Implementação de um controlador de semáforos numa FPGA com visualização num monitor VGA

Ana Cantanhede

***Resumo* - O presente artigo descreve a implementação de um controlador de semáforos numa FPGA, sendo a visualização do estado dos semáforos efectuada num monitor VGA, ligado à FPGA. O circuito controlador foi descrito na linguagem VHDL no ambiente ISE 6.3 da Xilinx e testado na placa TE-XC2Se da Trenz Electronic, que inclui uma FPGA XC2S300E da família Spartan-IIE da Xilinx.**

***Abstract* – This paper describes a FPGA-based implementation of a traffic-light controller. The traffic lights are displayed on a VGA monitor which is connected to the FPGA. The control circuit was described in VHDL language within the Xilinx ISE 6.3 environment and tested on TE-XC3Se board from Trenz Electronic which includes one XC2S300E FPGA of Spartan-IIE Xilinx family.**

I. INTRODUÇÃO

O presente artigo surge de um projecto [1] realizado no âmbito da cadeira de Computação Reconfigurável (licenciatura em Engenharia de Computadores e Telemática do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro), leccionada no 2º semestre do ano lectivo 2004/2005.

A. Objectivos

Pretende-se implementar um controlador de semáforos numa FPGA, sendo a visualização do estado dos semáforos efectuada num monitor VGA, ligado à FPGA. O circuito controlador foi descrito na linguagem VHDL no ambiente ISE 6.3 [2] da Xilinx e testado na placa TE-XC2Se [3] da Trenz Electronic, que inclui uma FPGA XC2S300E da família Spartan-IIE da Xilinx.

A configuração do cruzamento em que a implementação foi efectuada é a apresentada na figura 1.

Pretendia-se que, em funcionamento autónomo, o semáforo verde estivesse ligado durante 20 segundos na rua A e durante 10 segundos na rua B. A passagem de verde para vermelho era sempre antecedida por um período de 3 segundos em amarelo e havia ainda um período de 2 segundos que mediava o início de vermelho numa rua e o início de verde noutra. O circuito deveria ainda poder funcionar em modo manual, sendo controlado pelos interruptores da placa.

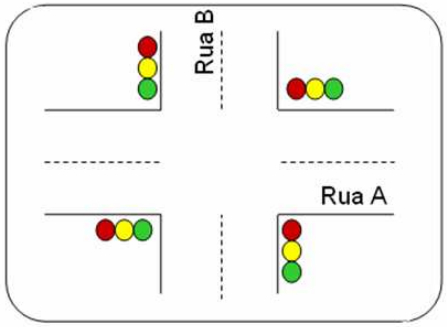

Fig. 1 – Configuração do cruzamento a implementar o controlador de semáforos

Por fim, era ainda necessário efectuar a simulação em ModelSim.

II. ARQUITECTURA DO SISTEMA

A placa TE-XC2Se, que foi utilizada para as experiências, é ilustrada na figura 2.

Ao conector VGA liga-se o cabo do monitor VGA e cada interruptor é utilizado para controlar uma das cores de um dos semáforos, no caso do sistema estar a funcionar em modo manual. A correspondência entre cada interruptor e a luz do semáforo de cada rua é a apresentada na tabela 1.

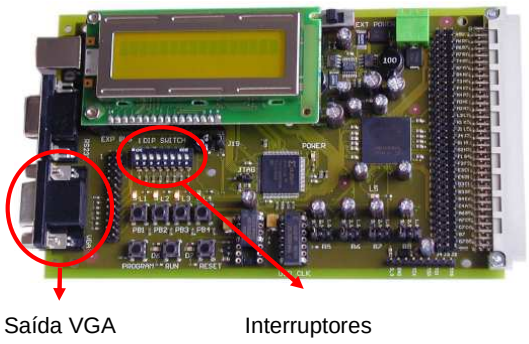

Fig. 2 – Placa TE-XC2Se, da *Trenz*, utilizada nas experiências

Nº Interruptor	Cor do Semáforo / Rua
1	Vermelho / A
2	Amarelo / A
3	Verde / A
4	Vermelho / B
5	Amarelo / B
6	Verde / B
7	———————————
8	Modo Manual (0) ou Automático (1)

Tabela 1 – Correspondência entre o número do interruptor e a luz do semáforo da rua que controla.

Adicionalmente, e uma vez que apenas faz sentido estar no máximo uma das luzes ligadas em cada uma das ruas, apenas foram implementadas as combinações de cores que obedeciam a este princípio.

Na figura 3 é apresentada uma fotografia do sistema que foi montado de modo a realizar este trabalho.

Fig 3. – Sistema montado para realizar a experiência. No monitor da esquerda é apresentado o cruzamento já descrito e a luz dos semáforos ligados de cada uma das ruas

De referir que apenas um monitor VGA é suficiente. Porém, de modo a ser mais rápido efectuar testes (de forma a não estar constantemente a comutar o cabo do monitor VGA), pode-se utilizar um monitor auxiliar que serve apenas para efectuar a visualização dos resultados.

III. IMPLEMENTAÇÃO

De forma a implementar o controlador de semáforos, foram construídos quatro módulos, nomeadamente:

- *char_rom* – contém a forma gráfica de cada uma das luzes dos semáforos (define como desenhar um círculo no monitor);
- *cpld* – descreve a leitura do estado de cada um dos interruptores;
- *statemachine* – contém a máquina de estados finitos utilizada para controlar os semáforos (em modo automático) e processa também a saída de acordo com o estado dos interruptores (em modo manual);
- *vga* – de acordo com o estado de cada semáforo, essa informação é mostrada no monitor.

Estes módulos são interligados conforme mostra o esquemático apresentado na figura 4.

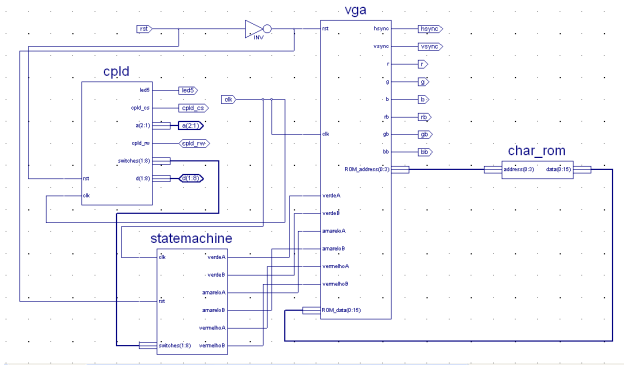

Fig. 4 – Esquemático do sistema implementado

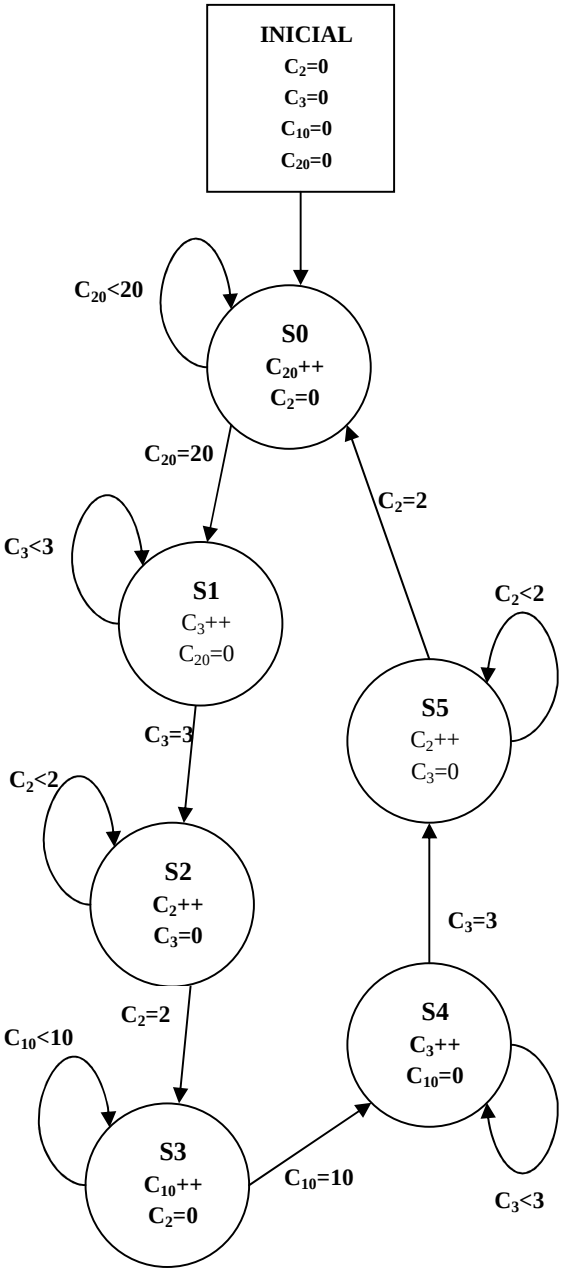

Fig 5. – Máquina de estados finitos implementada

Estado	Rua A	Rua B
Inicial	Inactivo	Inactivo
S0	Verde	Vermelho
S1	Amarelo	Vermelho
S2	Vermelho	Vermelho
S3	Vermelho	Verde
S4	Vermelho	Amarelo
S5	Vermelho	Vermelho

Tabela 2 – Relação entre cada estado e a luz do semáforo activa em cada uma das ruas

A máquina de estados finitos implementada, de modo a modelar o sistema pretendido, é a apresentada na figura 5.

Uma vez que temos 4 tempos diferentes envolvidos (nomeadamente, 2, 3, 10 e 20 segundos, foram criados 4 contadores, respectivamente, C_2, C_3, C_{10} e C_{20}). Na tabela 2 é apresentada a relação entre o estado do diagrama de estados e o estado dos semáforos em cada rua.

No módulo *statemachine*, com a ajuda dos sinais de *clock* e *reset* e com o estado dos interruptores é gerada a cor do semáforo de cada rua.

De forma a implementar a funcionalidade requerida, como já foi referido, foram necessários quatro contadores. Uma vez que a placa utilizada possui geradores de relógio de 25 e de 48 MHz, e de forma a simplificar o incremento dos contadores, utilizou-se o de 25 MHz pois, com este gerador de relógio, a diferença entre o tempo real e o tempo obtido pelo sistema é menor.

O código utilizado na implementação do módulo *statemachine* é o seguinte:

```
process(state, switches, clk_aux )
begin
    if (clk_aux'event and clk_aux='1') then
 if (switches(8) = '1') then -- modo automático
            case state is
                when INICIO =>
        -- estado inicial – semáforos apagados
                        verdeA   <= '0';
                        verdeB   <= '0';
                        amareloA <= '0';
                        amareloB <= '0';
                        vermelhoA <= '0';
                        vermelhoB <= '0';
                        c2 <= 0;
                        c3 <= 0;
                        c10 <= 0;
                        c20 <= 0;
                        next_state <= S0;
        -- luz verde ligada na rua A e luz vermelha ligada na rua B
                when S0 => verdeA   <= '1';
                        verdeB   <= '0';
                        amareloA <= '0';
                        amareloB <= '0';
                        vermelhoA <= '0';
                        vermelhoB <= '1';
                        if (c20 < 20) then
                                c20 <= c20 + 1;
                                next_state<=S0;
                        else
                                c20 <= 0;
                                next_state<=S1;
                        end if;
                when S1 =>
        -- amarelo ligado na rua A e vermelho ligado na rua B
                                ...     -- outros estados
                end case;
    else  -- modo manual
next_state <= INICIO; -- de forma a garantir que quando o interruptor
-- for alterado para o modo automático, a máquina de estados volta ao
-- estado inicial
            case switches(1 to 6) is
                -- tudo apagado
                when "000000" =>  verdeA   <= '0';
                                  verdeB   <= '0';
                                  amareloA <= '0';
                                  amareloB <= '0';
                                  vermelhoA <= '0';
                                  vermelhoB <= '0';

                -- A - todos apagados,  B - verde ligado
                when "000001" =>  verdeA   <= '0';
                                  verdeB   <= '1';
                                  amareloA <= '0';
                                  amareloB <= '0';
                                  vermelhoA <= '0';
                                  vermelhoB <= '0';
                when "000010" =>
                                  ...     -- outros casos
                end case;
            end if;
    end if;
end process;
```

De referir que, quando mais do que um dos interruptores é ligado, numa mesma rua, mantem-se ligada a luz correspondente ao interruptor que foi ligado em primeiro lugar.

O código apresentado é também o utilizado na simulação efectuada no Modelsim.

Relativamente ao módulo *vga*, a estratégia utilizada foi dividir o monitor em pequenas áreas de 16x16 pixels, de forma a que cada área representasse um caracter (no caso em estudo, uma das cores de um dos semáforos). Para cada uma das áreas com a mesma cor de semáforo foi definida a cor do fundo e do *foreground*. De seguida, de acordo com o estado dos sinais que representam a cor dos semáforos em cada uma das ruas, são apresentadas no monitor as cores apropriadas.

IV. SIMULAÇÃO

O código utilizado para efectuar a simulação no ModelSim é o já apresentado, sendo que o ficheiro *stimulus* utilizado tem o seguinte conteúdo:

vsim -t sec work.statemachine
view w
*add w **
force rst 1 0 sec, 0 1 sec
force clk 0 0 sec, 1 1 sec -repeat 2 sec

force switches "00000000" 0 sec, "10000000" 2 sec, "01001000" 3 sec, "11111111" 10 sec, "00100100" 60 sec, "00000001" 70 sec

De notar que a simulação foi efectuada em segundos de modo a apresentar os resultados exactos e não os aproximados (o que acontece na placa).

Na figura 6 é apresentada parte do resultado da simulação.

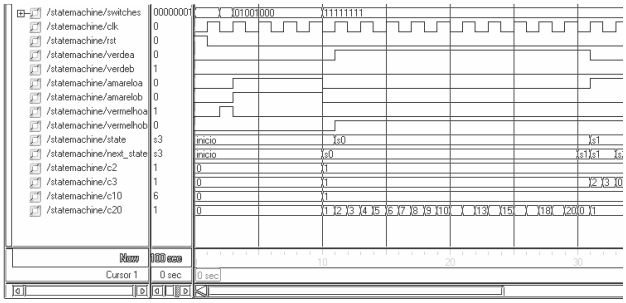

Fig. 6 – Resultado de uma simulação efectuada no ModelSim

V. CONCLUSÃO

Foram sentidas algumas dificuldades no início do trabalho, nomeadamente na forma de visualizar diferentes áreas do monitor com cores diferentes. Esta dificuldade foi ultrapassada, tal como outras menores que foram surgindo.

Conseguiu-se implementar o controlador de semáforos tal como foi descrito nos objectivos, pelo que a avaliação do trabalho elaborado é bastante positiva.

REFERÊNCIAS

[1] http://www.ieeta.pt/~iouliia/Courses/CR/index.html
[2] www.xilinx.com
[3] http://www.trenz-electronic.de/down/xc2se/pbxc2se.pdf