Chronos – Sistema Reconfigurável de Monitorização Para Aplicações Ultra-low-Power

João Barata, Ricardo Martins, e Rogério P. Dionísio, *membro, IEEE*

***Resumo* – O projecto *Chronos* consiste num sistema onde se integra um microprocessador assíncrono implementado em FPGA.**
O microprocessador é capaz de executar as funções dos microprocessadores RISC tipo PIC da Microchip®, possuindo unidades de entrada/saída (I/O) para ligação a sensores e capacidade de comunicação com um computador pessoal (PC).

Palavras-chave – Monitorização, Sistema Embebido, Microprocessador Assíncronos, FPGA, VHDL.

I. Introdução

Os sistemas embebidos[1] têm um papel de elevada importância no mundo moderno das tecnologias, no entanto uma das áreas que mais evoluiu com o seu advento, e que tem conseguido um espectro mais alargado de aplicações, foi a monitorização. A maioria dos sistemas embebidos possuem um relógio de sistema que rege o seu funcionamento a nível temporal, ou seja, são sistemas síncronos. Existem contudo algumas aplicações em que se torna vantajosa a utilização de sistemas sem sinal de relógio, os chamados sistemas assíncronos.

Neste trabalho, desenvolveu-se uma implementação assíncrona, em FPGA, de um microprocessador capaz de processar comandos idênticos aos dos microcontroladores PIC da Microchip®, que são do tipo RISC.

O microprocessador desenvolvido foi utilizado numa aplicação simples de monitorização, que tem como finalidade a verificação prática do funcionamento do sistema. Foi também elaborado um programa básico, em PC, que comunica com o microprocessador através da interface paralela (LPT).

Artigo elaborado em Fevereiro de 2008 por:
João Barata, engenheiro de firmware e hardware no departamento de Investigação e Desenvolvimento em Electrónica, da *Mecalbi, Lda* (barata@simplesnet.pt).
Ricardo Martins, engenheiro de firmware e hardware no departamento de Investigação e Desenvolvimento em Electrónica, da *Vitor Cardoso, Lda* (martins@simplesnet.pt).
Rogério Dionísio, docente do departamento de Engenharia Electrotécnica da Escola Superior de Tecnologia de Castelo Branco (rdionisio@ieee.org).

[1] Sistema microprocessado de aplicação dedicada

II. Vantagens e Desvantagens dos Circuitos Assíncronos

Devido ao facto de não possuírem relógio interno, os circuitos assíncronos têm um consumo de potência bastante inferior aos dos circuitos síncronos, pois em tecnologia CMOS, o consumo de potência, quando em estado estático, é quase nulo. Nos circuitos síncronos, como o sinal de relógio percorre todo o circuito a cada impulso, o seu consumo é muito maior.
Também relacionado com o relógio temos a vantagem, nos circuitos assíncronos, de não haver problemas de deslizamento de relógio ("Clock Skew"), não sendo portanto necessários mecanismos de distribuição de relógio.
Nos sistemas assíncronos todas as partes do circuito operam a diferentes velocidades, não mudando os seus estados em tempos dependentes do sinal de relógio global. Este facto faz com que as emissões electromagnéticas se dispersem, ou seja, a potência do ruído electromagnético emitido se encontra dispersa no espectro, ao invés dos sistemas síncronos, que concentram a potência de ruído electromagnético emitido ao redor da frequência do relógio do sistema. Temos então que os sistemas assíncronos são menos ruidosos que os sistemas síncronos.
Nos sistemas assíncronos, cada bloco pode transferir dados à sua velocidade natural, não estando dependente dos módulos que o sucedem, isto devido ao facto da transmissão de dados ser efectuada por eventos e não por esperas de impulsos de relógio, como acontece com os circuitos síncronos. Num circuito síncrono, a transferência de dados entre blocos internos é efectuada ao ritmo de transmissão do bloco mais lento, o que impõe uma dependência de tempos entre blocos e por conseguinte uma menor eficiência temporal. Nos sistemas assíncronos não existe o conceito de caminho crítico, que define o atraso máximo de transmissão de um bloco, pois, tal como já foi dito, a transmissão de dados é efectuada por eventos.
Existem inúmeras vantagens dos circuitos assíncronos face aos síncronos, no entanto são os circuitos síncronos que predominam no projecto de hardware. Isto deve-se ao facto dos circuitos síncronos serem mais simples, pois o projectista, no seu desenvolvimento, não necessita de se preocupar com os estados dinâmicos do sistema, pois tudo é regido pelo sinal de relógio. Esta complexidade, e o facto de existir uma grande deficiência no mercado, de ferramentas CAD para projecto de circuitos assíncronos, fazem com que estes sejam muito pouco

usados, sendo os circuitos síncronos os mais usados em circuitos digitais.

III. FPGA

Um FPGA (Field Programmable Gate Array) é um dispositivo lógico programável, ou seja, é um hardware reconfigurável, que consiste, tal como o seu nome indica, numa matriz de células lógicas programáveis.
Estes dispositivos semicondutores foram criados pela Xilinx,inc., na década de 80, e são largamente utilizados no processamento de informação digital, estando a ser aplicados nos mais diversificados aparelhos electrónicos, tais como, televisões, telemóveis, receptores de satélite, descodificadores de sinal etc. Os FPGAs são também muito úteis na prototipagem rápida de circuitos integrados. O mercado actual dos FPGAs assenta essencialmente em quatro fabricantes, sendo estes, por ordem de dimensão, Altera, Xilinx, Lattice e Actel.

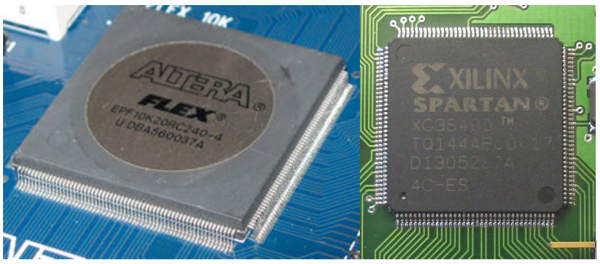


figura 1: FPGAs

Internamente, um FPGA é constituído basicamente por três tipos de componentes:
1) CLB (Configuration Logical Blocks): São blocos lógicos idênticos, constituídos pela reunião de flip-flops e a utilização de lógica combinatória. Um CLB pode ser configurado para definir qualquer função lógica.
2) IOB (Input/Output Blocks): São circuitos responsáveis pela interface entre as saídas/entradas do FPGA e as saídas/entradas das combinações de CLBs. São basicamente buffers, que funcionam como um pino bidireccional de entrada e saída do FPGA
3) Switch Matrix (Comutadores de interconexão): São as ligações utilizadas para interligar os CLBs e os IOBs. Estas conexões internas são definidas pela programação do FPGA, consoante as necessidades.

Na figura 2 pode ser visualizado o esquema básico de um FPGA.

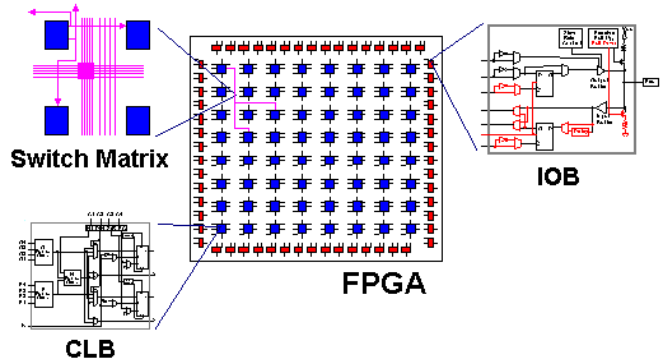


figura 2: Esquema básico de um FPGA

De frisar que existem vários modos possíveis de implementação das funções lógicas dentro dos blocos lógicos do FPGA, no entanto, o mais utilizado pelos maiores fabricantes é o recurso a blocos de memória LUT (Look-Up Table). As LUTs contêm células de armazenamento que são utilizadas para implementar pequenas funções lógicas. Devido ao facto de estas células de armazenamento serem do tipo volátil, ao faltar a alimentação ao FPGA é perdida toda a informação.
A correcta configuração das ligações entre CLBs e IOBs, bem como a configuração das funções dos CLBs, faz com se consiga, através de um FPGA qualquer hardware digital, que utilize lógica, quer combinatória, quer sequencial. A definição da configuração do FPGA, ou seja, da definição do hardware é feita através de linguagens de definição de hardware.

IV. LINGUAGEM DE DESCRIÇÃO DE HARDWARE

Uma linguagem de descrição de hardware é uma linguagem que permite descrever o funcionamento lógico de um dispositivo programável, como é o caso das FPGAs. Existem várias linguagens de descrição de Hardware, sendo as mais conhecidas, a VHDL, a Verilog e a SystemC.
A linguagem utilizada para a realização deste projecto foi a VHDL (VHSIC Hardware Description Language) que é uma linguagem para descrição de hardware de circuitos integrados de elevada frequência. Esta é uma das mais utilizadas, muito devido ao facto de ter sido uma das primeiras a surgir, tendo sido desenvolvida pelo departamento de defesa dos Estados Unidos da América, nos meados dos anos 80, com a finalidade de documentar de forma fácil e portável, o comportamento de ASICs que compunham os equipamentos vendidos às Forças Armadas americanas.

V. IMPLEMENTAÇÃO

A. Intruções

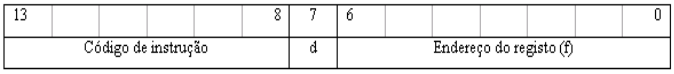
Cada instrução é constituída por uma palavra de 14 bits, dividida em Código de operação que indica o tipo de instrução e um ou mais operandos que definem a execução da instrução.
As operações dividem-se em três categorias básicas: orientadas ao byte, orientadas ao bit e operações com literais ou de controlo.
Para as operações orientadas ao byte ‘f’ representa o endereço do registo que será utilizado na operação e ‘d’ indica o destino do resultado da operação. Se ‘d’ for zero o resultado será deixado no acumulador (W), se for ‘1’ o resultado será guardado no registo especificado em ‘f’.
Nas operações orientadas ao bit, ‘b’ indica qual o bit que será afectado pela operação e ‘f’ indica qual a posição do registo f que será utilizada na operação.
Para as operações com literais ou de controlo, ‘k’ indica uma constante de oito ou dez bits, ou um literal.
A tabela 1 descreve os campos de um Código de instrução:

Campo	Descrição
f	Endereço do registo (0x00 a 0x7F)
W	Acumulador
b	Posição de um bit
k	Literal ou constante
x	Não considerado (Don´t care)
d	Selecção do destino. Se 'd' = 0, o resultado é colocado no acumulador, se 'd' = 1 o resultado é colocado no registo 'f'

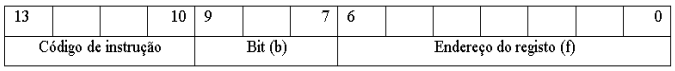
tabela 1: Campos do código de instrução

O formato das instruções, é assim diferente, consoante a categoria da instrução:

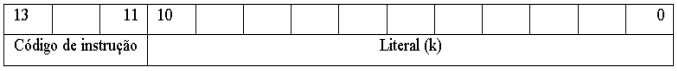
Para instruções orientadas ao byte:



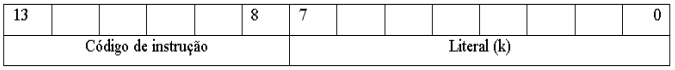
Para instruções orientadas ao bit:



Para operações com literal, ou de controlo:
- Geral



- Instrução CALL e GOTO



As Instruções implementadas baseiam-se no "Instruction Set" do microcontrolador PIC16F84 da Microchip™, e encontram-se descritas na tabela 2.

Mnemonicas		Descrição	Opcode (14-bit)			
			MSB			LSB
Instruções orientadas ao byte						
ADDWF	f,d	Adiciona W a f	00	0111	dfff	ffff
ANDWF	f,d	AND de W com f	00	0101	dfff	ffff
CLRF	f	Apaga f	00	0001	1fff	ffff
CLRW	-	Apaga W	00	0001	0xxx	xxxx
COMPF	f,d	Complemento de f	00	1001	dfff	ffff
DECF	f,d	Decrementa f	00	0011	dfff	ffff
DECFSZ	f,d	Decrementa f, salta se 0	00	1011	dfff	ffff
INCF	f,d	Incrementa f	00	1010	dfff	ffff
INCFSZ	f,d	Incrementa f, salta se 0	00	1111	dfff	ffff
IORWF	f,d	OR de W com f	00	0100	dfff	ffff
MOVF	f	Move f para W	00	1000	xfff	ffff
MOVWF	f	Move W para f	00	0000	1fff	ffff
NOP	-	Não executa nada	00	0000	0000	0000
RLF	f,d	Roda f à esquerda	00	1101	dfff	ffff
RRF	f,d	Roda f à direita	00	1100	dfff	ffff
SUBWF	f,d	Subtrai W de f	00	0010	dfff	ffff
SWAPF	f,d	Troca os nibles de f	00	1110	dfff	ffff
XORWF	f,d	XOR de W com f	00	0110	dfff	ffff
Para instruções orientadas ao bit						
BCF	f,b	Coloca bit de f a '0'	01	00bb	bfff	ffff
BSF	f,b	Coloca bit de f a '1'	01	01bb	bfff	ffff
BTFSC	f,b	Testa bit de f, salta de '0'	01	10bb	bfff	ffff
BTFSS	f,b	Testa bit de f, salta de '1'	01	11bb	bfff	ffff
Operações com literal, ou de controlo						
ADDLW	k	Adiciona literal a W	11	111x	kkkk	kkkk
ANDLW	k	AND de literal com W	11	1001	kkkk	kkkk
CALL	k	Chama sub-rotina	10	0kkk	kkkk	kkkk
GOTO	k	Salto incondicional	10	1kkk	kkkk	kkkk
IORLW	k	OR de literal com W	11	1000	kkkk	kkkk
MOVLW	k	Coloca literal em W	11	0000	kkkk	kkkk
RETLW	k	Retornar com literal em W	11	01xx	kkkk	kkkk
RETURN	-	Retornar da sub-rotina	00	0000	0000	1000
SUBLW	k	Subtrair W do literal	11	1110	xkkk	kkkk
XORLW	k	XOR de literal com W	11	1010	kkkk	kkkk

tabela 2: Instruções implementadas no microprocesador

Do "instruction set" do PIC16f84 não foram implementadas as funções relacionadas com o Watch Dog Timer (protecção para quando PIC bloqueia) e a função de SLEEP (função que coloca PIC em standby), pois são funções muito específicas do PIC. Não foram também implementadas as funções relacionadas com as interrupções, que poderão ser uma implementação futura.

VI. DESCRIÇÃO DO SISTEMA

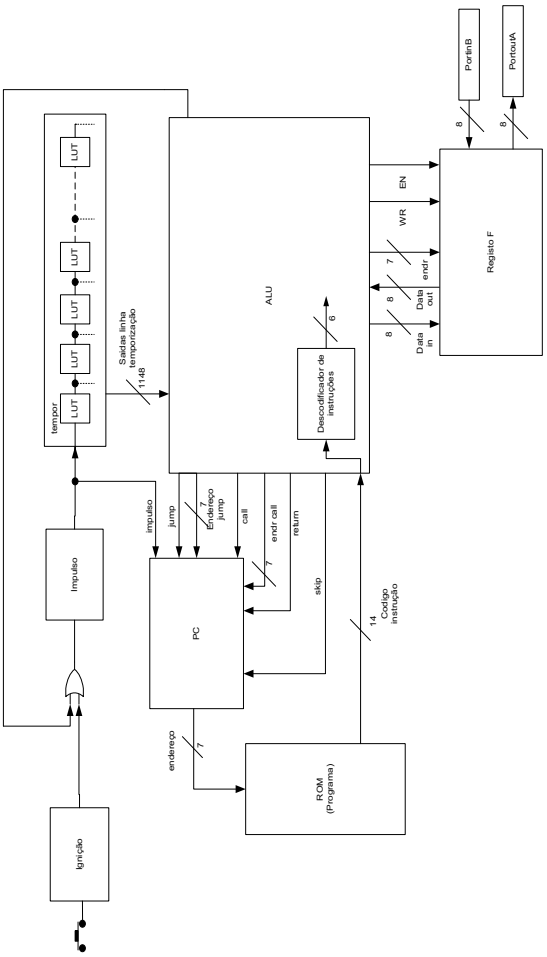


figura 3: diagrama de blocos do sistema

O sistema desenvolvido, esquematizado na figura 3, é constituído por oito blocos funcionais, principais: Ignição; Impulso; tempor; PC; ROM; ALU; Descodificador e RegistoF.
De seguida é descrito cada bloco constituinte do sistema, nomeadamente, entradas e saídas e princípio de funcionamento base.
Todo o processamento é controlado por um impulso que funciona como um sinalizador de início e de fim de uma operação. Este impulso é regenerado no bloco Impulso, a cada operação. Este circula através da linha de atraso tempor e volta ao início através do sinal de Done. A quantidade de LUT's percorridas, depende da instrução executada, e é controlada pela ALU.

Ignição
O bloco Ignição é o bloco que dá o impulso que despoletará todo o funcionamento do microprocessador. Este gera um impulso de duração muito curta, mas suficiente para que o bloco Impulso gere o sinal que irá controlar todas as operações.

Impulso
Este bloco, tal como foi já referido anteriormente, tem como função criar um impulso quadrado de duração limitada, aquando receber um impulso de início, do bloco de Ignição e regenerá-lo quando este regressar através do sinal de Done que indica o fim de operação.

Tempor
O bloco tempor consiste num conjunto de LUT's encadeadas em série e que formam uma linha de atraso. É esta linha de atraso que define o tempo de cada operação, dependendo este, da posição da linha à qual, a ALU ligue o sinal de Done.
Optou-se por uma linha de atraso única, e não uma para cada função, com vista a poupar espaço da FPGA, e devido ao facto de não ser necessário, pois como o microprocessador desenvolvido não possui pipeline, a execução simultânea de duas funções nunca se verifica.

ROM
A ROM é o bloco onde estão armazenados os códigos de instrução que serão executados ao longo do programa. Cada posição de ROM tem um tamanho de 14 bits, uma vez que é este o tamanho dos códigos de instrução utilizados, tal como foi já referido anteriormente. Esta possui apenas uma entrada onde o PC irá indicar a posição de memória da instrução a executar, e uma saída onde é colocado conteúdo dessa posição de memória, a qual corresponde a um código de instrução.

PC
O Contador de Programa é o bloco que controla qual a instrução a executar. Este a cada impulso incrementa a posição de memória da ROM uma unidade. No entanto é controlado também pela ALU, uma vez que existem determinadas instruções que na sua execução implicam que a próxima execução, não seja a instrução seguinte, mas sim outra posição distinta. Mais adiante será explicado o funcionamento deste tipo de instruções.

ALU
A ALU é o bloco principal do sistema, pois é neste bloco que são realizadas todas as operações, é este que controla todo o sistema.
Durante a execução do programa, esta recebe um código de instrução de 14 bits, a partir do qual reconhece qual a instrução a executar. Consoante a instrução em causa, o sinal Done é ligado a uma determinada posição da linha atraso, posição essa, que define o tempo de execução de cada instrução.

Descodificador
O descodificador é um bloco interno da *ALU*. É este bloco que lhe permite reconhecer de forma inequívoca, qual a instrução a executar. Recebe como entrada o *Código de instrução* com 14 bit, e a partir deste, gera um código de 6 bit correspondente à instrução em causa.

Registo F

Este bloco consiste numa memória RAM com 128 posições, em que cada posição tem 8 bits. Algumas das posições de memória correspondem a portos[2] de entrada ou de saída. Neste tipo de mapeamento de portos ler ou escrever num porto, é como ler ou escrever numa posição do registo, sendo utilizadas as mesmas instruções.

VII. Sistema de monitorização e processamento com o microprocessador

A. *Placa de desenvolvimento*

Para a implementação do microprocessador utilizaram-se as ferramentas de síntese de circuitos Project Navigator™ da Xilinx [1]. O circuito resultante foi implementado num FPGA Spartan II-E XC2S200E-PQ208, inserida numa placa de desenvolvimento D2-SB da Digilent [2].

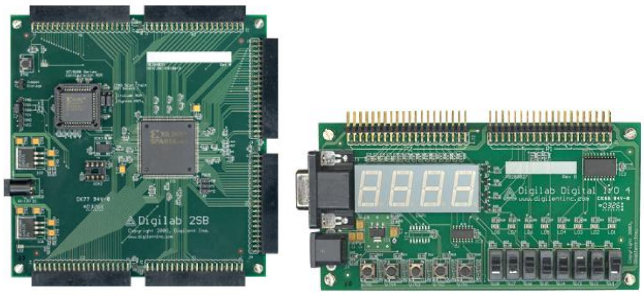


figura 4: Placa de desenvolvimento D2-SB com placa periférica DIO-4

A placa de desenvolvimento DIO-4 consiste numa placa de expansão da D2-SB, funcionando como interface entre o utilizador e o FPGA.

B. *Esquema básico do sistema*

O diagrama de blocos básico do sistema, é o apresentado na figura 5.

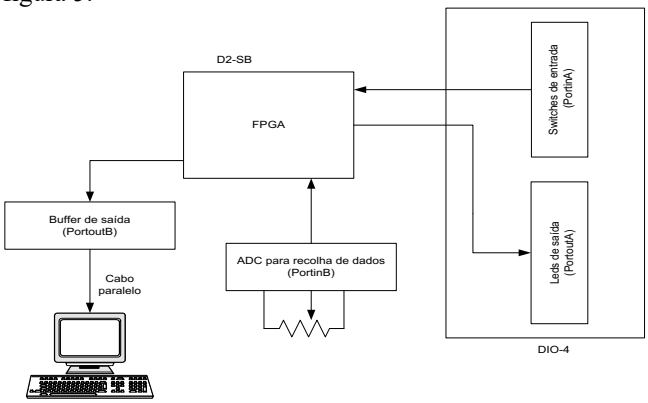


figura 5: Diagrama de blocos do sistema

A recolha de dados é feita com um ADC de oito bit, o *ADC0804* [3] e lida pelo microprocessador, através do porto *PortinB*.

[2] Memory maped IO

A ligação ao Computador, é feita através do porto paralelo (LPT), passando por um buffer *74LS244* [4], de forma a adaptar os níveis de tensão do FPGA (3V) e evitar que seja solicitada demasiada corrente do FPGA sob pena de que este seja danificado.

C. *Porto paralelo*

A interface paralela (LPT) constitui uma das interfaces mais utilizadas, em todo o mundo. Cerca de 99% dos computadores possuem esta interface.

Modos de funcionamento da porta paralela:
- Transmissão unidireccional

No modo SPP (Standard Parallel Port) o ritmo de transmissão de dados, pode chegar aos 150 kB/s.
- Transmissão bidireccional

No modo EPP (Enhanced Parallel Port), utilizando um cabo adequado, o ritmo de transmissão poderá chegar aos 2 MB/s.
No modo ECP (Enhanced Capabilities Port) tem as mesmas características que a EPP, porém, utiliza DMA (acesso directo à memória), sem a necessidade do uso do processador, para a transferência de dados.
Cada porto paralelo é constituído por 3 registos de 8 bits
- Registo de dados
- Registo de estado
- Registo de controlo

A extensão do cabo para interligar um computador a um periférico, é de no máximo 8 m. Na prática, utiliza-se um cabo com extensão menor. Quanto maior a extensão do cabo, maior é a interferência na transmissão dos dados.

D. *Aplicação de monitorização, no computador*

O software de programação escolhido para a elaboração da aplicação foi o C++ Builder™. Esta opção deveu-se principalmente à enorme diversidade de ferramentas que este software disponibiliza para a elaboração de aplicações.
Uma outra vantagem deste software é o facto de permitir a utilização directa de funções do sistema operativo, o que permite interagir directamente com este, melhorando o desempenho da aplicação.
A interface da aplicação desenvolvida é a apresentada na figura 6.

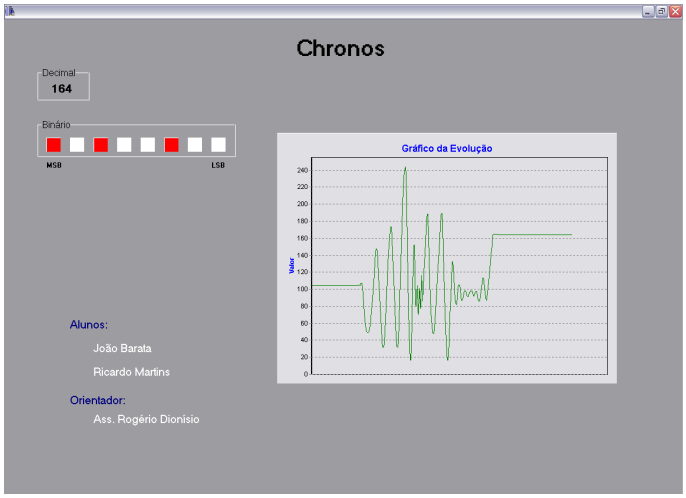


figura 6: Interface da aplicação desenvolvida

A aplicação apresenta o valor decimal enviado pelo microprocessador, e o seu código binário. Apresenta também um gráfico da evolução do valor, ao longo do tempo.
A aquisição do valor do registo do porto paralelo é feita através da seguinte função:

```
VALOR=DlPortReadPortUchar((unsigned
short)(PORTO_PARALELO));
```

Esta função advém da instalação de um *Driver* de sistema para C++, que possibilita o acesso directo aos registos da LPT, em qualquer sistema operativo *Windows*.

E. Instruções executadas no microprocessador

O conjunto de instruções executadas no microprocessador realiza a seguinte sequência de operações:

- Lê o valor do ADC
- Ao valor lido soma-lhe o valor definido, em binário, pela posição dos switches da placa DIO-4.
- O resultado da soma é mostrado nos LEDs da placa DIO-4 e enviado para a aplicação do computador, descrita anteriormente.

Na tabela 3 é apresentada a sequência de instruções executadas.

Código assembly	Código máquina	Descrição
MOVF 06h	00 1000 0000 0110	Lê o valor do porto 6
ADDWF 02h,0	00 0111 0000 0010	Soma o conteúdo do acumulador ao valor do porto 2
MOVWF 05h	00 0000 1000 0101	Escreve o valor do acumulador no porto 5
MOVWF 03h	00 0000 1000 0011	Escreve o valor do acumulador no porto 3
GOTO 00h	10 1000 0000 0000	Volta à posição inicial

tabela 3: Instruções executadas

VIII. Conclusão

Neste projecto foi desenvolvida uma implementação assíncrona, em FPGA, de um microprocessador. Este microprocessador é capaz de processar comandos idênticos aos dos microcontroladores PIC da Microchip®.
O microprocessador desenvolvido foi utilizado numa aplicação simples de monitorização, que tem como finalidade a verificação prática do funcionamento do sistema. Foi também elaborado um programa básico, em PC, que comunica com o microprocessador através da interface paralela (LPT).
Durante o desenvolvimento do microprocessador assíncrono deparou-se com uma das principais desvantagens dos sistemas assíncronos, que foram os problemas relacionados com tempos definidos para a execução dos blocos funcionais.

IX. Referências

[1] – Xilinx, Inc., “Xilinx Sysnthesis Tecnology (XST) User Guide”, Xilinx, Inc., 2002 (http://support.xilinx.com/support/techsup/tutorials/index.htm)
[2] – Digilent, “Digilent D2-SB System Board Reference Manual”, Digilent, Inc., 2004 (http://www.digilentinc.com/Data/Products/D2SB/D2SB-rm.pdf)
[3] – National Semiconductor, “ADC0804 - 8-Bit µP Compatible A/D Converters”, National Semiconductor, 1999 (http://www.national.com/mpf/DC/ADC0804.html)
[4] – Texas Instruments, “SN74LS244N - Octal Buffer/Line Driver With 3-State Outputs”, Texas Instruments, 1985 (http://focus.ti.com/lit/ds/sdls144b/sdls144b.pdf)