DSP	– <i>Digital Signal Processing</i>
FIFO	– <i>First-In First-Out</i>
MAC	– <i>Multiply-And-Accumulate</i>
PCB	– <i>Print Circuit Board</i>
MCM	– <i>Multichip Module</i>
PLL	– <i>Phase-Lock Loop</i>
DLL	– <i>Delay-Lock Loop</i>
BST	– <i>Boundary Scan Test</i>
EDA	– <i>Electronic Automation Design</i>
HDL	– <i>Hardware Description Language</i>
RTL	– <i>Register Transfer Level</i>
PLI	– <i>Programmable Language Interface</i>

API	– <i>Application Programming Software</i>
SDF	– <i>Standard Delay Format</i>
OVI	– <i>Open Verilog International</i>
IEEE	– <i>Institute of Electrical and Electronic Engineers</i>
VHDL	– <i>Very High Speed Integrated Circuit HDL</i>
VITAL	– <i>VDHL Initiative Toward ASIC Librares</i>
VHSIC	– <i>Very High Speed Integrated Circuit</i>
SDRAM	– <i>Synchronous Dynamic Ramdom Access Memory</i>
HSMC	– <i>High SpeedMezzanine Card</i>
USB	– <i>Universal Serial Bus</i>
LED	– <i>Ligth-Emitting Diode</i>
ECC	– <i>Error Correction Code</i>
SMA	– <i>SubMiniature type A</i>
I <sup>2</sup> C	– <i>Inter-Integrated Circuit</i>
ADC	– <i>Analog-to-Digital Converter</i>
SD	– <i>Secure Digital</i>
VGA	– <i>Video Graphics Array</i>
DAC	– <i>Digital-to-Analog Converter</i>
IrDA	– <i>Infrared Data Association</i>
DIP	– <i>Dual In-line Package</i>

UART	– <i>Univesal Asynchronous Reiceiver/Transmitter</i>
OLED	– <i>Organic Ligth-Emitting Diode</i>
SPI	– <i>Serial Peripheral Interface</i>
PMOD	– <i>Peripheral Module</i>
PWM	– <i>Pulse-Width Modulation</i>
EQFP	– <i>Enhanced Quad Flat Pack</i>
RGB	– <i>Red-Green-Blue</i>
SOIC	– <i>Small-Outline Integrated Circuit</i>
AS	– <i>Active Serial</i>
PIC	– <i>Programmable Interface Controller</i>
TQFP	– <i>Thin Quad Flat Pack</i>
MSSP	– <i>Master Synchronous Serial Port</i>
LVC MOS	– <i>Low Voltage Complementary Metal-Oxide Semiconductor</i>
LV TTL	– <i>Low Voltage Transistor-Transistor Logic</i>
EAGLE	– <i>Easily Applicable Graphical Layout Editor</i>

# 1. INTRODUÇÃO

O uso de dispositivos lógicos programáveis, nomeadamente de *Field-Programmable Gate Array* (FPGA), tem sofrido uma expansão considerável nos últimos anos, devido, em parte, a um aumento da sua densidade e complexidade destes dispositivos.

## 1.1. CONTEXTUALIZAÇÃO

Com o intuito de abrir horizontes e explorar novas ferramentas de trabalho surgiu o interesse de estudar os dispositivos programáveis designados por FPGA. As vantagens oferecidas por este tipo de chip são essencialmente a rapidez de execução de tarefas, e a possibilidade de poder ser reconfigurada no circuito onde se encontra inserida.

Após um estudo mais aprofundado destes chips e das suas aplicações conseguiu-se vislumbrar suas as enormes potencialidades, que permitiram concluir que são uma ferramenta muito versátil, que no final acabaram por ser o impulsionador e principal factor motivacional para a execução desta dissertação.

A finalidade desta tese é a concepção de uma ferramenta didáctica (*board*) baseada numa FPGA, que permita aos alunos do ISEP familiarizarem-se com estes dispositivos, e criar projectos e aplicações avançadas.

## 1.2. OBJECTIVOS

Na concepção desta plataforma didáctica existem muitos aspectos a ter em conta para se atingir o produto final desejado. É necessário definir o tipo de utilizador final, bem como as funcionalidades pretendidas e o custo de fabrico.

Este trabalho tem com objectivos:

- Criar uma *board* que permita implementar aplicações mais complexas;
- O custo de concepção tem ser reduzido;
- Validar o modelo criado;

Do ponto de vista do utilizador final a *board* deve:

- Ser de fácil utilização;
- Ser versátil;
- Ser robusta;

No entanto estes objectivos funcionam como uma linha guia, já que durante a fase de projecto poderão surgir novas especificações e/ou limitações que poderão levar à alteração das especificações acima referidas.

### 1.3. CALENDARIZAÇÃO

Etapa	Meses							
	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro	Novembro
Seleccção do tema da tese								
Estudo teórico								
Estado da arte								
Definição de especificações para o projecto								
Seleccção de componentes								
Concepção do circuito eléctrico do sistema								
Concepção da placa de circuito impresso								
Escrita do Relatório								

**Tabela 1** Calendarização da tese

### 1.4. ESTRUTURA DO RELATÓRIO

No primeiro capítulo deste relatório estão definidos os objectivos deste projecto, e esta representada a calendarização das várias fases do mesmo.

O segundo capítulo consiste numa breve explanação da história dos dispositivos lógicos programáveis, que tem como objectivo elucidar o leitor relativamente à razão que levou a que fossem criados estes dispositivos, sem deixar de referir as suas funcionalidades e aplicações.

Neste segundo capítulo está também presente um estudo da arte, porque como é lógico, é necessário saber o que já existe no mercado, para não criar um dispositivo desactualizado e obsoleto.

No terceiro capítulo são definidas ao pormenor as especificações pretendidas para sistema didáctico a conceber. É neste capítulo que também é definida a arquitectura pretendida para o sistema, descrevendo de uma forma geral as funcionalidades de cada bloco do sistema.

O quarto capítulo começa com uma descrição dos componentes escolhidos para implementar a arquitectura pretendida para a ferramenta didáctica, seguido da concepção do esquema eléctrico e da placa de circuito impresso, terminando com uma análise de custos de fabrico.

No quinto capítulo deste trabalho é feita a validação do modelo criado, através do fabrico de um protótipo do sistema didáctico.

O sexto capítulo contém as conclusões que foram retiradas da realização do trabalho, bem como algumas perspectivas para trabalhos futuros.

No sétimo capítulo estão listadas as referências bibliográficas que serviram de base para a concepção deste trabalho.





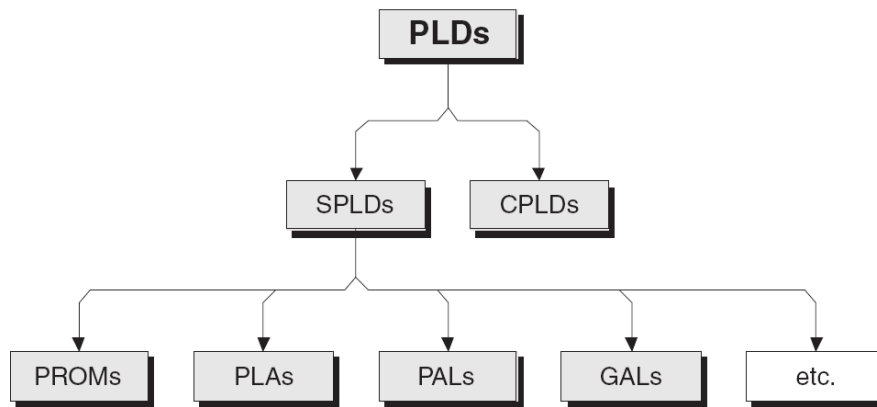


## 2. ESTADO DA ARTE

Os primeiros transístores eram fornecidos como componentes discretos, que eram encapsulados individualmente em pequenas latas de metal. Com o passar do tempo alguém achou que seria boa ideia fabricar circuitos inteiros numa porção de material semi-condutor. Depois da concepção do primeiro circuito integrado foi uma questão de tempo até surgissem os primeiros circuitos integrados programáveis, que eram designados de forma genérica de *Programmable Logic Devices* (PLD). Os primeiros componentes desta família começaram por aparecer em 1970, sob a forma de *Programmable Read Only Memory* (PROM), que eram dispositivos relativamente simples. Só mais tarde, no fim da década de 70 é que surgiram dispositivos mais complexos que acabaram por obter a designação de *Complex Programmable Logic Devices* (CPLD) para se distinguirem dos mais simples, agora referenciados como *Simple Programmable Logic Devices* (SPLD).

A figura 1 permite observar de forma mais clara como o enquadramento destas categorias no universo dos PLD, sem esquecer de agrupar os vários dispositivos nas respectivas famílias.

[1]



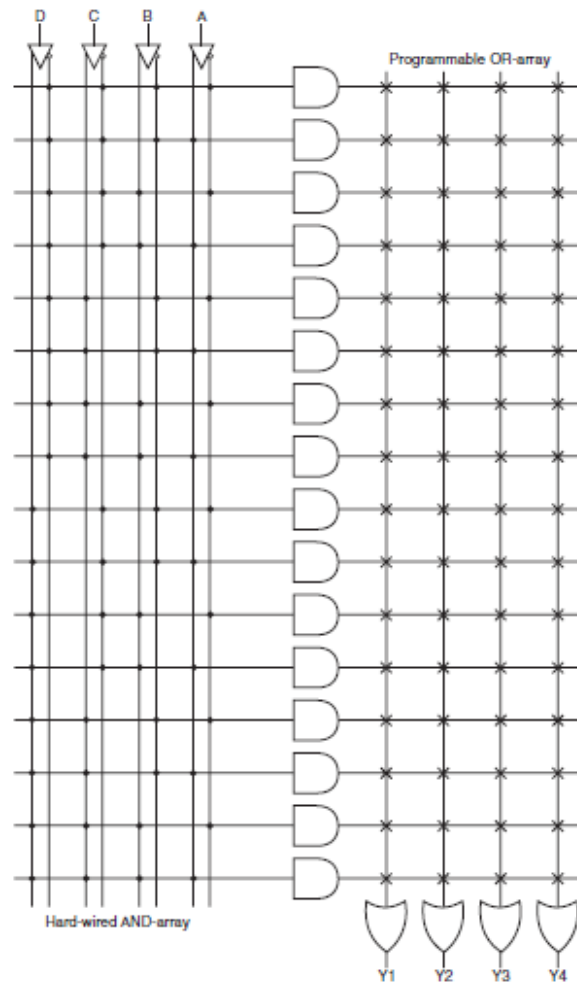
**Figura 1 Famílias de dispositivos PLD. [1]**

## **2.1. PROGRAMMABLE READ ONLY MEMORY (PROM)**

As PROM's podem ser consideradas as predecessoras dos PLD, já que inicialmente não se encontravam projectadas para esse fim. No entanto A arquitectura de uma PROM permite ao utilizador efectuar uma implementação em hardware de funções de lógica combinatória, de um dado número de entradas.

Quando usada como um dispositivo de memória, as  $n$  entradas da ROM (chamadas linhas de endereço) e  $m$  saídas (chamadas linhas de dados), podem ser usadas para guardar palavras de  $2^n m$  bits. Quando usada como um PLD, ela pode implementar  $m$  diferentes funções combinatórias, sendo cada função a combinação das  $n$  variáveis de entrada. Qualquer função concebível de  $n$  variáveis booleanas poder ser colocada em qualquer das  $m$  saídas. Um dispositivo ROM com  $n$  entradas e  $m$  saídas tem  $2^n$  portas AND *hard-wired*, ou seja com ligações já definidas, no andar de entrada e  $m$  portas OR programáveis na saída. Cada porta AND tem  $n$  entradas e cada porta OR tem  $2^n$  entradas, e desta forma cada porta OR pode ser usada para gerar qualquer função concebível de  $n$  variáveis booleanas. A matriz de portas AND produz todos os termos mínimos, para um dado número de variáveis de entrada, sendo depois as portas OR programáveis as responsáveis por permitir que apenas os termos mínimos desejados apareçam nas suas saídas. Na figura 2 está representada a arquitectura interna de

uma PROM de 4 entradas, uma matriz de 16 portas AND *hard-wired* e com uma matriz de 4 portas OR programáveis. [1] [2]



**Figura 2** Arquitectura de uma PROM de 4 entradas. [2]

Na figura, uma cruz (X) indica uma ligação fusível intacta (não programada) e um ponto (•) indica a conexão *hard-wired*. A programação destes dispositivos é feita através da injeção de pulsos de tensão elevados, de forma a fundir as ligações fusíveis desejadas, para implementar as funções pretendidas.

Uma das maiores desvantagens das PROM's é a sua utilização ineficaz da capacidade lógica. Não é uma solução económica usar estes dispositivos numa aplicação onde existem apenas

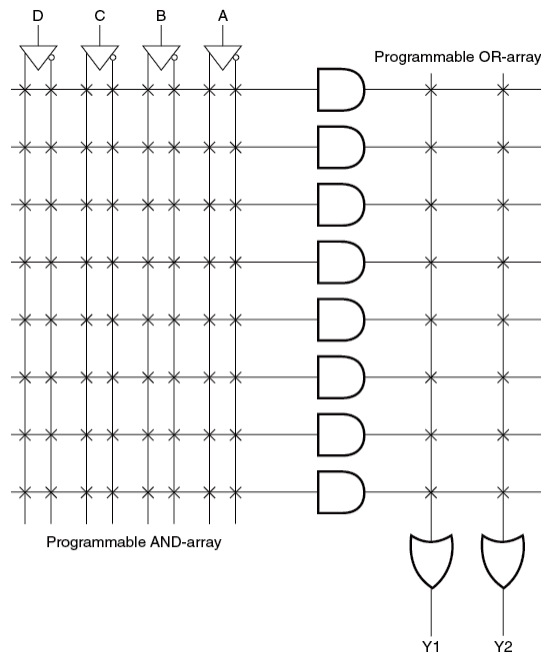
alguns termos mínimos. Outras desvantagens da sua utilização são o alto consumo de energia e uma cobertura segura das transições de lógica assíncrona, já que as saídas podem não apresentar nada, quando as entradas variam. Por norma são muito mais lentas que os circuitos dedicados para a implementação de lógica, já para não falar que não conseguem implementar lógica sequencial devido a não possuírem internamente *flip-flops*.

## **2.2. PROGRAMMABLE LOGIC ARRAY (PLA)**

Em resposta às limitações da arquitectura da PROM dispositivo, o próximo passo na evolução dos PLD foi a concepção da *Programmable Logic Array* (PLA). Estes dispositivos são os SPLD mais versáteis, já que permitem mais configurações por parte do utilizador, isto porque ambas as matrizes AND e OR são programáveis. Ao contrário da PROM, o número de funções AND na matriz de AND's é independente do número de entradas do dispositivo. Se pretendido, podem-se criar portas AND adicionais por uma simples introdução de mais linhas na matriz. De forma idêntica, o número de funções OR na matriz de portas OR é independentes quer do número de entradas do dispositivo, quer do número de funções na matriz. Se por ventura existir a necessidade, podem ser adicionadas mais portas OR, simplesmente por acréscimo de mais colunas à matriz.

Na figura 3 esta representada a arquitectura interna de uma PLA com 4 linhas de entrada, uma matriz de 8 portas AND no andar de entrada, e uma matriz de 2 portas OR no andar de saída.

[1] [2]



**Figura 3** Arquitectura interna da PLA. [2]

Apesar da sua versatilidade, as PLA nunca atingiram um nível significativo de cota de mercado. Apesar disso alguns fabricantes experimentaram diferentes arquitecturas destes dispositivos durante algum tempo, já que nestes dispositivos não existe a obrigatoriedade de existir uma matriz de AND's a alimentar uma matriz de OR's. Uma das arquitecturas alternativas que teve bastante utilização foi a de ter uma matriz de AND's a alimentar uma matriz de NOR, embora teoricamente pudessem ser criadas arquitecturas OR-AND, NAND-OR ou NAND-NOR. Uma das razões para que as PLA tenderam a possuir arquitecturas AND-OR e AND-NOR, deve-se ao facto tornava mais fácil o mapeamento das funções lógicas para a sua estrutura, já que por norma era utilizada a representação de soma de produtos para representar as funções lógicas.

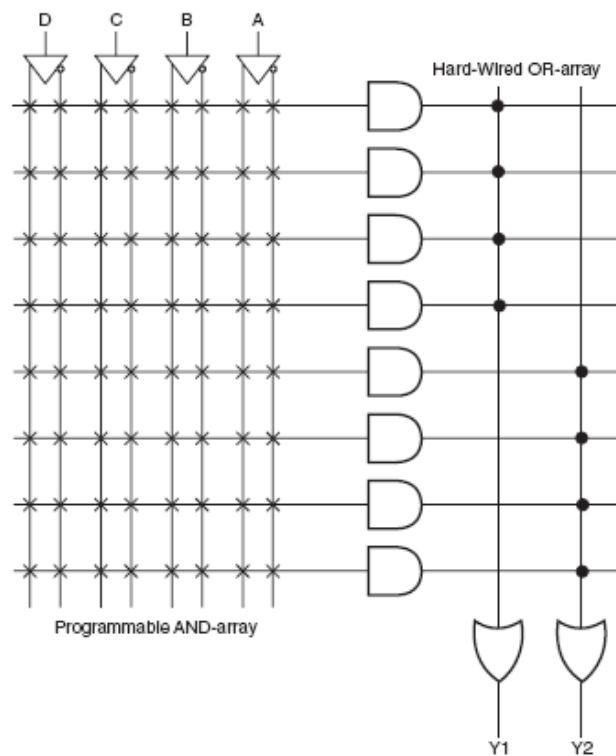
Apesar de uma PLA fazer um uso melhor da sua capacidade lógica, esta possui algumas contrapartidas. Uma das desvantagens é que os sinais demoram relativamente mais tempo a atravessar ligações programáveis, ao contrário das homólogas predefinidas, o que significa que as PLA são significativamente mais lentas do que as PROM. [1] [2]

### 2.3. *PROGRAMMABLE ARRAY LOGIC (PAL)*

Com o intuito de resolver os problemas de velocidade da PLA, foi introduzido um novo tipo de dispositivo chamado de uma *Programmable Array Logic* (PAL). Em termos de conceito, é o exacto oposto de uma PROM, já que ao contrário desta possui uma matriz programável de portas AND na entrada, e uma matriz fixa (não programável) de portas OR na saída.

A vantagem da PAL em relação à PLA é que é mais rápida, já que apenas uma das matrizes é programável. Por outro lado, tem a desvantagem de ser mais limitada porque apenas permite implementar um número restrito de somas de produtos.

A figura 4 ilustra a arquitectura interna de uma PAL de 4 linhas de entrada, uma matriz de 8 portas AND na entrada, e uma matriz de 2 portas OR na saída. [1] [2]



**Figura 4** Arquitectura interna da PAL. [2]

## 2.4. *GENERIC ARRAY LOGIC (GAL)*

A *Generic Array Logic (GAL)* é a evolução das PAL. Este dispositivo difere da PAL no facto de a matriz de AND's da entrada, que pode ser programada. Como se baseiam na tecnologia *Electrically-Erasable Programmable Read Only Memory (EEPROM)*, a GAL podem ser reprogramada, coisa que até agora não era possível fazer, já que os SPLD existentes se baseavam essencialmente na tecnologia de ligações fusíveis. Uma vez efectuada a fusão destas a ligações não havia como voltar atrás, e qualquer erro de programação implicava descartar o dispositivo. Esta característica torna a GAL muito atractiva para efeitos de prototipagem, já que quaisquer erros que ocorram sob funcionamento podem ser eliminados através de nova programação. De notar ainda que uma GAL pode substituir varias PAL. Na figura 5 está representado o diagrama de blocos de uma GAL22V10 (22 entradas e 10 saídas)

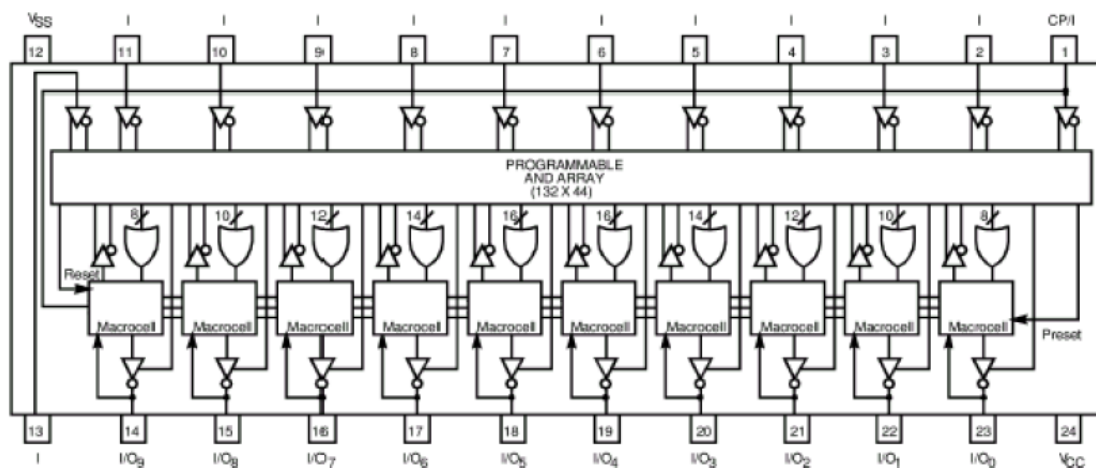


Figura 5 Diagrama de blocos da GAL22V10. [2]

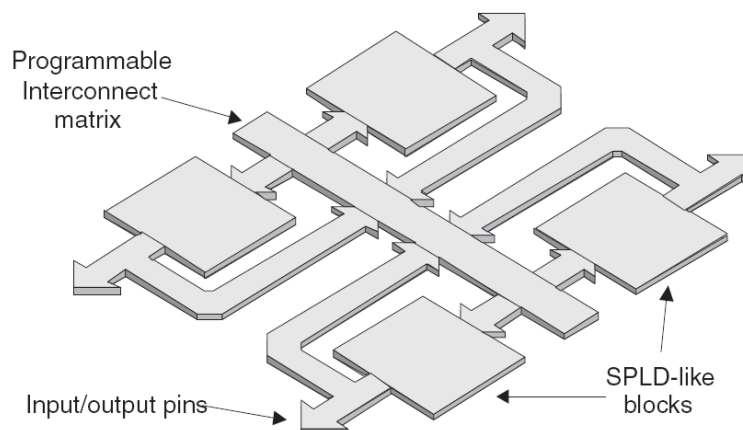


## 2.5. COMPLEX PROGRAMMABLE LOGIC DEVICE (CPLD)

No final da década de 70 começaram a surgir dispositivos mais complexos, que acabaram por obter a designação de CPLD. Mas o grande avanço tecnológico deu-se em 1984, quando a Altera introduziu no mercado um CPLD que combinava a tecnologia *Complementary Metal-Oxide Semiconductor* (CMOS) com a tecnologia EEPROM. O uso da tecnologia CMOS permitiu obter uma tremenda densidade e complexidade, com um consumo relativamente pequeno de potência.

Embora cada fabricante opte pela sua própria arquitectura, um CPLD consiste, de uma forma genérica, num número de blocos SPLD, por norma PAL's, que partilham entre si uma matriz de ligações programável.

A figura 5 fornece uma representação genérica do que é a arquitectura de um CPLD.



**Figura 6** Arquitectura de um CPLD. [1]

Cada bloco SPLD possui ligações que podem ser programadas. A matriz de ligações pode não assegurar 100% de conexões, ou seja, algumas conexões entre as saídas dos blocos de lógica, podem não ser suportadas por um dado CPLD.

Enquanto a complexidade de um dispositivo do tipo PAL possa rondar algumas centenas de portas lógicas, uma CPLD pode atingir uma complexidade de dezenas de milhares de portas lógicas.

### **2.5.1. MATRIZ DE INTERLIGAÇÕES DE UM CPLD**

A matriz de interligações programáveis *Programmable Interconnect Matrix* (PIM) permite unir os pinos de entrada / saída às entradas do bloco lógico, ou as saídas do bloco lógico às entradas de outro bloco lógico ou até mesmo para as entradas do mesmo bloco. A maioria dos CPLD's usa uma de duas configurações para a matriz:

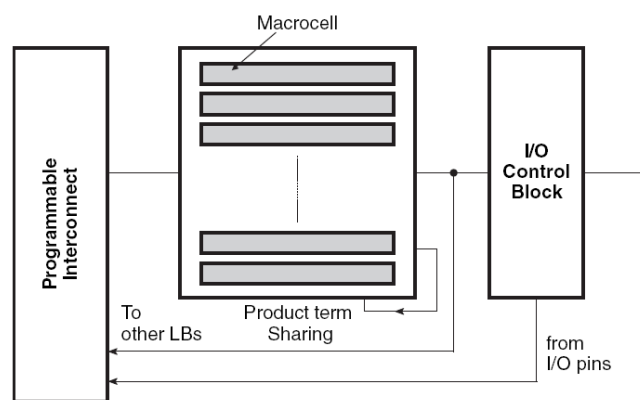
- Interligação mediante matriz.
- Interligação via *multiplexers*.

A primeira é baseada numa matriz de linhas e colunas com uma conexão de células programáveis em cada intersecção. Como no GAL esta célula pode ser activada para ligar/desligar a linha correspondente e coluna. Esta configuração permite plena interconexão entre as entradas e saídas do dispositivo ou blocos lógicos. No entanto, estas vantagens levam a reduções no desempenho do dispositivo enquanto aumenta o consumo de energia e tamanho do componente.

Na interligação através de *multiplexers*, existe um multiplexador por cada entrada para o bloco de lógica. As vias de interligação programáveis são conectadas às entradas de um número fixo de *multiplexers* cada bloco lógico. As linhas de selecção destes *multiplexers* são programadas para permitir apenas uma via da matriz de interligação por cada *multiplexer*. Vale ressaltar que nem todas as vias são conectadas às entradas de cada *multiplexer*. A capacidade para efectuar mais ligações aumenta usando *multiplexers* maiores, permitindo que qualquer combinação de sinais a partir da matriz de interligação, possa ser ligado a qualquer bloco lógico. No entanto, o uso de grandes *multiplexers* aumenta o tamanho do dispositivo e reduz o seu desempenho.

### 2.5.2. BLOCOS DE LÓGICA DE UM CPLD

Um bloco lógico é semelhante a um SPLD, cada um tem uma série de portas AND e OR na forma de soma de produtos, uma configuração para a distribuição destas somas de produtos, e macrocélulas. O tamanho do bloco lógico é uma medida da capacidade da CPLD, porque daqui depende o tamanho da função booleana que pode ser implementada dentro do bloco. Os blocos lógicos têm geralmente 4 a 20 macrocélulas.



**Figura 7 Estrutura de um bloco de lógica de um CPLD. [2]**

As macrocélulas de um CPLD são semelhantes às de um PLD. Estas também são equipadas com registos, controlo de polaridade, e *buffers* de saída em alta impedância. Normalmente, um CPLD tem macrocélulas de I / O, macrocélulas de entrada, e macrocélulas internas ou ocultas (*buried macrocells*), enquanto um 22V10 só tem macrocélulas de entrada / saída. Uma macrocélula interna é similar a uma macrocélula de entrada/saída, só que esta não pode ser directamente ligada a um pino de saída. A saída de uma macrocélula interna vai directamente para a matriz de interligação programável. Por isso, é possível lidar com equações e armazenar o valor de saída destas internamente usando os registos destas macrocélulas.

A figura 8 ilustra uma arquitectura genérica de uma macrocélula. [2]

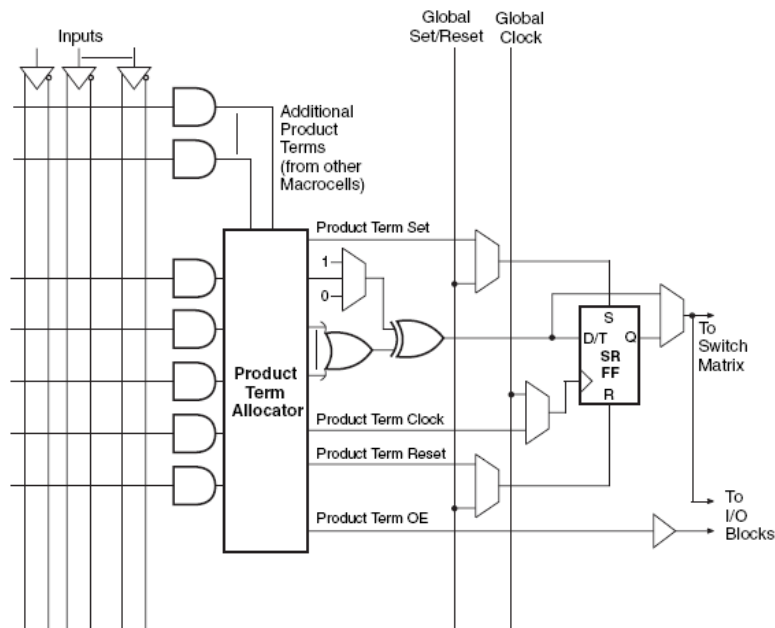


Figura 8 Arquitectura de uma Macro célula. [2]

## 2.6. FIELD-PROGRAMMABLE GATE ARRAY(FPGA)

A primeira FPGA foi criada pela *Xilinx* e tornou-se disponível no mercado em 1984.

Este dispositivo surgiu para preencher a lacuna que existia, até então, no universo dos circuitos integrados digitais. De um lado do espectro encontravam-se os SPLD's e CPLD's, que eram extremamente configuráveis e possuíam tempos de projecto rápidos, mas não conseguiam implementar funções muito longas e complexas. Do outro lado do espectro existiam os *Application-Specific Integrated Circuit* (ASIC). Estes conseguiam suportar funções de extrema complexidade e de superior extensão, mas o seu projecto era excessivamente longo e dispendioso. Além disso, assim que se fabrica o ASIC, o projecto fica estático no silício, ou seja, impossível de ser alterado.

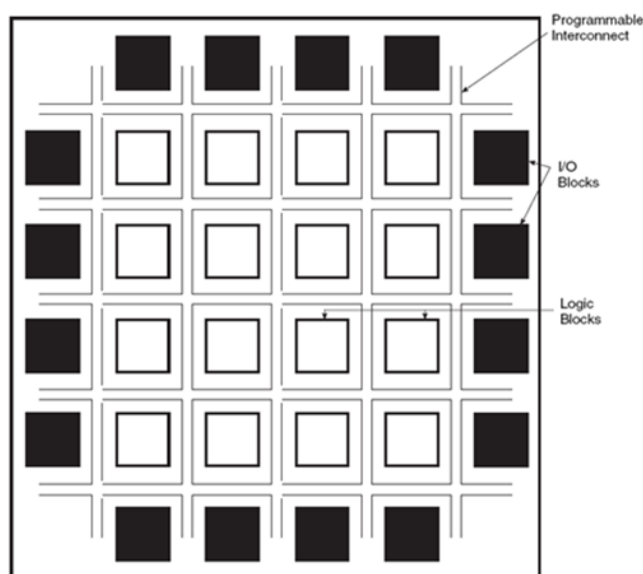
Assim sendo, uma FPGA não é mais que um dispositivo que representa um ponto intermédio entre estes dois extremos, já que concentra em si as características de permitir elevado índice de configuração e de permitir a implementação de funções de enorme complexidade.

A parte do nome “*Field-Programmable*” refere-se ao facto de poder ser programada no campo (“*in the field*”), ou seja, programada ao contrário dos circuitos integrados que saem de fábrica com a sua função definida. Em suma, isto significa que as FPGA podem ser configuradas em laboratório, ou reprogramadas no circuito em que se encontram inseridas, o que faz com que estes dispositivos possam ser frequentemente chamados de *In-System Programmable* (ISP).

Dependendo da tecnologia que usam, as FPGA podem ser programadas apenas uma vez, ou ser reprogramadas vezes sem conta.

A sua arquitectura baseia-se essencialmente por 3 componentes chave, sendo eles os blocos de lógica, os blocos Input/Output (I/O) e ligações programáveis.

Na figura 9 está representada a arquitectura básica de uma FPGA.

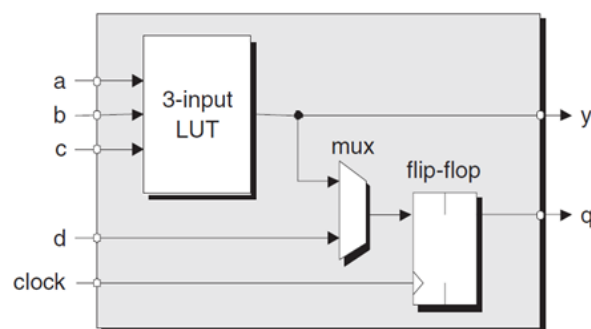


**Figura 9** Arquitectura básica de uma FPGA. [2]

Como se pode observar na figura os blocos de lógica e de I/O encontram-se dispostos em forma de matriz, tendo entre eles a matriz de conexões programáveis.

Como a arquitectura deste dispositivo é essencialmente dominada pelas ligações programáveis, sendo os blocos de lógica relativamente simples, como será demonstrado abaixo neste documento. É através da programação dessas ligações que é efectuada a conexão destes blocos, permitindo criar funções mais complexas, e torna-las disponíveis para exterior pelos pinos de I/O. A forma como é efectuada a programação depende destas ligações depende da tecnologia da FPGA, o que não cria qualquer problema a nível de programação de código, já que o mesmo é transversal a todas as tecnologias e famílias de FPGA.<sup>1</sup>

Na figura 10 ilustra a estrutura básica de um bloco de lógica, que neste caso consiste numa *Lookup Table* (LUT) de 3 entradas, um registo que pode funcionar como *flip-flop* ou *latch* e um *multiplexer*. [1] [2] [3]



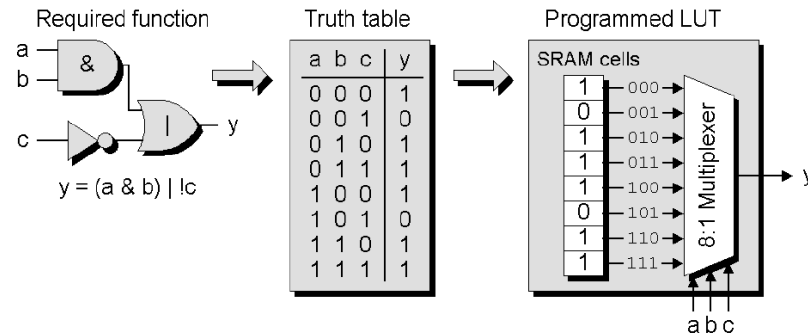
**Figura 10** Estrutura base de bloco de lógica programável. [1]

Uma LUT não é nada mais do que uma pequena memória de um de bits, com as suas linhas de endereçamento a representar as linhas de entrada do bloco de lógica, e a saída de um bit a representar na sua saída. Uma LUT com n entradas consegue implementar qualquer função lógica de n entradas, através da programação nessa memória, da tabela de verdade da função desejada.

---

<sup>1</sup> Esta transversalidade por norma só ocorre dentro dos dispositivos do mesmo fabricante, já que a linguagem de programação varia de fabricante para fabricante.

Na figura 11 ilustra o princípio de funcionamento de uma LUT, através da implementação de uma tabela de verdade da função criada pela associação das portas lógicas.



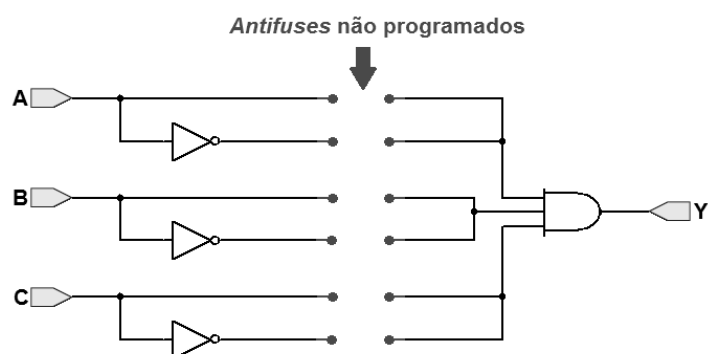
**Figura 11 Configuração de uma LUT. [1]**

### 2.6.1. TECNOLOGIAS DE PROGRAMAÇÃO

A forma como são criadas as ligações da matriz de ligações da FPGA, depende da tecnologia que esta usa. Cada tecnologia tem características próprias, que a tornam específica para determinada aplicação.

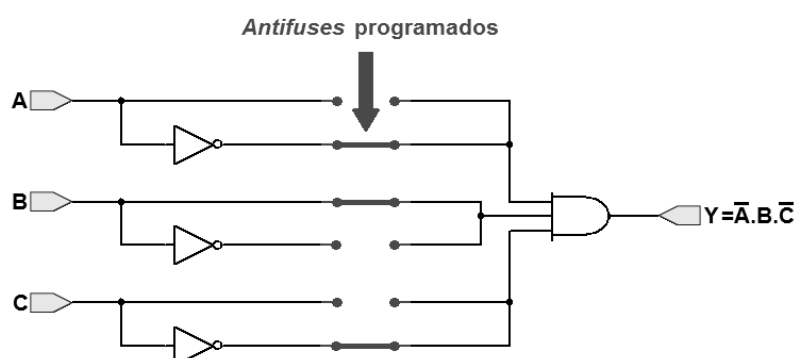
#### ➤ **ANTIFUSE**

Em alternativa a tecnologia de ligações fusíveis, existe o seu oposto, o *antifuse*, onde cada caminho configurável tem associada uma ligação chamada de *antifuse*. Enquanto não se encontra programado, um *antifuse* tem uma resistência tão elevada que pode ser considerado como sendo um circuito aberto, como se pode observar na figura 12.



**Figura 12** Ligações *antifuse* não programadas.

Esta é a forma em que o dispositivo se encontra quando sai de fábrica, no entanto pode-se programar de forma selectiva estes *antifuse* através da aplicação de pulsos de tensão e corrente elevados, nas entradas do dispositivo, criando assim ligações, como ilustrado na figura 13.

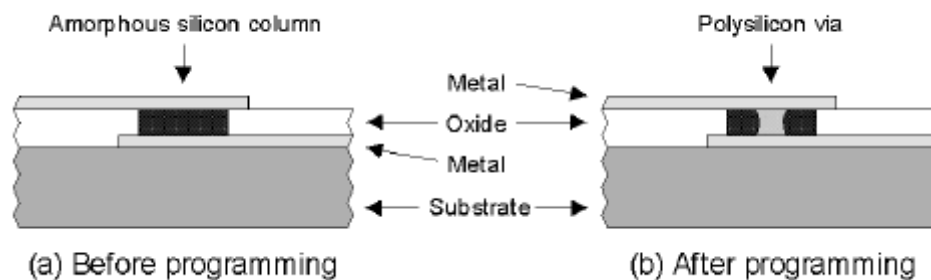


**Figura 13** Implementação de uma função lógica através da programação dos *antifuse*.

Um *antifuse* começa por ser uma coluna microscópica de silício amorfo (não cristalino) que se encontra localizada entre duas pistas metálicas. Enquanto não se encontra programado, o silício amorfo funciona como isolador, com uma resistência elevada, que excede um bilião de ohms.

Na figura 14 estão representados os estados possíveis de um *antifuse*.



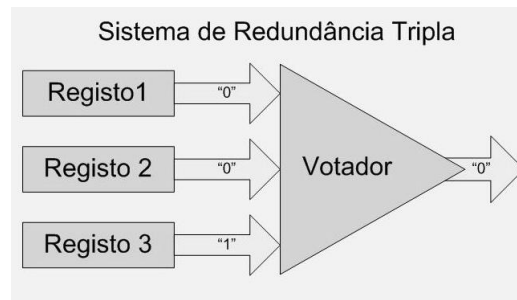


**Figura 14 Estados possíveis de *antifuse*. a) *antifuse* no estado não programado; b) *antifuse* após programação. [1]**

O acto de programar este elemento cria na realidade uma ligação, que é chamada de via. Isto é conseguido convertendo o silício amorfo em silício policristalino (figura 11 (b)). Como a criação dessa via tem um carácter definitivo, a sua programação é irreversível, por isso os dispositivos que usam esta tecnologia são dispositivos OTP.

Ao contrário das FPGA que usam SRAM, que são programados no sistema onde se encontram, os dispositivos baseados na tecnologia *antifuse* são programados *off-line*, através do uso de um programador especial. Apesar deste facto e de serem dispositivos OTP, estes oferecem uma serie de vantagens. Em primeiro lugar não são dispositivos voláteis, já que a sua configuração permanece intacta, mesmo que removida a alimentação do sistema, o que significa que a FPGA se encontra imediatamente disponível, assim que é restituída a alimentação do circuito. Ainda ligado ao aspecto da não volatilidade, estes dispositivos não requerem o uso de uma memória externa de configuração, para guardar a sua configuração, o que representa uma poupança em termos de custo de componentes, e de espaço na *board* do sistema. Outra vantagem meritória de ser referida é que, devido à sua estrutura interna, estes dispositivos são relativamente imunes aos efeitos de radiação. Isto tem particular interesse na indústria militar e aeroespacial, já que o estado de uma célula SRAM pode ser alterado, se a célula se encontrar sujeita a radiação. Apesar de as ligações em si serem imunes a radiação, é necessário ter em conta que quaisquer *flip-flops* presentes nestes dispositivos são sensíveis à radiação, por isso, para *chips* que vão operar em ambientes de radiação intensiva, é requerido que os *flip-flops* sejam protegidos através de um sistema de redundância tripla. Este sistema baseia-se essencialmente na existência de três cópias de cada registo, onde o resultado final é

dado através da “votação”. Idealmente os três registos deveriam ter valores idênticos, mas se um deles altera o seu estado, o valor final da saída vai ser ditado pelo valor que estiver em maioria nos registos. A figura 15 ilustra de forma mais clara este tipo de filosofia.



**Figura 15 Exemplo de um sistema de redundância tripla.**

Neste exemplo os dois primeiros registos têm na sua saída o valor lógico “0” e o terceiro apresenta uma alteração do seu estado tendo na sua saída o valor lógico “1”. Após a “votação” efectuada por cada registo, o “votador” compara os valores concluindo que o valor que se encontra em maioria o valor lógico “0”, que vai ser colocado na saída deste sistema.

A maior vantagem das FPGA’s baseadas na tecnologia *antifuse* será talvez o facto de a sua configuração se encontrar enterrada dentro deles. Por defeito é possível que o programador efectue a leitura de dados existentes dentro da FPGA, visto que é assim que ele funciona. À medida que cada *antifuse* é processado, o programador continua a testar quando aquele elemento foi programado, e de seguida passa ao próximo *antifuse*. Além disso, o programador pode ainda ser usado para testar se a configuração foi correctamente implementada. De forma a executar esta tarefa é exigido ao programador que esteja habilitado a ler os estados dos *antifuses* e compara-los com os estados que se encontram definidos no ficheiro de configuração. Uma vez programado o dispositivo, existe ainda a possibilidade de programar um grupo especial de *antifuses* que previne que seja feita leitura da configuração da FPGA. Isto permite que mesmo que seja removido o topo do dispositivo de forma a ter acesso ao seu interior, que não seja possível fazer distinção entre os *antifuses* programados e não programados, aliado ao facto de que estes se encontram enterrados nas camadas internas de

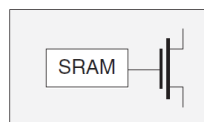
metalização<sup>2</sup>, torna impossível que seja efectuada engenharia revertida do projecto implementado. [1] [2] [3]

➤ **STATIC RANDOM ACCESS MEMORY (SRAM)**

Existem essencialmente dois tipos de dispositivos semicondutores de memória *Random Access Memory* (RAM), sendo eles a *Dynamic Random Access Memory* (DRAM) e a *Static Random Access Memory* (SRAM).

No caso da DRAM cada célula de memória é formada por um par transístor - condensador, o que consome pouca área de silício. O qualificativo “*Dynamic*” é usado porque ao longo do tempo o condensador perde carga, o que implica que cada célula tenha que ser carregada periodicamente se se pretender reter os dados nela armazenados. Esta operação, que é conhecida como “*refreshing*”, é um pouco complexa e requer numa quantidade substancial de circuitos adicionais. Quando o custo desses circuitos é amortizado ao longo de dezenas de milhões de bits de memória DRAM, esta tecnologia torna-se rentável. No entanto esta tecnologia tem pouco interesse no que diz respeito a lógica programável.

No caso da SRAM o qualificativo “*Static*” refere-se ao facto de assim que seja carregado um valor na célula SRAM, ele permanecerá inalterável, ate que seja introduzido outro valor na célula ou ate que seja removida a alimentação do sistema. Na figura 16 está representada uma célula programável baseada em SRAM.



**Figura 16 Célula programável baseada em SRAM. [3]**

---

<sup>2</sup> Camadas superiores que ligam os transístores as resistências, por norma estas camadas são de alumínio e são intercaladas por camadas de dióxido de silício.

A célula em si é constituída por um elemento de armazenamento SRAM com múltiplos transístores, sendo que essa célula controla um outro transístor, que dependendo do conteúdo dessa memória vai estar ao corte ou à condução.

Uma desvantagem de ter um dispositivo baseado nesta tecnologia é que cada elemento SRAM é constituído por quatro ou seis transístores, com uma configuração em *latch*, o que faz com que ocupe uma área maior a nível de silício. Outra desvantagem é que todos os dados de configuração são perdidos se se remover a alimentação do sistema, o que significa que estes dispositivos têm que ser reprogramados assim que se volta a restabelecer alimentação. Para que isto não aconteça, e para que não se percam os dados de configuração, pode utilizar-se uma pilha de *backup* para manter a FPGA alimentada, ou então utilizar uma memória externa que retém os dados mesmo que seja removida a alimentação, que depois efectua a programação da FPGA.

Apesar desta desvantagem estes dispositivos podem ser reprogramados sempre que se quiser, as vezes que se quiserem. Esta característica permite que sejam implementadas e testadas rapidamente novas ideias e projectos. Para além disso possibilita que a FPGA seja programada inicialmente para fazer um auto teste ou teste do sistema em que se encontra inserida, e ser programada posteriormente para executar a função pretendida

Existe no entanto outra preocupação no que diz respeito às FPGA baseadas em tecnologia SRAM, é que pode ser difícil proteger a *Intellectual Property* (IP) e projecto do projectista. Isto acontece quando o ficheiro de configuração é guardado em algum tipo de memória externa. Não existem actualmente ferramentas comercializáveis que permitam através da leitura da informação do ficheiro de configuração gerar o esquemático do projecto implementado. No entanto existem numerosas empresas espalhadas pelo mundo que são especialistas em engenharia revertida, que são especializadas em recuperar IP's, já para não falar que existem países em que os governos que ignoram o roubo de IP's, desde que exista dinheiro envolvido. Por isso se um projecto provar ser rentável, certamente que existe alguém ansioso por replica-lo.

Um aspecto positivo é que as FPGA de hoje em dia, baseadas em SRAM, suportam o conceito de "*bitstream encryption*". Neste caso os dados de configuração são encriptados antes de serem armazenados no dispositivo de memória externa. A chave de encriptação é guardada num registo SRAM especial, através da porta *Joint Test Action Group* (JTAG). Em

conjunção com alguma lógica adicional, a chave da encriptação permite que os dados de configuração encriptados sejam descriptados, a medida que vão sendo carregados para a FPGA.

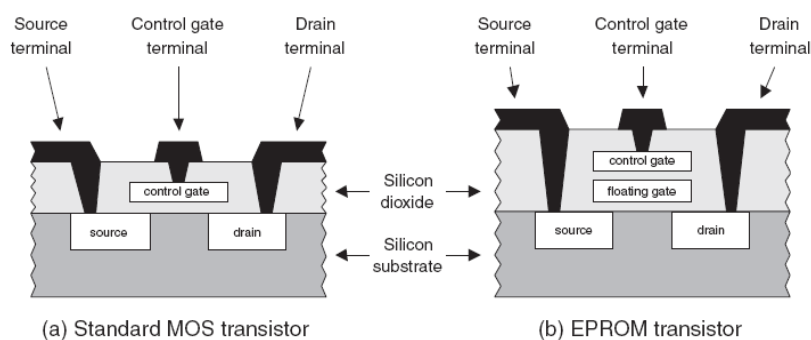
A principal desvantagem deste esquema é que ele requer o uso de uma bateria de *backup* para que a FPGA consiga reter a chave de encriptação mesmo que seja removida a alimentação.

Actualmente esta tecnologia de FPGA é a mais utilizada de todas as tecnologias de programação. [1] [2] [3]

➤ ***ELECRICALLY ERASEABLE PROGRAMMABLE READ ONLY MEMORY (EEPROM)***

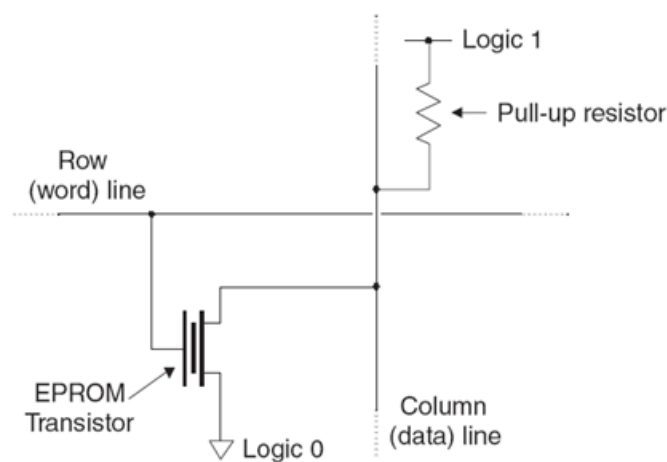
Com o uso de tecnologias como ligações fusíveis e ligações *antifuse*, uma vez programado o dispositivo, não havia como voltar atrás. Por esta razão surgiu a necessidade de se criar uma tecnologia que permitisse apagar os dados de configuração e voltar a programar o dispositivo.

Uma alternativa é a tecnologia *Eraseable Programmable Read Only Memory* (EPROM). Um transístor EPROM tem essencialmente a estrutura de um transístor *Metal Oxide Semionductor* (MOS), mas com a adição de uma *gate* flutuante que se encontra isolada por isolada pelas camadas de dióxido de silício, como se pode observar na figura 17.



**Figura 17 Transístor MOS Standard Versus Transístor EEPROM. a) Transístor MOS Standard. b) Transístor EEPROM. [1]**

No seu estado não programado, a gate flutuante do transistor encontra-se descarregada e por isso não afecta o funcionamento normal da gate de controlo. Para programar o transistor é necessário aplicar uma tensão relativamente elevada (na ordem dos 12V) entre o dreno e a *gate*. Isto faz com que o transistor se encontre em saturação, e que a força energética dos electrões force a sua passagem pelo dióxido de silício ate atingir a gate flutuante, sendo este processo conhecido por “*hot electron injection*”. Quando o sinal de programação é removido, a gate flutuante mantém-se carregada com uma carga negativa. Esta carga é muito estável e não se dissipa em menos de uma década, se o dispositivo funcionar em condições normais. A carga armazenada na gate flutuante proíbe o normal funcionamento da gate de controlo, o que permite distinguir quais as células que já estão programada das que não se encontram programadas. Isto significa que podemos usar este tipo de transistor para construir uma célula de memória.



**Figura 18 Célula de memória baseada em transistor EEPROM. [1]**

Na figura 18, acima apresentada, está representado o esquema eléctrico de uma célula de memória baseada num transistor EPROM. Como se pode constatar a programação já não é efectuada através de ligações fusíveis ou *antifuse*. Neste caso cada linha no estado irá polarizar todos os transístores que estão conectados a ela, colocando todas as colunas no estado lógico “0” através dos transístores que lhe estão associados. Para programar o dispositivo usam-se as gates flutuantes associadas aos transístores desejados,

consequentemente desactivando esses transístores, o que faz com que essas células aparentem conter o valor lógico “1”.

Uma vantagem destas células é facto que serem mais pequenas do que as ligações fusíveis e *antifuse*, o que faz com que ocupem menos espaço a nível de silício.

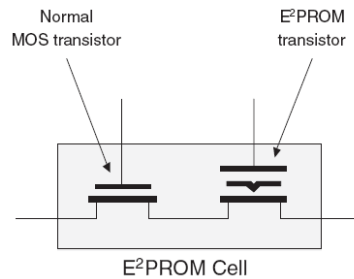
Uma célula EPROM pode ser apagada através da descarga dos electrões na gate flutuante da célula. A energia necessária para descarregar esses electrões é fornecida por radiação ultravioleta. Uma EPROM é encapsulada num encapsulamento de plástico ou cerâmica, com uma janela de quartzo no topo retira-lo do circuito onde se insere, removendo de seguida o autocolante que cobre a já, que por normas esta coberta por um autocolante. Para apagar o dispositivo é necessário nela de quartzo, e por fim coloca-se o dispositivo num contentor com uma fonte intensa de luz ultravioleta.

Os principais problemas das EPROM são essencialmente o custo dos seus encapsulamentos com janela de quartzo, e o tempo que demora a apagar um dispositivo deste tipo, que ronda os 20 minutos.

A evolução da tecnologia permitiu que se conseguisse conceber transístores cada vez mais pequenos. Com as estruturas no dispositivo a tornaram-se cada vez mais pequenas e o aumento da densidade (numero de transístores) grande percentagem da *die* encontra-se coberta por metal. Isto faz com seja mais difícil absorver a luz ultra violeta, aumentando assim o tempo de exposição necessário para apagar o dispositivo.

O próximo salto tecnológico deu-se com o surgir das *Electrically Erasable Programmable Read Only Memory* (EEPROM).

O tamanho de uma célula EEPROM é cerca de 2,5 vezes maior do que uma célula EPROM, isto porque uma célula EEPROM contem dois transístores, como se pode observar na figura 19.



**Figura 19 Célula e memória EEPROM. [1]**

O transistor EEPROM é similar ao transistor EPROM, por isso também possui uma gate flutuante, mas neste caso as camadas de isolamento de dióxido de silício são mais finas. O segundo transistor (o transistor MOS) pode ser utilizado para apagar electricamente a célula EEPROM.

A configuração de dispositivos que usam a tecnologia EEPROM é similar aos seus homólogos SRAM da maneira em que as suas células de configuração também se encontram interligadas por uma cadeia longa de descoladores de registos. Estes podem ser configurados *off-line*, usando um programador específico. Algumas versões destes dispositivos são ISP, mas o tempo de programação de uma FPGA EEPROM é três vezes superior ao tempo de programação de uma FPGA SRAM.

Uma vez programada FPGA, os dados nela contidos são não voláteis, o que permite que mesmo que o circuito fique sem alimentação, a configuração do dispositivo não é perdida, e a FPGA encontra-se logo disponível para executar a sua função assim que a alimentação é restituída.

No que diz respeito à protecção, alguns dispositivos usam o conceito *multibit key*, que no é mais que uma chave cujo comprimento pode ir de 50 a centenas de bits. Após a programação do dispositivo, pode-se carregar a chave definida pelo utilizador, para proteger os dados de configuração. Depois de carregada a chave é única forma de ler dados do dispositivo ou de escrever no dispositivo, é carregando uma cópia da chave através do porto JTAG.

A célula EEPROM, constituída por dois transístores é 2,5 vezes maior do que uma célula EPROM, mas mesmo assim é mais pequena do que uma célula SRAM, o que permite com



que a restante lógica se encontre mais próxima e compacta, diminuindo assim os atrasos nas interligações.

Uma desvantagem destes dispositivos é o facto de serem necessárias cerca de mais 5 etapas no processo de fabrico, quando comparado com o fabrico de tecnologia CMOS Standard, o que faz com que se encontrem atrasados, em termos de evolução tecnológica, uma ou mais gerações, em relação à tecnologia SRAM. [1] [3] [4] [5]

## ➤ **FLASH**

A tecnologia *Flash* é uma variação da tecnologia EEPROM. Originalmente o nome “*Flash*” foi atribuído a esta tecnologia para reflectir o facto de que o tempo de apagar o dispositivo era mais rápido do que o de uma EPROM. Componentes baseados na tecnologia Flash podem assumir varias arquitecturas. Alguns possuem uma célula apenas com um transístor com gate flutuante como utilizado na tecnologia EPROM, mas com camadas de isolamento mais finas, similar ao transístor da tecnologia EEPROM. Estes dispositivos podem ser apagados electricamente, mas apenas se podem apagar grandes partes do dispositivo de cada vez ou então apagar o disposto por completo.

Uma arquitectura alternativa é o uso de uma arquitectura similar à EEPROM, ou seja uma célula com dois transístores, o que permite apagar e reprogramar o dispositivo ao nível da *Word e não do byte*.

Inicialmente cada célula apenas conseguia armazenar apenas um bit, mas com a evolução tecnológica surgiram novas técnicas que permitem a expansão de memória de uma célula.

Uma técnica consiste em guardar vários níveis de carga na gate flutuante do transístor FLASH, de forma a conseguir representar dois bits. Uma outra forma de abordar este problema consiste em crias dois nos de armazenamento discretos por baixo da gate, fazendo com que a célula suporte dois bits.

No que diz respeito as vantagens e desvantagem, estas são essencialmente as mesmas proporcionadas pela tecnologia EEPROM, com a pequena diferença de que o tempo que demora a apagar o dispositivo ser menor.

A programação destes dispositivos é efectuada da mesma forma que a programação dos dispositivos que usam a tecnologia EEPROM. [3] [4] [5] [6]

### 2.6.2. BLOCOS DE I/O

As FPGA de hoje em dia podem ter mais de 1000 pinos, que por norma se encontram dispostos em forma de matriz (*array*), ao longo da base do encapsulamento.

Os blocos I / O são os meios em que os dados são enviados a partir da lógica interna para fontes externas e do qual os dados são recebidos de fontes externas.

Dependendo do que se pretender fazer, do dispositivo que se usa, o ambiente em que a placa de circuito impresso vai encontrar inserida, o projectista vai ter em conta estas características aquando a selecção do tipo de sinais e standards que o sistema vai utilizar.

O grande problema é que existem grandes variedades de standards, o que torna difícil criar uma FPGA's especiais para acomodar cada variação de standard. É por esta razão que cada pino de I/O pode ser configurado para aceitar e gerar sinais para qualquer que seja o standard pretendido. Estes pinos I/O's são agrupadas em bancos de I/O, como se pode observar na figura 20.

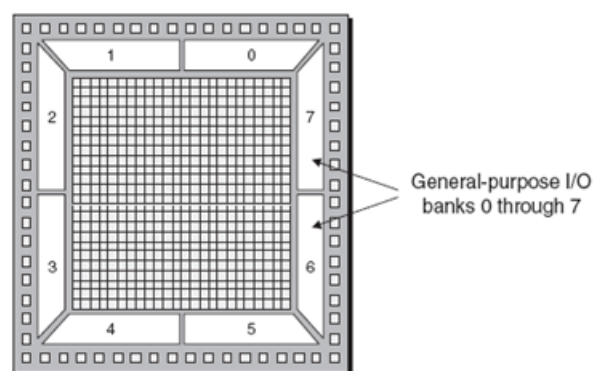


Figura 20 Bancos de *general purpose* I/O de uma FPGA. [3]

Uma característica interessante é que cada banco de I/O pode ser configurado de forma independente, para suportar standards. Para além de permitir que a FPGA comunique com dispositivos que usem standards de I/O diferentes, possibilita que a FPGA seja usada para fazer interface entre diferentes standards de I/O, bem como para fazer a tradução de diferentes protocolos que sejam baseados em standards eléctricos particulares.

Os sinais usados para conectar os dispositivos de hoje em dia, numa placa de circuito impresso, tem tempos de transição de estados muito rápidos. Para prevenir que estes sejam reflectidos para trás, é necessário adicionar resistências aos pinos de I/O da FPGA, contudo esta técnica tornou-se problemática quando o número de pinos aumentou e a distância entre eles diminuiu. É por esta razão que as FPGA de hoje em dia possuem resistências interna, cujo valor pode ser definido pelo utilizador, de forma a acomodar vários standards num pino de I/O. [2] [3]

#### ➤ TENSÃO DO NÚCLEO VERSUS TENSÃO DAS I/O'S

Nos primórdios da lógica combinatória, a tensão usada na alimentação dos integrados que implementavam a lógica era de 5V. Para além disso os seus sinais de I/O variavam entre 0V (zero lógico) e 5V (um lógico), o que tornava relativamente fácil trabalhar com eles.

Com o passar do tempo, a geometria dos chips tornou-se cada vez mais pequena devido ao facto de os transístores se tornarem mais pequenos, o que se traduziu numa diminuição de custos, aumento de velocidade de processamento e menor consumo. Contudo estas mudanças na estrutura dos chips requereram novas tensões de alimentação, sendo que o valor destas tem vindo a diminuir ao longo do tempo.

Um aspecto importante é que esta tensão é utilizada para alimentar a lógica interna da FPGA, sendo isso conhecido por “*core voltage*”. No entanto isto pode tornar-se problemático, devido ao facto de os standards de I/O terem níveis de tensão significativamente diferentes da tensão de alimentação da lógica interna, por esta razão cada banco de I/O possui pinos adicionais de alimentação. [2] [3] [4]

### 2.6.3. ARQUITECTURAS DE FPGA'S

É muito comum classificar os recursos fornecidos pela FPGA como *fine-grained* ou *coarse-grained*. Uma FPGA é essencialmente constituída por blocos programáveis relativamente simples que se entram embebidos numa rede de interconexões programáveis.

No caso da arquitectura *fine-grained*, cada bloco de lógica pode ser usado para implementar funções muito simples. Por exemplo pode ser configurado para implementar uma porta lógica primitiva com 3 entradas, ou um elemento de armazenamento, como um *flip-flop* D.

Para além de serem usadas para implementar *glue logic*<sup>3</sup> e máquinas de estados irregulares, as arquitecturas *fine-grained* são particularmente eficientes na implementação de *systolic algorithms* (funções que beneficiam de implementações com paralelismo massivo). Estas arquitecturas também oferecem algumas vantagens no que diz respeito a tecnologias tradicionais de síntese de lógica, que são orientadas para arquitecturas *fine-grained* de ASIC.

No início dos anos 90 existiu grande interesse nas FPGA com arquitecturas *fine-grained*, mas com o tempo estas foram desaparecendo, deixando apenas as arquitecturas *coarse-grained*.

No que diz respeito a arquitecturas *coarse-grained*, cada bloco de lógica contém uma quantidade relativamente elevada de lógica quando comparado com os homólogos da arquitectura *fine-grained*. Por exemplo os blocos podem conter LUT's de quatro entradas, quatro *multiplexers*, quatro *flip-flops* do tipo D.

Um aspecto importante a ter em conta aquando a implementação de uma arquitectura *fine-grained* é que esta requer um número relativamente elevado de conexões a entrar e sair do bloco, isto quando comparado com as funções que cada bloco pode implementar. À medida que aumenta a granularidade dos blocos (para *medium-grained* ou *coarse-grained*), a quantidade de conexões que interligam os blocos torna-se menor, quando comparado com a quantidade de funcionalidades que pode suportar. Este aspecto torna-se importante, já que o principal responsável por atrasos relacionados com a propagação dos sinais ao longo da FPGA é a matriz de interligações da FPGA.

---

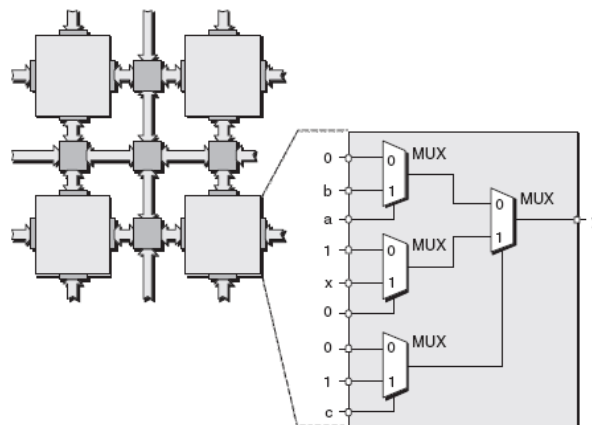
<sup>3</sup>É um circuito lógico configurável que tem como função fazer o *interface* circuitos integrados

Actualmente um grande número de fabricantes começaram a fabricar dispositivos *coarse-grained* que contem blocos de lógica extremamente complexos, onde cada bloco é um elemento de processamento capaz de implementar desde funções aritméticas como *Fast Fourier Transform* (FFT) ate um microprocessador completo. Estes dispositivos não são classificados como FPGA, por esta razão e para não criar confusão, as FPGA's baseadas em LUT são várias vezes classificadas como *médium-grained*, permitindo que o termo *coarse-grained* seja utilizado para classificar estes dispositivos. [2] [3]

➤ **BLOCOS DE LÓGICA BASEADOS EM MULTIPLEXER VERSUS BLOCOS DE LÓGICA BASEADOS EM LOOKUP TABLE**

Relativamente a arquitecturas *médium-grained* existem essencialmente dois tipos de arquitecturas de blocos de lógica, uma baseada em *multiplexers* e outra baseada *lookup tables*.

No caso de uma arquitectura baseada por *multiplexer* o bloco é constituído por uma serie de portas lógicas e *muxes*, o que torna muitas vezes possível ter acesso a valores intermédios de sinais que interligam as portas lógicas e os *muxes*. Na figura 21 está representado um exemplo de uma arquitectura baseada em *muxes*.

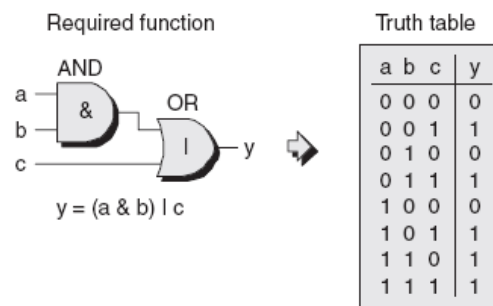


**Figura 21 Bloco de lógica baseado em *multiplexers*. [3]**

Neste caso cada bloco de lógica pode ser fragmentado em pequenos blocos, podendo cada um deles ser usado para implementar uma função simples. Esta característica faz com que esta forneça melhor performance em projectos baseados em pequenas funções lógicas, e utilização de silício. Estas arquitecturas oferecem também vantagens na implementação de lógica de controlo que usem instruções do tipo “*if - then*”. Contudo este tipo bloco de lógica não fornece *carry logic chains* de alta velocidade, ao controlo dos homólogos baseados em LUT’s, que lideram quando a aplicação envolve processamento aritmético.

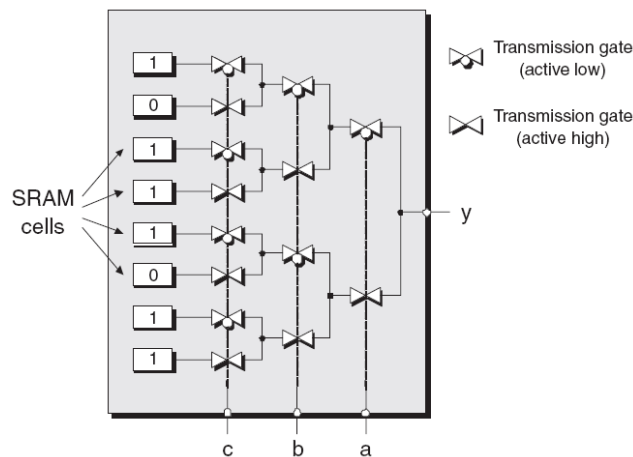
O conceito por detrás da LUT consiste em usar um grupo de sinais de entrada é usado para criar um índice da *lookup table*. O conteúdo da tabela é disposto de tal forma que permite que cada célula indexada por cada combinação das entradas, guarde o valor pretendido.

Na figura 22 esta representada uma função lógica de três entradas e a respectiva tabela de verdade, que vai ser implementada numa LUT.



**Figura 22 Função lógica e respectiva tabela de verdade. [3]**

A implementação desta função pode ser conseguida carregando uma LUT de três entradas com os valores apropriados. Supondo que a LUT é constituída por células SRAM (embora possa ser constituída por células de outra tecnologia). A técnica que é mais comumente usada é utilizar as entradas para seleccionar as células SRAM, através de uma ligação em cascata de *transmission gates*, como se encontra ilustrado na figura 23.

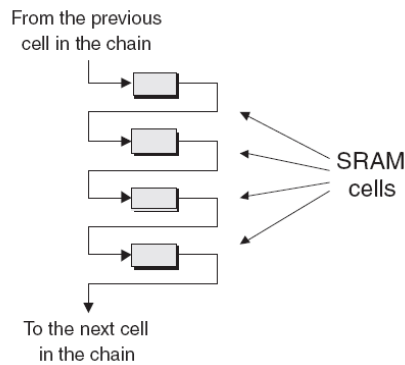


**Figura 23** LUT baseada em *transmission gates*. [3]

Se uma *transmission gate* é activada, ela faz um *bypass* do sinal que se encontra na sua entrada, para a sua saída. Se a *transmission gate* é desactivada, ela é imediatamente desconectada do fio que esta a controlar.

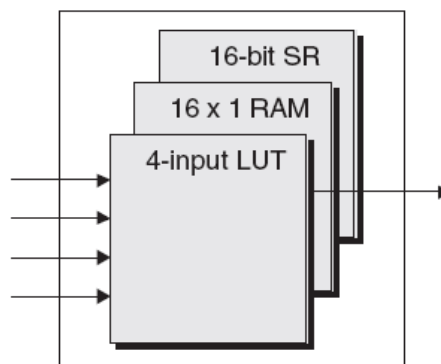
O facto de o núcleo da LUT ser um dispositivo baseado em tecnologia SRAM, que contem uma serie de células SRAM, oferece uma serie de possibilidades interessantes. Para além do papel principal de funcionar como LUT, alguns dispositivos permitem que a *lookup table* seja usada como um pequeno bloco de memória RAM. Este tipo de memória é denominado de memória RAM distribuída, em primeiro lugar porque as LUT se encontram espalhadas pela superfície do chip, e em segundo lugar para haver uma distinção entre estes pequenos aglomerados de memória e os grandes blocos de memória RAM embebida.

Outra possibilidade advém do facto de todas as células de configuração da FPGA, incluindo as das LUT's, se encontrarem interligadas através de uma cadeia (*chain*) longa, como ilustrado na figura 24.



**Figura 24 Células de configuração ligadas em cadeia. [3]**

Alguns dispositivos permitem que as células SRAM que formam a LUT sejam tratadas de forma independente do corpo principal da cadeia, o que permite que funcione como um deslocador de registos (*shift register*). Por estas razões uma LUT pode ser considerada como sendo multifacetada, como ilustrado na figura 25.



**Figura 25 LUT representada como um elemento multifacetado. [3]**

Uma arquitectura baseada numa *lookup table* oferece vantagens para aplicações de circuitos lógicos de bastantes camadas, processamento de funções aritméticas e transmissão de grande quantidade de dados.

No entanto se pretender implementar uma função lógica simples, como por exemplo uma porta AND, vai existir um desperdício de recursos e de termos, porque vamos usar a LUT por



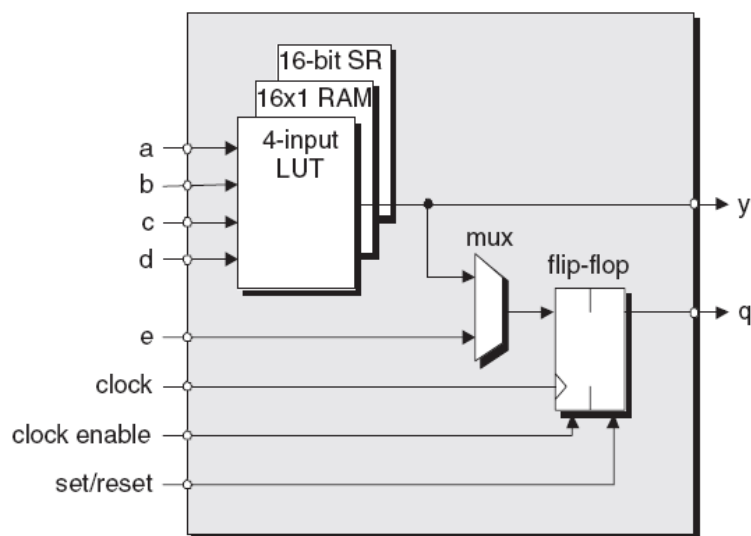
completo, o que cria uma grande probabilidade de existirem atrasos para uma função tão simples.

### ➤ **CONFIGURABLE LOGIC BLOCK (CLB) VERSUS LOGIC ARRAY BLOCK(LAB)**

Apesar de uma LUT poder ser multifacetada, não é o suficiente para criar um bloco de lógica. Por norma um bloco de lógica contém outros elementos como *multiplexers* e registos.

Um problema com as FPGA é que cada fabricante atribui a sua nomenclatura a esta estrutura, o que pode ao olhar inexperiente causar alguma confusão.

O fabricante de FPGA Xilinx denomina o núcleo do bloco de lógica de *Logic Cell* (LC). Entre outras coisas a LC contém uma LUT multifacetada de quatro entradas, um *multiplexer* e um registo.



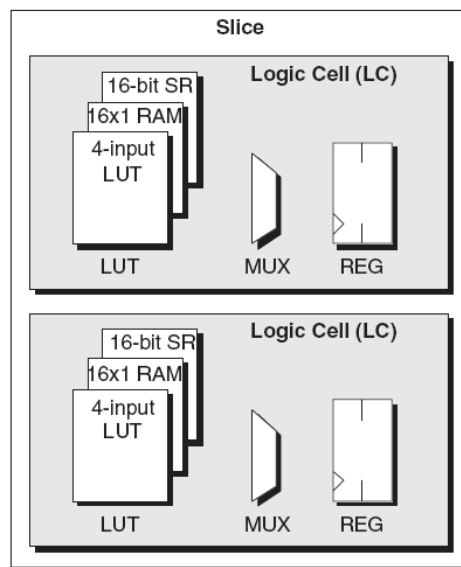
**Figura 26** Vista simplificada de uma LC da Xilinx. [3]

O registo pode ser configurado como *flip-flop* ou como *latch*. A polaridade do *clock* (flanco ascendente ou flanco descendente) pode ser configurada, para activar e desactivar os sinais de *set/reset*.

Para além destes componentes a LC possui ainda outros elementos, incluindo *fast carry logic*, normalmente utilizada em operações aritméticas.

No caso da Altera o núcleo do bloco de lógica equivalente chamado de *Logic Element* (LE). Existem várias diferenças entre dispositivos da Xilinx e da Altera, no entanto os conceitos que ambas aplicam são similares.

O nível acima na hierarquia é o que a Xilinx chama de “*slice*”. Neste caso uma *slice* é constituída por duas LC, como se pode observar na figura 27.



**Figura 27** *Slice* da Xilinx, constituída por duas LC. [3]

As ligações entre as LC não estão representadas na figura com o intuito de manter a simplicidade, mas é necessário ter em conta que apesar cada LUT, *multiplexer* e registo terem as suas próprias entradas e saídas, a *slice* tem apenas um *clock*, *clock enable* e sinais de *set/reset* que são comuns a ambas as LC's.

Subindo mais ainda mais um nível na hierarquia, atinge-se o que a Xilinx chama de *Configurable Logic Block* (CLB), e ao que a Altera se refere como *Logic Array Block* (LAB).

No que diz respeito à constituição do CLB, algumas FPGA's da Xilinx tem CLB's apenas com duas *slices*, enquanto que outras têm CLB's com quatro *slices*. O número de *slices* varia consoante a família da FPGA Comparativamente com a arquitectura básica da FPGA, um CLB igual a um bloco de lógica. A figura 28 ilustra esta analogia.

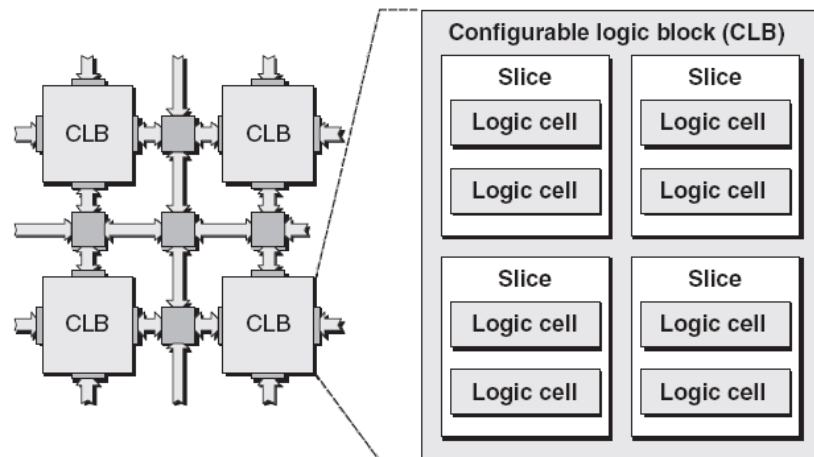


Figura 28 CLB que contem quatro *slices*. [3]

Para além das *slices* existe também dentro do CLB interligações programáveis com velocidade superior. Estas são utilizadas para fazer a conexão com entre *slices* adjacentes.

A razão para existir este tipo de arquitectura LC-*slice*-CLB é o facto de existir uma arquitectura equivalente em termos de ligações. Existem ligações rápidas entre os LC numa *slice*, depois ligações ligeiramente mais lentas interligam as *slices*, seguidas das ligações que interligam os CLB's. A ideia por detrás desta arquitectura é otimizar a troca de informação, tornando mais fácil conectar todas as estruturas, sem ter atrasos causados pelas ligações. [2]  
[3]

#### ➤ **FAST CARRY LOGIC**

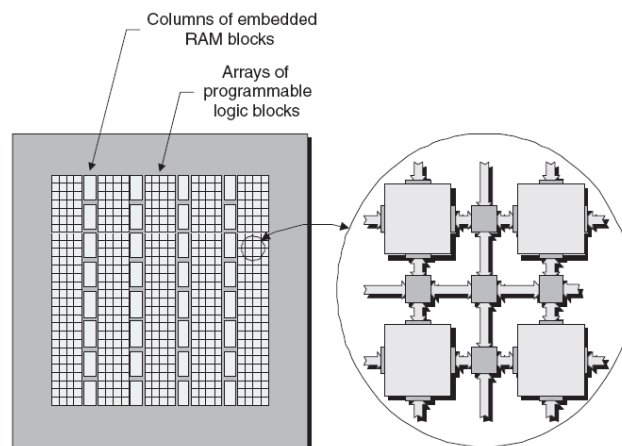
Uma característica chave das FPGA modernas é o facto de possuírem uma lógica e ligações especialmente dedicadas à implementação de *fast carry logic*. Este tipo de lógica encontra-se

dentro da LC, que é posteriormente complementada por interligações que conectam as duas LC que estão dentro da *slice*, que conectam *slices* do CLB E os CLB entre si.

Esta lógica especial e ligações dedicadas melhoram a performance de funções lógicas como contadores e funções aritméticas como somadores. A disponibilidade destas ferramentas aliada a *shif registers* e multiplicadores embidos, tornam a FPGA ideal para desempenhar *Digital Signal Processing* (DSP).

#### ➤ **EMBEDDED RAMs**

Um grande número de aplicações requer o uso de memória, por isso as FPGA de hoje em dia possuem, blocos de memória RAM embebidos que são denominados de *e-RAM* ou *block RAM*. Da arquitectura do componentes, estes blocos podem encontrar-se posicionados na periferia do dispositivo, espalhados pela superfície do dispositivo, ou organizados em colunas, como representado na figura 29.



**Figura 29** Arquitectura de uma FPGA com *block RAM*. [3]

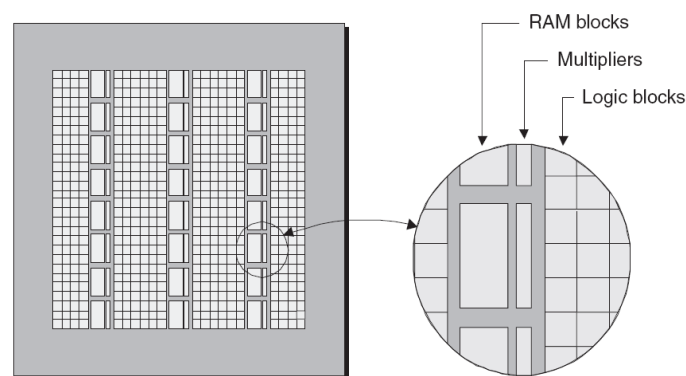
Dependendo do dispositivo, um bloco de memória RAM pode armazenar desde um milhão de bits até dezenas de milhares de bits. Para além disso o dispositivo pode ter embebido desde dezenas a centenas destes blocos, providenciando uma capacidade total de armazenamento que pode ir de centenas de milhares de bits até vários milhões de bits.

Cada bloco de memória RAM pode ser usado individualmente, ou múltiplos blocos podem ser combinados de forma a implementar um bloco maior. Estes blocos são tipicamente

utilizados para implementar memória RAM com um ou dois portos, funções *First-In First-Out* (FIFO) e máquinas de estados. [3]

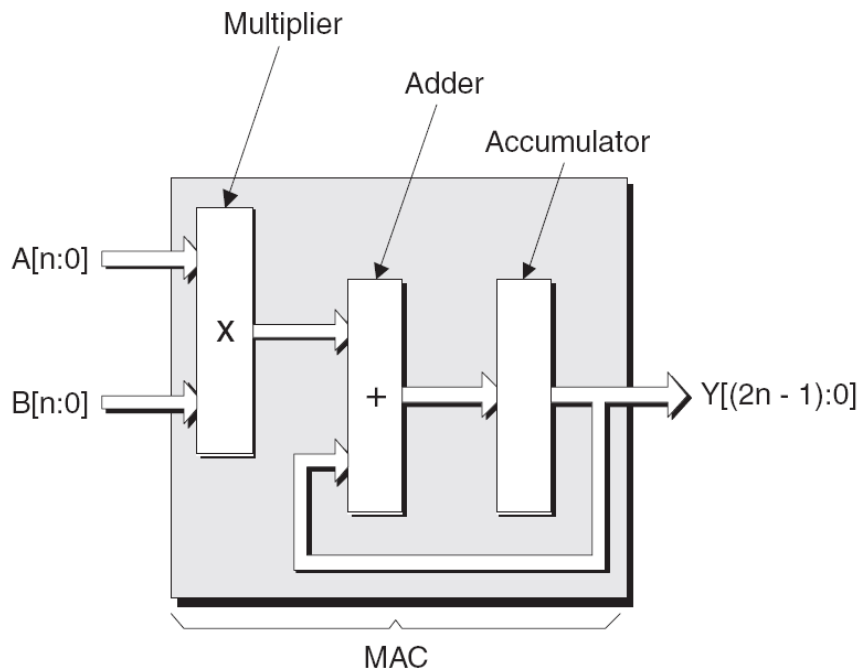
➤ **EMBEDDED MULTIPLIERS, ADDERS E MULTIPLY-AND-ACCUMULATE (MAC)**

Algumas funções como a multiplicação tornam-se relativamente lentas quando são implementadas através da conexão de blocos de lógica. O facto de estas funções serem necessárias em muitas aplicações, algumas das FPGA de hoje em dia incluem blocos especiais de multiplicação. Estes estão tipicamente localizados na proximidade dos blocos de memória RAM embebida, como se pode observar na figura 30.



**Figura 30 Vista aproximada dos blocos de multiplicação. [3]**

De forma idêntica, algumas FPGA oferecem também blocos de somadores. Uma operação muito utilizada em aplicações DSP chamada de *multiply-and-accumulate*, que se encontra representada na figura 31. Como o próprio nome sugere, esta função multiplica dois números e soma o resultado dessa operação com o valor guardado num acumulador.



**Figura 31 Funções que formam um bloco MAC. [3]**

Se a FPGA apenas tiver blocos de multiplicação embebidos, para implementar esta função é necessário combinar um multiplicador com um somador, que é formado de blocos de lógica programável, enquanto o resultado da operação pode ser armazenado numa associação de *flip-flop*, num bloco de memória RAM, ou em blocos de memória RAM distribuída. A implementação torna-se mais fácil se a FPGA fornecer blocos de soma, algumas até possuem embebido o próprio bloco MAC.

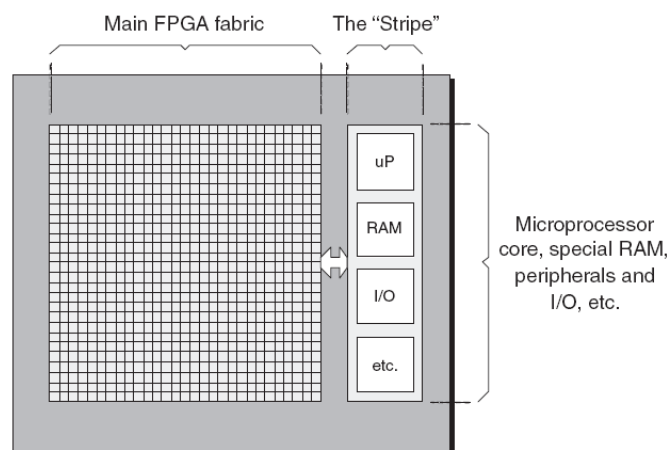
#### ➤ **EMBEDDED PROCESSING CORES (HARD AND SOFT)**

Quase todo o projecto de um circuito eléctrico pode ser realizado em *hardware* (usando portas lógicas, registos, etc.) ou por *software* (através de instruções a serem usadas por um microprocessador). Uma das principais critérios que cria divisão é o qual a rapidez que se pretende para a execução das funções que se pretende executar:

- Lógica do picosegundo e nanosegundo: esta funciona com extrema rapidez, o que dita que tem que ser implementada por *hardware* na FPGA.
- Lógica do microsegundo: esta lógica funciona a uma velocidade bastante elevada e pode ser implementada tanto por *hardware* como por *software*, o que leva muitas vezes a uma de tempo no que diz respeito à decisão de qual caminho seguir)
- Lógica do milissegundo: É a lógica normalmente utilizada para implementar interfaces, como a leitura do estado de interruptores, fazer piscar *Light-Emitting Diodes*(LEDs). É muito trabalhoso tornar o *hardware* mais lento para implementar este tipo de operações, através do uso de contadores para gerar esse atraso. Por isso muitas vezes é preferível implementar este tipo de funções como instruções de um microprocessador, que possui velocidade reduzida quando comparado com *hardware* dedicado, mas que permite a execução de funções de extrema complexidade

O que sucede na realidade é que de uma forma ou de outra, a maioria dos projectos usa um microprocessador. Até recentemente estes eram vistos como componentes discretos, mas as FPGA's de tecnologia de topo que existem actualmente, possuem embebidos um ou mais microprocessadores, que são tipicamente categorizados como *microprocessor cores*. Neste caso faz sentido que estas aplicações mais lentas sejam implementadas nestes núcleos, em vez de serem implementadas num microprocessador externo a FPGA. Isto fornece um grande número de vantagens, como redução de custos em termos de aquisição de componentes, diminui o número de pistas, vias e pinos numa *Print Circuit Board* (PCB), e torna a placa mais pequena e mais leve.

Um *hard microprocessor core* é implementado como um bloco dedicado e predefinido de *hardware*. Existem duas formas de integrar um núcleo de processamento numa FPGA. A primeira consiste em colocar uma “faixa” de *hardware* ao lado do circuito da FPGA, chamado de “*stripe*”, como se encontra representado na figura 32.



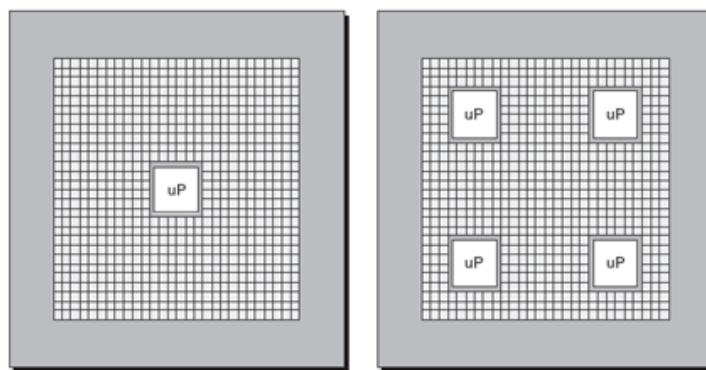
**Figura 32** *Chip com um microprocessor core embebido, fora do circuito principal da FPGA.* [3]

Neste cenário, todos os componentes são tipicamente concebidos no mesmo *chip* de silício, embora poderiam ser formados por dois *chips* e ser encapsulado num Multichip Module (MCM). O circuito principal da FPGA também pode conter memória RAM, multiplicadores e outras ferramentas embebidas, que não estão representadas nesta ilustração para termos de simplificação e ao mesmo tempo para dar relevância ao aspecto do núcleo de processamento embebido.

Uma vantagem desta implementação é que o circuito principal da FPGA é idêntico para dispositivos com ou sem microprocessador embebido, o que torna o processo mais fácil para as ferramentas de projecto usadas pelos engenheiros. Outra vantagem é que os fabricantes podem adicionar à *stripe* funções como memória ou periféricos especiais, de forma a complementar o *microprocessor core*.

Uma alternativa a esta arquitectura é embeber um ou mais *microprocessor cores* no circuito principal da FPGA. A figura 33 ilustra este tipo de arquitectura, bem como a forma como são inseridos os núcleos na FPGA.





**Figura 33** *Microprocessor cores* embebidos no circuito principal da FPGA. [3]

Mais uma vez são omissos os blocos de memória RAM, blocos multiplicação e outras ferramentas, para simplificar a figura.

Neste caso as ferramentas de projecto são capazes de se aperceber da presença destes blocos no circuito principal da FPGA. Neste caso qualquer memória que o núcleo necessite de usar é formada pelos blocos de memória RAM adjacentes e quaisquer funções de periféricos são implementadas pelos blocos de lógica que lhe são contíguos.

Esta arquitectura permite obter melhor performance em termos de velocidade, já que o núcleo se encontra inserido no circuito principal da FPGA.

Em oposição à opção de embeber fisicamente um microprocessador na FPGA, é possível configurar um grupo de blocos de lógica programável para agir como um microprocessador. Este é tipicamente designado de “*soft core*”, mas podem ser mais especificamente categorizados quer por “*soft core*” quer por “*firm core*”, dependendo da forma como o microprocessador é mapeado para os blocos de lógica.

*Soft cores* são mais simples e mais lentos do que os seus homólogos *hard cores*.<sup>4</sup> Contudo, estes possuem vantagens relativamente aos *hardware cores*, já que se pode implementar um núcleo sempre que se entender e que haja necessidade, já para não falar que se podem

---

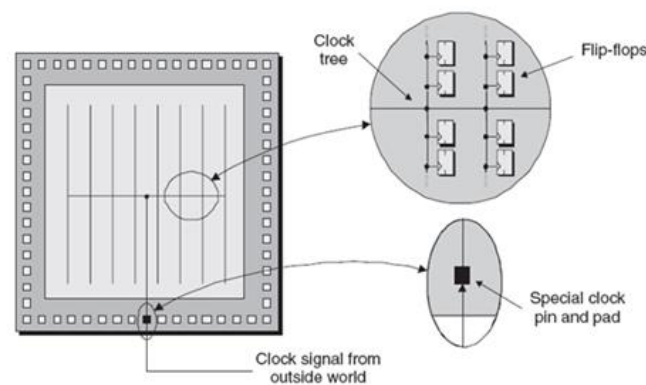
<sup>4</sup> Um *soft core* funciona de 30% a 50% da velocidade de um *hard core*

implementar tantos núcleos que se desejar até se ficar sem recursos, ou seja desde que existam blocos de lógica para o fazer. [3]

### ➤ *CLOCK TREES E CLOCK MANAGERS*

Todos os elementos dentro da FPGA, por exemplo os registos que actuam como *flip-flop* dentro do bloco de lógica programável, têm que ser controlados através de um *clock*. Este sinal por norma vem de fora da FPGA através de um pino de entrada dedicado, e posteriormente é encaminhado pelo dispositivo e conectado aos registos apropriados.

Considere-se a representação simplificada de uma árvore de *clock* da figura 34.



**Figura 34** *Clock tree*. [3]

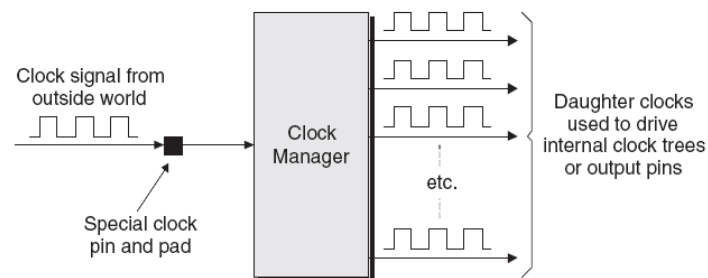
Nesta figura apenas estão representadas uma *clock tree* e os registos a que ela está conectada.

Este circuito é chamado de “*clock tree*” porque o sinal ramifica-se uma e outra vez. Esta estrutura é utilizada para garantir que todos os *flip-flop* recebem o sinal de *clock* ao mesmo tempo. Se o circuito de distribuição de *clock* fosse apenas uma linha longa a levar o *clock* a todos os *flip-flop*, um após o outro, então o *flip-flop* mais próximo do pino de *clock* iria receber o sinal de *clock* muito mais cedo do que o que se encontra no fim da linha, ou seja, o fenómeno conhecido como *skew*.

A árvore de *clock* é implementada usando pistas especiais e encontra-se separada das interconexões de uso geral, usadas para conectar os blocos de lógica e outros blocos embebidos na FPGA.

O cenário apresentado na figura 31 muito simplista, já que na realidade estão disponíveis vários pinos de *clock* e por conseguinte existem vários domínios de *clock* (*clock trees*) dentro do dispositivo.

O pino de *clock* pode no entanto ser configurado para controlar uma função especial (bloco) chamado de *clock manager* que gera um número de *daughter clocks*, como se encontra representado na figura 35.

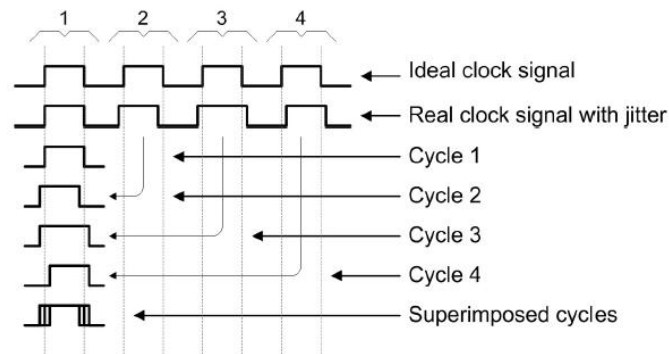


**Figura 35** *Clock manager e daughter clocks gerados.* [3]

Estes *daughter clocks* podem ser usados para alimentar as árvores internas de *clock* ou então pinos de saída, podendo fornecer *clocks* para outros dispositivos. Cada família de FPGA tem o seu tipo de *clock manager*, sendo que podem existir múltiplos blocos de *clock manager* num dispositivo, onde esses diferentes *clock managers* poderão suportar apenas algumas das seguintes características:

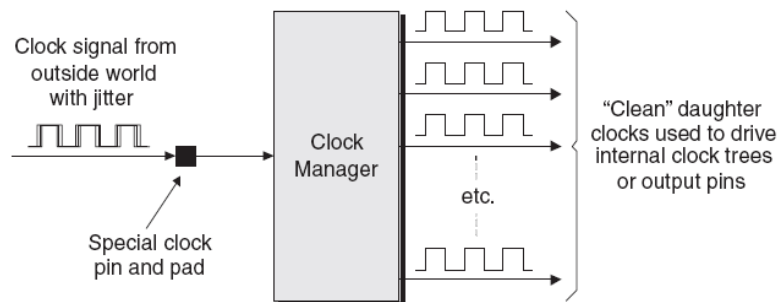
- *Jitter removal*: Para propósito de exemplo imagine-se um sinal de *clock* com uma frequência de 1 MHz. Num ambiente ideal cada *clock edge* vinda de fora do dispositivo, iria chegar precisamente 1 milésima de segundo depois da sua predecessora. No mundo real as *clock edges* podem chegar um pouco adiantadas ou um pouco atrasadas.

Uma forma de visualizar este efeito, conhecido como *jitter*, imagine-se que se sobreponham varias *edges* uma por cima da outra, o resultado seria um sinal de relógio difuso, como se pode observar na figura 36.



**Figura 36** Clock difuso criado pelo fenómeno de *jitter*. [3]

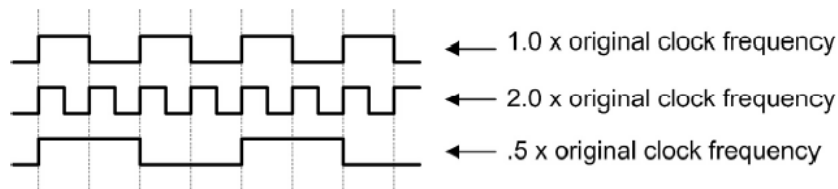
O *clock manager* da FPGA pode ser usado para detectar e corrigir o este *jitter* e fornecer um *daughter clock* “limpo”, a ser usado dentro do dispositivo, como o ilustrado pela figura 37.



**Figura 37** Uso do *clock manager* para remover o *jitter*. [3]

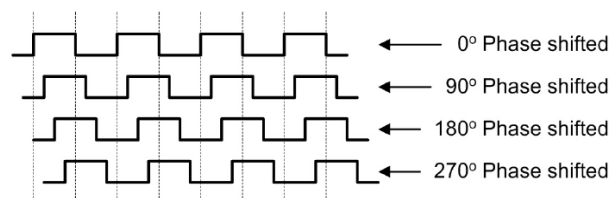
- *Frequency syntesis*: pode acontecer que o sinal que é introduzido na FPGA não seja o desejado pelos engenheiros de projecto. Neste caso o *clock manager* pode ser usado para gerar *daughter clocks* sinais com frequências que derivam da multiplicação ou divisão de frequência do sinal original.

Como exemplo desta ferramenta considere os sinais da figura 38. O primeiro é um *daughter clock* com frequência igual à do sinal original, o segundo é um *daughter clock* com uma frequência que é o dobro da do sinal original e o terceiro é um *daughter clock* com uma frequência que é metade do sinal original.



**Figura 38** Uso do *clock manager* para efectuar uma síntese de frequência. [3]

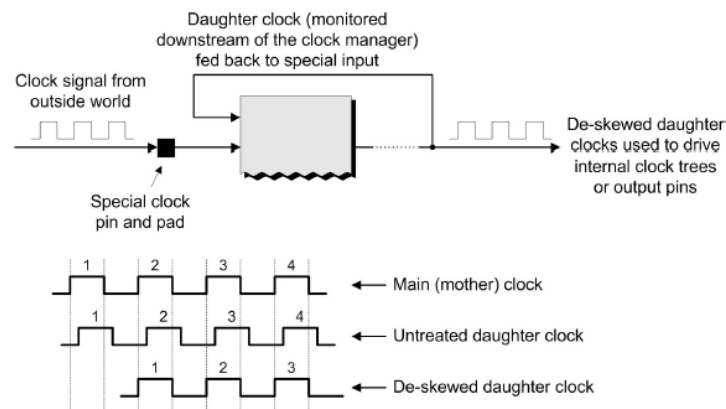
- *Phase shifting*: certos projectos requerem *clocks* desfasados entre si. Alguns *clock managers* permitem seleccionar desfasamentos fixos com valores comuns como 120° e 240 °(esquema de *clock* trifásico), ou 90°, 180° e 270 (esquema de *clock* quadrifásico como o representado na figura 39). Existem ainda outros que permitem seleccionar a quantidade de desfasamento do sinal original.



**Figura 39** Uso do *clock manager* para criar *daughter clocks* desfasados. [3]

- *Auto-skew correction*: para manter a simplicidade, vamos assumir que estamos a falar de um *daughter clock* que foi configurado para ter a mesma frequência e fase que o sinal principal de *clock* que é introduzido na FPGA. Por defeito o *clock manager* vai também adicionar um atraso a medida que faz a sua sintetização. Outro aspecto também bastante significativo é que vai também ser introduzido um atraso pelas portas e ligações que fazem a distribuição do *clock*. O resultado, se nada for feito, é um *daughter clock* que vai estar bastante atrasado em relação ao sinal principal de *clock*. A diferença entre estes dois sinais é conhecida por *skew*.

Dependendo da forma como o *clock* principal e o *daughter clock* são usados na FPGA, o *skew* pode causar muitos problemas. Por isso o *clock manager* pode permitir que uma entrada especial alimente o *daughter clock*. Neste caso o *clock manager* vai comparar os dois sinais e adiciona um atraso adicional ao *daughter clock* de forma a ele ficar realinhado com o *clock* principal, como ilustrado na figura 40.



**Figura 40** Monitorização e correcção de um *daughter clock*. [3]

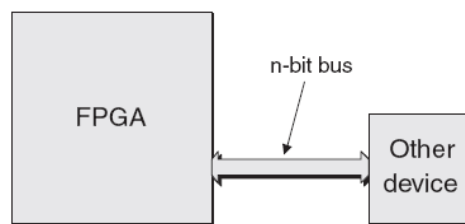
Para ser mais exacto apenas o primeiro *daughter clock* (*zero phase shifted*) e os restantes *daughter clocks* vão ser alinhados com este primeiro *daughter clock*.  
Proponentes

Algumas FPGA tem *clock managers* são baseados em *Phase-Locked Loop* (PLL), enquanto que outras possuem são baseados em *Delay-Locked Loop* (DLL) digital. Os

PLL's têm sido usados desde 1940 em implementações no domínio analógico, mas mais recentemente houve o interesse de fazer coincidir as fases de sinais digitais. Por isso os PLL's podem ser aplicados quer no domínio analógico, já os DLL's estão restringidos ao domínio digital, devido a sua natureza. Os defensores do DLL's afirmam que estes apresentam vantagens em termos de precisão, estabilidade, gestão de consumo e insensibilidade ao ruído. [3]

#### ➤ **GIGABIT TRANSCEIVERS**

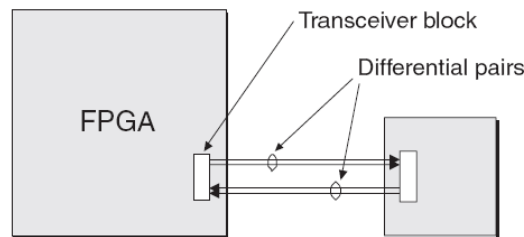
A forma tradicional de mover grandes quantidades de dados entre dispositivos é usando um *bus* (figura 41), um conjunto de sinais que transportam dados semelhante e executam a mesma função.



**Figura 41** Uso de um bus para comunicação entre dispositivos. [3]

Os primeiros sistemas baseados em microprocessadores usavam um bus de 8 bits para transmissão de dados. À medida que foi surgindo a necessidade de enviar dados cada vez mais rápido, o tamanho do *bus* aumentou para 16 bits, depois para 32 bits, posteriormente para 64 bits e por aí adiante. O problema é que isto requer que o dispositivo tenha um grande número de pinos, e por sua vez, requer um grande número de pistas para conectar os dois dispositivos. Desenhar estas pistas para que tenham o mesmo comprimento e impedância é uma tarefa muito trabalhosa e torna as placas de circuito impresso muito complexas. Para além disso torna-se extremamente difícil gerir problemas relacionados com a integridade do sinal, quando se está a utilizar um grande número de pistas de *bus*.

Por estas razões as FPGA's de hoje em dia podem incluir blocos especiais de *gigabit transceivers*. Estes blocos usam um par diferencial para transmitir para transmitir dados (Tx) e outro para receber dados (Rx), como representado na figura 42.



**Figura 42** Uso de transceivers de alta velocidade para comunicar entre dispositivos. [3]

Estes *transceivers* funcionam a velocidades incrivelmente elevadas, permitindo receber e transmitir bilhões de bits de dados, por segundo. Para além disso cada bloco pode suportar mais que um *transceiver* e a FPGA em si pode conter mais do que um destes blocos. [3]

#### **2.6.4. MODOS DE CONFIGURAÇÃO DE UMA FPGA**

Apesar de cada fabricante ter a suas nomenclaturas, técnicas e protocolos, a programação de uma FPGA resume-se essencialmente à criação de um ficheiro de configuração, normalmente designado de “bit file”. Este ficheiro contém a informação que vai ser carregada para a FPGA, para que esta seja configurada para executar a função pretendida.

No caso das FPGA baseadas em *antifuse* o ficheiro de configuração contém essencialmente a representação dos *antifuses* que vão ser criados. Nestes dispositivos é possível visualizar as células *antifuse* espalhadas pelo dispositivo em sítios estratégicos. Para efectuar a programação é necessário colocar a FPGA num dispositivo de programação especial, de seguida o bit file é carregado pelo computador anfitrião, sendo que o dispositivo de programação vai usar este ficheiro para se guiar quando começar a aplicar pulsos de tensão elevada e corrente elevada, para criar os *antifuses* pretendidos.



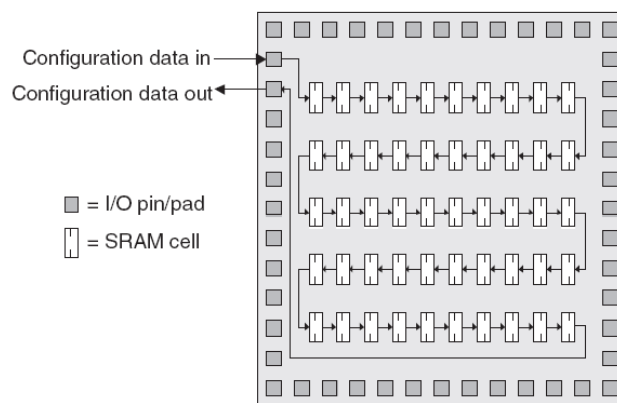
Uma forma simples de interpretar a programação é imaginar cada *antifuse* é olhar para cada um como se fosse uma localização virtual em termos de coordenadas “x” e “y” na superfície do dispositivo, onde esses valores de “x” e de “y” são valores inteiros, onde essa localização vai ser seleccionada quando se pretender programar esse *antifuse*.

Uma vez criados os *antifuse*, a FPGA é removida do dispositivo e colocada na placa de circuito impresso onde vai funcionar. Os dispositivos baseados em *antifuse* são OTP visto que uma vez programados o seu estado não pode ser alterado.

No caso das FPGA SRAM, o ficheiro de configuração contem uma mistura de dados de configuração (dados usados para programar directamente os elementos lógicos) e comandos de configuração (instruções que dizem ao dispositivo o que fazer com esses dados de configuração).

Estes dispositivos são *in-system programmable*, ou seja, são programados no circuito onde estão inseridos, e necessitam de ser reprogramados se se remover a alimentação.

Do ponto de vista do exterior, a FPGA baseada em tecnologia SRAM é vista como se fosse um deslocador de registos longo, como o representado na figura 43.



**Figura 43 FPGA SRAM vista como um deslocador de registos longo. [3]**

As FPGA iniciais usavam uma ferramenta chamada de porto de configuração. Mesmo hoje em dia quando existem tecnologias mais sofisticadas, como JTAG, o porto de configuração continua a ser largamente usado, devido a sua simplicidade.

As FPGA EEPROM e *Flash* usam a mesma analogia do deslocador de registos da tecnologia SRAM.

### ➤ PORTO DE PROGRAMAÇÃO

O porto de programação possui um pequeno grupo de pinos de configuração dedicados que são usados para indicar que tipo de configuração que vai ser usado. Nos primórdios apenas eram usados 2 pinos, que forneciam 4 modos de funcionamento, que estão ilustrados na tabela 2.

Mode Pins	Mode
0 0	Serial load with FPGA as master
0 1	Serial load with FPGA as slave
1 0	Parallel load with FPGA as master
1 1	Parallel load with FPGA as slave

**Tabela 2** Quatro modos de configuração originais. [3]

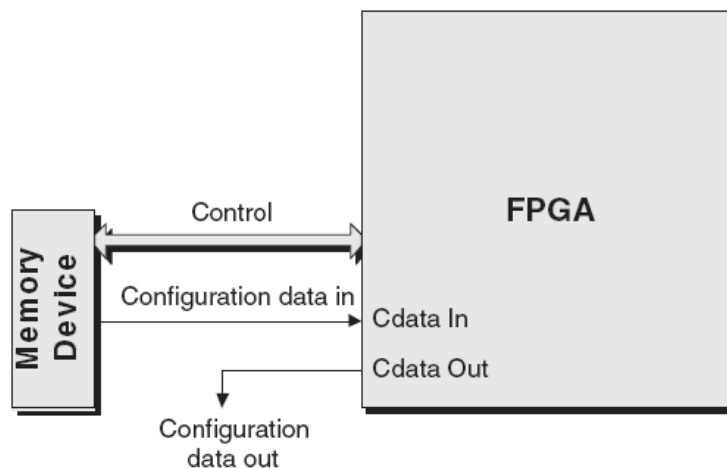
Os pinos que definem o modo de funcionamento são ligados aos valores lógicos “0” ou “1” consoante o modo de funcionamento pretendido. Para além destes pinos a FPGA possui um pino que indica quando se pode começar a efectuar a configuração e outro quando indica que a se terminou a configuração.

O porto de configuração possui ainda pinos adicionais para efectuar o controlo da informação que é carregada para a FPGA. O número de pinos do porto de configuração varia com o número de modos existentes.

Assim que acaba o processo de configuração, grande parte destes pinos pode ser usado como pino de I/O.[3]

### ➤ **SERIAL LOAD**

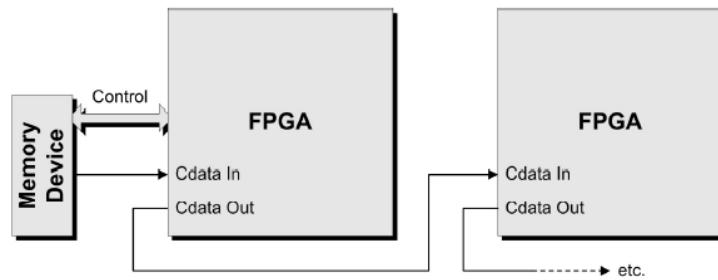
O modo mais simples de programação de uma FPGA é o *Serial Load*. Nos primórdios este método envolvia o uso de uma memória PROM, que foi posteriormente substituída por uma EPROM, que por sua vez foi substituída por uma EEPROM, que deu lugar a tecnologia FLASH. Esta memória de função específica tem apenas um pino de saída de dados que é ligado ao pino de entrada de dados de configuração, como o ilustrado na figura 44.



**Figura 44** *Serial Load* com a FPGA a funcionar como mestre. [3]

A FPGA usa também alguns bits para efectuar o controlo do dispositivo de memória externo, tais como sinal de *reset*, para dar a informação de que a FPGA está pronta para efectuar a leitura de dados e um sinal de *clock* para controlar a saída de dados.

O sinal de saída de dados de configuração proveniente da FPGA necessita de ser apenas conectada com o pino de leitura de dados de configuração do dispositivo de memória. Outro cenário que pode ocorrer é existirem múltiplas FPGA's no circuito. Neste caso cada FPGA poderá ter um dispositivo de memória dedicado e ser configuradas isoladamente, como o ilustrado na figura 42, ou então podem partilhar o mesmo dispositivo de memória e serem ligadas em cascata (*daisy-chain*) como ilustrado na figura 45.

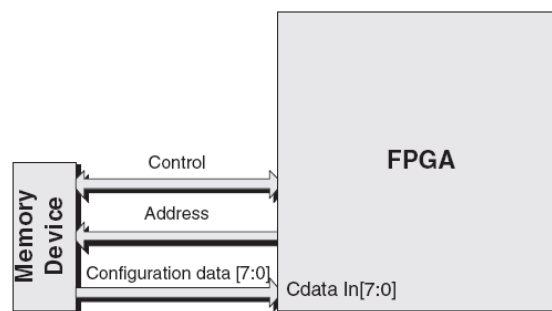


**Figura 45** FPGA's conectadas em *daisy-chain*. [3]

Neste cenário a FPGA que esta ligada directamente com o dispositivo de memória vai funcionar em *serial master mode*, enquanto a outra vai funcionar em *serial slave mode*. Esta topologia permite adicionar mais FPGA's em modo *serial slave load*. [3]

#### ➤ **PARALLEL LOAD COM A FPGA COMO MESTRE**

Em muitos aspectos este modo é semelhante ao anterior, excepto que os dados são lidos do dispositivo em blocos de 8 bits (1 byte), através de 8 pinos de saída. Para além de sinais de controlo, as FPGA iniciais forneciam ao dispositivo de memória externo qual o endereço do próximo byte a ser carregado, como o representado na figura 46.



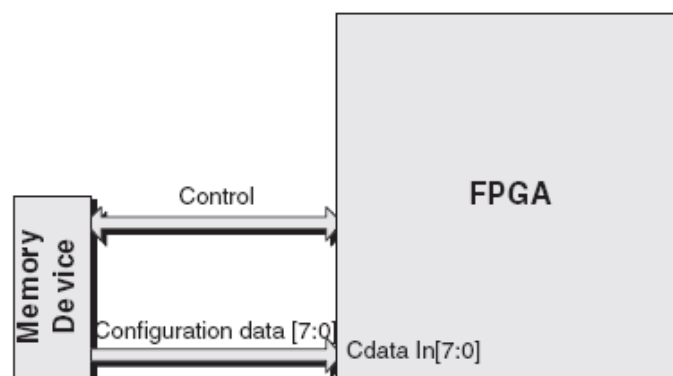
**Figura 46** *Parallel Load* com a FPGA a funcionar como mestre (técnica original). [3]

Isto funcionava usando um contador interno da FPGA que gera esses endereços (as FPGA originais tinham contadores de 24 bits que permitiam endereçar 16 milhões de bytes de dados). No início da sequência de configuração, este contador seria inicializado a zero. Depois de efectuada a leitura dos dados contidos no endereço indicado, o contador iria incrementar para ficar a apontar para o endereço do byte seguinte. Este processo vai continuar até serem lidos todos os dados do dispositivo de memória externa. Este tipo de técnica oferece vantagens em termo de velocidade.

Assim que termina o processo de configuração, os 8 pinos usados para a transferência de dados podem ser usados como pinos de I/O, o que na realidade não é o ideal. Como estes pinos têm pista ligadas ao dispositivo de memória externa, podem causar uma variedade de problemas de integridade de sinal.

A razão para a popularidade desta técnica junto das primeiras FPGA residia no facto de os dispositivos usados na topologia *serial load*, com a FPGA como mestre, possuírem um custo de aquisição muito elevado. Para além disso esta técnica permitia ao engenheiros utilizar dispositivos de memória *off-the-self*, que são relativamente mais baratos.

Actualmente os dispositivos de memória especializados para configurar as FPGA são baseados em tecnologia FLASH e relativamente baratos. Por isso as FPGA de hoje em dia utilizam uma variante da técnica original de *parallel load*. Neste caso o dispositivo de memória não necessita que seja fornecido externamente os endereços de memória, por isso FPGA já não necessita de um contador interno para gerar os mesmos, como se pode observar na figura 47.

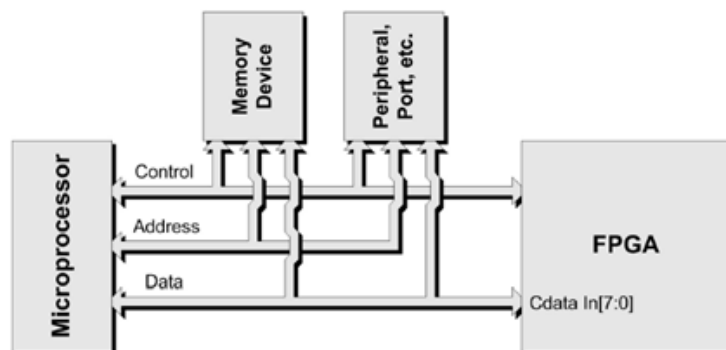


**Figura 47** *Parallel Load* com a FPGA como mestre (técnica moderna). [3]

### ➤ **PARALLEL LOAD COM A FPGA COMO ESCRAVO**

Os modos acima referidos, onde a FPGA funciona como mestre, são muito atractivos devido a sua simplicidade de implementação, já que só requerem o uso da FPGA e de um dispositivo de memória.

Contudo um grande número de placas de circuito impresso também incluem um microprocessador, o que muitas vezes leva os engenheiros a optar por ser o microprocessador a configurar a FPGA, como ilustra o diagrama da figura 48.



**Figura 48 Parallel Load com a FPGA como escravo. [3]**

A ideia desta topologia é ter o microprocessador a controlar a FPGA, sendo que é ele que vai informar a FPGA de quando quer iniciar o processo de configuração. Ele então efectua a leitura do dispositivo de memoria ou periférico apropriado, e começa a escrever os dados na FPGA, de seguida lê o próximo byte do dispositivo e memoria e escreve novamente na FPGA, repetindo este processo ate acabar a configuração da mesma.

Este cenário oferece uma serie de vantagens, no mínimo o facto de o microprocessador poder consultar o ambiente que o rodeia e seleccionar os dados de configuração que vão ser carregados para a FPGA.

### ➤ **SERIAL LOAD COM A PGA COMO ESCRAVO**

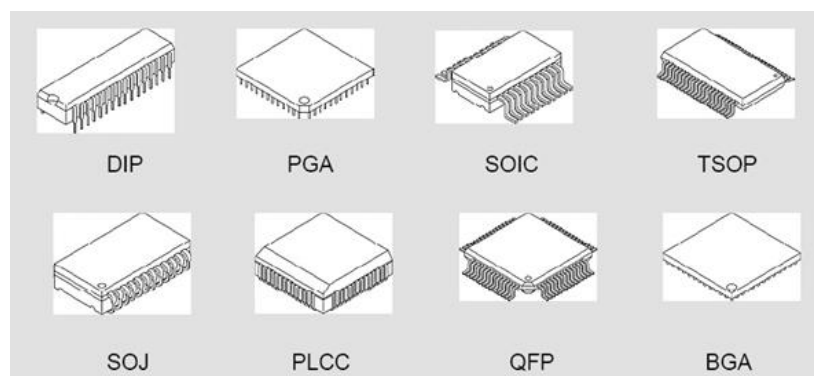
Este modo é em todo similar ao seu homólogo *parallel load*, apenas com a excepção de que só é usado um único bit para carregar os dados para a FPGA. O microprocessador é o responsável pela leitura dos dados de configuração do dispositivo apropriado, e de seguida converte-os numa serie de bits a serem carregados para a FPGA.

A principal vantagem deste modo é o facto de usar menos pinos de I/O da FPGA, já que só existe um pino de I/O da FPGA usado para configuração.

### ➤ **USO DO PORTO JTAG**

Como placas de circuito impresso se tornam mais complexas, a necessidade de teste completo torna-se cada vez mais importante. Avanços no *surface-mount package* e na concepção de placas de circuito impresso, resultaram em menores placas, tornando tradicional teste de sondas de métodos de teste e externa *bed-of-nails* mais difícil de implementar.

Como resultado, a redução de custos e reduções no tamanho da PCB, por vezes são compensadas por aumentos de custos nos tradicionais métodos de ensaio.



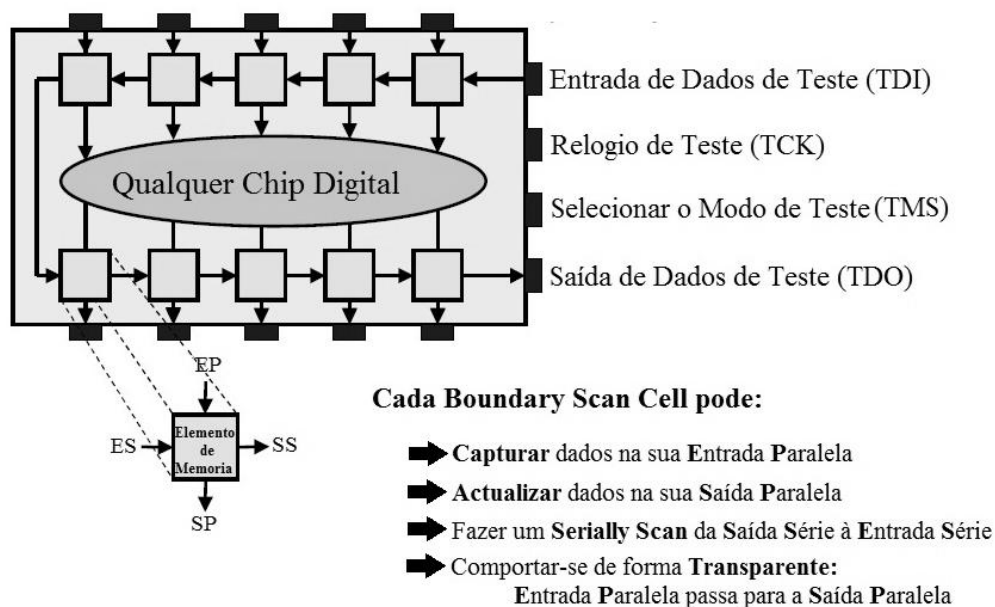
**Figura 49** Evolução do encapsulamento dos dispositivos. [7]

Na década de 1980, o JTAG desenvolveu uma especificação *Boundary Scan* para testes que mais tarde foi padronizado como o IEEE Standard 1149.1.

A arquitectura *Boundary Scan Test* (BST) oferece a capacidade de testar componentes de forma eficiente as placas de circuito impresso com espaçamento entre pontas de teste reduzido.

O conceito básico subjacente à tecnologia BST está representado na figura 50 e que consiste em associar uma célula dedicada ao teste, por cada pino funcional do componente. Estas células designam-se por células de varrimento periférico e a sua função principal consiste em desacoplar a lógica interna dos pinos. Quaisquer valores lógicos pretendidos podem ser aplicados nas saídas destas células, independentemente dos valores que estejam presentes nas suas entradas, tornando assim possível:

- Aplicar qualquer combinação nos pinos de saída do circuito integrado e nas entradas da sua lógica interna (através das saídas paralelas das células).
- Capturar os valores presentes nos pinos de entrada do circuito integrado e nas saídas da sua lógica interna (através das entradas paralelas das células). [7] [8]

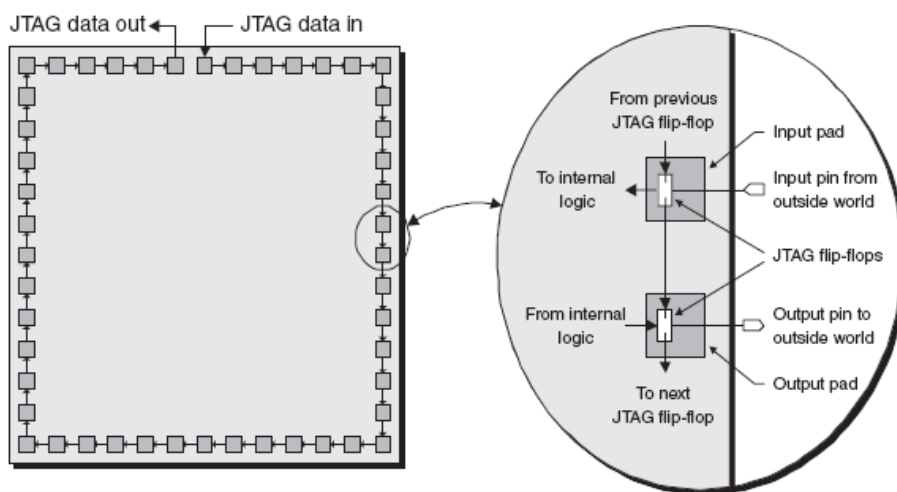


NOTA: Toda a lógica digital está contida no Boundary Scan Register

**Figura 50** Conceito básico subjacente à tecnologia BST. [8]



Assim como os demais dispositivos modernos, as FPGA's de hoje em dia também estão equipadas com o porto JTAG, ou seja possuem os pinos que requeridos pela norma IEEE 1149.1. Um desses pinos é usado para introduzir dados JTAG, e um outro é usado para saída de dados. Todos os pinos de I/O restantes da FPGA têm associado um registo JTAG (um flip-flop), onde estes registos estão interligados em *daisy-chain* como se pode observar na figura 51.



**Figura 51** Registos *Boundary Scan* do JTAG. [3]

A ideia por detrás do *Boundary Scan* é que é possível carregar dados em serie para o para os registos JTAG associados com os pinos de entrada, permitir que o dispositivo processar, guardar o resultado desse processamento nos registos JTAG, e por fim coloca-los disponíveis ao mundo exterior através do pino de saída do porto JTAG. Contudo os dispositivos JTAG podem conter alguma lógica de controlo associada ao porto JTAG, permitindo que este possa ser usado para efectuar mais que *Boundary Scan*. Por exemplo é possível emitir comandos especiais que são carregados para um registo especial JTAG de comando, através do pino de entrada deste porto. Um desses comandos permite conectar a cadeia de registos JTAG com a cadeia interna SRAM de configuração. Neste caso é possível então programar uma FPGA através do porto JTAG. [3]

## 2.6.5. LINGUAGENS DE PROGRAMAÇÃO DE FPGA

Com o aumento do tamanho e da complexidade dos projectos, os projectos de ASIC's realizados na forma esquemática atingiram um impasse. Tornou-se extremamente difícil visualizar, capturar, fazer a depuração, compreender o projecto e manter um nível de abstracção ao nível da porta lógica, quando um projecto compreendia mais que 5000 portas lógicas.

Para além disso capturar um projecto<sup>5</sup> de tamanho avultado ao nível da porta lógica é propício a existência de erros de projecto, já para não falar que é um processo extremamente demorado. Por estas razões alguns vendedores de ferramentas *Electronics Design Automation* (EDA) desenvolveram ferramentas de projecto com base na utilização de *Hardware Description Languages* (HDL).

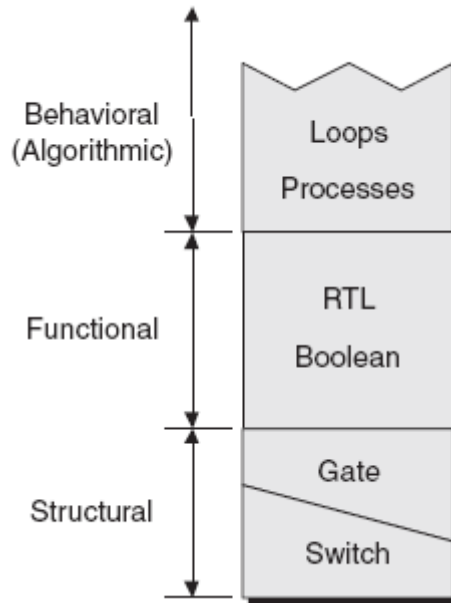
A ideia por detrás das linguagens de descrição de hardware é o facto que poder descrever o funcionamento de hardware. Neste contexto o termo “hardware” refere-se a qualquer parte física de um sistema electrónico, incluindo os circuitos integrados, placas de circuito impresso ate aos próprios cabos. Contudo no contexto de HDL, o hardware refere-se apenas porções de (componentes e fios) circuitos integrados e placas de circuito impresso.

Nos primórdios quem criava a ferramenta EDA criava a sua própria HDL. Algumas destas HDL eram concebidas para representar especificamente circuitos analógicos, enquanto que outras se limitavam apenas ao domínio digital. Contudo no ramo das FPGA's apenas tem relevância as HDL's que trabalham no domínio digital.

A primeira coisa que salta a vista é que a funcionalidade de um circuito digital pode ser representada em diferentes níveis de abstracção (representados na figura 52), e que as várias HDL's existentes suportam estes mesmos níveis de abstracção com maior ou menor grau.

---

<sup>5</sup> Captura de um esquema é um passo no ciclo de projecto *Electronic Design Automation* (EDA) em que o diagrama electrónico, ou esquemático electrónico do circuito projectado é criado por um projectista, com o auxílio de uma ferramenta de captura esquemática, também conhecida como editor esquemático.



**Figura 52** Diferentes níveis de abstracção. [3]

O nível mais baixo de abstracção para uma HDL é o nível do interruptor (*switch*), que se refere à capacidade de descrever o circuito como uma rede de transístores a funcionar como interruptor. Um nível abstracção ligeiramente a cima será o nível da porta lógica (*gate*), que se refere à capacidade de descrever o circuito como uma rede de portas lógicas e funções lógicas. Tanto o nível do interruptor como o da porta lógica podem ser classificados como representações estruturais.

O próximo nível de sofisticação do HDL é a capacidade de poder efectuar representações funcionais (*functional*), que compreende um grande número de construções.

No nível mais baixo esta a capacidade de representar equações booleanas. O nível de abstracção funcional engloba também as representações *Register Transfer Level* (RTL). O termo RTL abrange uma multiplicidade de manifestações, mas uma forma simplista de interpretar este conceito é considerando um projecto formado por um conjunto de registos interligado por lógica combinatória. Estes circuitos são muitas vezes controlados por um sinal de *clock* comum, por isso pode-se assumir que existem dois sinais chamados de CLOCK e CONTROL, juntamente com um conjunto de registos REGA, REGB, REGC e REGD, então todas as linhas de comando do tipo RTL são qualquer coisas do género:

```

when CLOCK rises
  if CONTROL == "1"
    then REGA = REGB & REGC;
    else REGA = REGB | REGD;
  end if;
end when;

```

**Figura 53** Exemplo de código de operações RTL. [3]

Neste caso, os símbolos *when*, *rise*, *if*, *then*, *else* e semelhantes, são palavras-chave, cuja semântica é definida pelo criador da linguagem HDL.

O nível mais alto de abstracção é o nível comportamental (*behavior*), que se refere a capacidade de descrever o comportamento do circuito, usando contrições abstractas como ciclos (*loops*) e processos (*processes*). Isto também abrange o uso de elementos de algoritmia como somadores e multiplicadores, em equações, similares as da figura 54.

```

Y = (DATA-A + DATA-B) * DATA-C;

```

**Figura 54** Exemplo de código de descrição funcional de um circuito. [3]

As primeiras linguagens de descrição de *hardware* apenas suportavam representações estruturais na forma de ligações de interruptores ou portas lógicas. Outras linguagens como ABEL, CUPL, e PALASM eram usadas para fazer a captura da funcionalidade requerida para os PLD's. Estas linguagens suportavam níveis diferentes níveis de abstracção, como equações booleanas, tabelas de verdade em forma de texto, descrições de maquinas de estados finitas em forma de texto.

A nova geração de HDL's, que era predominantemente direccionados para simulação lógica, e suportavam níveis de habitação como RTL e comportamental.

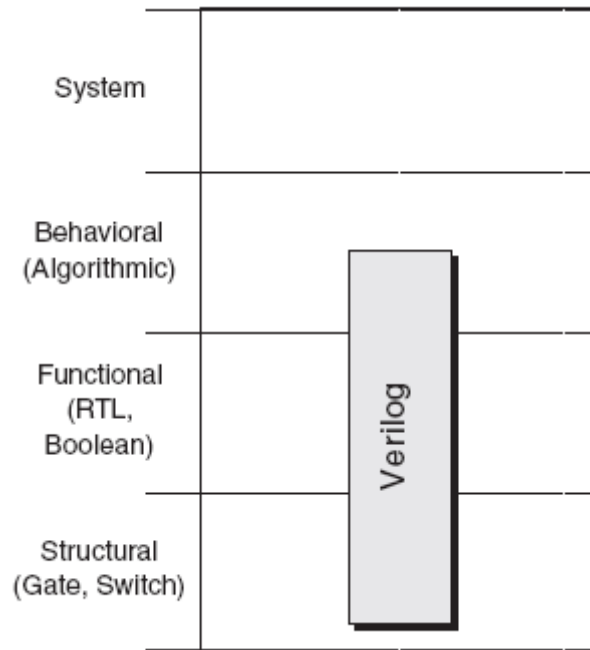
## ➤ VERILOG HDL

Em meados da década de 80, Phil Moorby criou uma nova HDL chamada de Verilog. Em 1985, a companhia onde trabalhava, Gateway Design Automation, introduziu esta linguagem no mercado, acompanhada de um simulador lógico chamado de Verilog-XL.

Um conceito interessante associado ao Verilog e ao Verilog-XL era o *Programmable Language Interface* (PLI) do Verilog. Na realidade o nome mais indicado para este interface é *Application Programming Software* (API), que não é mais do que uma livreria funções que permitem que softwares externos introduzam dados numa aplicação e aceder a dados dessa mesma aplicação. Por isso, o PLI Verilog é uma API que permite estender as funcionalidades do Verilog e do simulador.

Outra característica de grande relevância associada ao Verilog e Verilog-XL é a capacidade de ter a informação de tempos de atraso especificada num ficheiro de texto externo, denominado de *Standard Delay Format* (SDF). Isto permitia que ferramentas como *post-place-and route timing analysis packages* gerar o ficheiro SDF que poderia ser usado pelo simulador para fornecer respostas mais precisas.

Como linguagem, o Verilog era uma razoavelmente forte ao nível de abstracção estrutural (nível do interruptor e da porta lógica), era muito forte ao nível de abstracção funcional (equações booleanas e RTL), como se pode observar na figura 55.



**Figura 55 Níveis de abstração do Verilog. [3]**

Em 1989, a *Gateway Design Automation* foi adquirida pela *Cadence Design Systems*. O cenário mais previsível era o Verilog continuar a ser uma HDL proprietária, contudo em 1990 a *Cadence* colocou disponível ao domínio público o Verilog HDL, Verilo PLI e Verilog SDF.

Isto foi uma medida arrojada, já que significava que qualquer um poderia desenvolver um simulador Verilog e tornar-se um concorrente da *Cadence*. A razão pela qual a *Cadence* tomou esta iniciativa foi pelo facto de a linguagem VHDL começar a reunir muitos seguidores. A vantagem de tornar o Verilog uma linguagem de domínio público foi a de varias companhias que se encontravam a desenvolver ferramentas baseadas em HDL's, tais como aplicações de síntese de lógica, se sentirem agora confortáveis a usar o Verilog como sendo a sua linguagem de escolha.

Possuir uma única ferramenta de projecto que permitia a simulação, síntese, e uso muitas outras ferramentas, tornava mais fácil a vida de toda a gente.

O Verilog tornou-se rapidamente popular. O problema causado por esta popularidade residia no facto de várias empresas começarem a estender a linguagem em sentidos diferentes. De forma eliminar este tipo de problemas, foi criada em 1991 uma organização sem fins

lucrativos, chamada de *Open Verilog International* (OVI). Esta organização visava a criação de uma Verilog HDL e Verilog PLI standardizados.

A popularidade do Verilog continuou a subir de tal forma, que a OVI pediu *Institute of Electrical and Electronic Engineers* (IEEE) para criar um comité para fazer com que o Verilog fosse estabelecido como uma norma IEEE.

Conhecido como IEEE 1364, este comité foi criado em 1993, mas foi só em 1995 é que foi lançada oficialmente a norma IEEE do Verilog, que foi oficialmente chamada de IEEE 1364-1995.

Em 2001 foram efectuadas algumas pequenas alterações a esta norma, que é muitas vezes chamada de Verilog 2001. [3] [8]

➤ ***VERY HIGH SPEED INTEGRATED CIRCUIT HDL (VHDL) E VHDL INICIATIVE TOWARD ASIC LIBRARIES (VITAL)***

Em 1990 o departamento de defesa dos Estados Unidos da América lançou o programa *Very High Speed Integrated Circuit* (VHSIC), cujo principal objectivo era impulsionar o estado da arte da tecnologia de circuitos integrados digitais.

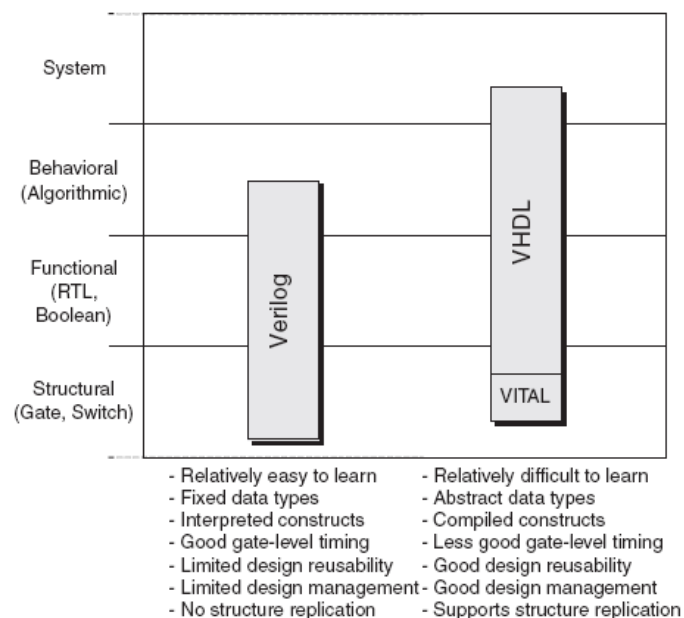
Este programa procurava resolver, entre outras coisas, o facto de ser difícil reproduzir circuitos integrados (e placas de circuito impresso) após longos ciclos de vida do equipamento militares em que estavam inseridos, isto porque a função dos componentes não era documentada em pormenor. Para além disso, disso diferentes componentes usados num sistema eram muitas vezes concebidos e testados por ferramentas de projecto diferentes.

De forma a resolver este problema, foi desenvolvido um novo tipo de HDL's chamado de VHSIC HDL (ou VHDL para encurtar) foi lançado em 1981. Um aspecto único deste processo é que a indústria de encontrou envolvida logo desde cedo neste projecto. Em 1983 uma equipa que era composta pela *Intermetrics*, IBM e *Texas Instruments* foi contratada para desenvolver a VHDL, tendo sido lançada oficialmente em 1985.

Um facto interessante foi o de o departamento de defesa dos Estados Unidos da América doar todos os direitos do VHDL ao IEEE em 1986, para encorajar a aceitação desta linguagem pela

indústria. Após algumas modificações efectuadas para colmatar alguns problemas conhecidos desta linguagem, o VHDL foi lançado como uma norma oficial chamada de IEEE 1076, no ano de 1987. Esta linguagem foi estendida em 1993 e novamente lançada em 1996.

Como linguagem o VHDL é muito forte no nível de abstracção funcional (equações booleanas e RTL), no nível de abstracção comportamental (algoritmos), e também suporta alguns designs ao nível do sistema. Contudo VHDL é bastante fraco no que diz respeito ao nível de abstracção estrutural (do interruptor e da porta lógica), principalmente no que diz respeito capacidade de criar a modelação de tempos de atraso. Tornou-se rapidamente visível que o VHDL não tinha precisão suficiente em termos de tempos, para ser usado como um simulador *sign-off*. Por esta razão foi lançada em 1992 na *Design Automation Conference* a iniciativa *VHDL Initiative Toward ASIC Libraries* (VITAL). VITAL foi um esforço para aumentar as funcionalidades do VHDL em termos de modelações de tempos em ambientes de projecto de ASIC e FPGA. O resultado final engloba uma livreria de funções primitivas e métodos associados para efectuar o *back-annotating* de informações relativas a atrasos, em modelos de livrerias, onde o mecanismo de atraso era baseado pelo mesmo formato tabular subjacente, usado pelo Verilog, como se pode observar na figura 56. [3] [8] [9]



**Figura 56 Níveis de abstracção, Verilog Versus VHDL. [3]**



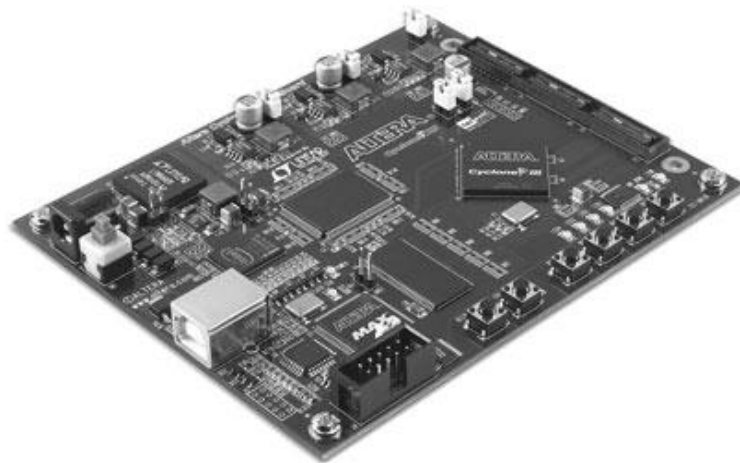
## 2.6.6. **KITS DE DESENVOLVIMENTO BASEADOS EM FPGA**

Antes de começar a concepção do protótipo, é necessário avaliar o que o mercado tem para oferecer, para depois decidir quais as melhores especificações para o produto final.

Neste tópico os *kits* estão organizados por fabricantes.

### ➤ **ALTERA**

- *CYCLONE III STARTER DEVELOPMENT KIT*



**Figura 57 Cyclone III Starter Development Kit da Altera. [10]**

Este *kit* baseia-se na FPGA EP3C25F324 da família *Cyclone III* da Altera. Esta possui 25000 LE's, 66 blocos de memória M9K, 16 blocos de multiplicação de 18×18, 4 PLL's

O *kit* oferece 214 I/O's disponíveis para o utilizador.

A configuração deste dispositivo é efectuada através do circuito USB-Blaster™ embebido na placa do *kit*. Este circuito inclui o CPLD EOM3128A da Altera, e permite que sejam enviados os ficheiros de configuração via Universal Serial Bus (USB).

Como dispositivos de armazenamento o kit oferece uma memória *Synchronous Dynamic Random Access Memory* (SDRAM) DDR de 256 Mbits, 1 Mbyte de memória SRAM síncrona, e 16 Mbytes de memória *Flash*.

O sinal de *clock* é produzido por um oscilador de 50MHz embutido na placa.

Como *interface* com o utilizador a kit possui 6 botões de pressão (4 configuráveis pelo utilizador) e 7 LED's (4 configuráveis pelo utilizador)

Para *interface* com o exterior possui conectores *High Speed Mezzanine Card* (HSMC) e Universal Serial Bus (USB) tipo B.

O custo de aquisição é de \$199. [10]

- *CYCLONE III FPGA DEVELOPMENT KIT*



**Figura 58 Cyclone III Development Kit da Altera.[10]**

Este *kit* de desenvolvimento baseia-se na FPGA EP3C120F780 da família *Cyclone III* da Altera. Esta possui 119 088 LE's, 3981312 bits de memória RAM, 576 blocos de multiplicação de 18×18, 4 PLL's

Este *kit* fornece 214 I/O's disponíveis para o utilizador.

A configuração deste dispositivo é efectuada através do circuito USB-Blaster™ embebido na placa do *kit*. Este circuito inclui o CPLD MAX II da Altera, e permite que sejam enviados os ficheiros de configuração através de um dispositivo de memória *Flash* e de um computador a funcionar como *host*.

Como dispositivos de armazenamento o *kit* oferece uma memória SDRAM DDR2 de 256 Mbytes, 8 Mbyte de memória SRAM síncrona, e 64 Mbytes de memória *Flash*.

O sinal de *clock* é produzido por 2 osciladores de 50MHz e 125MHz respectivamente, embutidos na placa. O circuito também possui conectores *SubMiniature version A* (SMA), que permitem a entrada de *clock* externo e a saída do *clock* do *kit*.

Como interface com o utilizador a *kit* possui vários botões de pressão, interruptores e LED's, um *Liquid Crystal Display* (LCD) de 2 linhas e 16 caracteres, um *Graphics Liquid Crystal Display* (GLCD) de 128×64.

Possui 2conectores HSMC, um conector USB tipo B e conector RJ45 (Ethernet).

Este *kit* permite alimentação através de fonte de alimentação externa e suporta os sistemas Americano, Europeu e do Reino Unido.

O custo de aquisição é de \$995. [10]

#### ➤ **EASY FPGA**

- EZ1CUSB



**Figura 59 Kit EZ1CUSB da Easy FPGA. [11]**

O *kit* de desenvolvimento EZ1CUSB fornece uma solução completa e de baixo custo para o desenvolvimento de projectos e aplicações baseadas em FPGA. Este *kit* utiliza FPGA da família *Cyclone* da Altera e o controlador USB FT2232C da FTDI.

Este oferece 128 I/O's disponíveis para utilizador e é configurável via USB.

Este produto possui 2 versões. A versão EZ1CUSB-6 utiliza a FPGA EP1C6Q240 que tem 5980 LE's, 20 blocos de memória M4K, 92160 bits de memória e 2 PLL's tem um custo de aquisição de \$199.

A versão EZ1CUSB-12 utiliza a FPGA EP1C12Q240 que tem 12060 LE's, 52 blocos de memória M4K, 239616 bits de memória, 2 PLL's e tem um custo de aquisição de \$299. [11]

- EZ1KUSB



**Figura 60 Kit EZ1KUSB da Easy FPGA. [11]**

O *kit* de desenvolvimento EZ1KUSB fornece uma solução completa e de baixo custo para o desenvolvimento de projectos e aplicações baseadas em FPGA. Este *kit* utiliza FPGA da família *Acex* da Altera e o controlador USB FT245BM da FTDI. O kit de desenvolvimento

EZ1KUSB oferece 58 I/O's (compatíveis com 5V) disponíveis para utilizador e é configurável via USB.

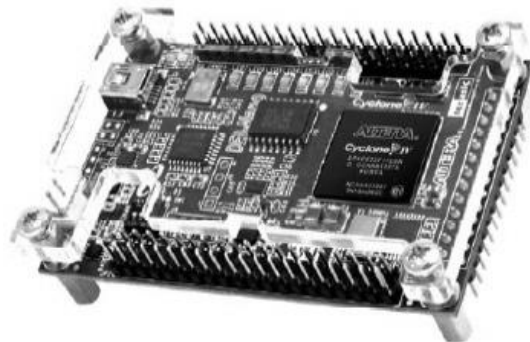
Este *kit* possui 3 versões. A versão EZ1KUSB-10 utiliza a FPGA EP1K10TC144 que tem 10000 portas lógicas, 576 LE's, 12220 bits de memória e tem um custo de aquisição de \$169.

A versão EZ1KUSB-30 utiliza a FPGA EP1K30TC144 que tem 30000 portas lógicas, 1728 LE's, 24576 bits de memória e um custo de aquisição de \$179.

A versão EZ1KUSB-50 utiliza a FPGA EP1K50TC144 que tem 50000 portas lógicas, 2880 LE's, 40960 bits de memória e um custo de aquisição de \$189. [11]

#### ➤ TERASIC

- DE0-NANO BOARD.



**Figura 61 Kit DE0-Nano Board da Terasic. [12]**

Este *kit* baseia-se na FPGA EP4CE22F17C6 da família *Cyclone IV* da Altera. Esta possui 22320 LE's, 594 Kbits de memória RAM, 66 blocos de multiplicação de 18×18, 4 PLL's

Este *kit* fornece 72 I/O's disponíveis para o utilizador.

A configuração deste dispositivo é efectuada através do circuito USB-Blaster™ embebido na placa do *kit*. Este circuito a memória *Flash* de configuração serie EPCS16.

Como dispositivos de armazenamento o *kit* oferece uma memória SDRAM de 32 Mbytes, uma memória EEPROM *Inter-Integrated Circuit* (I<sup>2</sup>C) de 2Kbits.

O sinal de *clock* é produzido por um oscilador de 50MHz embutido na placa.

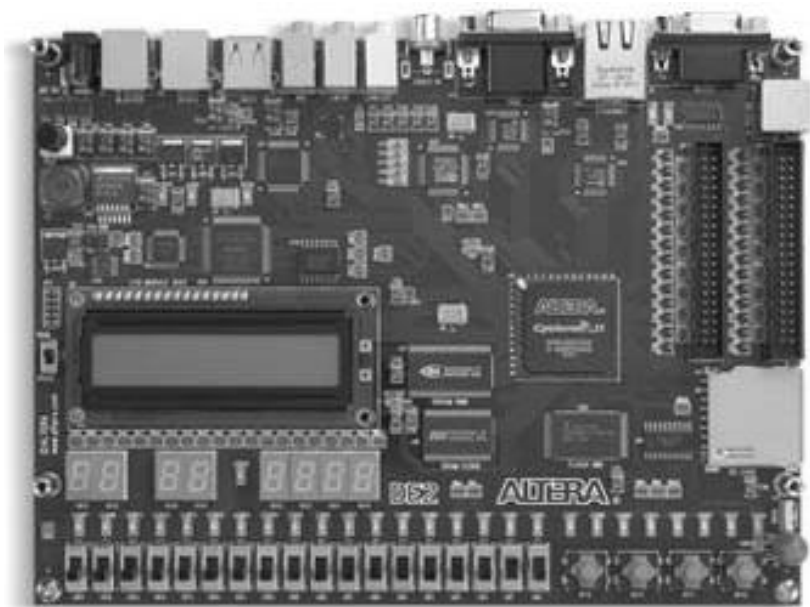
Como *interface* com o utilizador possui 2 botões de pressão, 4 *Dual In-line Package* (DIP) *switch*, 8 LED's.

Este kit também tem embebido o acelerómetro digital ADI ADXL345, de 3 eixos, com uma resolução de 13 bits, e o *Analog-to-Digital Converter* (ADC) ADC128S02, que tem 8 canais e 12 bits de resolução.

Este kit permite alimentação através de fonte de alimentação externa através de 2 pinos (que pode ir de 3,6V a 5,7V), bem como alimentação via USB (5V), através de uma ficha USB mini-AB.

O custo de aquisição deste *kit* varia, sendo ele \$79 para o publico geral e \$59 para publico académico.

- DE2 BOARD



**Figura 62** *Kit DE2 Board da Terasic.* [12]

Este *kit* baseia-se na FPGA EP2CE35 da família *Cyclone II* da Altera. Esta possui 33216 LE's, 483840 bits de memória RAM, 70 blocos de multiplicação de  $18 \times 18$ , 4 PLL's

A configuração deste dispositivo é efectuada através do circuito USB-Blaster™ embebido na placa do *kit*. Este circuito a memória flash de configuração serie EPCS16.

Como dispositivos de armazenamento o *kit* oferece uma memória SDRAM de 8 Mbytes, 512 Kbyte de Memoria SRAM, 4 Mbytes de memória *Flash*, e uma *slot* para cartão *Secure Digital* (SD)

O sinal de *clock* é produzido por 2 osciladores de 50MHz e 27MHz respectivamente, embutidos na placa.

Como interface com o utilizador possui 4 botões de pressão, 18 interruptores, 9 LED's configuráveis pelo utilizador, 8 *displays* de 7 segmentos e um LCD.

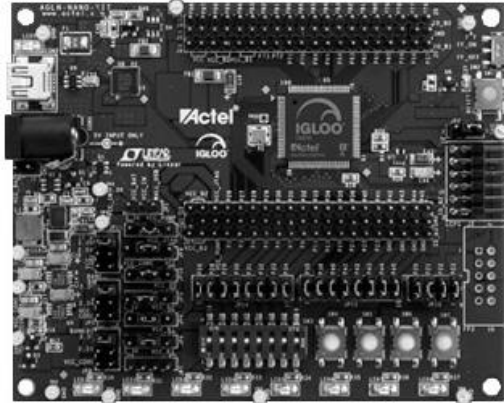
Este *kit* também tem embebido o acelerómetro digital ADI ADXL345, de 3 eixos, com uma resolução de 13 bits, e o *Analog-to-Digital Converter* (ADC) ADC128S02, que tem 8 canais e 12 bits de resolução.

Este *kit* de desenvolvimento oferece como *interfaces* para o exterior, um controlador USB mestre/escravo com 2 conectores USB (um de tipo A e um do tipo B); uma entrada de TV com descodificador associado; uma ficha *Video Graphics Array* (VGA) com 3 *Digital-to-Analog Converter* (ADC) de 10 bits; um conector Ethernet; *jacks* para microfone, linha de entrada e saída de som com qualidade de CD 24 Bits; um *transceiver* RS-232 com uma ficha DB9 associada, 2 conectores PS/2 para rato e teclado; um *transceiver Infrared Data Association* (IrDA) e 2 *pin headers* de 40 pinos para expansão.

O custo de aquisição deste *kit* varia, sendo ele \$495 para o publico geral e \$269 para publico académico. [12]

➤ **ACTEL**

- *IGLOO NANO STARTER KIT*



**Figura 63 IGLOO Nano Starter Kit da Actel. [13]**

Este *kit* baseia-se na FPGA *low-power* AGLN250-VQG100 da família IGLOO da Actel. Esta possui 250000 *system gates*, 2048 macrocélulas, 36 Kbits de memória RAM e 1 PLL com 68 I/O's.

A configuração deste dispositivo é efectuada através do circuito USB.

O sinal de *clock* é produzido por um oscilador de 20MHz incluído no circuito.

Como *interface* com o utilizador possui 4 botões de pressão e 8 LED's configuráveis pelo utilizador.

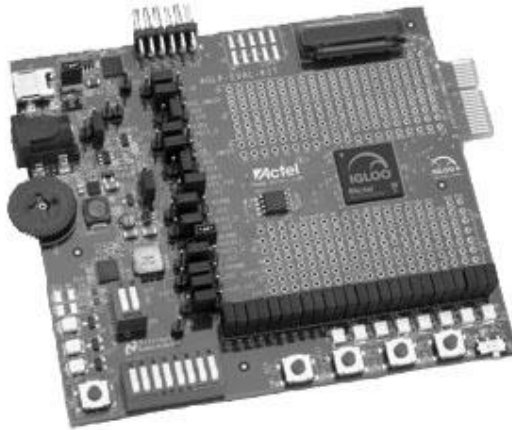
Este *kit* oferece uma conexão USB-to-*Universal Asynchronous Receiver/Transmitter* (UART) para comunicação com *Hyperterminal*.

Enquanto se encontrar ligado a uma ficha USB, o *kit* não necessita de alimentação externa, embora permita essa opção.

O custo de aquisição é de \$99. [13]



- IGLOO *Plus Starter Kit*



**Figura 64 IGLOO Plus Starter Kit. [13]**

Este *kit* baseia-se na FPGA *low-power* AGLP125V2-CSG289 da família IGLOO da Actel. Esta possui 125000 *system gates*, 1024 macrocélulas, 36 Kbits de memória RAM e 1 PLL com 71 I/O's. Esta FPGA possui saídas com *Schmitt Trigger*, para fornecer uma maior protecção contra ruído.

A configuração deste dispositivo é efectuada através do circuito USB.

O sinal de *clock* é produzido por um oscilador de 20MHz incluído no circuito.

Como *interface* com o utilizador possui 5 botões de pressão (4 configuráveis pelo utilizador) e 11 LED's (8 configuráveis pelo utilizador) e um *display* de baixo consumo de *Organic Light-Emitting Diode* (OLED) de 96×16 pixels.

Este kit oferece uma conexão USB-to-UART para comunicação com *Hyperterminal*.

As bloco de saídas da podem ser configuradas independentemente para funcionarem nos níveis de tensão de 2,5V ou 3,3V.

Esta PCB suporta o porto JTAG, sendo que os pinos do mesmo podem funcionar a qualquer nível de tensão, desde que ele se enquadre entre 1,2V e 1,5V.

Enquanto se encontrar ligado a uma ficha USB, o *kit* não necessita de alimentação externa, embora permita essa opção.

O custo de aquisição é de \$299. [13]

➤ XILINX

- Avnet Spartan-6 LX9 *MicroBoard*

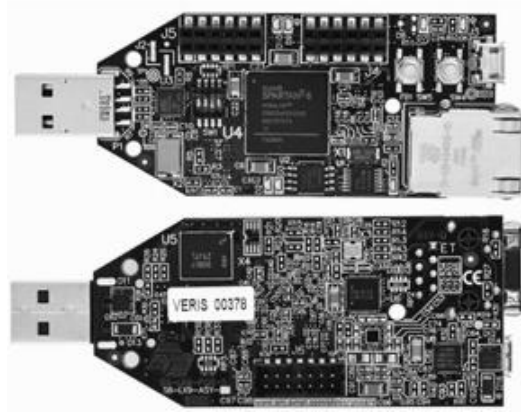


Figura 65 Kit Avnet Spartan-6 LX9 *MicroBoard* da Xilinx. [14]

Este kit baseia-se na FPGA XC6SLX9-2CSG324C da família Spartan 6 Xilinx. Esta possui 9152 LC's, 90 Kbits de memória RAM distribuída, 576Kbits *block* RAM, 16 blocos de multiplicação de 18×18, 2 PLL's

A configuração deste dispositivo é efectuada através do circuito USB.

Este kit oferece possui um conversor USB-to-UART, e um conversor USB-to-JTAG.

Como dispositivos de armazenamento o kit oferece uma memória SDRAM de 64 Mbytes, de memória *Flash Serial Peripheral Interface* (SPI) de 128Mb Mbytes.

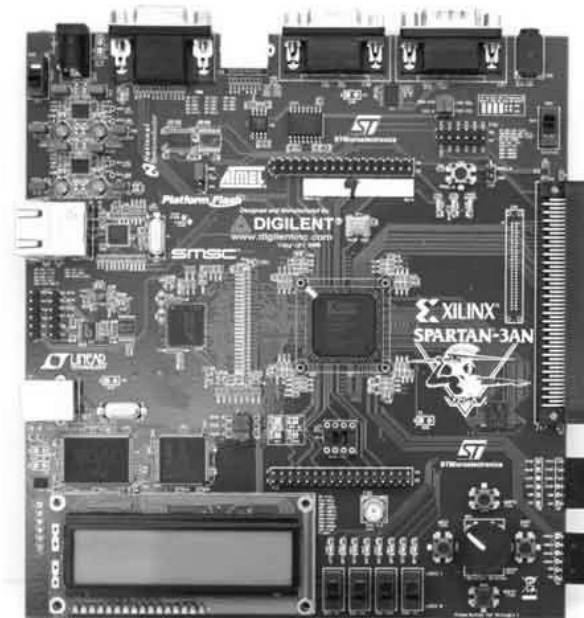
O sinal de *clock* é produzido gerador de *clock* programável.

Como *interface* com o utilizador possui 2 botões de pressão, 4 LED's e 4 DIP *switch*.

Este *kit* de desenvolvimento oferece como interfaces para o exterior um conector Ethernet e dois conectores *Peripheral Module* (PMOD) utilizados para expansão.

O custo de aquisição é de \$89. [14]

- *Spartan 3AN Evaluation Kit*



**Figura 66** *Spartan 3AN Evaluation Kit* da Xilinx. [14]

Este kit baseia-se na FPGA XC3S700AN-FG484 da família Spartan 3 Xilinx. Esta possui 13248 LC's, 92 Kbits de memória RAM distribuída, 360 Kbits de *block* RAM, 20 blocos de multiplicação de  $18 \times 18$ .

A configuração deste dispositivo é efectuada através do circuito USB, usando a conexão JTAG-USB.

Como dispositivos de armazenamento o kit oferece uma memória SDRAM de 32 Mbytes DDR2, duas memórias *Flash* SPI de 16Mb Mbytes, uma memória *Flash* PROM de 4Mbits e uma memória *parallel Flash* de 32 Mbits.

O sinal de *clock* é produzido por um oscilador de 50MHz embutido na placa, permitindo incluir outro oscilador numa outra *slot* existente.

Como interface com o utilizador possui 4 botões de pressão, 8 LED's, 4 interruptores, um botão rotativo para seleccionar funções e um LCD de 2 linhas e 16 caracteres.

Este kit possui também dispositivos analógicos, como um ADC de 4 canais, um DAC de 2 canais e um amplificador de sinal.

Em termos de conectores, a *board* oferece um conector Ethernet 10/100, um porto serie RS-232, um conector PS/2 para rato ou teclado, um conector VGA capaz de representar 4096 cores, 20 I/O disponíveis para p utilizador, um *mini-jack stereo* para efectuar áudio *Pulse-Width Modulation* (PWM).

O custo de aquisição é de \$189. [14]

# 3. PROJECTO DO SISTEMA DIDÁCTICO

Após um longo estudo das FPGA e do que o mercado tem para oferecer em termos de *kits* de desenvolvimento, seguiu-se a etapa de planeamento de projecto, sendo o primeiro tópico relevante a abordar é os requisitos de hardware pretendidos para o projecto.

## 3.1. REQUISITOS DE *HARDWARE* DO SISTEMA DIDÁCTICO

Estas especificações são ainda genéricas, visto que ainda não foram seleccionados os componentes que vão ser em particular para cumprir as directivas pretendidas.

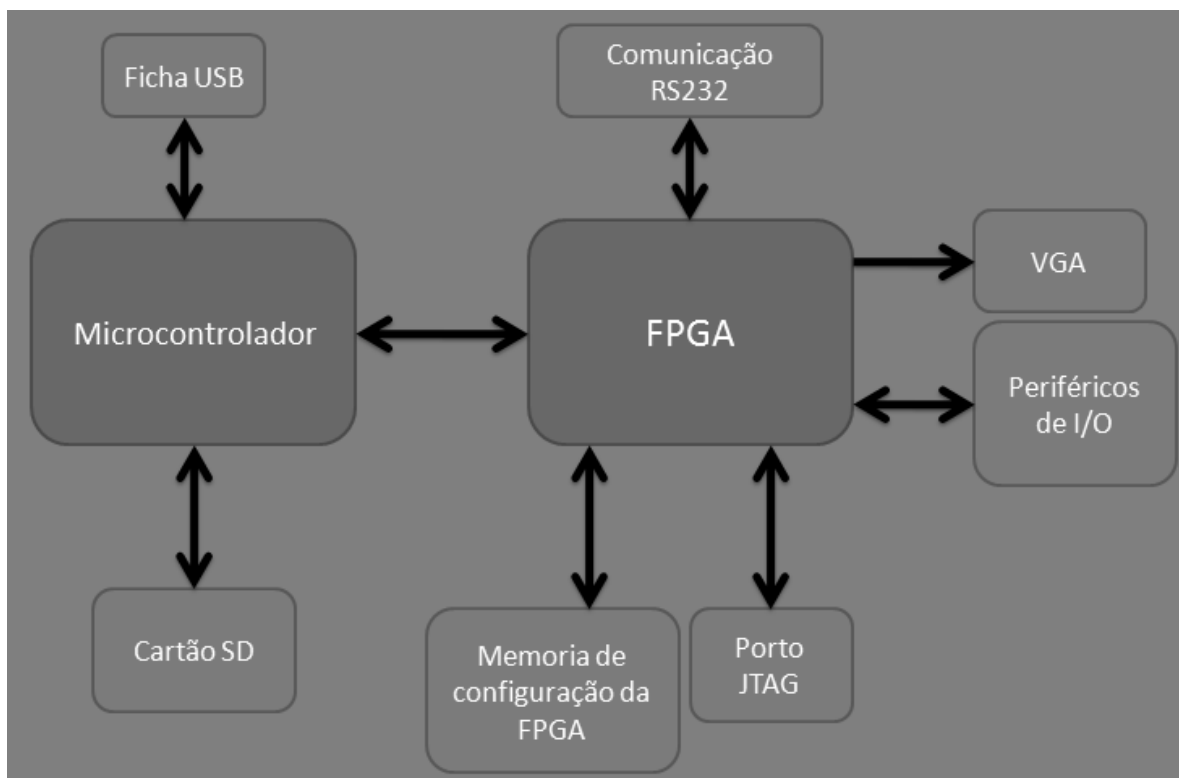
A ferramenta didáctica tem que possuir:

- FPGA de alta densidade
- Dispositivos de memória
  - SD card slot
  - Memória Flash
  - Memória de inicialização de FPGA

- Sistema de Clock
  - Oscilador
  - Entrada externa de *clock*
  - Saída de *clocks*
  - Interface para seleccionar o *clock* do sistema
- Interface de utilizados
  - Botões de pressão com *Schmitt Triggers* associados
  - *Dip switches* de 8 interruptores para similar estados lógicos
- Saídas do sistema
  - Conector RS232
  - Conector VGA
  - Conector USB
  - 2 *jacks* 3,5mm *sterio*
  - LCD
  - *Display* 7 segmentos
  - LED's configuráveis o utilizador
- Alimentação
  - Através de alimentação externa.
  - Pilha de *backup*
- Outras características
  - DAC dedicado ao conector VGA
  - ADC's

### 3.2. ARQUITECTURA DO SISTEMA

Na figura 67 está representada, de forma genérica, a arquitectura pretendida para o sistema. Nesta representação já estão também presentes algumas das escolhas para execução do projecto, como por exemplo a inclusão de um microcontrolador no sistema.



**Figura 67 Arquitectura do sistema didáctico.**

Na figura acima representada, para além de ilustrar a arquitectura do sistema, deixa transparecer as ferramentas que o sistema possui, bem como a forma como vai operar.

Como esta descrição é ainda genérica, e um pouco vaga, nos tópicos abaixo apresentados vão oferecer uma abordagem mais pormenorizada dos componentes do sistema.

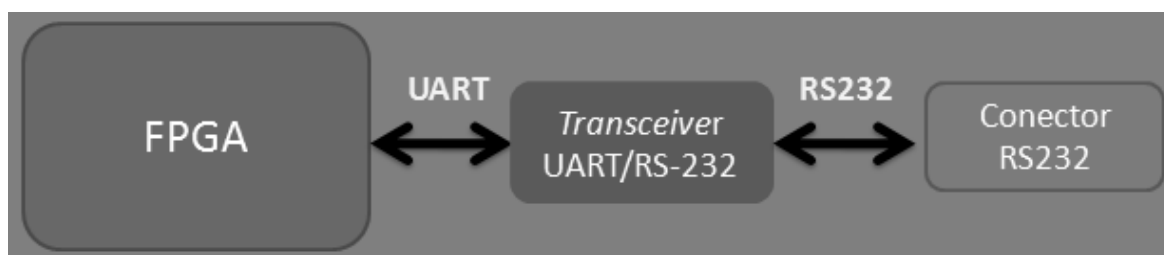
### 3.2.1. FPGA

A FPGA vai ser o cérebro deste sistema, sendo que é ela que comandar todos os periféricos incluindo o microprocessador.

Apesar de a FPGA ser uma FPGA de alta densidade, houve a preocupação não a sobrecarregar, deixando o controlo de alguns dispositivos para o microprocessador, isto com o intuito de deixar o maior número de recursos (portas lógicas) disponíveis para o utilizador da placa.

A FPGA será configurada através do porto JTAG. Mediante a tecnologia da FPGA a ser utilizada, o sistema vai incluir uma memória flash de configuração da FPGA, para evitar perda de dados caso seja removida a alimentação. Isto será apenas necessário se a tecnologia da FPGA for SRAM, que é um tipo de memória volátil.

No que diz respeito a comunicação com os periféricos e para o exterior, a FPGA comunica para o exterior via UART com um *transceiver* UART/RS232, que transforma essa comunicação serie, no protocolo RS232, como representado na figura 68.



**Figura 68 Implementação do protocolo RS232.**

A comunicação com o microprocessador é essencialmente uma comunicação UART, no entanto a FPGA tem um pino conectado a um pino de interrupção externa do microcontrolador, de forma a obter maior controlo sobre o mesmo.



Os restantes periféricos estão ligados directamente aos pinos da FPGA, com a excepção dos botões de pressão que ligam a um *Schmitt Triggers*.

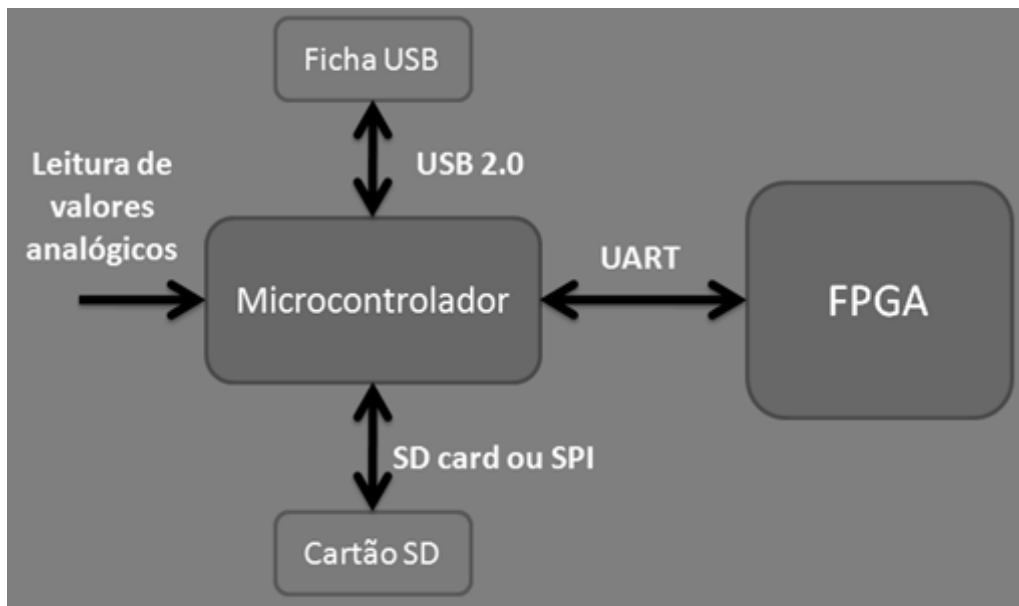
### **3.2.2. MICROCONTROLADOR**

O microcontrolador vai funcionar como um assistente da FPGA, estando essencialmente dedicado as funções de converter os dados enviados pela FPGA via UART no protocolo USB. O facto de usar o microcontrolador para efectuar esta função dispensa a inclusão de um circuito específico para efectuar a conversão UART – USB, o que no final se traduz numa poupança em termos de componentes e espaço na placa de circuito impresso. A conversão UART-USB poderia ser efectuada por chip dedicado a essa função, mas chegou-se a conclusão que é preferível incluir um microcontrolador que suporte esta função, visto que não trás grande acréscimo no seu custo, o facto de suportar esta função, e assim dispensa-se o uso de outro componente, o que leva a poupança no custo de aquisição do mesmo e de espaço na placa do *kit* didáctico.

Outra função que o microcontrolador vai desempenhar é escrever no cartão de memória SD o que lhe for enviado pela FPGA, isto permite poupar recursos da FPGA, já que o controlo do mesmo acaba por ser efectuado pelo microprocessador. O protocolo de comunicação com o cartão SD será o protocolo *SD card* ou o protocolo serie SPI.

O microcontrolador vai também funcionar como um ADC, convertendo os valores das suas entradas analógicas em valores digitais e envia-los para a FPGA, já que esta não possui ADC's embebidos.

A figura 69 ilustra as funções a desempenhar pelo microcontrolador.



**Figura 69 Funções do microcontrolador.**

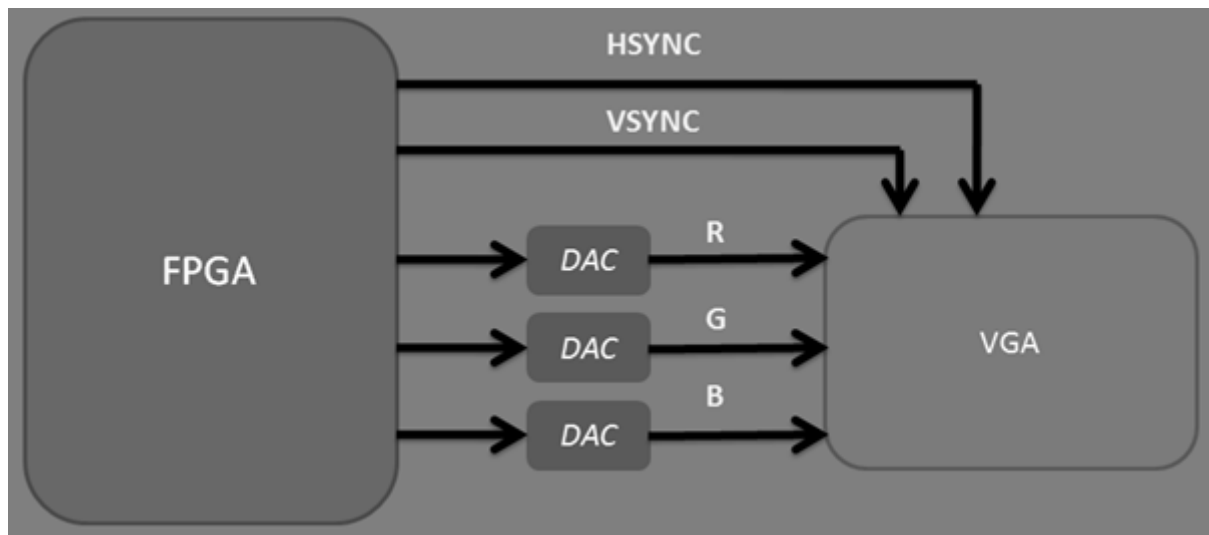
No entanto o mesmo não está limitado a estas funções e permite também uma expansão das mesmas, se o utilizador assim o pretender.

### **3.2.3. CONECTOR VGA**

Esta *interface* permite conectar a FPGA com um monitor ou projector, permitindo realizar projectos mais que requeiram transmitir informação visual de forma mais elaborada.

A FPGA não está directamente conectada, visto não possuir pinos analógicos. Entre a FPGA e o conector VGA vai existir um DAC por cada pino relacionado com a geração do modelo de cores *Red-Green-Blue* (RGB) que efectua a conversão dos valores digitais provenientes da FPGA em valores analógicos que vão ser introduzidos nesses pinos. Os restantes pinos de controlo (Hsync e Vsync) estão directamente conectados com a FPGA.

A figura 70 ilustra de forma mais clara o interface da FPGA com o conector VGA.

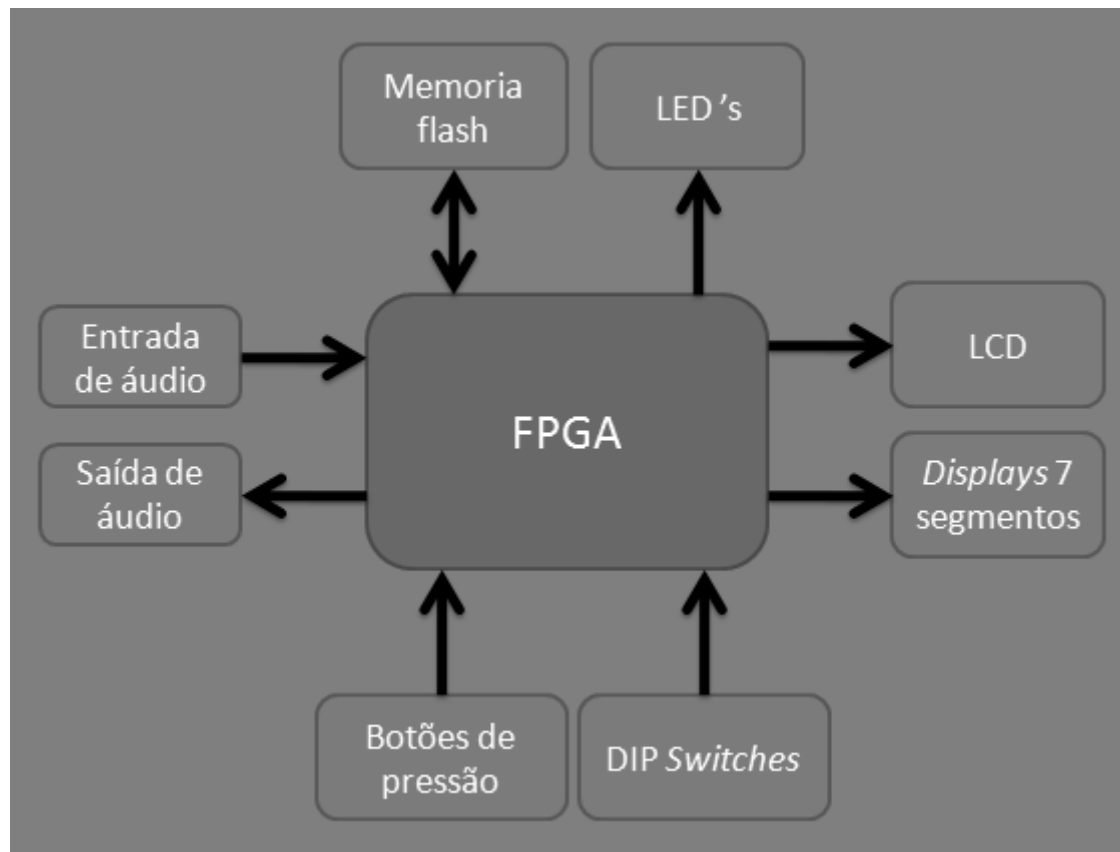


**Figura 70 Interface da FPGA com o conector VGA.**

#### **3.2.4. PERIFÉRICOS DE I/O**

No que diz respeito a periféricos de entrada e de saída, pretende-se oferecer o maior número possível de periféricos e de tipos de periféricos, para que a ferramenta didáctica seja o mais versátil possível.

Na figura 71 estão representados os periféricos de I/O que se pretende incluir no sistema.



**Figura 71** Periféricos de I/O.

Existe ainda um ponto a ter em conta no que diz respeito aos periféricos que se prende essencialmente com o custo de introdução no sistema. Ao associar mais periféricos ao sistema estamos a tornar o sistema mais caro.

## 4. IMPLEMENTAÇÃO DO PROJECTO

Depois da definição das especificações pretendidas e da arquitectura do sistema, o passo seguinte é a escolha dos componentes que vão de encontro a estas directivas.

Inicialmente foi concebida uma placa de circuito impresso com a dimensão de 16cm×10cm (formato *eurocard*), mas chegou-se à conclusão que teria mais vantagens dividir o sistema em duas placas com dimensões de 8cm×10cm (formato *semi-eurocard*) interligadas por meio de uma ligação *piggyback*. A ideia era incluir o maior número de ferramentas na placa inferior, funcionando esta como base do sistema, e a superior como uma expansão já que apenas iria conter periféricos. Isto torna o sistema mais compacto e versátil, já que a placa da base permite que lhe seja conectada qualquer placa de expansão.

No entanto para cumprir estes requisitos de espaço e manter o baixo custo do sistema, foi necessário abdicar de alguns periféricos de I/O, nomeadamente da memória Flash, dos *displays* de 7 segmentos, dos DIP *switch* e da entrada e saída de áudio.

## **4.1. SELECÇÃO DE COMPONENTES**

O primeiro componente a ser seleccionado é a FPGA que vai ser o núcleo deste sistema, e por requer mais atenção e cuidado no processo de escolha.

### **4.1.1. FPGA**

No que diz respeito à escolha de uma FPGA há que ter em conta os seguintes aspectos:

- Número de LC ou LE
- Número e tamanho dos blocos de memória RAM
- Número e tamanho dos blocos de multiplicação
- Número e tamanho dos blocos de soma
- Número e tamanho dos blocos de MAC
- Número de I/O's
- Tecnologia da FPGA
- Consumo da FPGA
- Linguagem de programação
- Encapsulamento da FPGA
- Custo de aquisição

Como o objectivo do sistema é ser um sistema didáctico, interessa que seja rapidamente programável, de forma a existir uma aprendizagem mais rápida.

Este tópico vai ser essencialmente condicionado pela tecnologia do dispositivo. Como a tecnologia *antifuse* não é uma tecnologia que permite reprogramação, é logo excluída a partida, restando as tecnologias SRAM, EEPROM e Flash. Como o tempo de programação destas duas últimas tecnologias (EEPROM e Flash) é três vezes superior ao da SRAM, é mais que evidente que é preferível optar por esta tecnologia. No entanto a tecnologia SRAM tem limitações e desvantagens, essencialmente por ser um tipo de tecnologia em que os dados são voláteis. Para não existir perdas dos dados é necessário que o dispositivo esteja sempre alimentado. Este problema pode ser resolvido colocando uma bateria de *backup* ou adicionar uma memória não volátil de inicialização da FPGA.

Ainda relacionado com o aspecto da programação, dependendo do fabricante, a linguagem de programação pode variar (tipicamente entre Verilog e VHDL). Devido ao facto de possuir conhecimentos em VHDL, e de ter trabalhado com dispositivos e ferramentas do fabricante Altera, na cadeira de Síntese de Alto Nível de Componentes Programáveis, leccionada no ISEP pelo engenheiro José Viera dos Santos, que levou a que fosses escolhido este fabricante.

Depois de um estudo das famílias da Altera optou-se por seleccionar uma FPGA de uma das famílias *Cyclone*, *Cyclone II*, *Cyclone III* *Cyclone IV*; que são as FPGA da Altera com menos consumo e custo de aquisição.

Após um estudo destas famílias e dos dispositivos que nelas estão incluídos, foi escolhida a FPGA EP3C5E144C7 (representada na figura 72) para ser a base do sistema didáctico.



**Figura 72** FPGA EP3C5E144C7N da Altera.[19]

Esta FPGA tem o encapsulamento *Enhanced Quad Flat Pack* (EQFP) de 144 pinos, e usa a tecnologia SRAM. Este dispositivo tem 5136 LE, 46 blocos de memória M9K, 423936 bits de RAM, 23 multiplicadores de 18×18 e 2 PLL's. [15]

➤ **LOGIC ELEMENTS E LOGIC ARRAY BLOCKS**

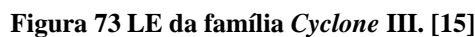
Os LAB's desta FPGA são constituídos por 16 LE's e um elemento de controlo do LAB. Um LE é o elemento lógico mais pequeno na família *Cyclone III*. Cada LE tem 4 entrada, uma LUT de 4 entradas, um registo e lógica de saída.

Cada LE tem as seguintes características:

- Uma LUT de 4 entradas que permite implementar qualquer função de 4 variáveis
- Um registo programável
- Uma conexão *carry chain*
- Tem a possibilidade de alimentar as seguintes interligações:
  - Local
  - Coluna
  - Linha
  - Ligação em cadeia de registos
  - *Direct Link*
  - *Register packing support*
  - *Register feedback support*

Na figura 73 encontra-se representado o LE da família *Cyclone III*.





Para funções combinatórias, a saída da LUT faz *bypass* ao registo e ataca directamente as saídas do LE.

100

funções distintas. O controlo síncrono do LAB não se encontra disponível, quando se utiliza do *register packing*.

O modo de *register feedback* permite que a saída do registo alimente a LUT do mesmo LE.

Como referido anteriormente, o LAB é constituído por um 16 de LE's, sendo que cada LAB tem as seguintes ligações:

- Sinais de controlo do LAB
- Das *carry chains* provenientes dos LE's
- Das *register chains*
- Com as interligações locais

As interligações locais são usadas para transferir os sinais entre os LE's do mesmo LAB. As ligações das *register chain* transferem os dados do registo de um LE para o registo do LE adjacente, dentro do mesmo LAB.

Na figura 74 encontra-se representada a estrutura de um LAB da família *Cyclone III*. [15]

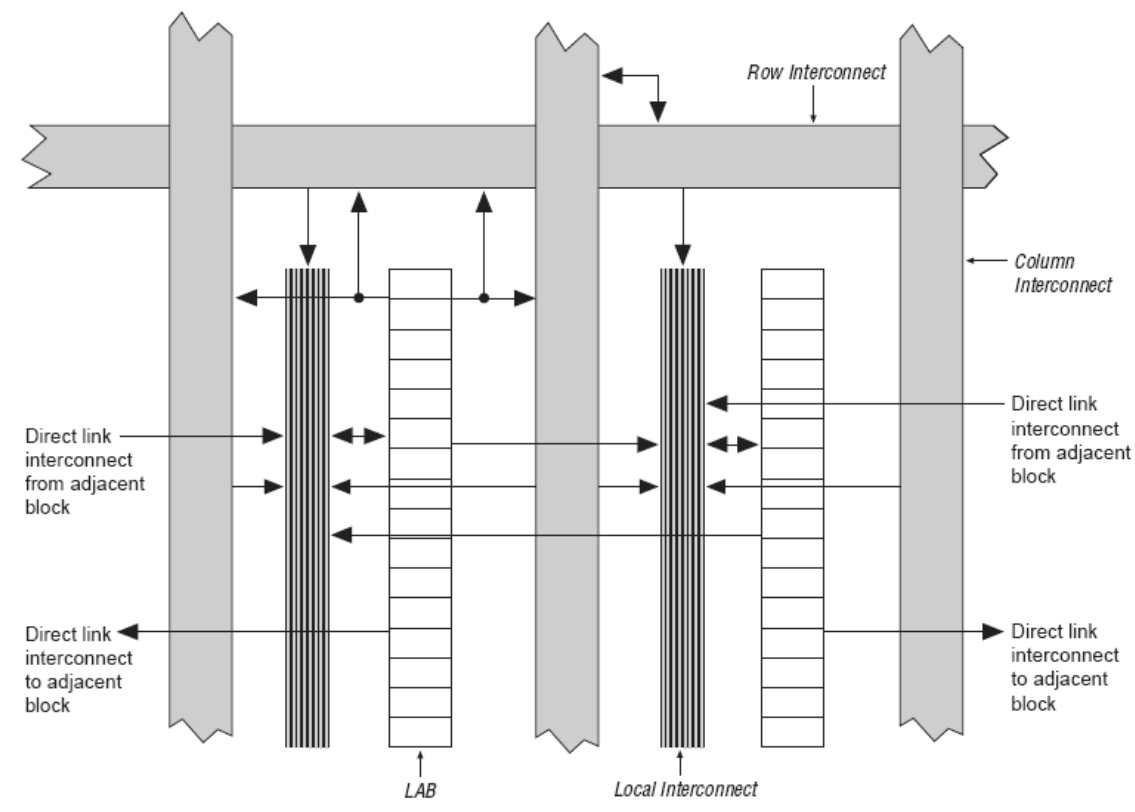


Figura 74 Estrutura de um LAB da Cyclone. [15]

#### 4.1.2. SELECÇÃO DA MEMÓRIA DE INICIALIZAÇÃO DA FPGA

Como a FPGA é baseada na tecnologia SRAM, encontra-se sujeita a perda de dados, já que este tipo de memória é volátil. Uma solução para este problema é usar uma memória não volátil de inicialização da FPGA.

O Altera possui memórias Flash cuja que permitem armazenar os dados de configuração da FPGA, e carrega-los para a FPGA assim que for restabelecida a alimentação do sistema.

Estes dispositivos têm várias versões, onde a principal diferença é a capacidade da memória. A Altera disponibiliza estas memórias com capacidades de 1 Mbit, 4 Mbits, 16 Mbits, 64 Mbits e 128Mbits. Tendo em conta o número de portas lógicas e LE's que a FPGA EP3C5E144C7 possui, uma memória de 16 Mbits é mais que suficiente para acomodar os dados de configuração. Na figura 75 está representado o dispositivo EPCS16 que a memória

flash da Altera de 16 Mbits com um encapsulamento *Smal-Outline Integrated Circuit* (SOIC) de 8 pinos.



**Figura 75 Memória Flash de inicialização de FPGA EPCS16SI8N. [19]**

Esta memória é configurada através de um *interface* denominado de Active Serial (AS), e tem um interface fácil de usar, já que apenas requer 4 pinos.

Esta memória também permite que o processador embebido NIOS da altera, acesse a ela através do *interface* AS.

Este dispositivo permite mais de 100000 ciclos de programação ou de apagar dados.

Assim como a FPGA este dispositivo também é ISP e permite também ser programado usando qualquer um dos cabos de programação da Altera. [16]

#### **4.1.3. MICROCONTROLADOR**

O microcontrolador, como já foi referido, vai funcionar como assistente da FPGA, e como referido no capítulo de projecto este dispositivo tem que possuir as seguintes especificações:

- Suportar comunicação USB 2.0
- Suportar o protocolo SD *card* ou SPI
- Ter ADC's embebidos

Devido a familiarização com o dispositivo 18F4550 da Microchip, e visto que ele preenche os requisitos acima referidos, foi escolhido para ser microcontrolador a ser incluído no sistema.

Para não ocupar muito espaço na placa de circuito impresso é preferível usar a versão *Surface-Mount Device* (SMD), daí ser usado o *Programmable Interface Controller* (PIC) 18F4550 com encapsulamento *Thin Quad Flat Pack* (TQFP)



**Figura 76 Programmable Interface Controller (PIC) 18F4550 da Microchip.[19]**

Este microcontrolador é um dispositivo muito versátil devido a quantidade de ferramentas que oferece, no entanto no que diz respeito a este projecto apenas algumas das suas características são relevantes, sendo elas:

- O facto de suportar o protocolo USB 2.0 e possuir um *transceiver* USB embebido.
- Ter um *Master Synchronous Serial Port* (MSSP) que suporta os protocolos serie SPI e I<sup>2</sup>C, bem como o modo mestre e modo escravo destes protocolos.
- Ter 13 ADC's de 10 bits.

Como referido anteriormente no capítulo de projecto, o microcontrolador não está limitado as funções que desempenha nesta plataforma didáctica, já que possibilita a expansão das suas funções. Se esse for o caso é aconselhável consultar o *datasheet* do fabricante com o objectivo de forma a conhecer as restantes características deste dispositivo. [17]

#### 4.1.4. *TRANSCIVER RS232*

Em termos de *transceivers* RS232 não existiu a necessidade de escolher muito, e foi seleccionado o MAX232, que é o *transceiver* RS232 mais usado.

Para ocupar menos espaço em termos de placa de circuito impresso, optou-se pela versão SMD, que se encontra representada na figura 77.



**Figura 77** MAX232 com encapsulamento SOIC de 16 pinos.[19]

#### 4.1.5. **LCD**

Como LCD optou-se por usar um LCD de 2 linhas e 16 caracteres, ilustrado na figura 78.



**Figura 78** LCD de 2 linhas e 16 caracteres. [19]

Este *display* possui 16 pinos, sendo que 2 deles não têm função alguma, 2 deles são dedicados a alimentação (VCC e GND).

O controlo é efectuado através de 4 pinos. O pino CONTR que serve para controlar contraste do LCD, o pino RS que faz o reset do *display*, o pino R/W que serve para leitura e escrita no LCD e o pino E que faz o *enable* do *display*.

O barramento de dados deste *display* é constituído por 8 pinos (D0 a D7).

#### **4.1.6. DAC'S DE INTERFACE COM O CONECTOR VGA**

No que diz respeito ao DAC que vai efectuar o *interface* da FPGA com o conector VGA, existiam 2 soluções possíveis, usar um circuito integrado que implementasse um DAC ou então implementar um DAC usando resistências.

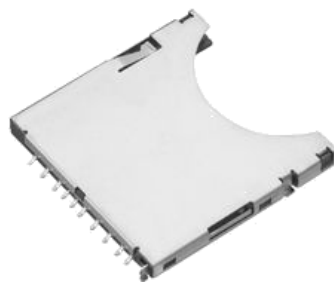
Com um DAC o sistema poderia suportar resoluções maiores e definir mais cores, mas seriam resoluções demasiado elevadas para um monitor analógico. Outro problema que o uso de um DAC apresenta é o facto de possuir muitos pinos de entrada, o que iria ocupar muitos recursos da FPGA, nomeadamente pinos de I/O, já que vai ser utilizado um DAC para cada pino do sistema de cores RGB. Uma solução para poupar recursos, seria usar um circuito integrado que implementasse um DAC com vários canais, ou então usar DAC's que suportassem protocolos serie (SPI ou I<sup>2</sup>C). No entanto estes tipos de dispositivos tem um custo elevado, e o investimento nestes componentes não iria ser compensado pelas mais-valias que poderiam trazer ao sistema.

Por estas razões e com o intuito de redução de custos, a solução seleccionada foi o uso de resistências, para implementar um DAC, que para além de permitir poupar dinheiro, permite poupar recursos da FPGA.

#### **4.1.7.        *SLOT* CARTÃO SD**

Relativamente ao cartão SD o formato de escolha foi o formato *standard*, por ser o formato mais barato de adquirir.

Na figura 79 esta uma imagem do *slot* para um cartão SD formato *standard* que vai ser utilizado na placa de circuito impresso.



**Figura 79 Slot cartão SD *standard*. [19]**

#### **4.1.8.        OSCILADOR**

Para sistema de *clock* do sistema optou-se por usar um oscilador de 50MHz, ilustrado na figura 80.



**Figura 80 Oscilador de 50MHz. [19]**



Estes componentes apenas necessitam de 3 pinos para funcionar, sendo 2 deles os pinos de alimentação (VCC e GND) e o outro a saída do *clock*.

No entanto a ideia é não soldar oscilador na placa mas sim soldar um pente fêmea para permitir a troca de oscilador deste oscilador por outro de forma a mudar a frequência de *clock* do sistema.

## **4.2. CONCEPÇÃO DO PROTÓTIPO DO SISTEMA**

Como foi anteriormente referido, inicialmente pretendia-se implementar o sistema numa placa de 16 cm×10cm, mas com o decorrer do trabalho chegou-se a conclusão que seria mais vantajoso dividir o sistema em dois. A ideia foi criar uma placa que funcionasse como o cérebro do sistema e que permitisse ao mesmo tempo ter disponíveis o máximo de pinos de I/O da FPGA e do PIC, que seria complementada por uma placa de periféricos que ligara a inferior através de *piggyback*.

Este sistema permite uma maior versatilidade do sistema, já que se pode criar placas de periféricos com fins específicos, para aplicar na placa base do sistema.

### **4.2.1. ESQUEMA DE LIGAÇÕES**

Como o sistema já tinha sido concebido inicialmente só numa placa de circuito impresso, o que corresponde a um só esquema eléctrico<sup>6</sup>, foi necessário pegar nesse esquema de ligações e dividi-lo pelas duas placas de circuito impresso, de forma a deixar apenas periféricos na placa superior.

---

<sup>6</sup> Ver em anexo o esquema ligações inicial

## ➤ PLACA INFERIOR

Este sistema vai ser alimentado através de alimentação externa, através de um transformador com uma tensão de 9V a.

Há no entanto que ter em atenção os níveis de tensão a que funcionam os vários dispositivos.

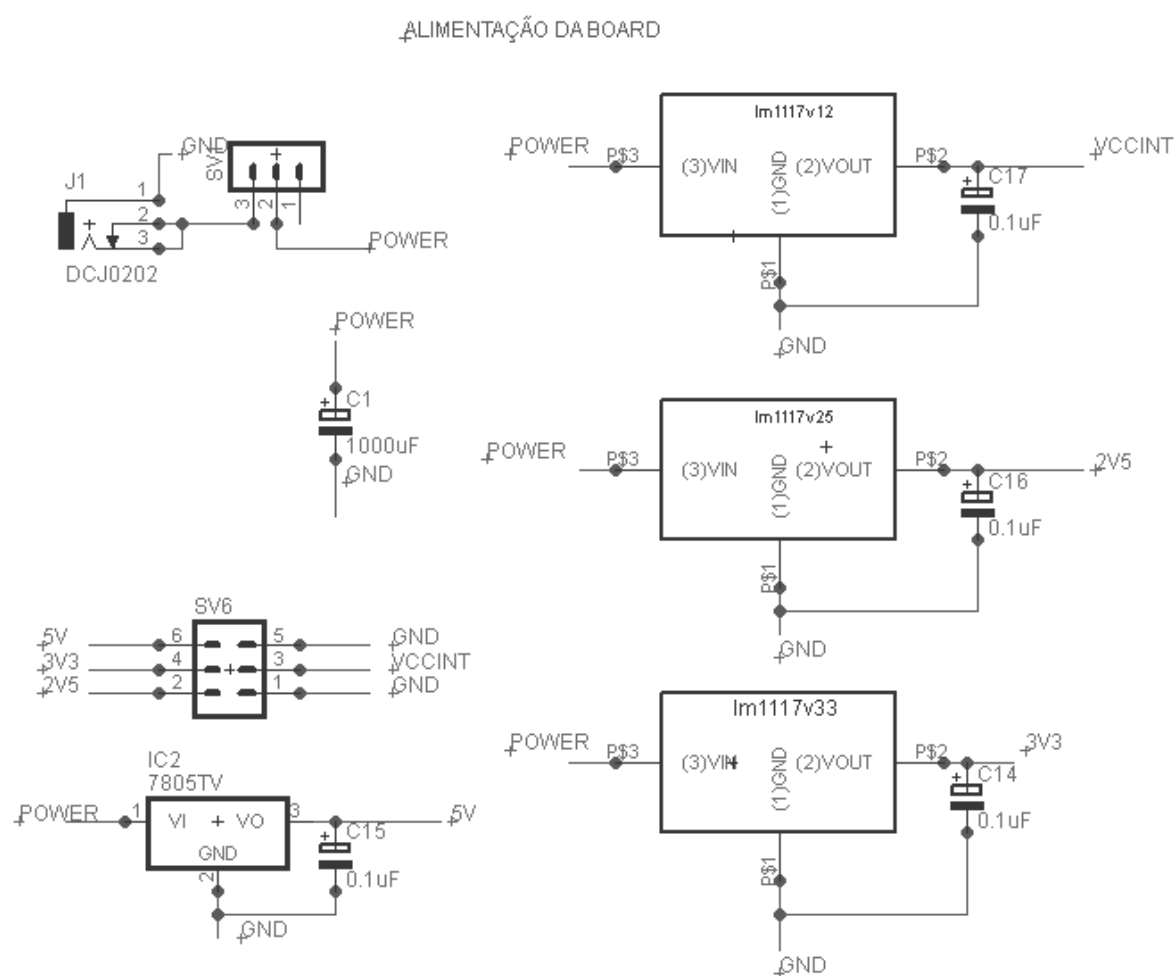
A FPGA é um dispositivo e baixo consumo já que funciona a níveis de tensão reduzidos. A lógica interna ( $V_{CCINT}$ ) suporta níveis de tensão que vão de -0,5V a 1,8V. Os buffers de saída ( $V_{CCIO}$ ) já funcionam a níveis de tensão mais elevados, suportam tensão desde -0,5V a 3,9V. Existe ainda outros pinos que necessitam de ser alimentados, sendo eles os pinos de alimentação do regulador do PLL ( $V_{CCA}$ ), que suportam tensões que vão desde -0,5V a 3,75V. [15]

Após consulta do *datasheet* do dispositivo EP3C5E144C7 e do documento *Cyclone III Family Pin Connection Guidelines*, chegou-se aos níveis de tensão aconselháveis seriam de 1,2V para  $V_{CCINT}$  e 2,5V para  $V_{CCA}$ . Como os pinos de saída suportam tensões mais elevadas de forma a poder suportar as normas de I/O mais comuns. Como a FPGA possui 8 bancos de saídas, poderia configurar-se os vários bancos para suportarem diferentes tensões, e por conseguinte diferentes normas de I/O, mas como isso poderia criar confusão aos utilizadores deste sistema didáctico, optou-se por manter um nível de tensão para todas as saídas. Sendo assim o nível de tensão das saídas ( $V_{CCIO}$ ) seleccionado foi de 3,3V, que é compatível com as tecnologias *Low Voltage Complementary Metal Oxide Semiconductor (LVCMOS)* e *Low Voltage Transistor-Transistor Logic (LVTTL)*. Posteriormente, se se pretender que as saídas deste kit suportem níveis de tensão mais elevados, pode-se conceber uma carta de expansão para esse efeito. [15]

Como o PIC vai estar conectado a FPGA, convém que funcione ao mesmo nível de tensão das suas saídas, o que não apresenta problema já que ele suporta uma tensão de alimentação ( $V_{DD}$ ) que vai desde os 2,5V a 5V, o que é perfeitamente compatível com os 3,3V a que vão funcionar as saídas da FPGA. [17]

Depois de definidos parâmetros de alimentação dos dispositivos, concebeu-se o bloco de alimentação da placa de circuito impresso, representado na figura 81.

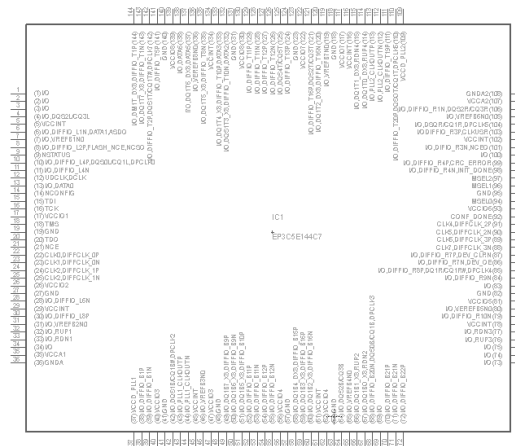
Por questões de familiarização com o software *Easily Applicable Graphical Layout Editor* (EAGLE), o circuito eléctrico foi realizado na versão 6.1 desta ferramenta.



**Figura 81 Bloco de alimentação da board.**

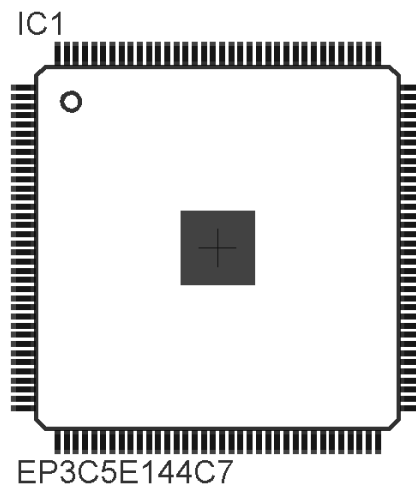
Este sistema tem então que ser alimentado a partir de alimentação externa, através de um transformador com tensão de 9 a 12V.

Como o EAGLE não possui o símbolo da FPGA, teve q ser criada uma livreria nova, com o símbolo deste componente, representado na figura 82<sup>7</sup>.



**Figura 82 Símbolo da FPGA EP3C5E144C7 da Altera.**

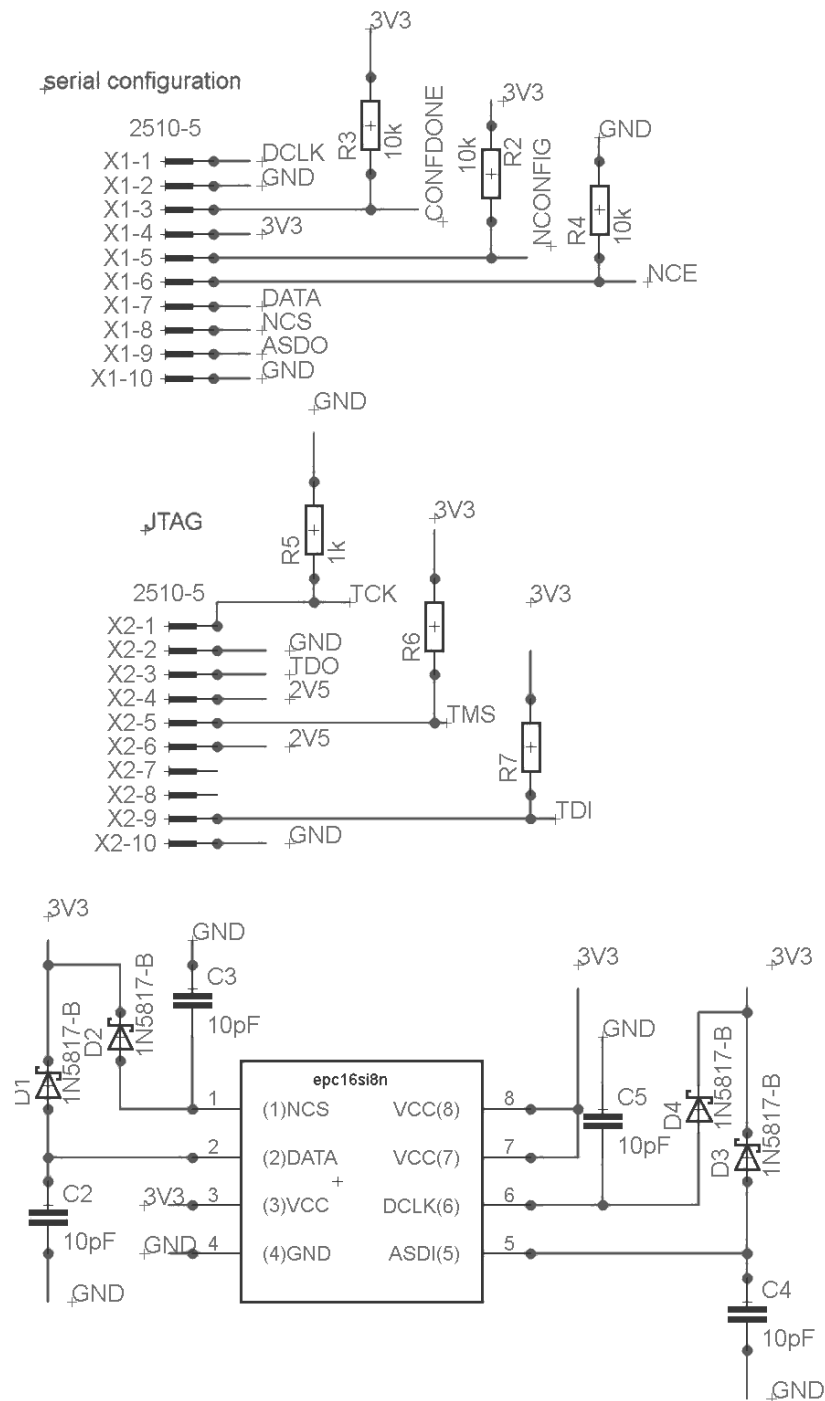
O correspondente símbolo do encapsulamento EQFP de 144 da FPGA, a utilizar na concepção da placa de circuito impresso, encontra-se representado na figura 83.



**Figura 83 Encapsulamento EQPF de 144 pinos da FPGA EC3P5E144C7.**

<sup>7</sup> O símbolo encontra-se em maior pormenor em anexo

O próximo passo foi criar o circuito de configuração da FPGA, que como foi referido anteriormente foi referido inclui um porto JTAG e uma memoria Flash de inicialização serie, representados na figura 84.



**Figura 84** Circuito de configuração da FPGA.

Como se pode observar na figura acima representada, o circuito de configuração é constituído por um porto JTAG, e uma ficha para programar a memória serie EPCS16SI8N. Este esquema engloba uma ficha de configuração AS que é usada para configurar a memória Flash, e uma JTAG, baseado no apresentado no *Cyclone III Device Handbook*.

Para a FPGA saber o modo de configuração a que vai ser sujeita é necessário levar ao calor lógico “1”ou “0” os pinos MSEL da FPGA. O modo de configuração escolhido foi o *Fast Active Serial Standard* ilustrado na tabela 3, abaixo apresentada.

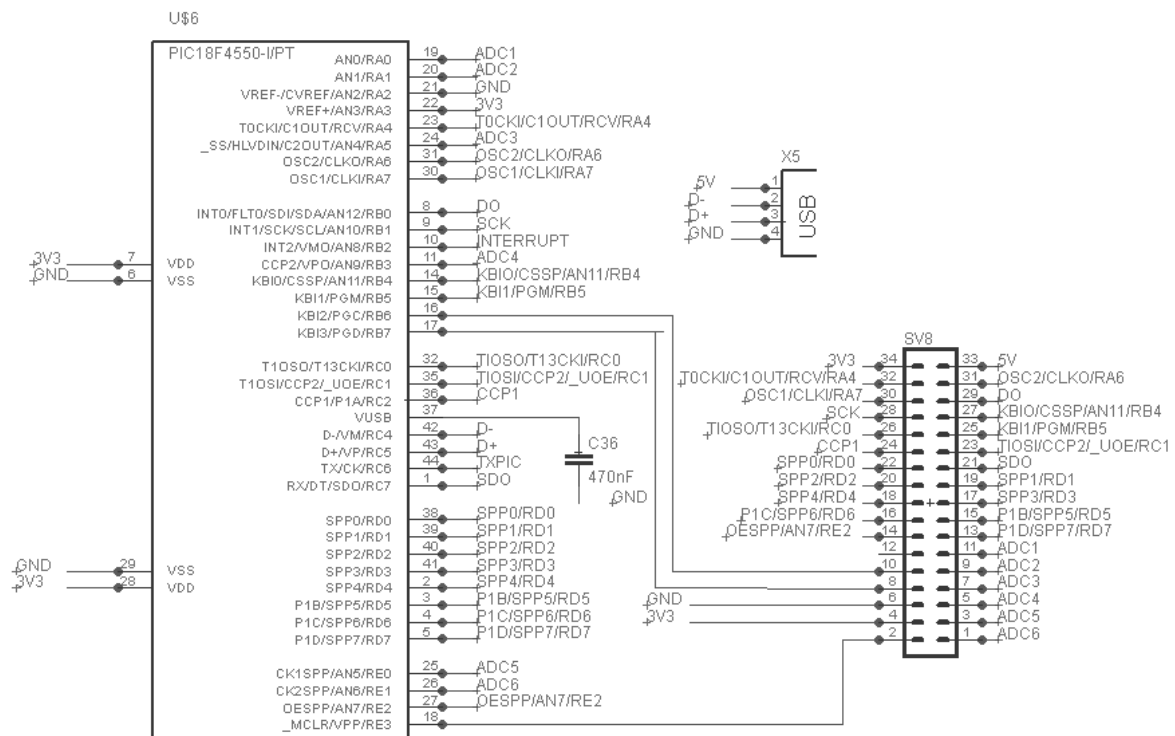
Configuration Scheme	MSEL				Configuration Voltage Standard (V) (2), (3)
	3	2	1	0	
Fast Active Serial Standard (AS Standard POR)	0	0	1	0	3.3
Fast Active Serial Standard (AS Standard POR)	0	0	1	1	3.0/2.5
Fast Active Serial Fast (AS Fast POR)	1	1	0	1	3.3
Fast Active Serial Fast (AS Fast POR)	0	1	0	0	3.0/2.5
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	0	1	1	1	3.3
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	1	0	1	1	3.0/2.5
Active Parallel ×16 Standard (AP Standard POR, for Cyclone III devices only)	1	0	0	0	1.8
Active Parallel ×16 Fast (AP Fast POR, for Cyclone III devices only)	0	1	0	1	3.3

**Tabela 3** Tabela de configuração dos pinos MSEL da FPGA. [15]

Como a memória vai ser alimentada a 3,3V a configuração a fazer corresponde à primeira linha da coluna.

Como foi referido anteriormente o PIC vai ser o assistente da FPGA, e vai estar responsável por escrever no cartão SD, fazer comunicação USB com o exterior, e ler valores analógicos mas suas entradas.

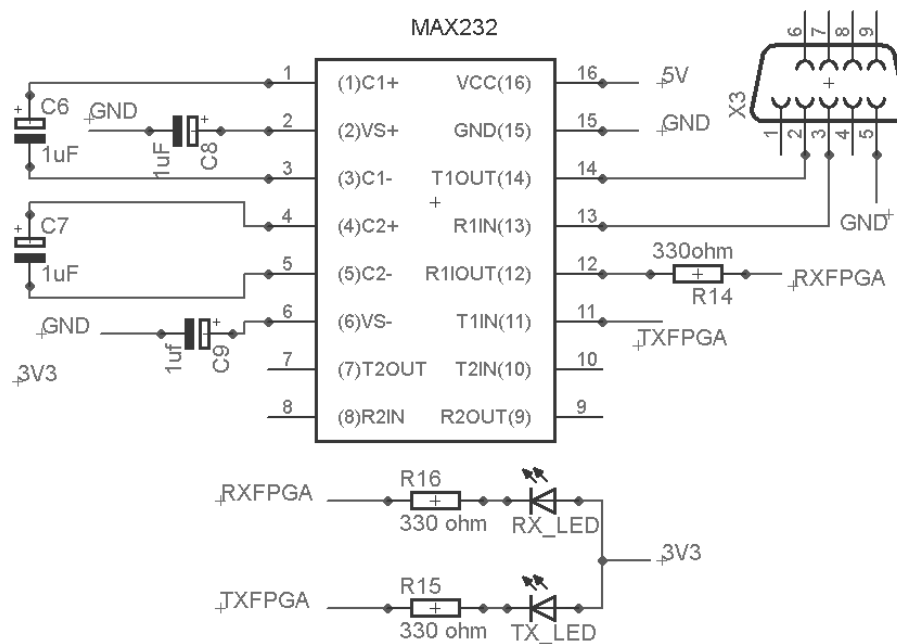
Na figura 85 está representado o esquema de ligações do PIC.



**Figura 85 Esquema de ligações do PIC.**

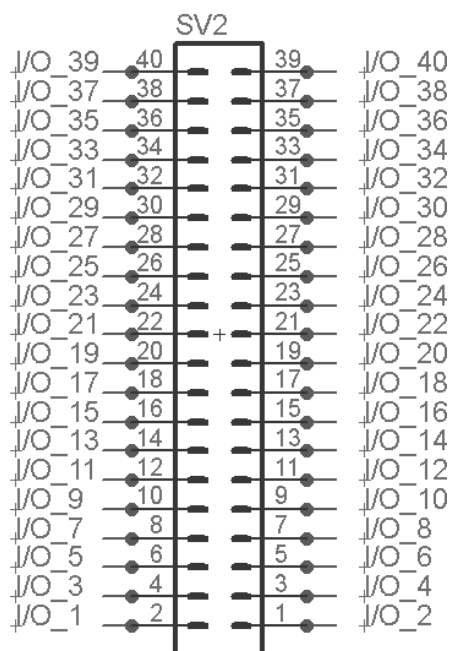
Como o cartão SD se vai encontrar na placa de circuito impresso de cima, a conexão do microcontrolador com o cartão SD vai ser feita através de um pente fêmea de 34 pinos representado na figura. Este pente também inclui os pinos de programação do pickit e entradas analógicas do PIC, bem como pinos para permitir a expansão das funcionalidades do microcontrolador.

O último tópico a abordar na concepção do sistema é a ligação dos periféricos de I/O do sistema à FPGA. Como foi referido anteriormente a maior parte deles vai estar colocado na placa superior, e interligam com a FPGA através de um *piggyback*. O único periférico que fica na placa inferior, para além da ficha USB, é o conector RS232, cujo esquema de ligações esta representado na figura 86.



**Figura 86** Esquema de ligações do conector RS232.

As restantes saídas da FPGA estão disponíveis para o utilizador através de um pente macho de 40 pinos, representado na figura 87.

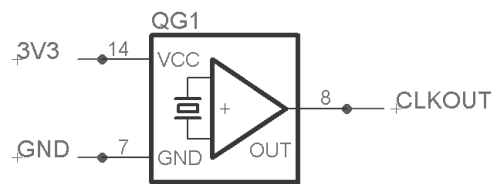


**Figura 87** Saídas disponíveis para o utilizador.



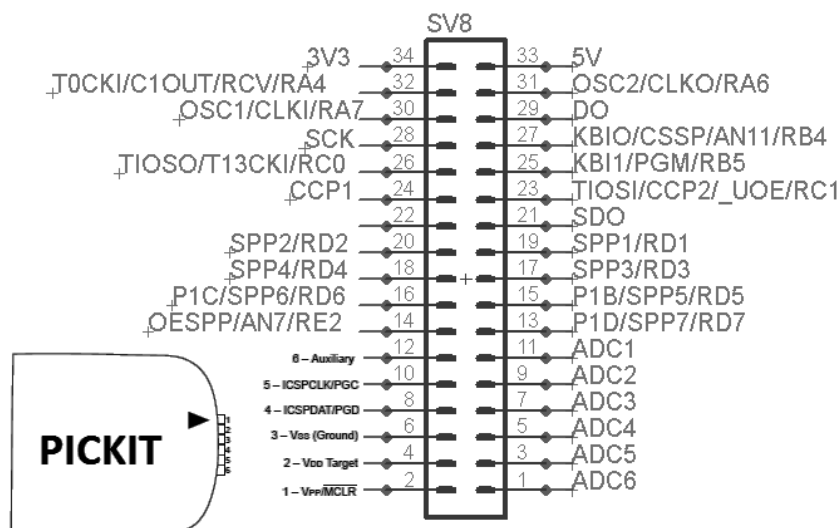
Estas saídas vão funcionar a uma tensão de 3,3V e poderão no máximo debitar uma corrente de 40mA.

O *clock* do sistema vai ser gerado por um oscilador que pode ser trocado, já que este vai encaixar num *socket* em vez de ser soldado na placa. Para além disso o sistema permite uma entrada de *clock* externo, possuindo um botão para seleccionar entre o *clock* proveniente do oscilador e o *clock* externo.



**Figura 88 Circuito do oscilador.**

O *piggyback* é efectuado em 2 pentes machos. Um deles leva os sinais que estão ligados ao PIC para a placa superior, tendo também nele incluídos os pinos de programação microcontrolador, como está representado na figura 89.



**Figura 89 Pente macho que interliga o PIC com a placa superior, com porto de programação incluído.**

O segundo pente macho (figura 90) é usado para interligar a FPGA com os periféricos existentes na placa superior.

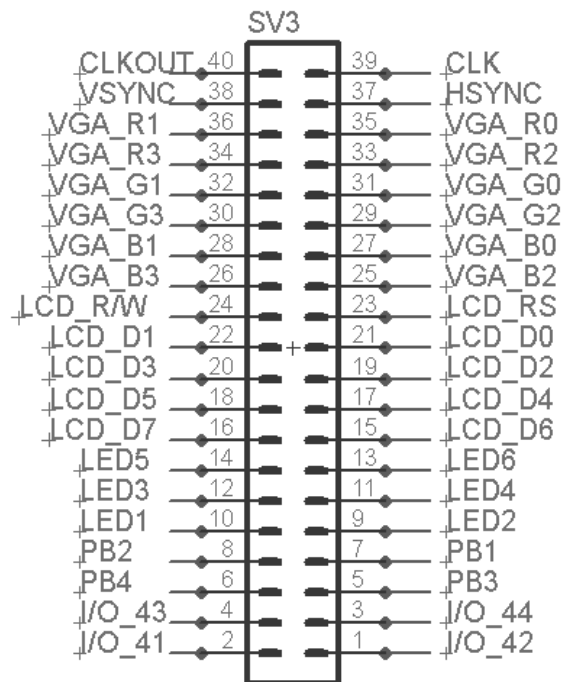
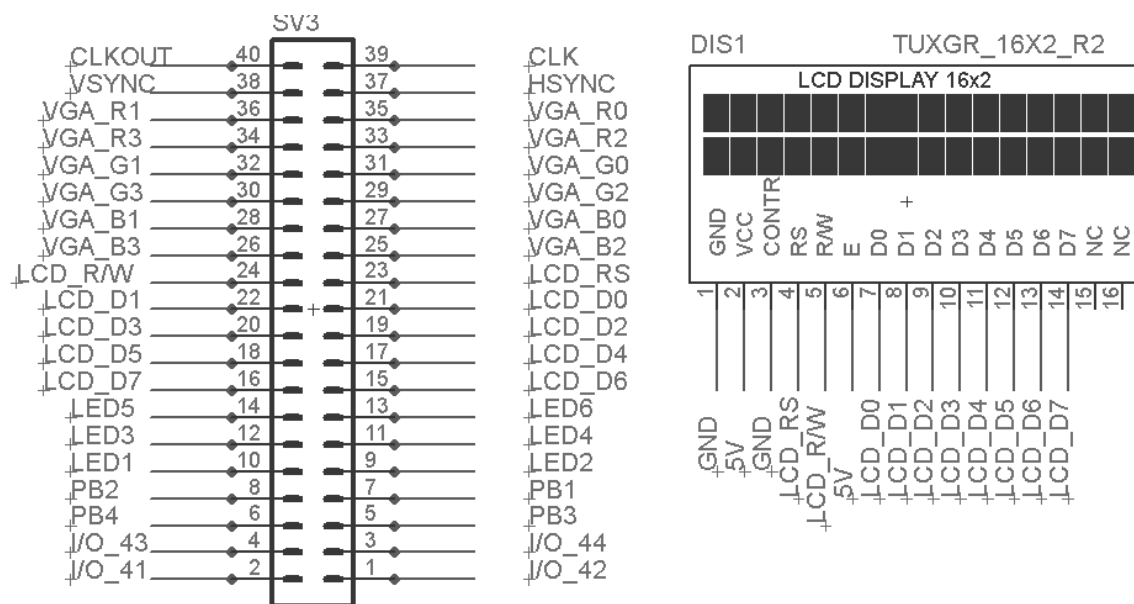


Figura 90 Pente macho q interliga a FPGA com os periféricos da placa superior.

## ➤ PLACA SUPERIOR

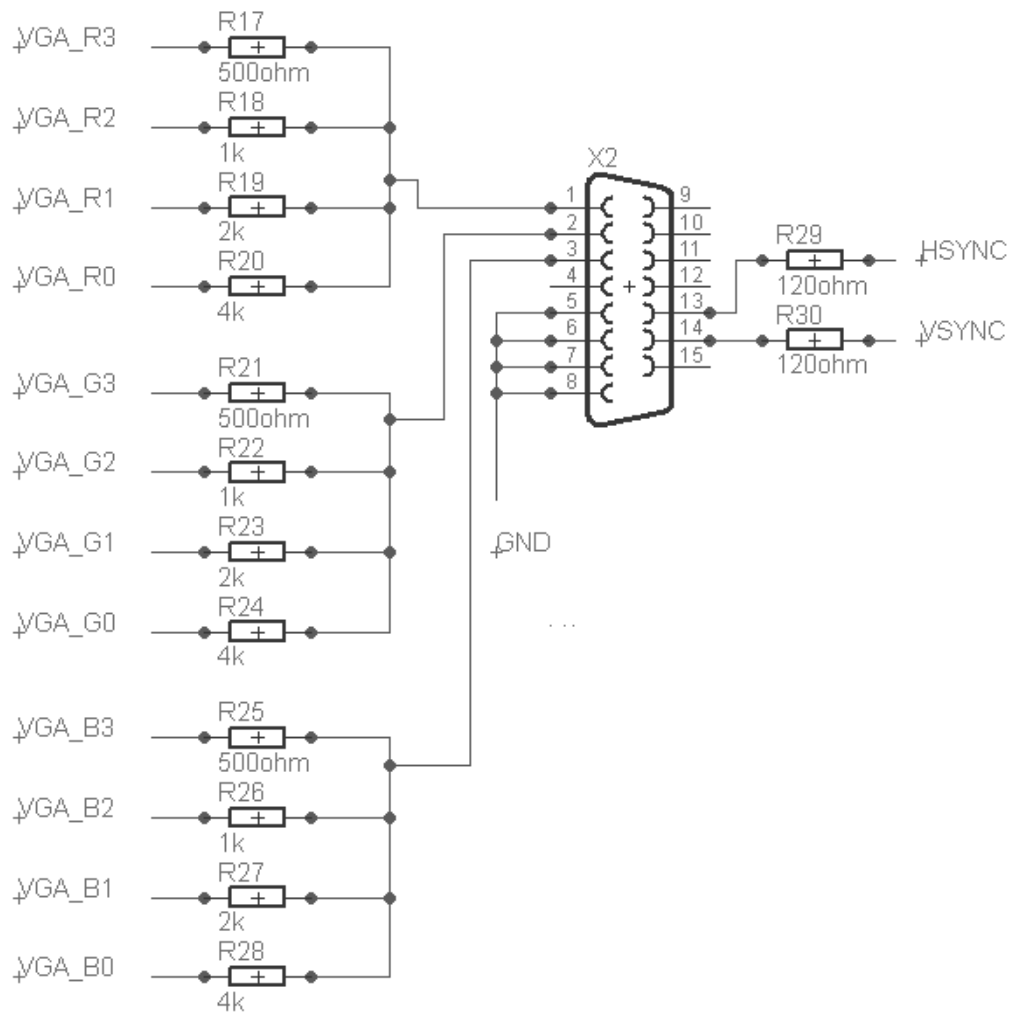
A placa superior é praticamente constituída por periféricos de I/O. Nela estão englobados Um LCD, uma ficha VGA, um *slot* de cartão SD, botões de pressão e LED's.

O LCD liga directamente a FPGA através do *piggyback* anteriormente referido, como se pode observar na figura 91.



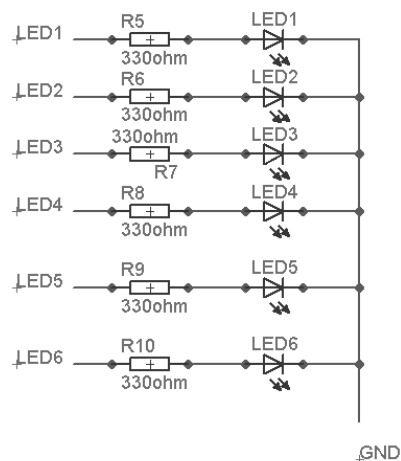
**Figura 91 Esquema de ligação do LCD.**

A conexão com a ficha VGA também é efectuada através do pente ilustrado na figura acima apresentada, no entanto neste caso a FPGA não liga directamente ao conector, já que tem existir pelo meio uma conversão de sinais digitais em sinais analógico. Esta conversão é efectuada através de DAC's, que por opção são implementados através de resistências, como se pode observar na figura 92.



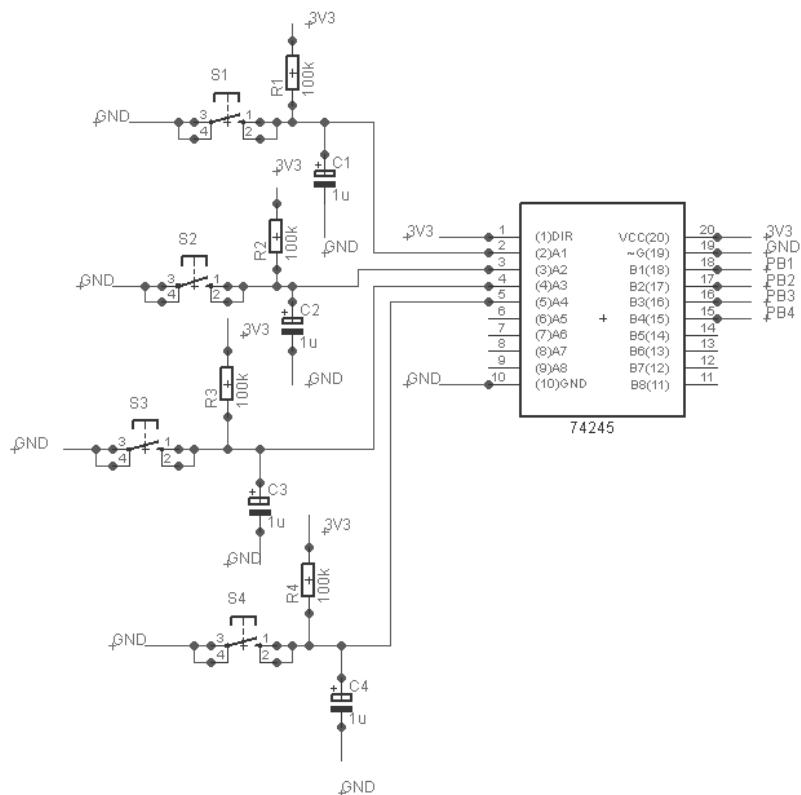
**Figura 92 Esquema de ligações do conector VGA.**

A FPGA encontra-se ainda conectada a 8 LED's de 3mm que podem ser configurados pelo utilizador, cujo esquema de ligações se encontra representado na figura 93.



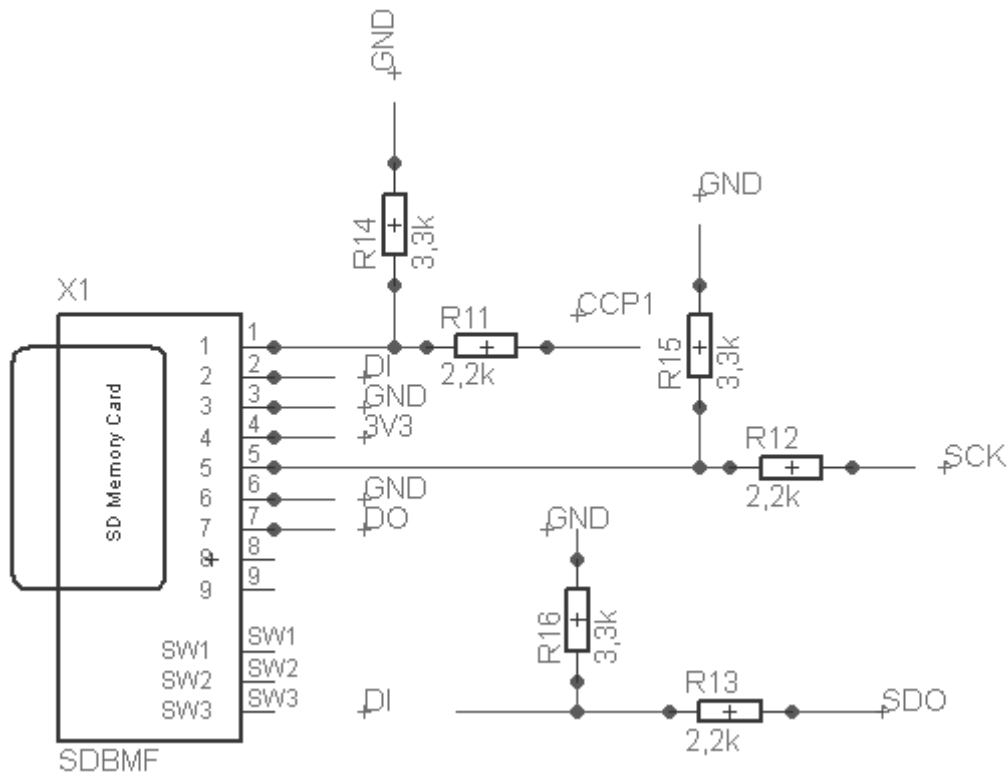
**Figura 93** Esquema de ligações do LED's.

Os últimos periféricos que se encontram ligados a FPGA são 4 botões de pressão com um *Schmitt Trigger* associado, como se pode contactar na figura 94.



**Figura 94** Esquema de ligações dos botões de pressão.

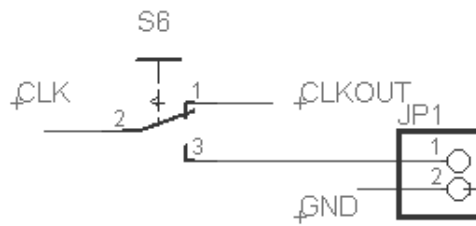
O sistema possui ainda uma *slot* de cartão SD que é controlado pelo PIC, como já foi várias vezes referido. A conexão deste periférico com o microcontrolador é feita através do pente que está conectado ao PIC através de *piggyback*, como se pode observar na figura 95.



**Figura 95** Esquema de ligações cartão SD.

Neste caso o cartão SD vai comunicar com o microcontrolador através do protocolo serie SPI. [18]

Como referido anteriormente, o sistema também permite fazer a selecção de qual será a fonte de *clock* do sistema, permitindo escolher entre o oscilador *onboard* ou uma fonte de *clock* externa. Na figura 96 encontra-se representado o mecanismo de selecção de *clock*.



**Figura 96 Mecanismo de selecção de *clock*.**

#### **4.2.2. PLACA DE CIRCUITO IMPRESSO**

Como foi anteriormente referido, inicialmente estava pensado conceber uma placa com a dimensão de 16cm×10cm<sup>8</sup>, mas optou-se por dividir o sistema em dois.

Depois da implementação do esquema eléctrico do sistema, pode então dar-se o passo seguinte, que seria a concepção das placas de circuito impresso. O projecto destas placas foi um processo demorado e penoso, devido a limitações impostas, quer de espaço para cada placa de circuito impresso quer do ao número de camadas usadas na sua concepção. No entanto, apesar da dificuldade, foi possível incluir todos os componentes e dispositivos pretendidos espaço disponível.

##### **➤ PLACA INFERIOR**

A placa inferior, como já fora referido, é o cérebro deste sistema, por conseguinte foi a mais trabalhosa de conceber. As principais dificuldades surgiram, como era esperado, nas ligações da FPGA, isto devido ao encapsulamento da FPGA e ao facto de só terem sido usadas duas camadas, ou seja, a placa concebida era apenas de dupla face.

Na figura 97 está representado o aspecto final da placa inferior do sistema.

---

<sup>8</sup> O *layout* da PCB inicial também se encontra disponível em anexo

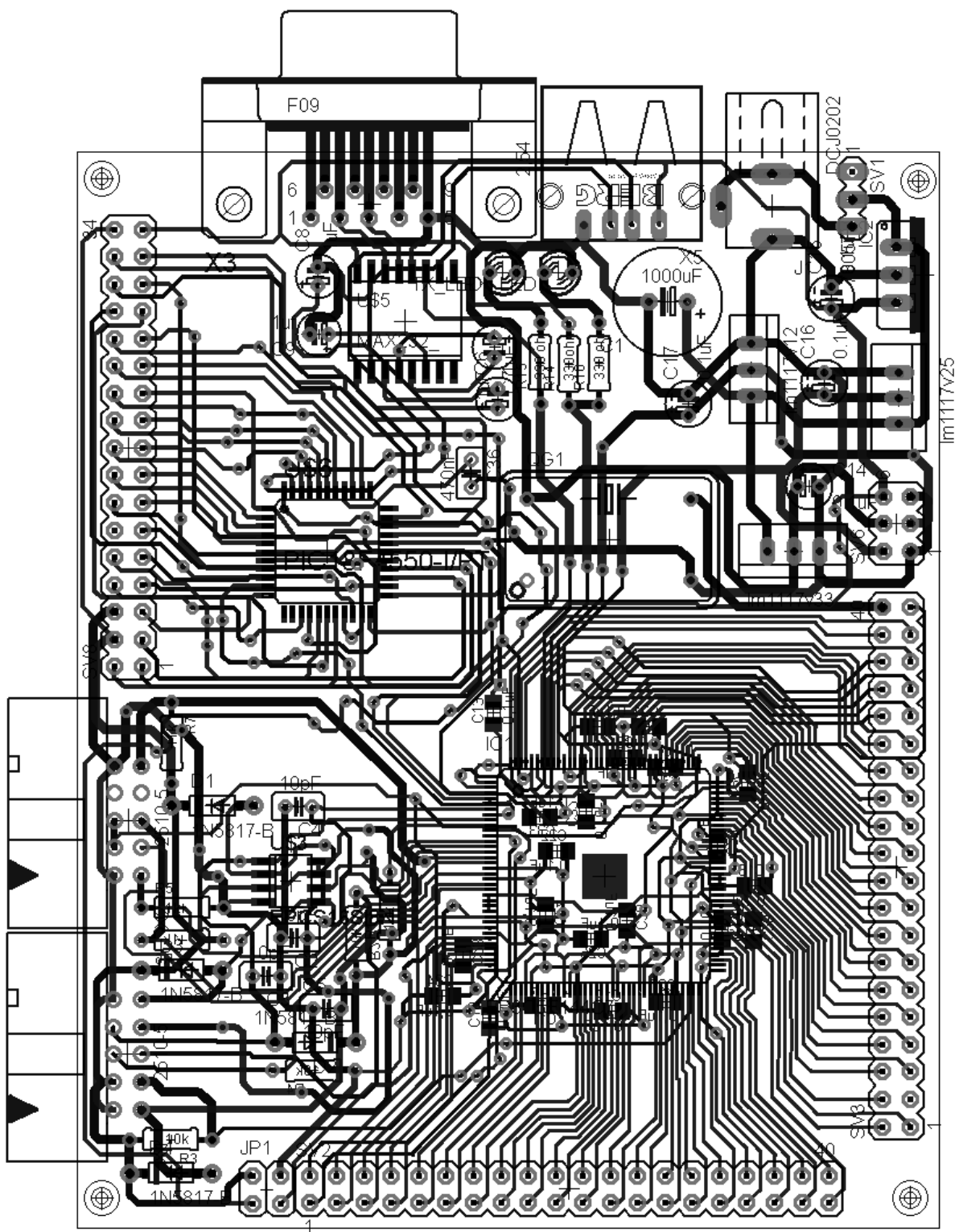
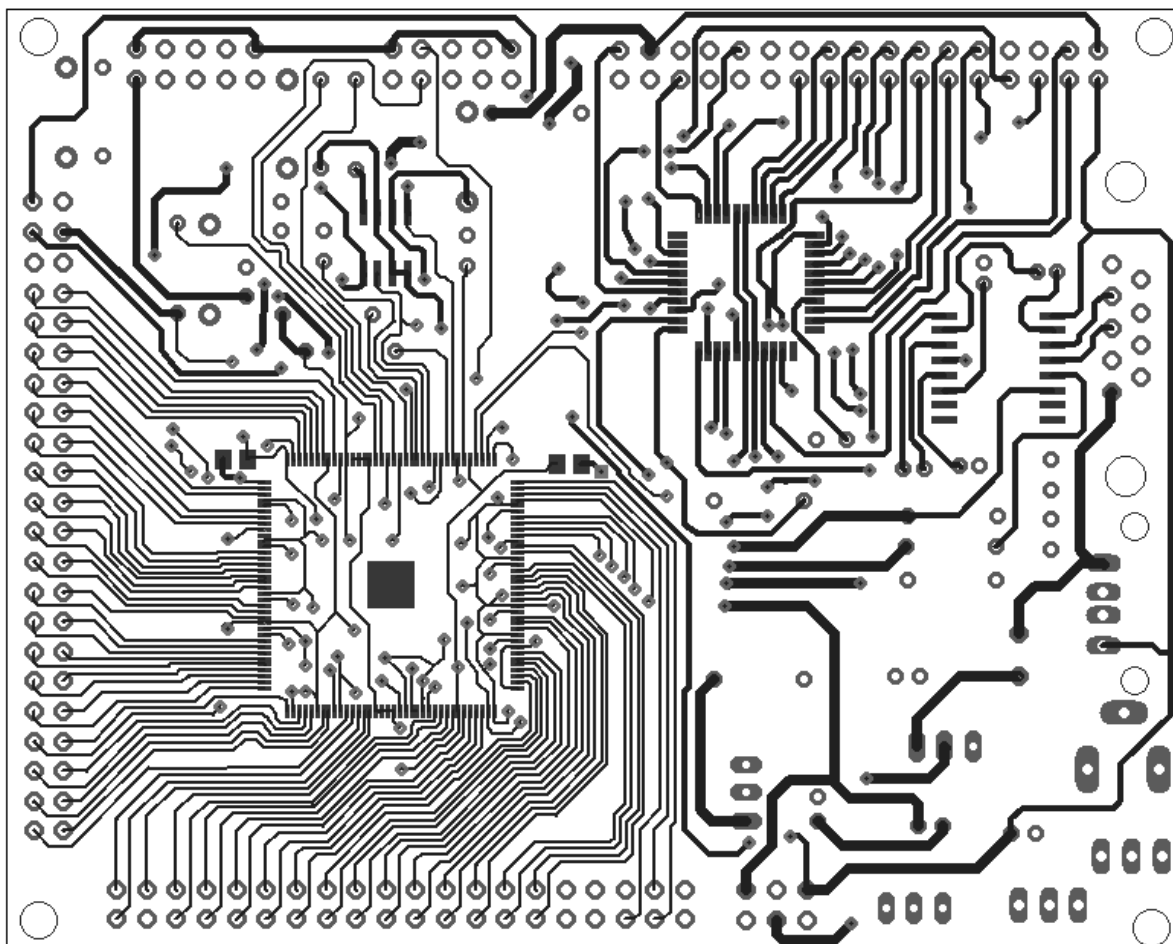


Figura 97 Aspecto final da placa inferior do sistema.

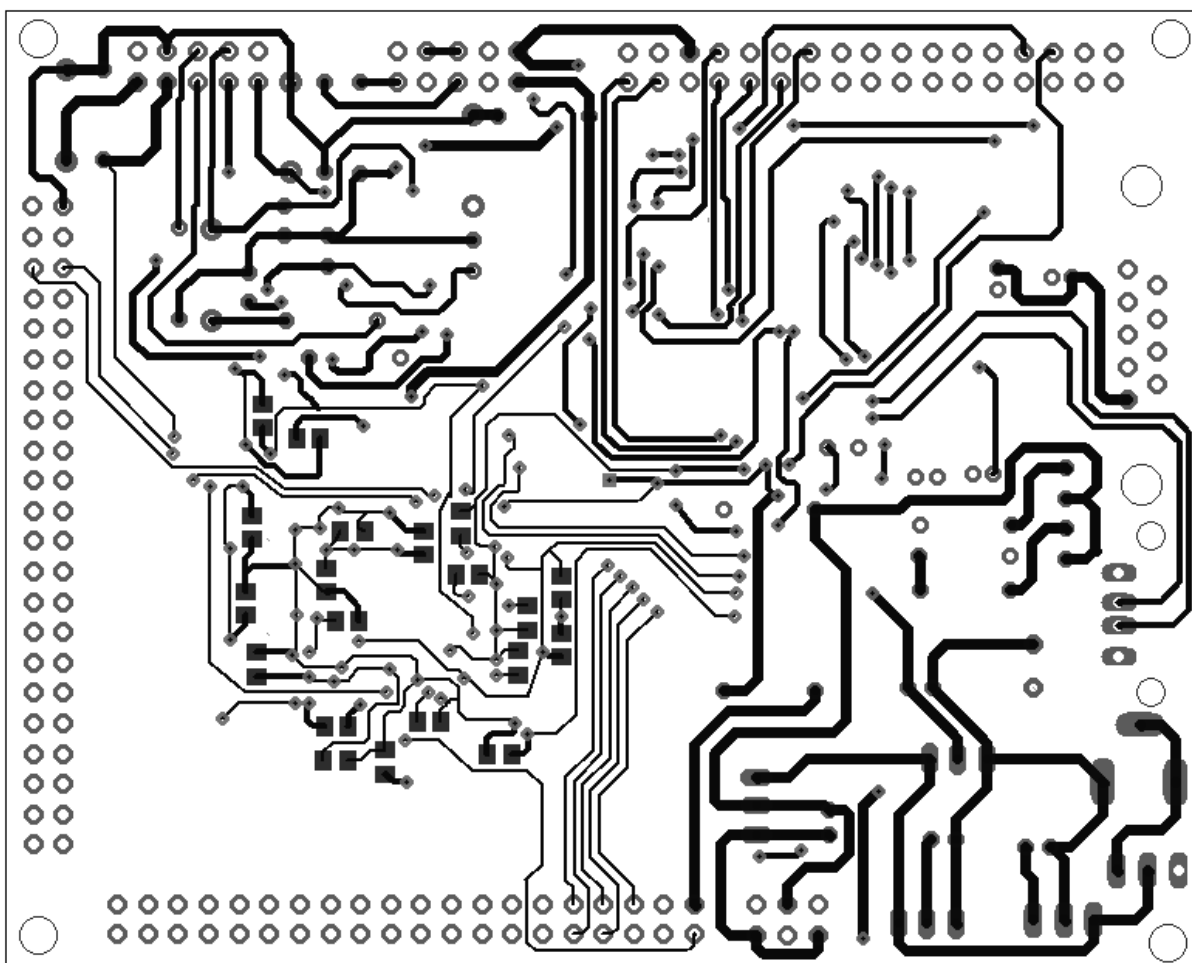


Como já fora referido anteriormente, a placa de circuito impresso foi concebida com apenas 2 faces. Na figura 98 está representada a face de cima da placa.



**Figura 98** *Layout da face de cima (TOP) da placa de circuito impresso inferior.*

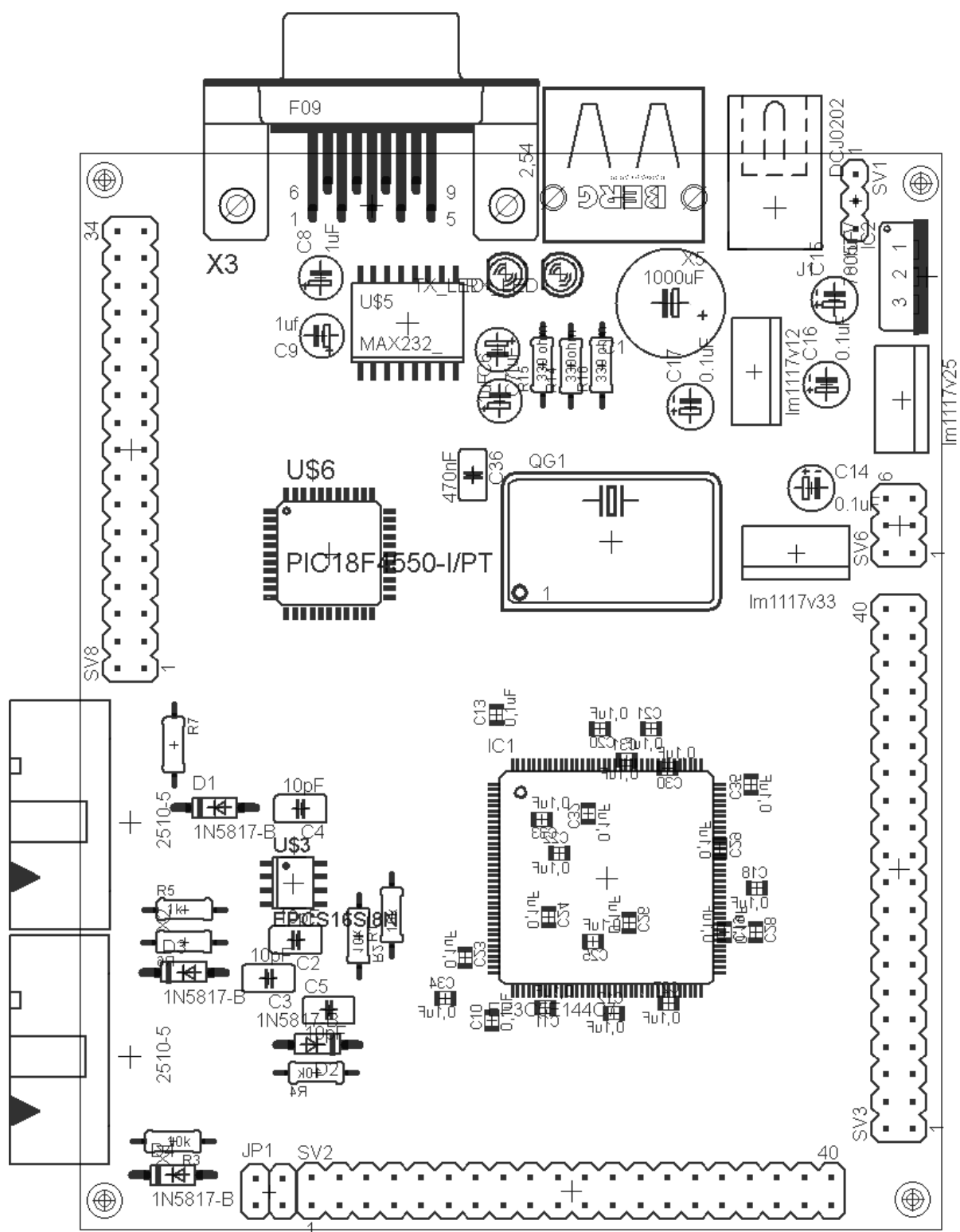
E na figura 99 esta representada a face de baixo da placa de circuito impresso.



**Figura 99** *Layout* da face de baixo (BOTTOM) da placa de circuito impresso inferior.

Como se pode constatar a face de cima tem um maior número de ligações. Isto deve-se ao facto a maior partes dos componentes serem SMD, por opção de projecto, já que a versão SMD de um componente é bastante mais reduzida.

Na figura 100 consegue-se visualizar de forma melhor os componentes usados na placa inferior.



**Figura 100** Componentes da placa de circuito impresso inferior.

Na tabela 4, abaixo apresentada, estão discriminados os componentes usados na concepção da placa inferior, para se poder ter uma melhor percepção do que eles representam.

Componente	-	Descrição
IC1	-	FPGA EP3C5E144C7N
U6	-	PIC 18F4550 de encapsulamento TQFP 44 pinos
U3	-	Memória de inicialização da FPGA EPCS16SI8N de 16Mb
U5	-	MAX 232 encapsulamento SOIC de 16 pinos
LM1117V33	-	Regulador de tensão de 3,3V
LM1117V25	-	Regulador de tensão de 2,5V
LM1117V12	-	Regulador de tensão de 1,2V
IC2	-	LM7805 regulador de tensão de 5V
X1 e X2	-	Ficha IDC macho de 10 pinos para soldar na placa de circuito impresso
X3	-	Ficha DB9 macho para soldar na placa de circuito impresso
X5	-	Ficha USB tipo A para soldar na placa de circuito impresso
J1	-	Ficha de Transformador
C1	-	Condensador de 1000 $\mu$ F 16V
C2 a C5	-	Condensadores cerâmicos de 10pF
C6 a C9	-	Condensadores electrolíticos de 1 $\mu$ F 16V
C10 a C13	-	Condensador SMD cerâmico de 0,1 $\mu$ F
C14 a C17	-	Condensador electrolítico de 0,1 $\mu$ F 16V
C18 a C35	-	Condensador SMD cerâmico de 0,1 $\mu$ F
C36	-	Condensador ceramico de 470pF
R1 a R4	-	Resistências de 1/4 W de 10K $\Omega$
R5 a R7	-	Resistências de 1/4 W de 1K $\Omega$
R14 a R15	-	Resistências de 1/4 W de 330 $\Omega$
D1 a D4	-	Diodo 1N5517B
RX_LED	-	LED de 3mm vermelho
TX_LED	-	LED de 3mm verde
QG1	-	Oscilador de 50MHz
SV1	-	Pente macho de 3 pinos
SV2 e SV3	-	Pente macho duplo de 40 pinos
SV6	-	Pente macho duplo de 6 pinos
SV8	-	Pente macho duplo de 34 pinos
JP1	-	Pente macho duplo de 4 pinos

**Tabela 4** Lista de componentes da placa inferior.

## ➤ PLACA SUPERIOR

Como referido anteriormente, a placa superior vai ser constituída apenas com periféricos do sistema. Na figura 101 encontra-se representado o aspecto final da placa superior.

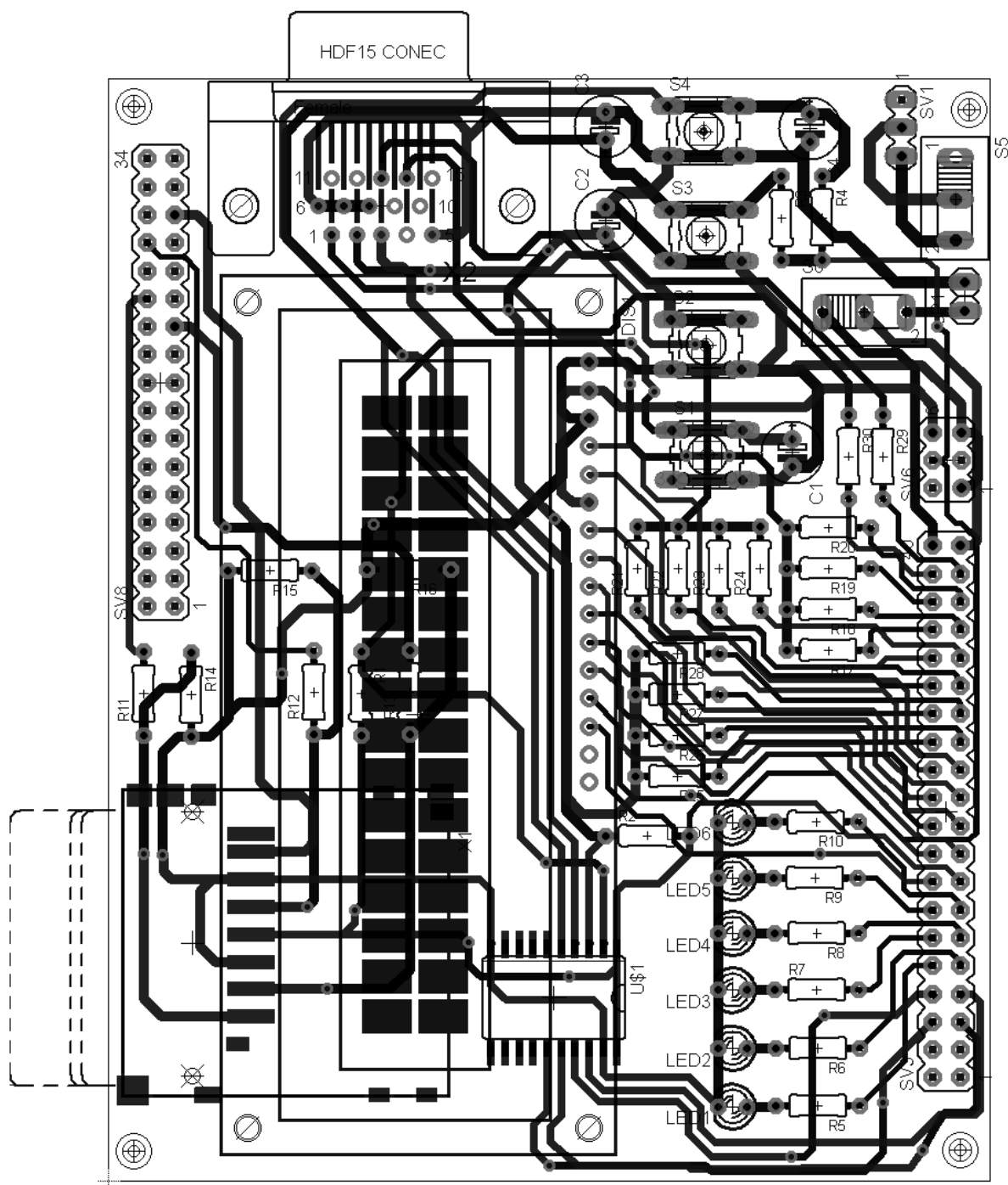
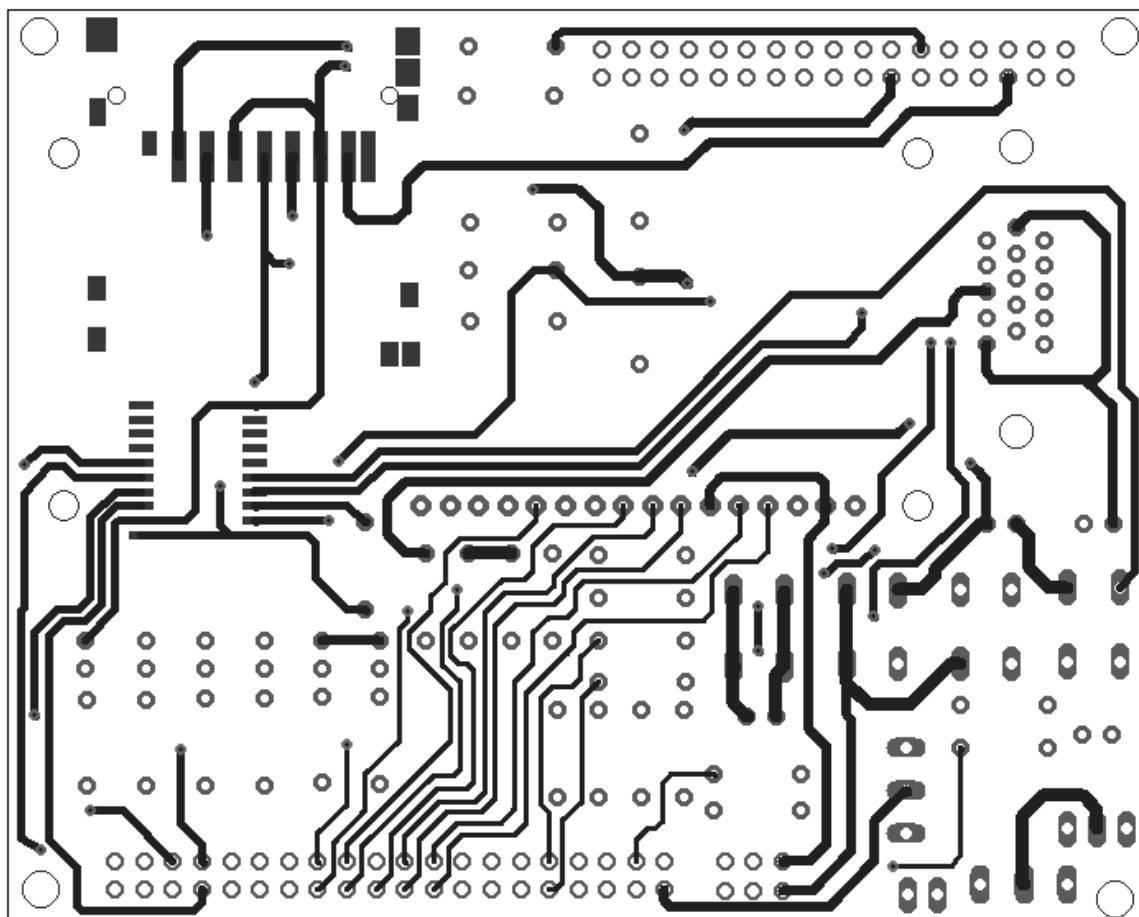


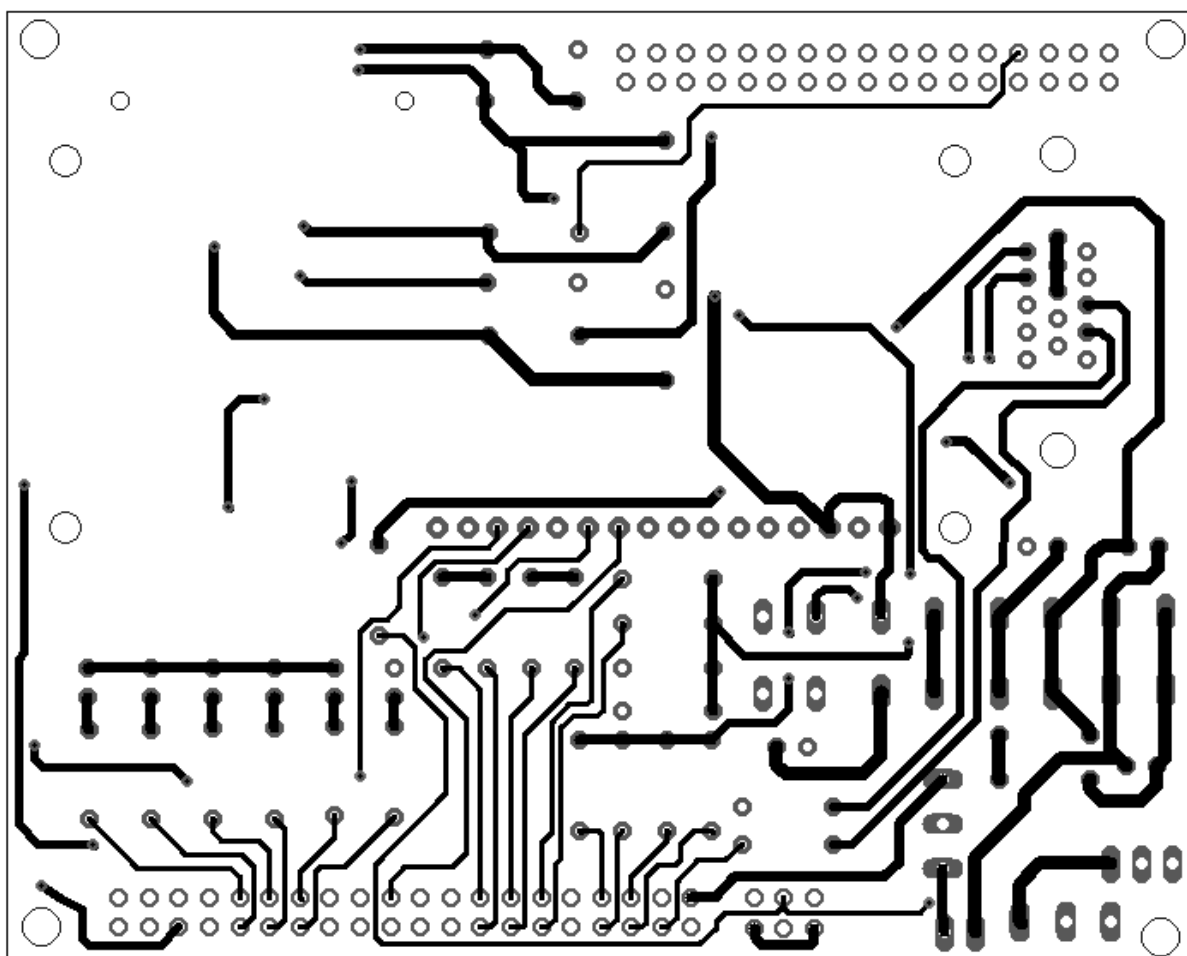
Figura 101 Aspecto final da placa superior.

Assim como a placa inferior, a superior também foi concebida só com duas faces, encontrando-se a face de cima representada na figura 102.



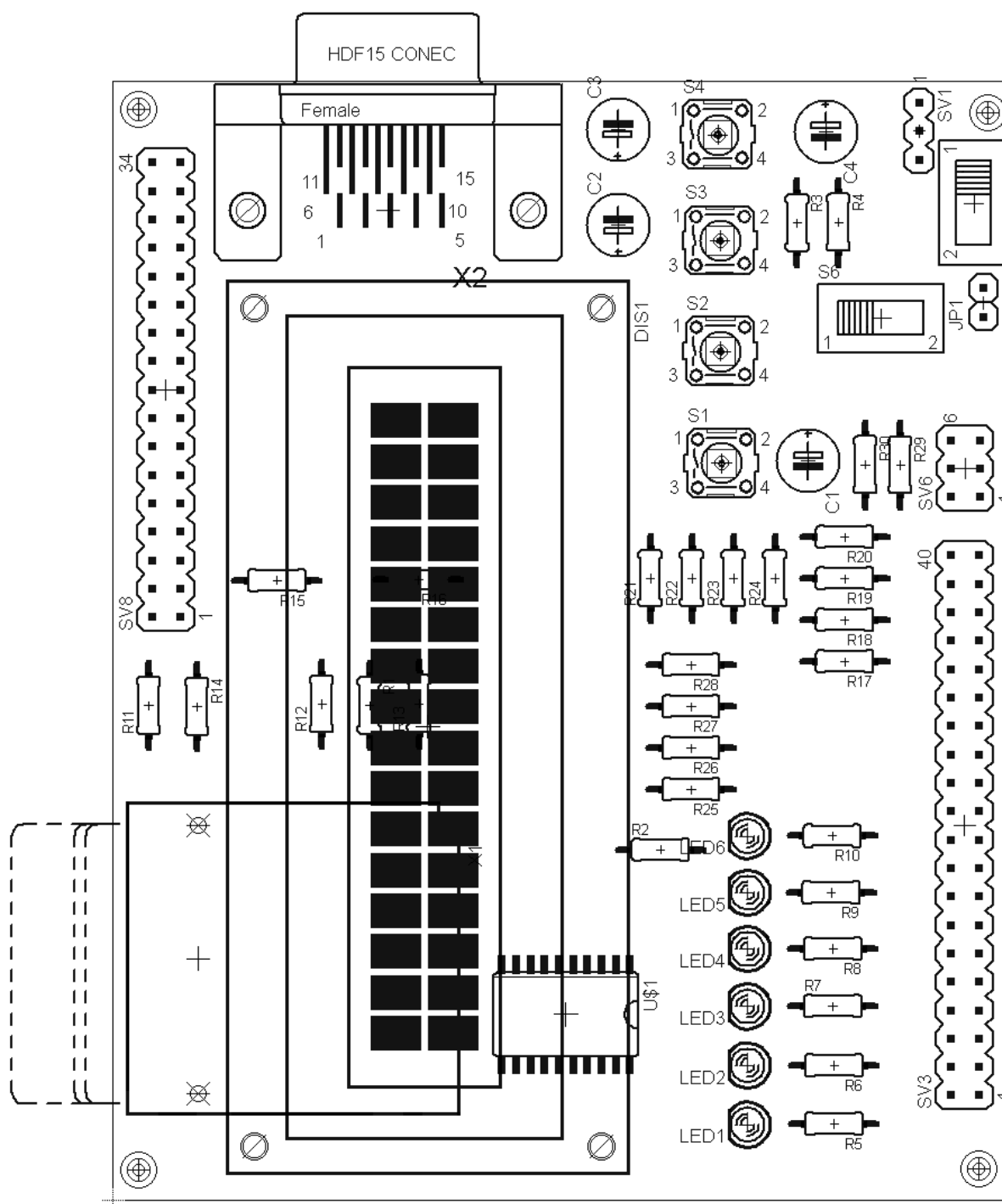
**Figura 102**      *Layout da face de cima (TOP) da placa de circuito impresso superior.*

Na figura 103 encontra-se ilustrada a face de baixo da placa superior do sistema.



**Figura 103**      *Layout da face de baixo (TOP) da placa de circuito impresso superior.*

A figura 103 ilustra, de forma mais clara aos componentes utilizados na placa superior.



**Figura 104** Componentes da placa de circuito impresso superior.

Na tabela 5, abaixo apresentada, estão discriminados os componentes usados na concepção da placa inferior, para se poder ter uma melhor percepção do que eles representam.



Componente	-	Descrição
74245	-	3 state octal buffer transceiver
C1 a C4	-	Condensadores electrolíticos de 1 $\mu$ F 16V
R1 a R4	-	Resistências de 1/4 W de 100K $\Omega$
R5 e R10	-	Resistências de 1/4 W de 330 $\Omega$
R11 a R13	-	Resistências de 1/4 W de 2,2K $\Omega$
R14 a R16	-	Resistências de 1/4 W de 3,3K $\Omega$
R17,R21,R25	-	Resistências de 1/4 W de 500 $\Omega$
R18,R22,R26	-	Resistências de 1/4 W de 1K $\Omega$
R19,R23,R27	-	Resistências de 1/4 W de 2K $\Omega$
R20,R24,R28	-	Resistências de 1/4 W de 4K $\Omega$
R20 e R30	-	Resistências de 1/4 W de 120 $\Omega$
J1	-	Pente fêmea de 2 pinos
S5 e S6	-	Interruptores de deslizar de 3 pinos e 2 posições
X1	-	<i>Slot</i> de cartão SD <i>Standard</i>
DIS1	-	LCD de 2 linhas e 16 caracteres
LED 1 a LED6	-	LED de 3mm verde
SV1	-	Pente fêmea de 3 pinos
SV3	-	Pente fêmea duplo de 40 pinos
SV6	-	Pente fêmea duplo de 6 pinos
SV8	-	Pente fêmea duplo de 34 pinos

**Tabela 5** Lista de componentes da placa superior do sistema.

#### 4.2.3. PROGRAMAÇÃO DOS DISPOSITIVOS DO SISTEMA

Este sistema é baseado em dois dispositivos programáveis distintos, a FPGA, a memória Flash de inicialização e o PIC, que usam linguagens diferentes, e por conseguinte ambientes de programação diferentes.

O PIC como já foi referido é programado através do cabo pickit. Na figura 105 está representada a versão 3 deste programador.



Figura 105 PICKIT III da Microchip.

A programação do microcontrolador é efectuada no *software* chamado de MPLAB ou então no *software* mikroC. Na figura 106 está representado um ambiente geral de trabalho do *software* MPLAB.

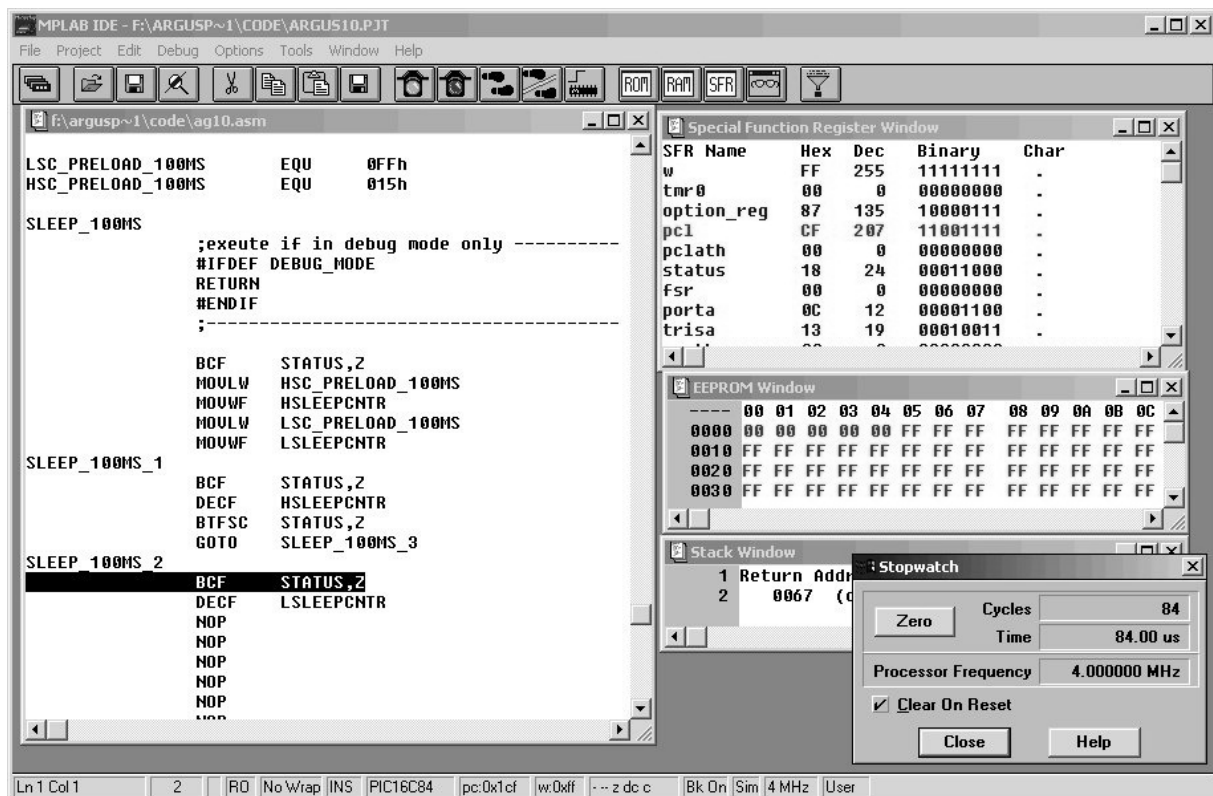
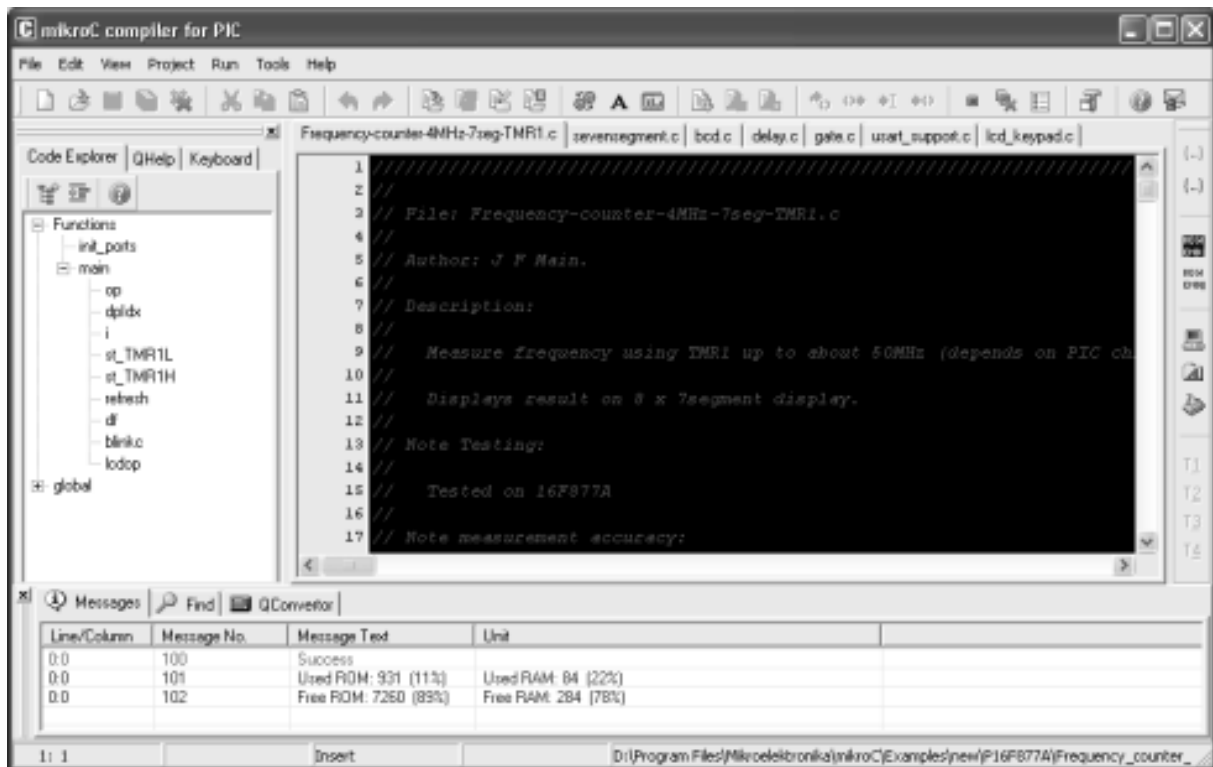


Figura 106 Ambiente de trabalho genérico do MPLAB.

O software mikroC é basicamente um compilado de C específico para os PIC e possui uma série de funções embudadas, incluindo umas dedicadas a controlar um cartão SD, o que é bastante relevante para este sistema. Na figura 107 está ilustrado ambiente base do mikroC.



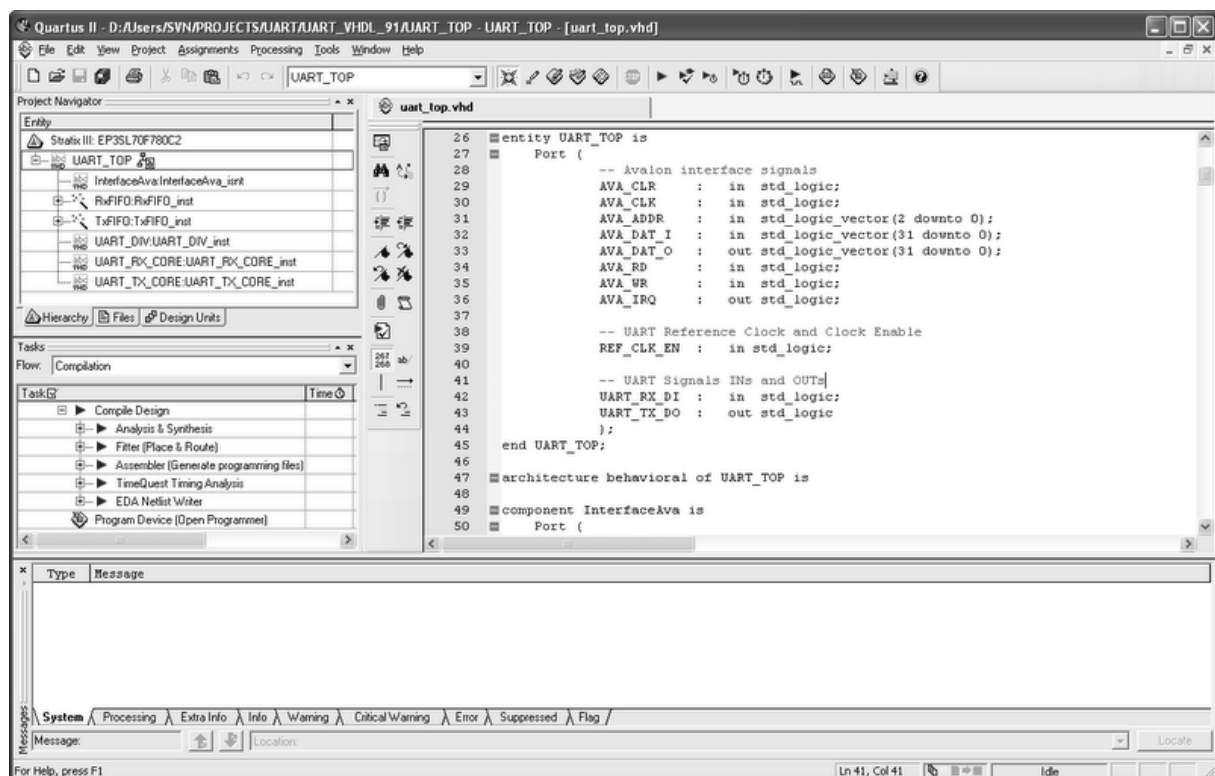
**Figura 107** Aspecto geral do compilador mikroC.

Tanto a FPGA como a memória de inicialização, podem ser programadas através de qualquer cabo do fabricante Altera, como por exemplo o USBBLATER apresentado na figura 108.



**Figura 108** USB-BLASTER da Altera.

O software utilizado para a programação destes dispositivos é o Quartus da Altera, ilustrado na figura 109.



**Figura 109** Aspecto genérico do software Quartus II.

A linguagem de programação utilizada para programar a FPGA e a memória de inicialização é o VHDL.

Este software de programação é extremamente poderoso, oferece uma serie de ferramentas, como a ferramenta *chip planner*, que permite dividir a FPGA em blocos e atribuir-lhe funções distintas. Estes blocos depois funcionam em paralelo, não existindo qualquer interferência entre eles.

#### **4.2.4. ESTUDO DE CUSTOS DE FABRICO DO SISTEMA DIDÁCTICO**

Como fazer só um protótipo é muito dispendioso, o estudo de custo de fabrico do sistema foi efectuado para um pedido de 10 sistemas didácticos, com o intuito de reduzir o custo de fabrico, com base numa encomenda em quantidade.

No que diz respeito ao fabrico de das placas de circuito impresso do sistema, foi efectuado um pedido de orçamento a empresa Circuitotal, sediada em Aveiro.

As especificações do pedido de orçamento foram:

- 20 Unidades
- Dimensões de 8cm×10cm
- Furos metalizados
- Mascara de soldadura

Em resposta ao pedido de orçamento, foi enviado um *email* com o custo total da encomenda e de cada unidade. O custo total da encomenda ficaria em 312,42€ com IVA incluído, ficando cada unidade ao preço de 15,621€. [19]

Na tabela 6 estão discriminados os custos de aquisição de todo o material necessário para concepção desta ferramenta.

Descrição	Qt.	Preço Unidade (€)	Total(€)
Placas de circuito impresso	20	15.621	312.42
FPGA EP3C5E144C7N	10	11.9493	119.493
PIC 18F4550 de encapsulamento TQFP 44 pinos	10	3.42078	34.2078
Memória de inicialização da FPGA EPCS16SI8N	10	11.51975	115.1975
MAX 232 encapsulamento SOIC de 16 pinos	10	0.583407	5.83407
Regulador de tensão de 3,3V	10	1.034044	10.34044
Regulador de tensão de 2,5V	10	1.034044	10.34044
Regulador de tensão de 1,2V	10	1.034044	10.34044
LM7805 que é um regulador de tensão de 5V	10	0.442046	4.42046
Ficha IDC macho de 10 pinos para soldar na PCB	10	2.129787	21.29787
Ficha DB9 macho para soldar na PCB	10	0.92158	9.2158
Ficha USB tipo A para soldar na PCB	10	1.12464	11.2464
Ficha de Transformador	10	0.71852	7.1852
Condensador de 1000µF 16V	10	0.1246476	1.246476
Condensadores cerâmicos de 10pF	40	0.0262416	1.049664
Condensadores electrolíticos de 1µF 16V	80	0.11715	9.372
Condensador SMD cerâmico de 0,1µF	230	0.0032802	0.754446
Condensador electrolítico de 0,1µF 16V	40	0.17182	6.8728
Condensador cerâmico de 470pF	10	0.01854094	0.1854094
Resistências de 1/4 W de 10KΩ	40	0.08647232	3.4588928
Resistências de 1/4 W de 1KΩ	60	0.08647232	5.1883392
Resistências de 1/4 W de 330Ω	90	0.00370975	0.3338775
Resistências de 1/4 W de 100KΩ	40	0.09152539	3.6610156
Resistências de 1/4 W de 2,2KΩ	30	0.00370975	0.1112925
Resistências de 1/4 W de 3,3KΩ	30	0.074195	2.22585
Resistências de 1/4 W de 500Ω	30	0.1345663	4.036989
Resistências de 1/4 W de 2KΩ	30	0.053108	1.59324
Resistências de 1/4 W de 4KΩ	30	0.433455	13.00365
Resistências de 1/4 W de 120Ω	20	0.00370975	0.074195
Pente macho de 50 pinos	1	3.8946908	3.8946908
Interruptores de deslizar de 3 pinos e 2 posições	20	0.77319	15.4638
Slot de cartão SD <i>Standard</i>	10	1.6537675	16.537675
LCD de 2 linhas e 16 caracteres	10	10.35237128	103.5237128
Diodo 1N5517B	40	0.4	16
LED de 3mm verde	90	0.0948915	8.540235
LED de 3mm vermelho	10	0.0948915	0.948915
Oscilador de 50MHz	10	2.05403	20.5403
Pente macho duplo de 40 pinos	35	2.38205	83.37175
Pente fêmea duplo de 40 pinos	35	6.46668	226.3338
3 state octal buffer transceiver	10	0.1404238	1.404238
TOTAL			1502.158009

**Tabela 6** Custo de aquisição dos componentes do sistema. [19] [20]

Como se pode observar na tabela acima apresentada, o custo de aquisição de componentes para 10 sistemas, obtido no site da Digikey, seria de cerca de 1502,16€, sendo que dos componentes para cada sistema ficaria por cerca de 150,22€. Apesar de estes preços já incluírem IVA e portes, eles só se referem a aquisição de componentes, pelo pode existir um valor acrescentado referente a mão-de-obra.

Mesmo com a hipótese de ter esse custo associado, o preço de concepção de um sistema tem um valor aceitável, devido ao número de ferramentas e funcionalidades que oferece.

Como as quantidades são relativamente baixas (menores que 500 unidades) estas encomendas são tratadas como material para concepção de protótipos, e não produção em grande escala, o que torna o custo de aquisição maior, pelo que seria mais rentável produzir o sistema em grande escala.

Outra forma de reduzir os custos seria apenas fabricar a placa inferior, que é a base do sistema, o que implicaria abdicar da placa dos periféricos, no entanto ficariam disponíveis para o utilizador todos os pinos associados por esses mesmos periféricos. Na tabela 7 estão discriminados os componentes necessários para fabricar 10 placas base do sistema.

Descrição	Qt.	Preço Unidade(€)	Total(€)
Placas de circuito impresso	10	17.466	174.66
FPGA EP3C5E144C7N	10	11.0902	110.902
PIC 18F4550 de encapsulamento TQFP 44 pinos	10	3.42078	34.2078
Memória de inicialização da FPGA EPCS16SI8N	10	11.51975	115.1975
MAX 232 encapsulamento SOIC de 16 pinos	10	0.583407	5.83407
Regulador de tensão de 3,3V	10	1.034044	10.34044
Regulador de tensão de 2,5V	10	1.034044	10.34044
Regulador de tensão de 1,2V	10	1.034044	10.34044
LM7805 que é um regulador de tensão de 5V	10	0.442046	4.42046
Ficha IDC macho de 10 pinos para soldar na PCB	10	2.129787	21.29787
Ficha DB9 macho para soldar na PCB	10	0.92158	9.2158
Ficha USB tipo A para soldar na PCB	10	1.12464	11.2464
Ficha de Transformador	10	0.71852	7.1852
Condensador de 1000µF 16V	10	0.1246476	1.246476
Condensadores cerâmicos de 10pF	40	0.0262416	1.049664
Condensadores electrolíticos de 1µF 16V	40	0.11715	4.686
Condensador SMD cerâmico de 0,1µF	230	0.0032802	0.754446
Condensador electrolítico de 0,1µF 16V	40	0.17182	6.8728
Condensador cerâmico de 470pF	10	0.01854094	0.1854094
Resistências de 1/4 W de 10KΩ	40	0.08647232	3.4588928
Resistências de 1/4 W de 1KΩ	30	0.08647232	2.5941696
Resistências de 1/4 W de 330Ω	30	0.00370975	0.1112925
Interruptor de deslizar de 3 pinos e 2 posições	10	0.77319	7.7319
Slot de cartão SD <i>Stardard</i>	10	1.6537675	16.537675
Diodo 1N5517B	40	0.4	16
LED de 3mm verde	10	0.0948915	0.948915
LED de 3mm vermelho	10	0.0948915	0.948915
Oscilador de 50MHz	10	2.05403	20.5403
Pente macho duplo de 40 pinos	35	2.38205	83.37175
TOTAL			851.4392411

**Tabela 7** Custo de aquisição de componentes para produção da placa base do sistema. [19] [20]

Como se pode observar na tabela 7, o custo total dos componentes para a concepção de 10 placas de sistema ficaria em 851,44€, ficando o custo de aquisição de componentes para uma placa em 85,144€, podendo ainda existir um custo associado a mão-de-obra.



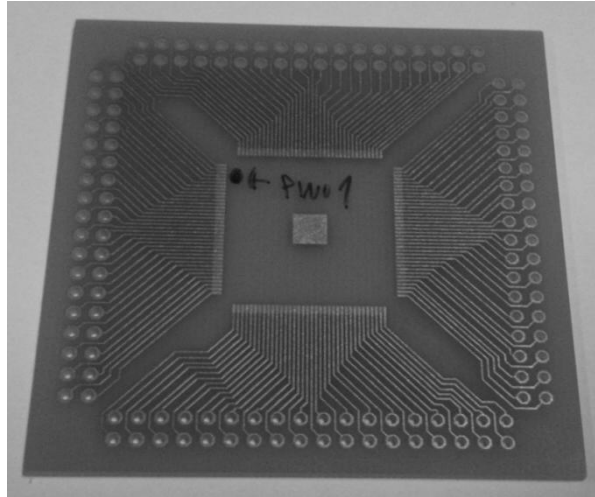
## 5. VALIDAÇÃO DO SISTEMA

Neste capítulo vai-se efectuar a validação do projecto do sistema didáctico.

Devido ao facto da grande parte do material necessário para validação desta ferramenta didáctica não estar disponível no DEE, aliado à importância deste tópico neste trabalho, foi tomada a iniciativa de adquirir por conta própria os componentes não existentes no DEE, a fim de se poder a continuar a desenvolver este trabalho.

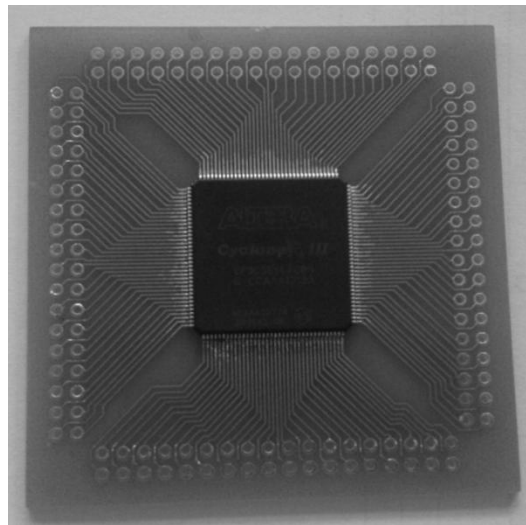
### 5.1. SOCKET FPGA

Como a FPGA é um dispositivo SMD depois de soldada é difícil e arriscado remove-la, e se por ventura o sistema não funcionar como o esperado a FPGA fica confinada nessa PCB. Foram estas as razões que levaram a que se opta-se por criar um socket para soldar a FPGA, socket esse que permitisse remover a FPGA da placa principal. O socket representado na figura 110 criado converte o *pinout* SMD da FPGA para 4 pentes de 36 pinos.



**Figura 110**      **Socket criado para a FPGA.**

A FPGA foi posteriormente soldada no socket como ilustra a figura 111.



**Figura 111**      **FPGA soldada no socket.**

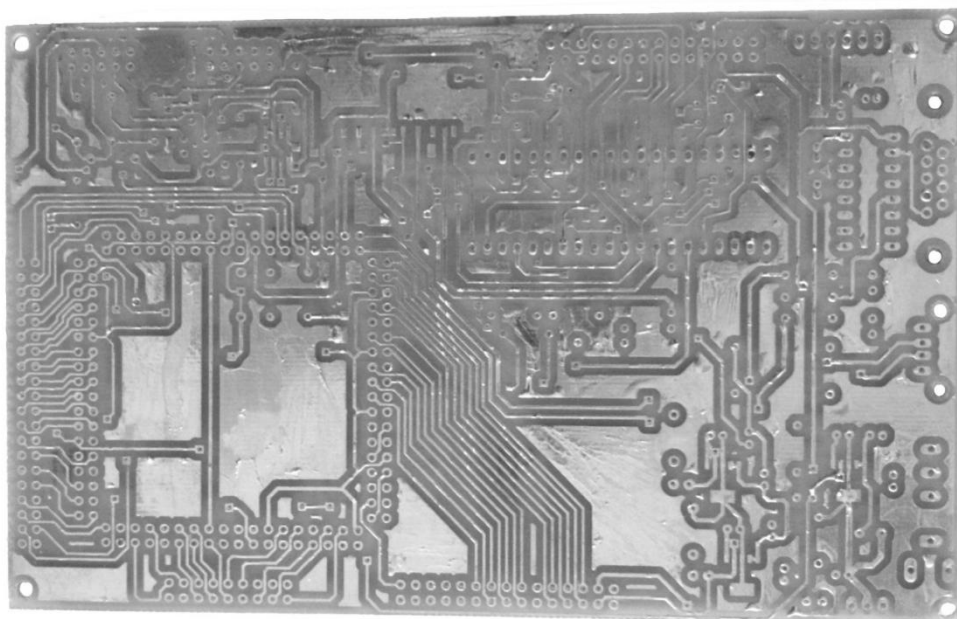
Com este socket já é possível remover a FPGA da PCB onde esta inserida.

Como este socket não encaixa na placa base do sistema, esta teve que ser refeita para o suportar, o que levou a um aumento o tamanho da placa para 16cm×8cm.

Para minimizar custos com componentes foram encomendadas amostras da Texas Instruments de reguladores de tensão, nomeadamente os reguladores de 3,3V (TPS79533), 2,5V (TPS79625) e de 1,2V que acabou por ser substituído por um de 1,6V (TPS79516), por não existir em amostras da Texas Instruments.

Tiveram que ser efectuadas outras alterações para acomodar o PIC 18F4550 e o MAX232, visto que o DEE só possui estes dispositivos em encapsulamento DIP.

Após refeito o esquema eléctrico, foi então criado a partir deste a placa do protótipo do sistema.<sup>9</sup> O próximo passo foi adquirir uma placa pré sensibilizada com as dimensões acima especificadas, que foi concebida na máquina do Laboratório de Sistemas Autónomos do ISEP. Para evitar a oxidação a placa foi estanhada em ambas as faces, sendo que a face de cima (TOP) está representada na figura 112.

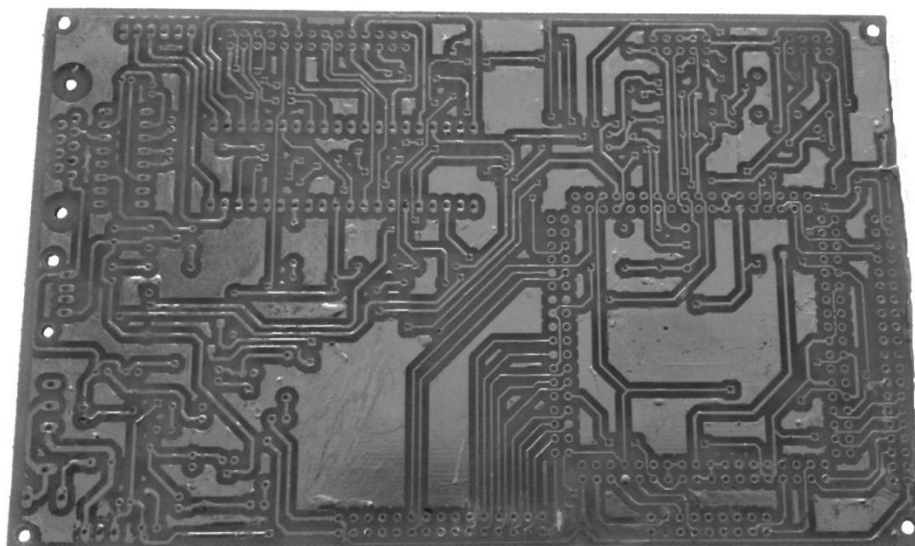


**Figura 112** Vista da face de cima (TOP) da placa do protótipo.

Na figura 113 está representada a outra face da placa, a face de baixo (BOTTOM).

---

<sup>9</sup> Tanto o esquema eléctrico, como o *layout* da placa do protótipo encontram-se disponíveis em anexo.



**Figura 113** Vista da face de baixo (BOTTOM) da placa do protótipo.

Apesar de todo o esforço empreendido, não foi possível concluir a montagem da placa atempadamente, pelo que a fase de validação ficou estagnada após o fabrico da PCB do protótipo.

## 5.2. LISTA DE ENCARGOS

Como referido anteriormente, grande parte dos componentes foram adquiridos por iniciativa do próprio, com o objectivo de construir e validar o protótipo do sistema didáctico, estando discriminada a lista de encargos na tabela 8.

Componente	Quantidade	Custo (€)
PCB do socket da FPGA	2	45
FPGA EP3C5E144C7N	1	35
Placa pré sensibilizada	1	6
Ficha de transformador	1	0,8
Oscilador	1	1
<b>TOTAL</b>		87,8

**Tabela 8** Lista de encargos

## 6. CONCLUSÕES

No decorrer do estudo realizado para conceber o sistema didáctico chegou-se a conclusão as FPGA são uma ferramenta muito versátil, e por isso mesmo o seu uso encontra-se em expansão de forma exponencial.

Relativamente à concepção do sistema, foi provado que era possível conceber um sistema com a FPGA de alta densidade, com um custo relativamente reduzido, quando comparado com a maior parte dos kits de desenvolvimento existentes no mercado, tendo em conta as funcionalidades que este apresenta.

Seria de grande interesse validar o sistema, através da concepção de um protótipo. Apesar de grande esforço financeiro realizado, não foi possível reunir fundos suficientes para concluir o protótipo deste sistema até à data de entrega deste documento.

No que diz respeito a trabalho futuro, o aspecto mais importante seria o da validação do sistema, que implicaria o teste de todas as interligações e desenvolvimento de rotinas de software dedicadas a comunicação entre a FPGA, PIC e periféricos.

Depois da validação do sistema didáctico, o passo seguinte seria fazer uma proposta ao DEE para produção desta ferramenta, com o objectivo de a incluir no programa de uma cadeira, dedicada a transmitir aos alunos do ISEP conhecimentos de VHDL e FPGA's.

Outro aspecto interessante a abordar seria o estudo da concepção de *daughter cards* para expansão das funções desta ferramenta didáctica. O conceito seria conceber placas separadas para adicionar e expandir as funções do sistema, que poderiam ser desenvolvidas por alunos do ISEP.

Um outro desenvolvimento que poderia resultar deste projecto poderia ser a concepção de um *kit* com três vertentes. Através de algumas modificações da placa base o sistema, todos os pinos do PIC poderiam ser colocados à disposição do utilizador, sem quebrar a ligação com a FPGA, o que permitia que o sistema funcionasse como dois *kits* independentes, um de microcontrolador e um de FPGA, possibilitando também que estas duas ferramentas funcionassem em conjunto.

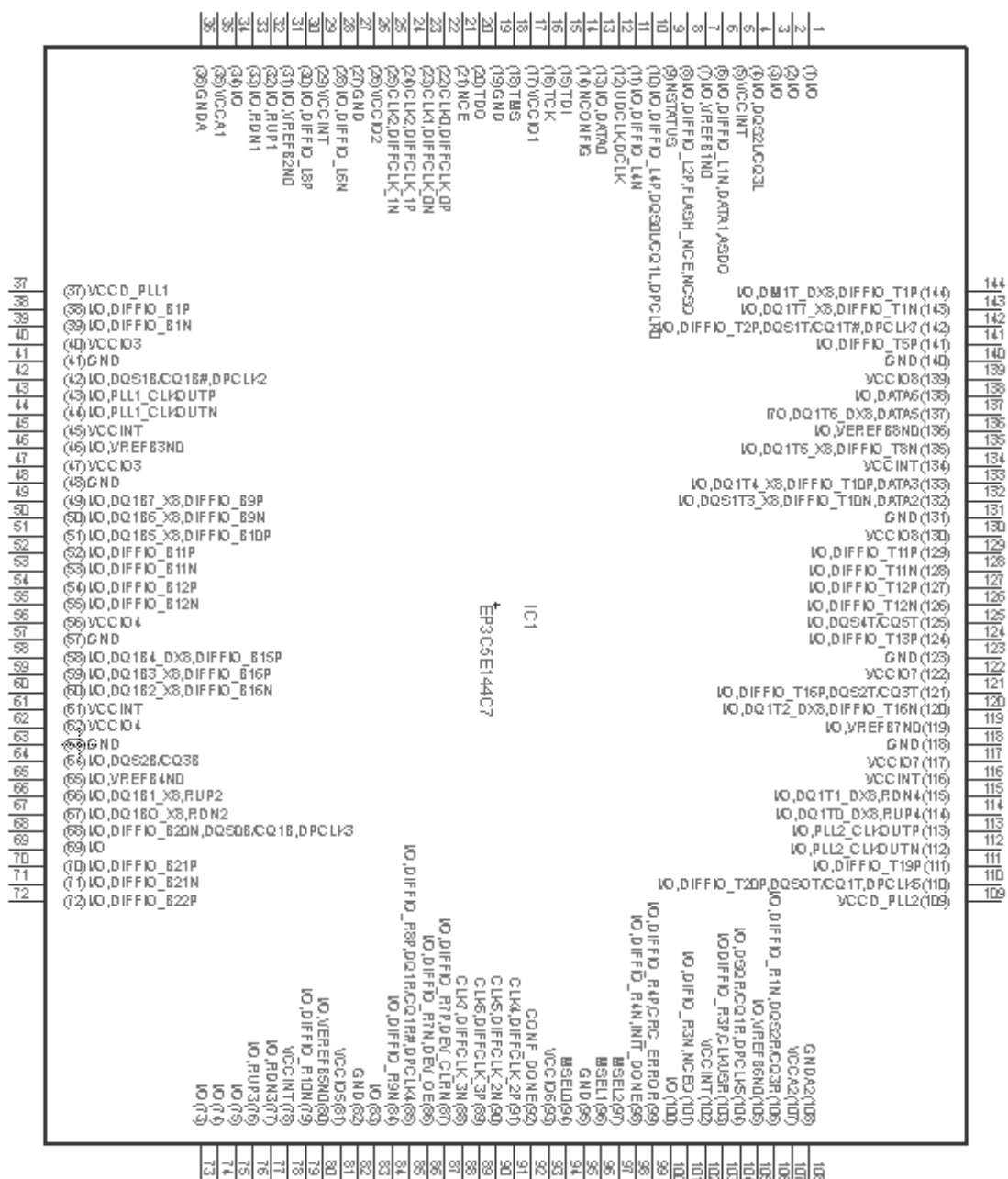
## 7. REFERÊNCIAS BIBLIOGRÁFICAS

- [1] Maxfield, Clive – *The Design Warrior's Guide to FPGA: Devices, Tools and Flows*. Elsivier,2004.
- [2] Maini, Anil K. – *Digital Systems: Principles, Devices and Applications*. John Willey & Sons Ltd,2007.
- [3] Maxfield, Clive – *FPGA World Class Design*. Elsivier.
- [4] Kaelslin, Hubert – *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrics*. Cambridge University Press,2008.
- [5] Sicard, Etienne ; Bendhia, Sonia Delmas – *Advance CMOS Cell Design*. McGraw-Hill.
- [6] Parker, Kenneth P. – *The Boundary Scan Handbook Analog and Digital*
- [7] Bleeker, Harry ; Eijnden, Peter Van der ; Jong, Frans de – *Boundary Scan Test A Pratical Approach*. 1993.
- [8] Smith, Gina R. – *FPGA 101: Everything you need to know to get started*. Elsivier,2010.

- [9] Ashenden, Peter J. – *The VHDL Cookbook*. 1990.
- [10] ALTERA – *Altera* [em linha]. San Jose, CA: Altera, act. 2012.[consult. 10 Ago. 2012] Disponível na Internet <URL [http://www.altera.com/products/devkits/kit-dev\\_platforms.jsp](http://www.altera.com/products/devkits/kit-dev_platforms.jsp)>.
- [11] EasyFPGA – *EasyFPGA* [em linha]. Foster City, CA: EasyFPGA, act. 2012.[consult. 11 Ago. 2012] Disponível na Internet <URL <http://www.easyfpga.com/>>.
- [12] TERASIC – *Terasic Technologies* [em linha]. Terasic, act. 2012.[consult. 11 Ago. 2012] Disponível na Internet <URL <http://www.terasic.com.tw/en/>>.
- [13] ACTEL – *Actel* [em linha]. Actel, act. 2012.[consult. 12 Ago. 2012] Disponível na Internet <URL <http://www.actel.com/>>.
- [14] XILINX – *Xilinx* [em linha]. Terasic, act. 2012.[consult. 12 Ago. 2012] Disponível na Internet <URL <http://www.actel.com/>>.
- [15] Altera – *Cyclone III Handbok : Volume I and Volume II*. Altera,101 Innovation Drive San Jose, CA. December 2011.
- [16] Altera – *Enhanced Configuration Devices (EPC4, EPC8 and EPC16) Data Sheet*. Altera,December 2009.
- [17] Microchip – *PIC18F2455/2550/4455/4550 Data Sheet*. Microchip Technology Inc., 2006
- [18] Ibrahim, Dogan – *Advanced PIC Microcontroller Projects in C: From USB to ZIGBEE with the PIC 18F Series*. Elsivier, 2008.
- [19] CIRCUITOTAL – *Circuitotal* [em linha]. Aveiro: Circuitotal, act. 2012.[consult. 27 Out. 2012] Disponível na Internet < <http://www.circuitotal.pt/pt/contact.htm>>.
- [20] DIGIKEY – *Digikey* [em linha]. Digikey, act. 2012.[consult. 27 Out. 2012] Disponível na Internet <URL <http://www.digikey.com/>>.

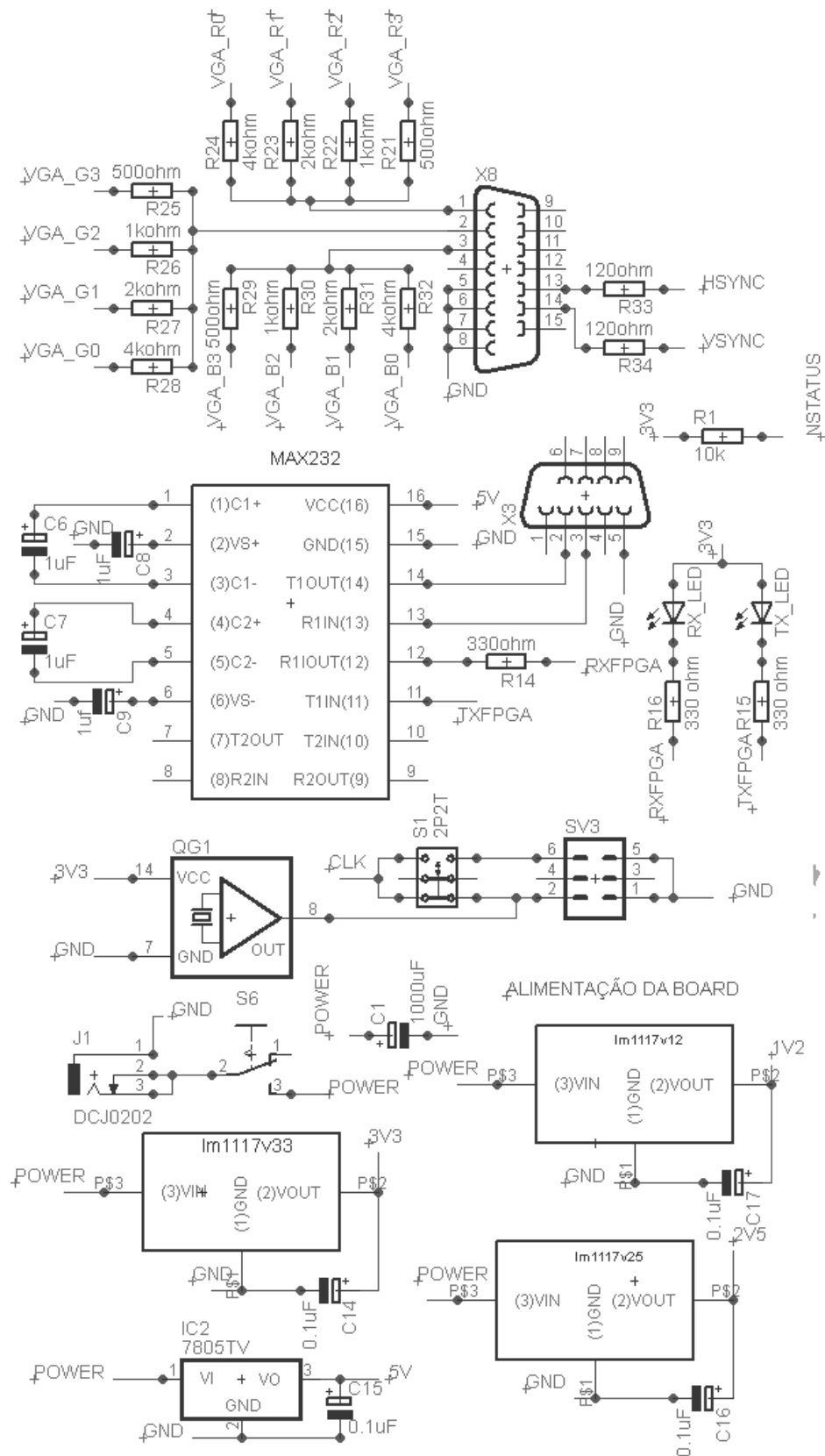


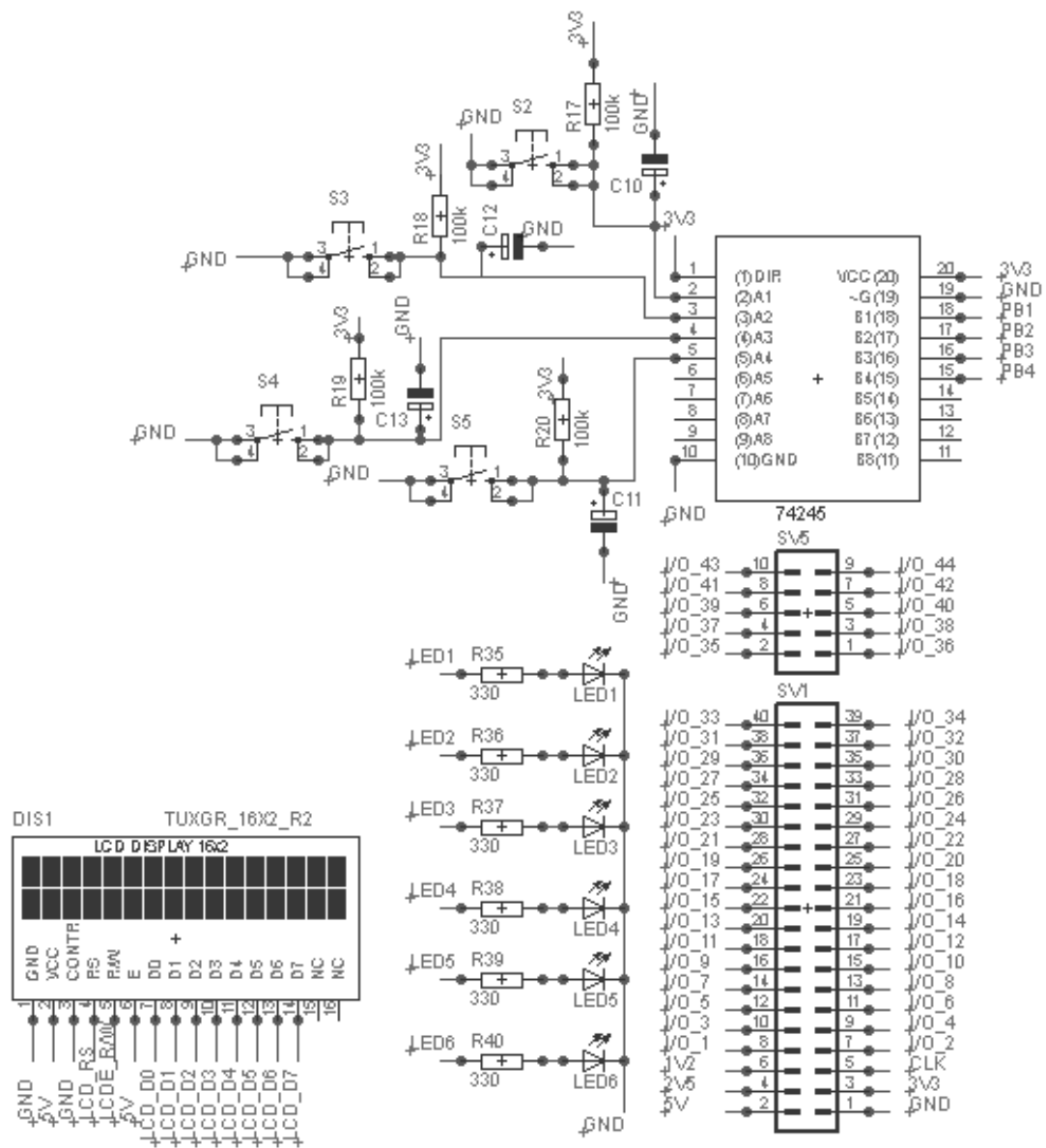
## Anexo A. Símbolo da FPGA EP3C5E144C7N



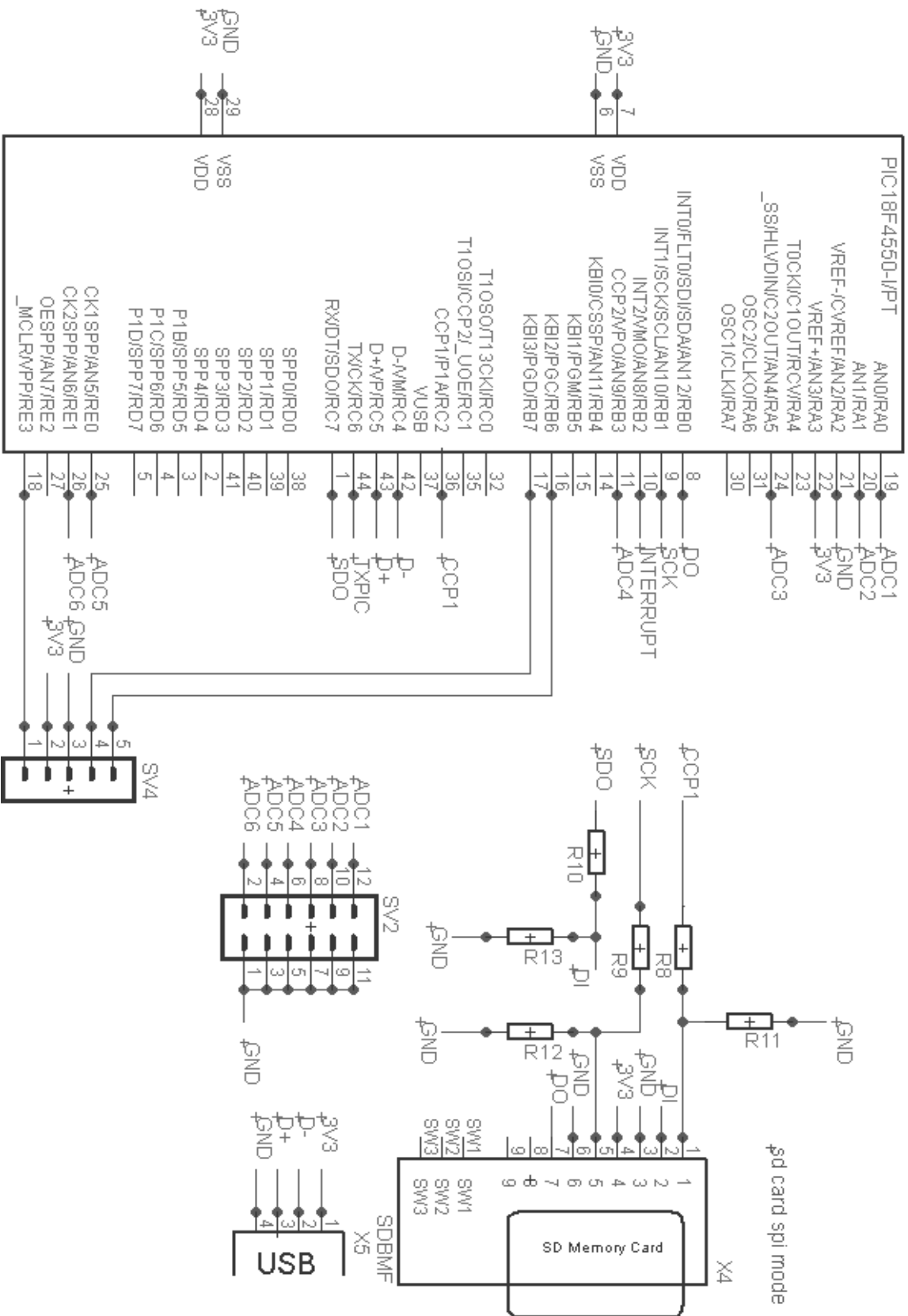
## Símbolo da FPGA ao pormenor

## Anexo B. Esquema de ligações da PCB inicial

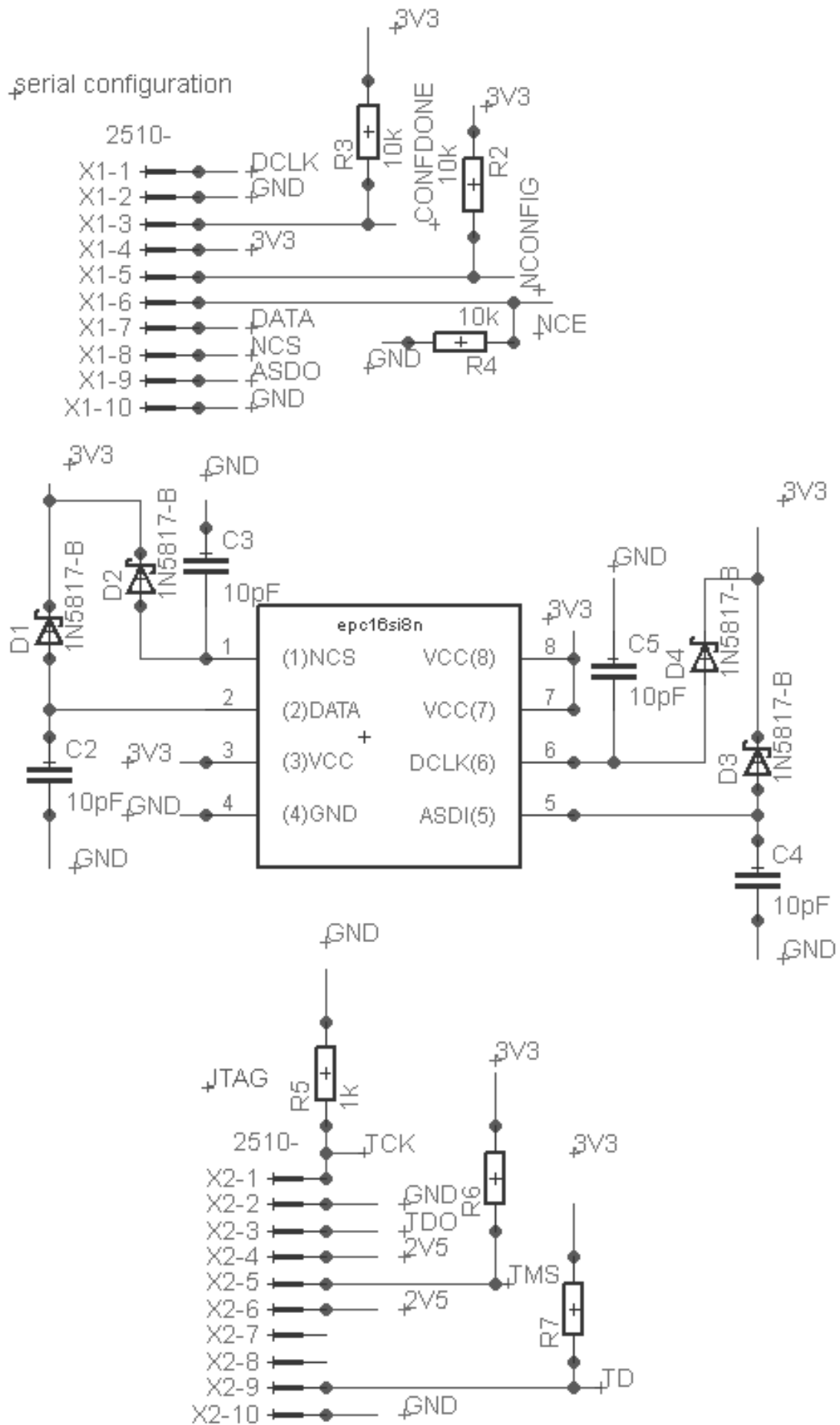




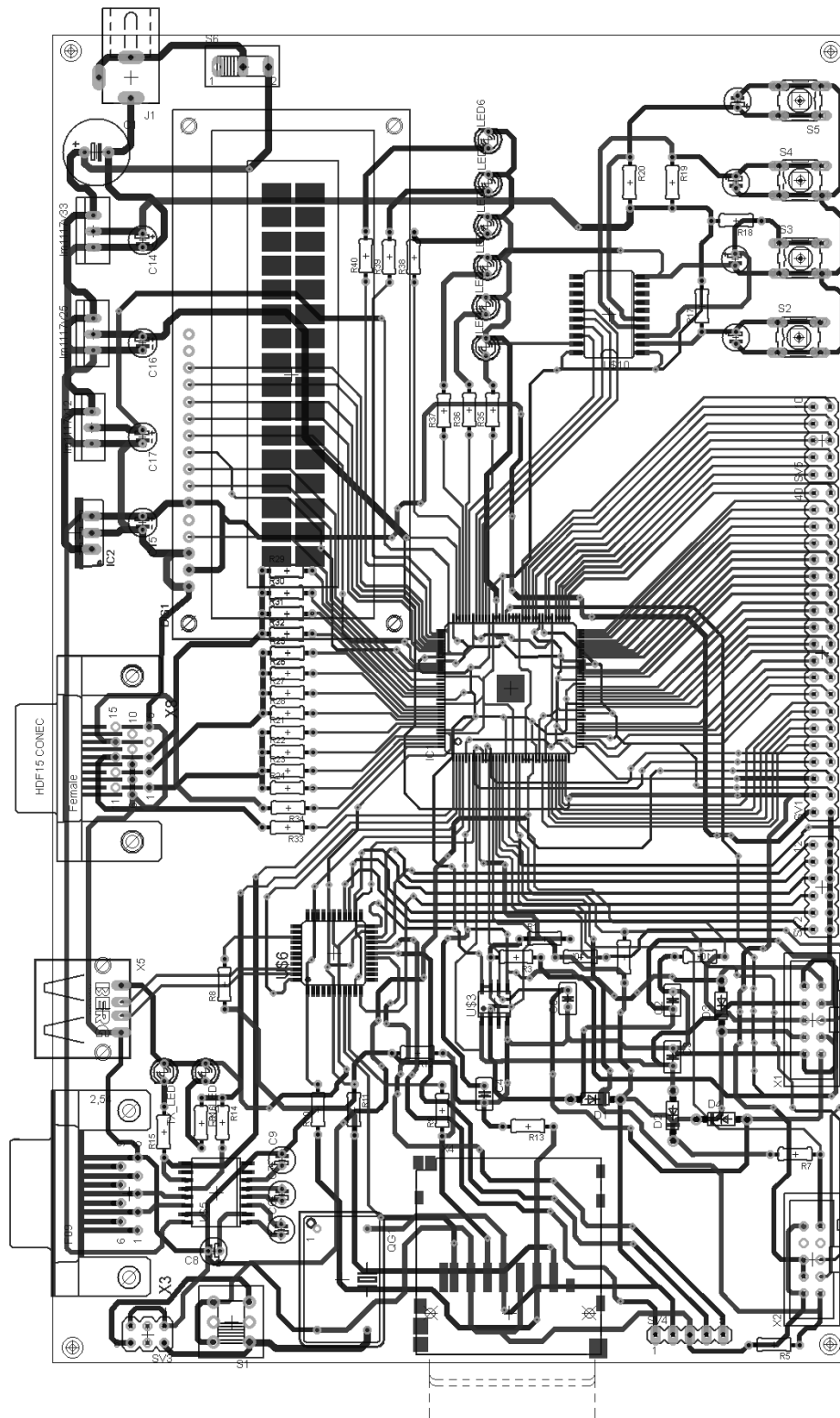
U\$6



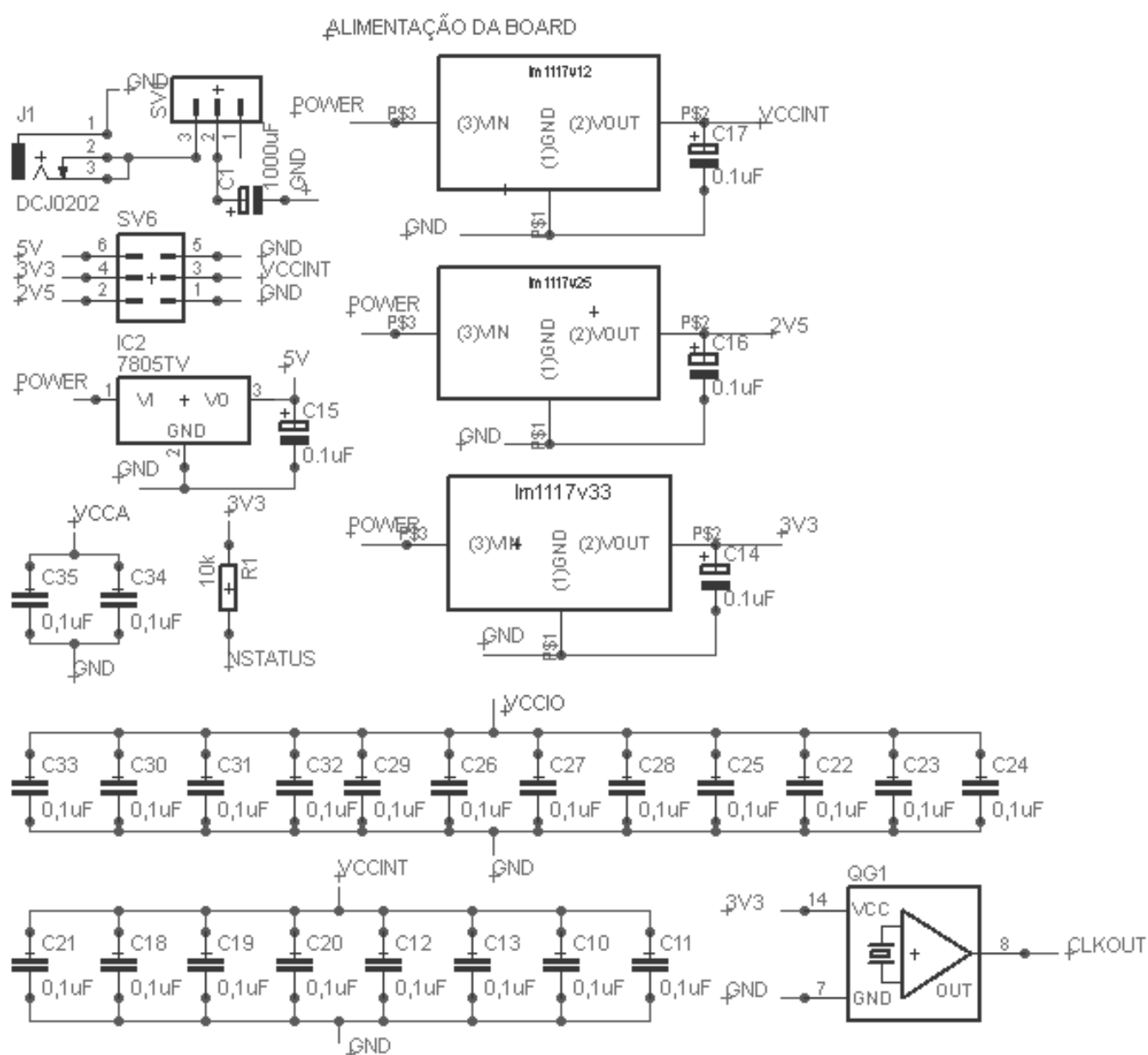




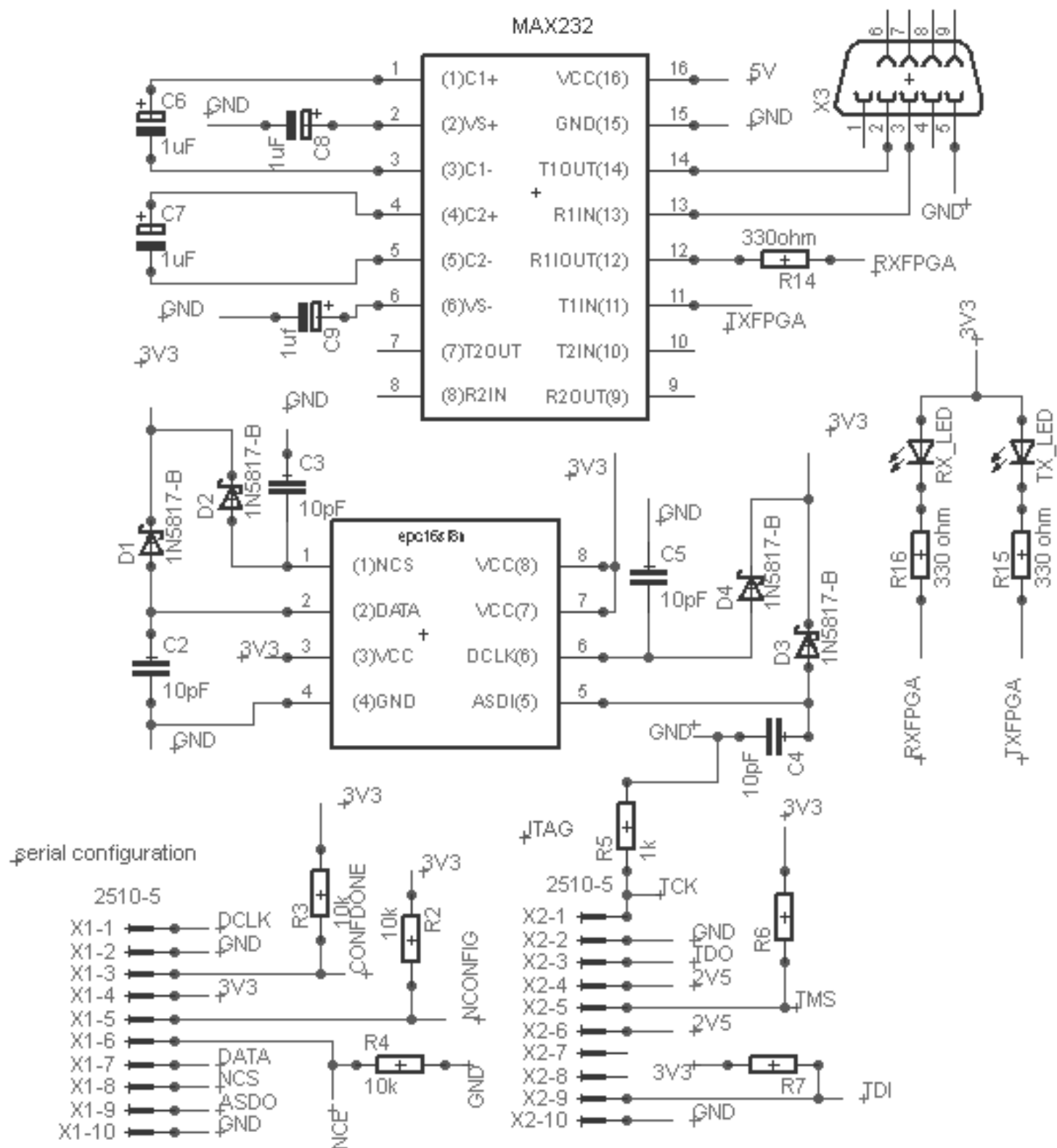
## Anexo C. *Layout* da PCB inicial

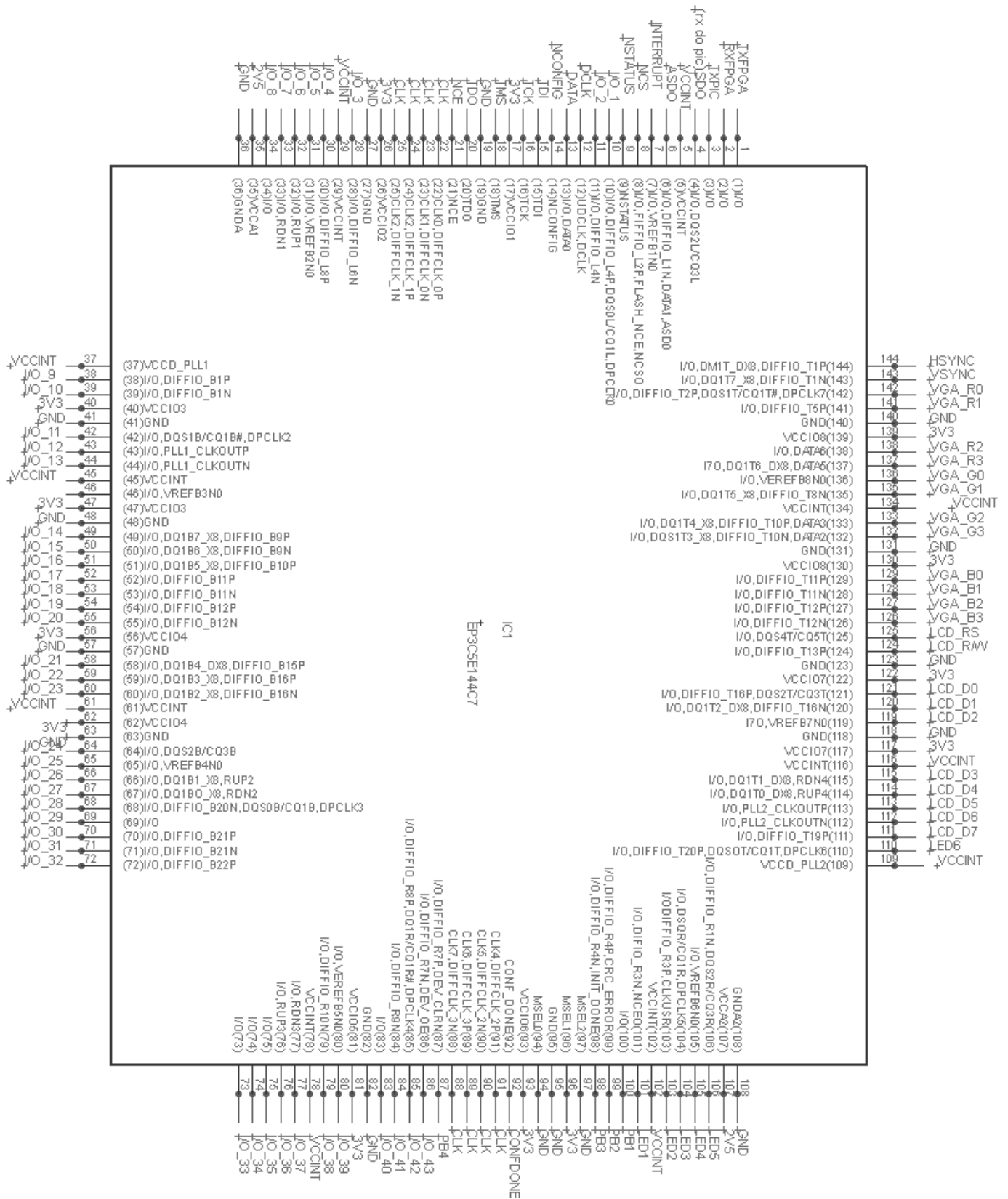


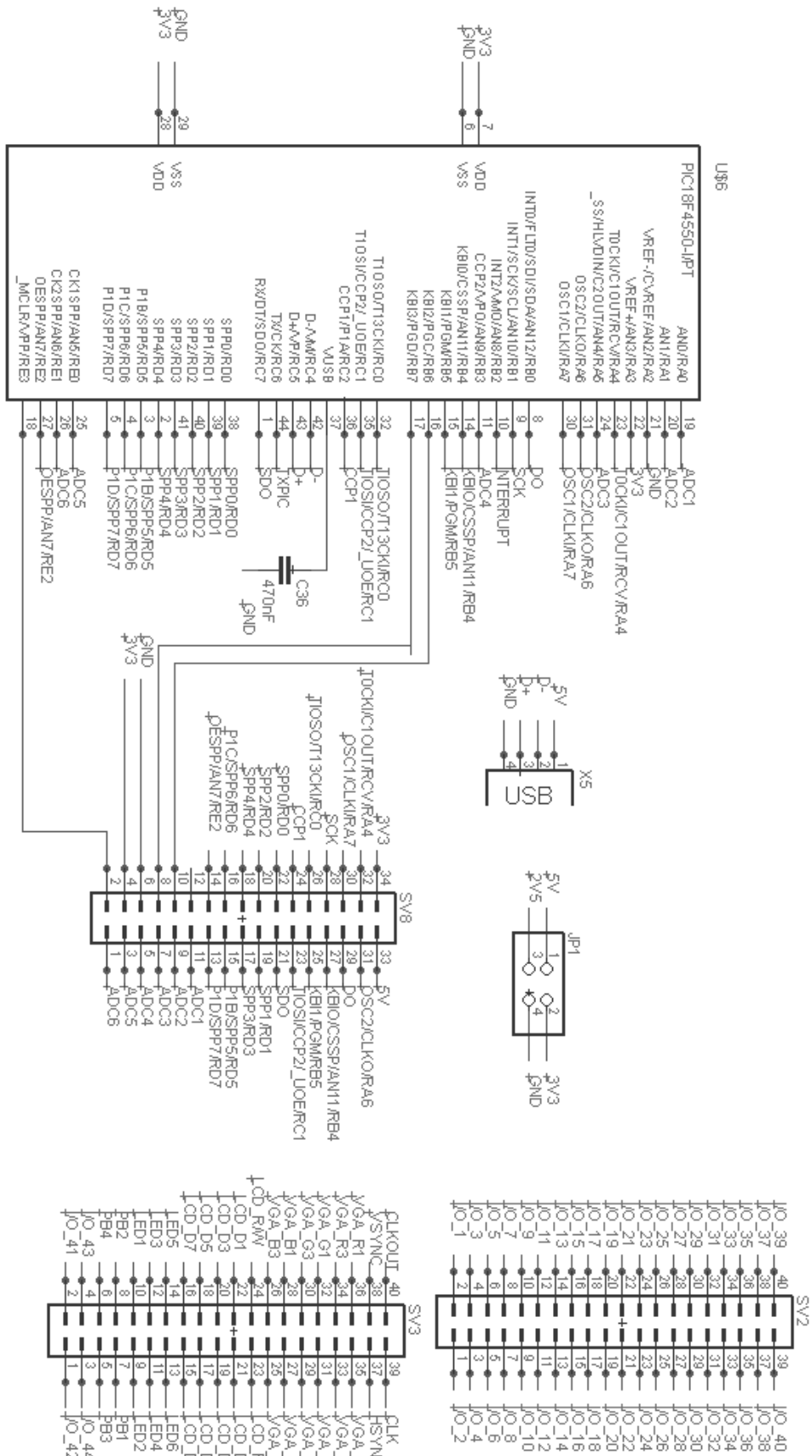
## Anexo D. Esquema de ligações da placa inferior



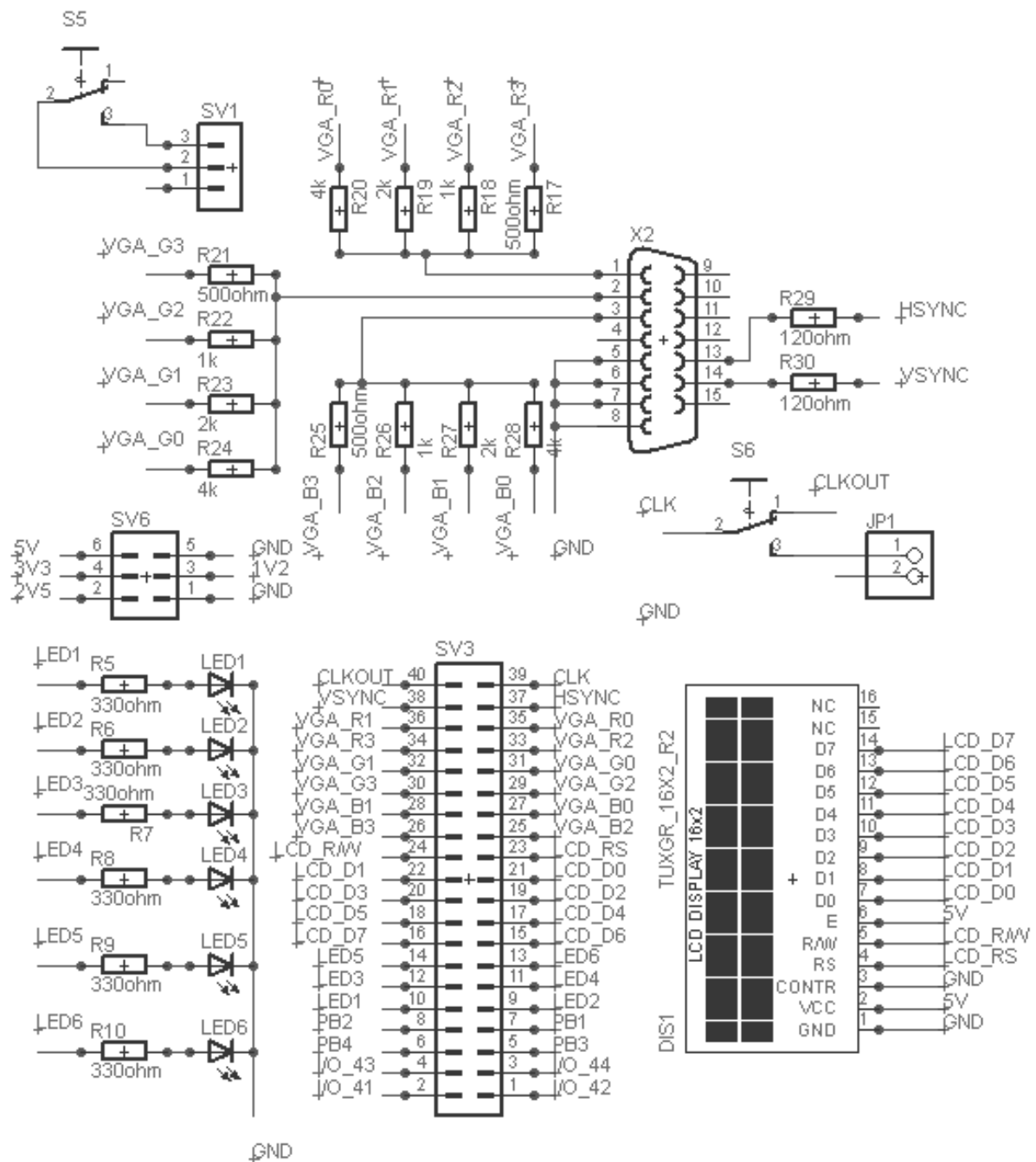


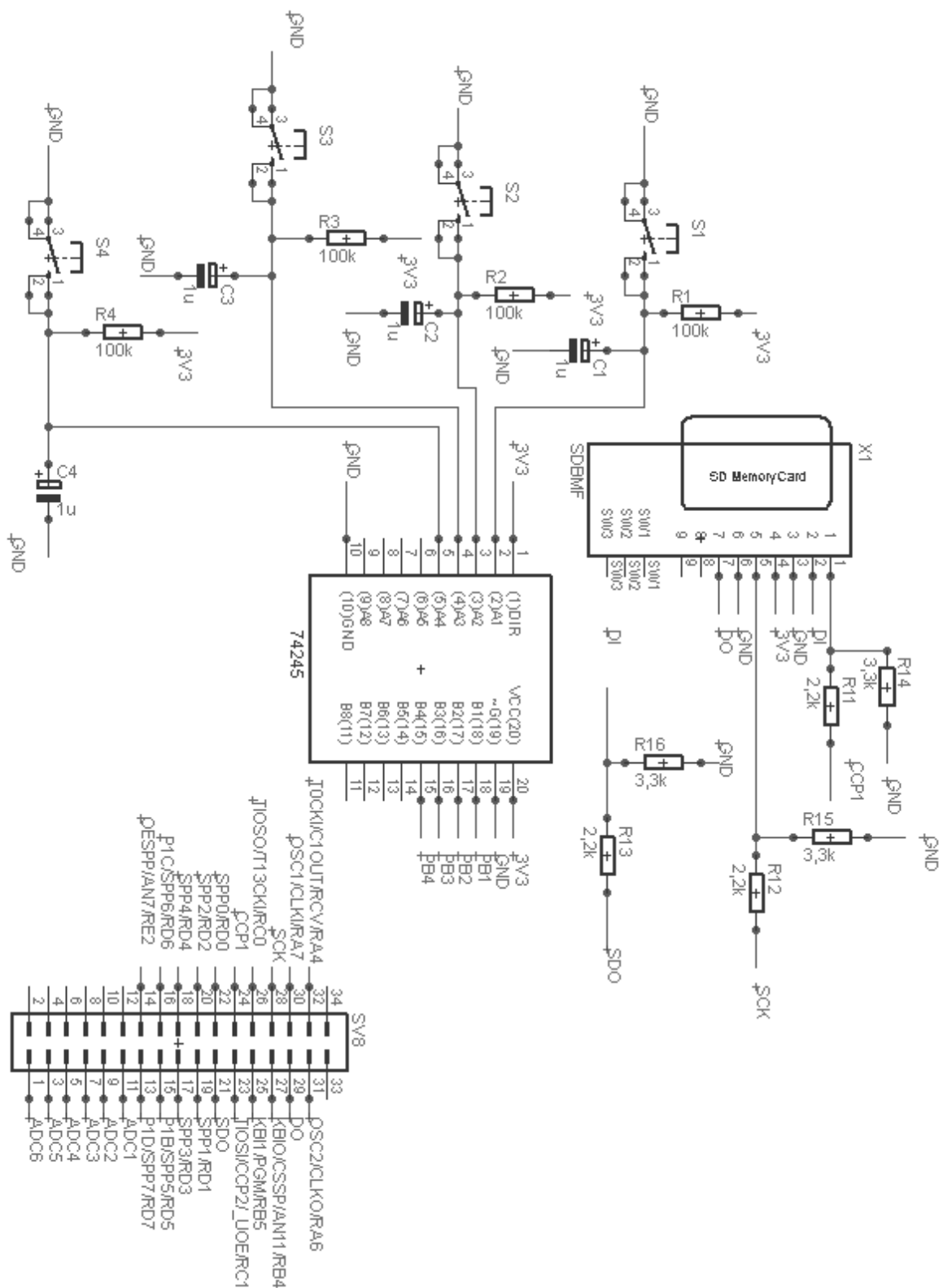




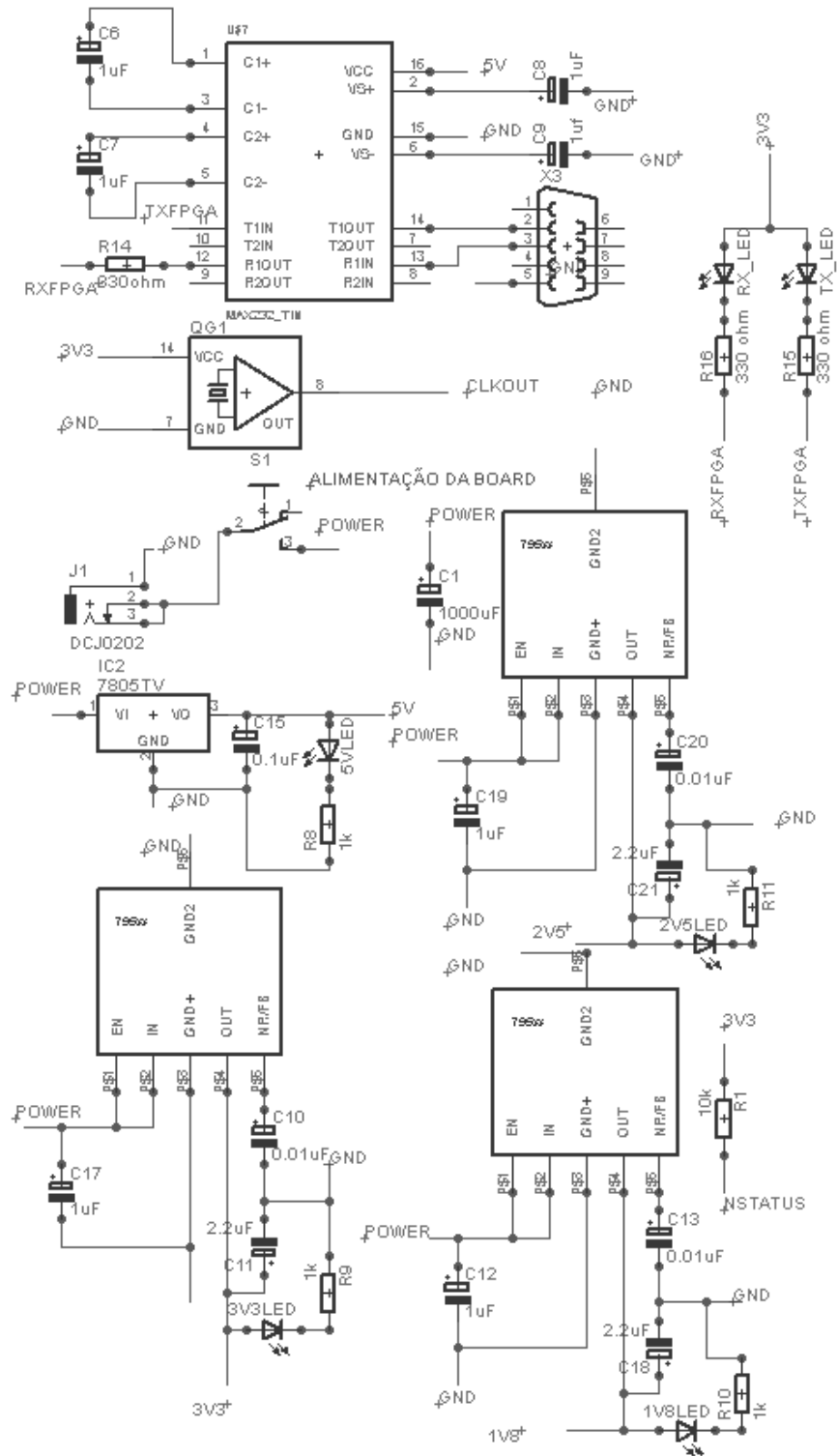


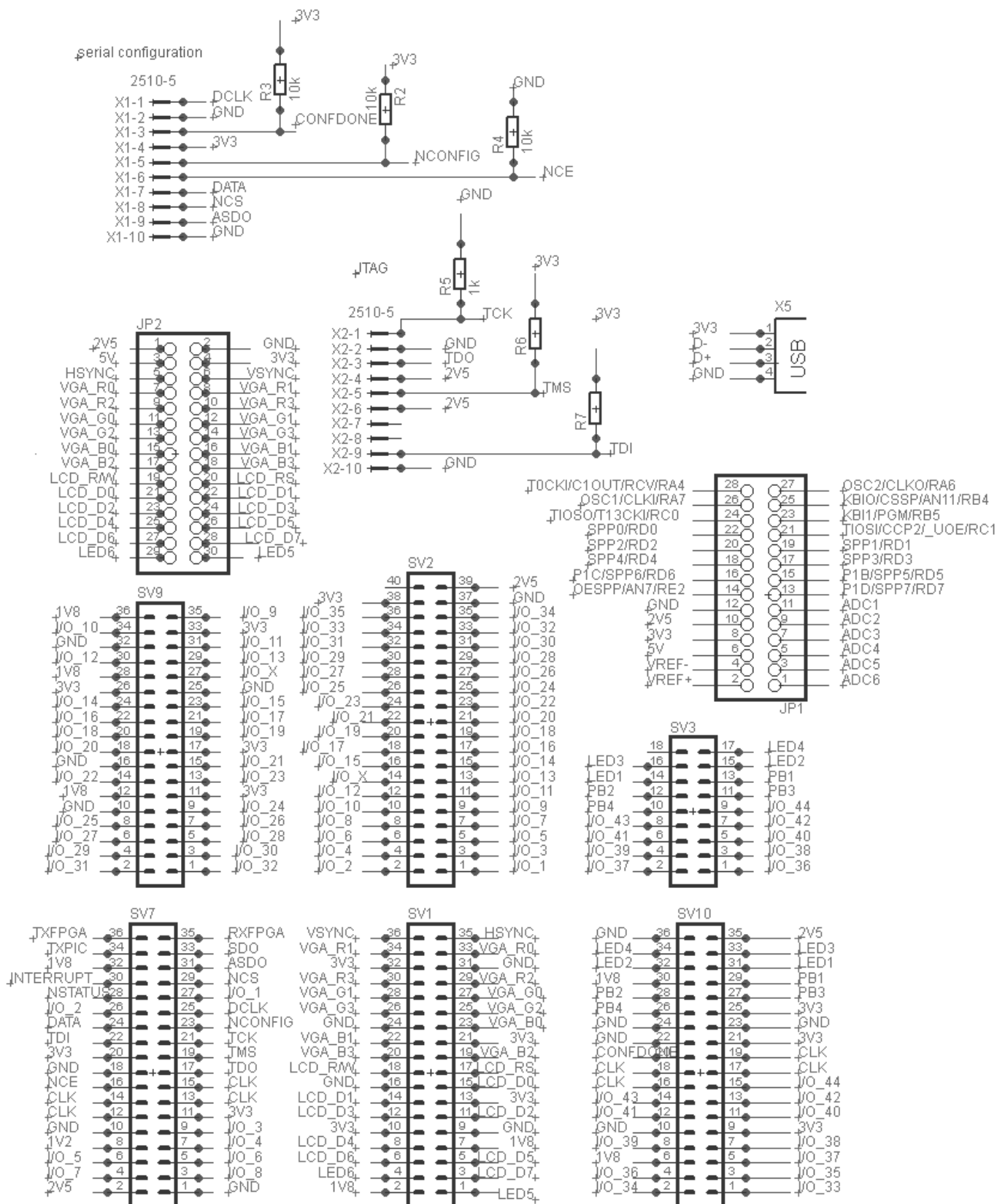
## Anexo E. Esquema de ligações da placa superior





## Anexo F. Esquema de ligações do protótipo

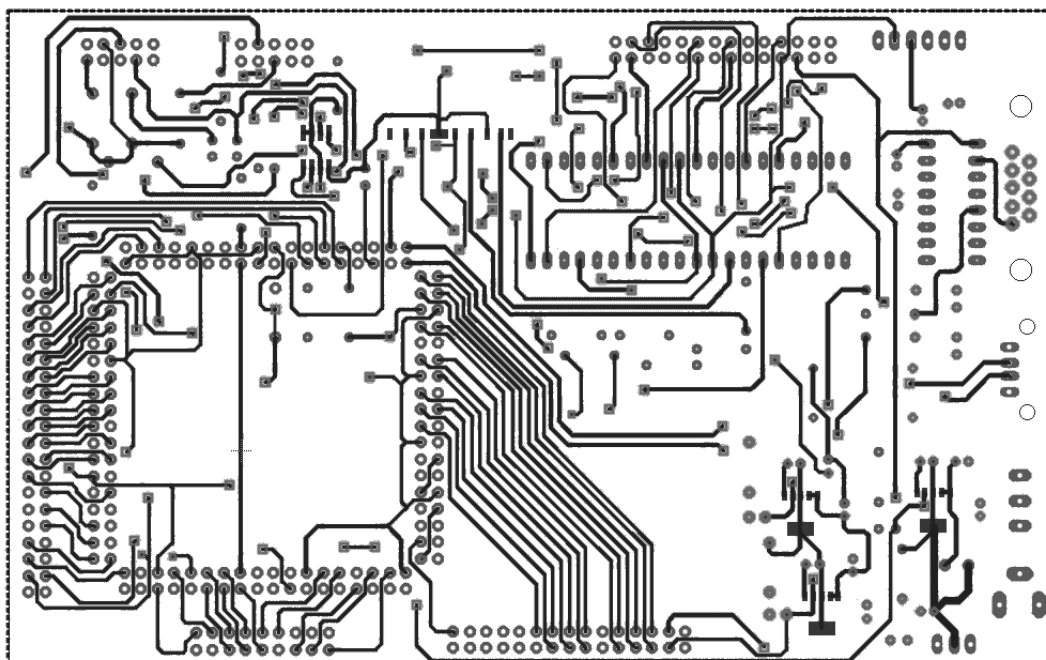




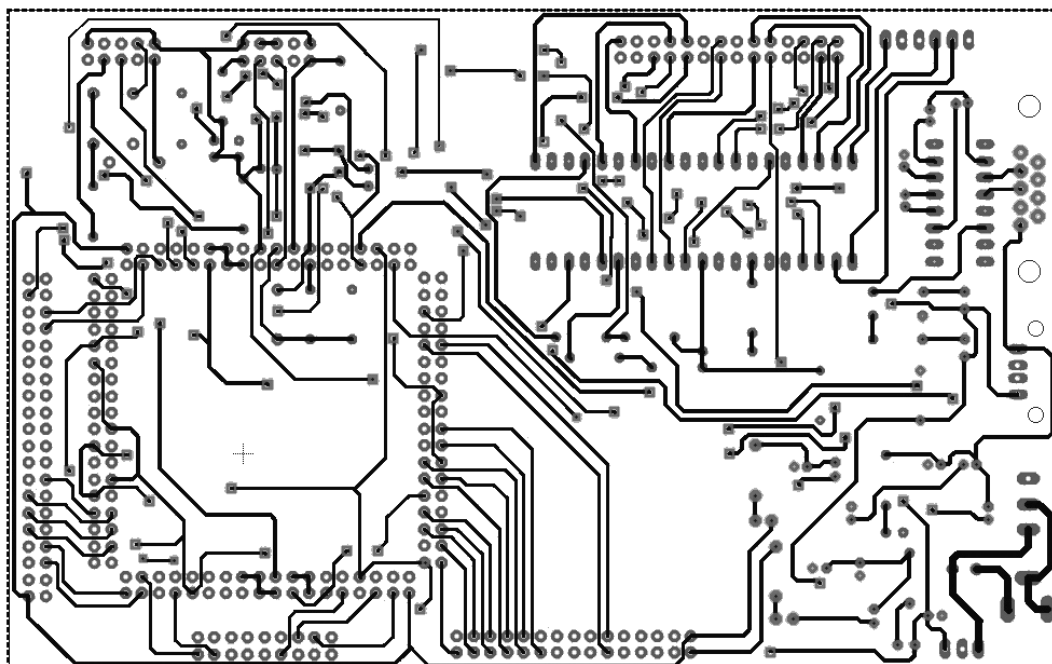




## Anexo G. *Layout* da PCB do protótipo



Face superior (TOP)



Face inferior (BOTTOM)