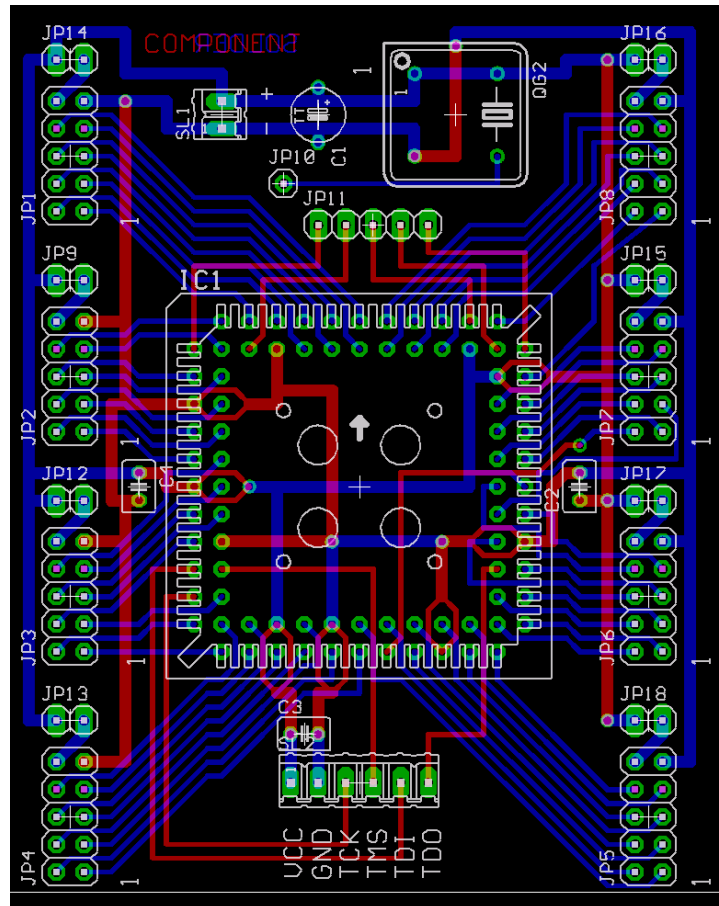


Sistema de Limpeza e Alimentação para uma Jaula



Curso:

Engenharia Electrotécnica e de Computadores

Elaborado por:

Miguel Fernandes nº28159

Diogo Cardoso nº28192

André Fidalgo nº28193

Docente Responsável:

Prof. Ricardo Gonçalves

Data de Entrega: 30/01/2009

Índice

Introdução	3
Enquadramento teórico	4
Implementação.....	7
Dimensionamento	7
1. Máquina de Estados	7
2. Tabela de verdade	9
3. Simplificações de funções através dos Mapas de Karnaugh.....	10
Montagem	12
1. Esquema	12
2. Explicação esquema	16
3. Lista de Material	18
Conclusão.....	19
Bibliografia.....	20

Introdução

Este trabalho tem como principal objectivo a montagem de um sistema de limpeza e alimentação para uma jaula, através de conceitos teóricos e práticos adquiridos nas aulas de Sistemas Lógicos ao longo do semestre.

O sistema funciona da seguinte maneira: Para ser iniciada a alimentação é necessário accionar um interruptor. Enquanto o animal está a ser alimentado, a jaula não pode ser limpa; O período de alimentação demora quatro ciclos de relógio; Para iniciar um novo período de alimentação será necessário terminar o anterior manualmente através do interruptor; Após oito refeições, a jaula será automaticamente limpa; O período de limpeza demora quatro ciclos; Apenas é possível iniciar um período depois de já ter terminado o anterior. O sistema tem de permitir visualizar se o animal está a ser alimentado, se a jaula está a ser limpa e a quantidade de refeições actual.

Para a resolução do problema que nos foi proposto começámos pela implementação de uma máquina de quatro estados distintos: estado inicial, estado de alimentação, estado de espera e estado de limpeza. De seguida fizemos a tabela de verdade onde temos os estados de transição. Depois, com base nos mapas de Karnaugh construídos a partir dos valores da tabela obtivemos as expressões simplificadas de que necessitávamos para a implementação dos Flip-Flops JK que nos permitem demonstrar em termos prático o funcionamento da máquina.

Enquadramento Teórico

Para a realização deste trabalho necessitámos dos conhecimentos adquiridos, tanto nas aulas teóricas, como da experiência adquirida nas aulas práticas.

Neste trabalho temos que ter em especial atenção os conhecimentos adquiridos sobre Álgebra de Boole, Portas Lógicas, Flip-Flops, mapas de Karnaugh e Contadores.

A Álgebra de Boole é um ramo da Matemática em que as variáveis só podem tomar dois valores (0 ou 1), obedecendo assim à lógica binária. Existem uma série de regras (propriedades, postulados, teoremas) associadas à Álgebra de Boole que permitem a manipulação das funções. Na Álgebra de Boole as variáveis, denominadas binárias, têm apenas dois significados lógicos: verdadeiro ou falso. Estes dois valores representam-se simbolicamente por 1 e 0, respectivamente. Os símbolos 1 e 0 não exprimem quantidades mas estados das variáveis.

Durante a realização deste trabalho, foram usadas as funções lógicas AND (e), OR (ou) e NOT (negação).

As portas lógicas são circuitos integrados que segundo as regras da álgebra de Boole realizam a tarefa de implementar as funções booleanas por estas idealizadas.



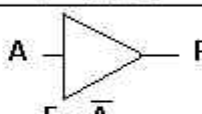
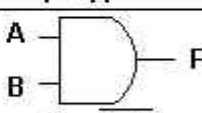

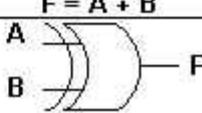
Nome	Símbolo	Entradas		Saídas
		A	B	F
E (AND)	 $F = A \cdot B$	0	0	0
		0	1	0
		1	0	0
		1	1	1
OU (OR)	 $F = A + B$	0	0	0
		0	1	1
		1	0	1
		1	1	1
INVERSORA (NOT)	 $F = \bar{A}$	0		1
		1		0
NÃO E (NAND)	 $F = \overline{A \cdot B}$	0	0	1
		0	1	1
		1	0	1
		1	1	0
NÃO OU (NOR)	 $F = \overline{A + B}$	0	0	1
		0	1	0
		1	0	0
		1	1	0
OU EXCLUSIVO (XOR)	 $F = A \oplus B = \bar{A}B + A\bar{B}$	0	0	0
		0	1	1
		1	0	1
		1	1	0

Figura 1 – Esquema breve sobre os símbolos e expressões algébricas de cada função lógica usadas nos respectivos circuitos integrados.

O método gráfico dos mapas de Karnaugh, permite-nos de uma forma simples realizar a tarefa de simplificar funções.

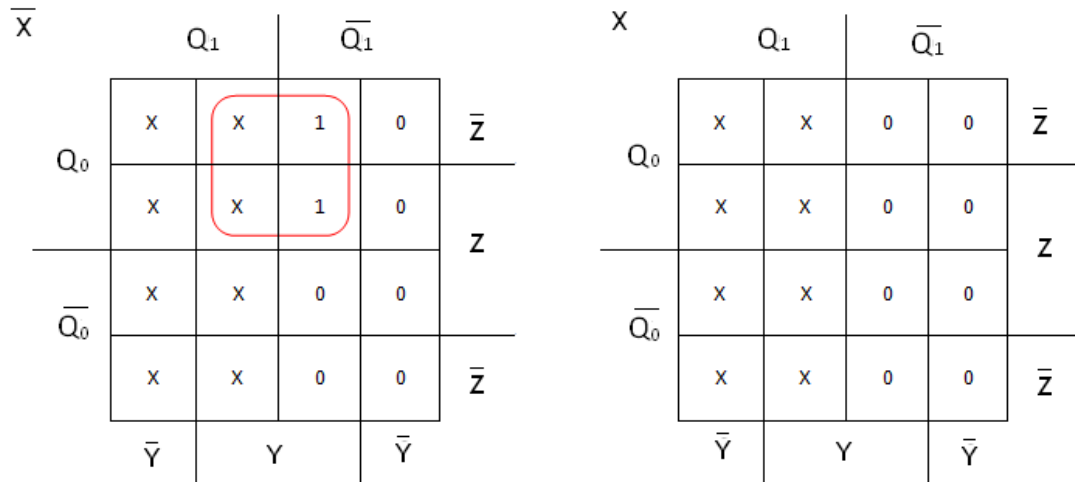
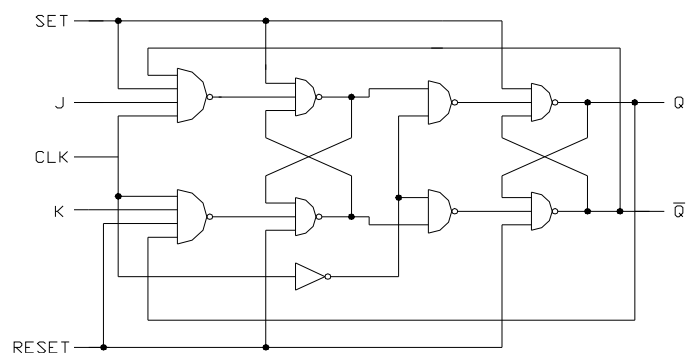


Figura 2- Exemplo de mapas de Karnaugh.

Os Flip-Flops são bastante úteis, sendo os mais importantes os tipos D e J-K, pois a partir destes dois tipos mais elementares conseguimos recriar os resultados de outros. Resumidamente são circuitos sequenciais constituídos por portas lógicas capazes de armazenar 1 bit que vai ser usado para determinar o estado seguinte da saída do Flip-Flop.

J	K	Q_n	Q_{n+1}
0	0	0	0 (Q_n)
0	0	1	1 (Q_n)
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Figuras 3 e 4 – Tabela de verdade do Flip-Flop JK (à esquerda) e esquema lógico do mesmo (à direita).

Q_t	\longrightarrow	Q_{t+1}	J_t	K_t
0	\longrightarrow	0	0	X
0	\longrightarrow	1	1	X
1	\longrightarrow	0	X	1
1	\longrightarrow	1	X	0

Figura 5 – Tabela de excitação do Flip-Flop JK.

Um contador é também um circuito sequencial, que basicamente faz aquilo que o nome diz, conta o número de impulsos na entrada e fá-los sair nas suas saídas na forma de um número em binário.

Conta	Saídas			
	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figura 6 – Tabela de um contador

O CPLD (Complex programmable logic device) é uma macrócelula que nos permite implementar funções lógicas combinacionais. Isto dá-nos a hipótese de “montar” o circuito que pretendemos de um modo muito mais rápido, fácil e prático, tornando-o menos propício a falhas, tais como maus contactos de fios, fios partidos, e outros acontecimentos similares que são bastante frequentes em montagens desta complexidade.

O CPLD é programado através de um programa de computador, neste caso o Xilinx, que nos permite implementar as funções, testá-las e encapsula-las para se posteriormente as quisermos voltar a usar. Após ser criado o esquema do circuito, basta “introduzi-lo” no CPLD através desse mesmo programa.

Implementação

Dimensionamento

1 - Máquina de estados

Após analisarmos o que nos foi proposto, evidenciámos que este projecto se baseia numa máquina de estados. Posto isto, procedemos à verificação de quantos estados são necessários para implementar o sistema de limpeza e alimentação e o modo como se interligam.

Para concluir, desenhámos o diagrama de estados da máquina que indica quais as condições necessárias à passagem entre os vários estados.

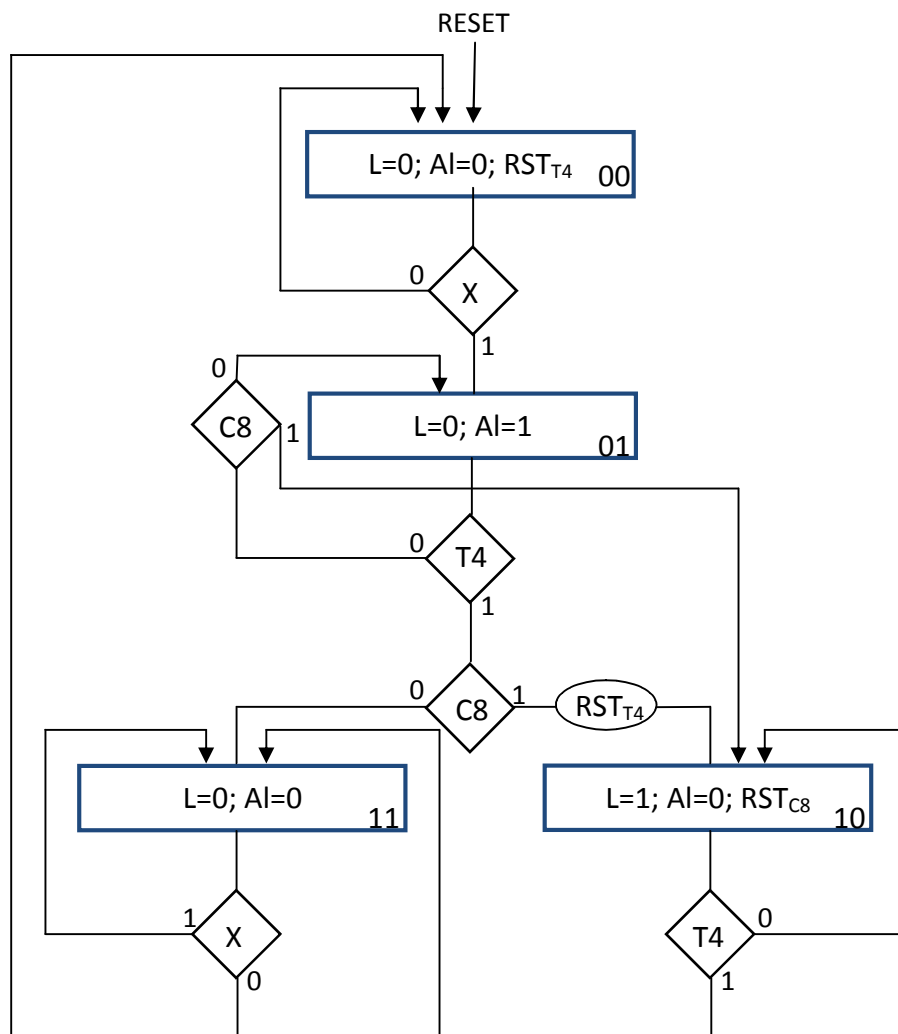


Figura 7 – Diagrama de estados

1.1 - Legenda do diagrama de estados

X – Interruptor;

T4 – Contador até 4 (ciclos que demora a alimentação/limpeza);

C8 – Contador até 8 (número de alimentações);

L – Limpeza da jaula;

AI – Alimentação do animal;

RST₄ – Reset ao contador de 4;

RST₈ – Reset ao contador de 8.

O estado actual encontra-se no canto inferior direito de cada estado.

2 - Tabela de verdade

Após a construção do diagrama de estados, foi feita a tabela da verdade que contém todos os inputs e outputs do nosso sistema, assim como o estado em que se encontra ($Q(t)$) e para onde deve evoluir o sistema ($Q(t+1)$). Como o sistema depende de quatro estados são necessários dois Flip-Flops JK, correspondendo cada um a uma variável (com duas variáveis obtêm-se 4 combinações diferentes). Analisando o diagrama de estados (Figura 7) é possível concluir que os inputs do sistema são o interruptor, o contador de 4 e o contador de 8. Quanto aos outputs temos o reset ao contador de 4, o reset ao contador de 8, a limpeza e a alimentação.

	Q(t)		Input			Q(t+1)		Entradas dos FF-JK				Output			
	Q1	Q0	X	T4	C8	Q1	Q0	J1	K1	J0	K0	L	AI	RST8	RST4
0	0	0	0	0	0	0	0	0	x	0	x	0	0	0	1
1	0	0	0	0	1	0	0	0	x	0	x	0	0	0	1
2	0	0	0	1	0	0	0	0	x	0	x	0	0	0	1
3	0	0	0	1	1	0	0	0	x	0	x	0	0	0	1
4	0	0	1	0	0	0	1	0	x	1	x	0	0	0	1
5	0	0	1	0	1	0	1	0	x	1	x	0	0	0	1
6	0	0	1	1	0	0	1	0	x	1	x	0	0	0	1
7	0	0	1	1	1	0	1	0	x	1	x	0	0	0	1
8	0	1	0	0	0	0	1	0	x	x	0	0	1	0	0
9	0	1	0	0	1	1	0	1	x	x	1	0	1	0	0
10	0	1	0	1	0	1	1	1	x	x	0	0	1	0	0
11	0	1	0	1	1	1	0	1	x	x	1	0	1	0	1
12	0	1	1	0	0	0	1	0	x	x	0	0	1	0	0
13	0	1	1	0	1	1	0	1	x	x	1	0	1	0	0
14	0	1	1	1	0	1	1	1	x	x	0	0	1	0	0
15	0	1	1	1	1	1	0	1	x	x	1	0	1	0	1
16	1	0	0	0	0	1	0	x	0	0	x	1	0	1	0
17	1	0	0	0	1	1	0	x	0	0	x	1	0	1	0
18	1	0	0	1	0	1	1	x	0	1	x	1	0	1	0
19	1	0	0	1	1	1	1	x	0	1	x	1	0	1	0
20	1	0	1	0	0	1	0	x	0	0	x	1	0	1	0
21	1	0	1	0	1	1	0	x	0	0	x	1	0	1	0
22	1	0	1	1	0	1	1	x	0	1	x	1	0	1	0
23	1	0	1	1	1	1	1	x	0	1	x	1	0	1	0
24	1	1	0	0	0	0	0	x	1	x	1	0	0	0	0
25	1	1	0	0	1	0	0	x	1	x	1	0	0	0	0
26	1	1	0	1	0	0	0	x	1	x	1	0	0	0	0
27	1	1	0	1	1	0	0	x	1	x	1	0	0	0	0
28	1	1	1	0	0	1	1	x	0	x	0	0	0	0	0
29	1	1	1	0	1	1	1	x	0	x	0	0	0	0	0
30	1	1	1	1	0	1	1	x	0	x	0	0	0	0	0
31	1	1	1	1	1	1	1	x	0	x	0	0	0	0	0

Figura 8 – Tabela de verdade

3 – Simplificação das funções através dos mapas de Karnaugh

Os mapas de Karnaugh servem para retirar através da tabela anterior as funções J_0 , K_0 , J_1 , K_1 e outputs simplificadas. Como temos 5 variáveis, vamos utilizar dois mapas de 4 variáveis para cada função.

<div><div>!Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>0</td><td>X</td><td>X</td><td>0</td><td>!X</td></tr><tr><td>1</td><td>X</td><td>X</td><td>1</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>1</td><td>X</td><td>X</td><td>1</td><td></td></tr><tr><td>0</td><td>X</td><td>X</td><td>0</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	0	X	X	0	!X	1	X	X	1	X	!C8	1	X	X	1		0	X	X	0	!X		!Q0		Q0		!Q0	<div><div>Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>1</td><td>X</td><td>X</td><td>0</td><td>!X</td></tr><tr><td>1</td><td>X</td><td>X</td><td>0</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>1</td><td>X</td><td>X</td><td>0</td><td></td></tr><tr><td>1</td><td>X</td><td>X</td><td>0</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	1	X	X	0	!X	1	X	X	0	X	!C8	1	X	X	0		1	X	X	0	!X		!Q0		Q0		!Q0	J0
	T4		!T4																																																																													
C8	0	X	X	0	!X																																																																											
	1	X	X	1	X																																																																											
!C8	1	X	X	1																																																																												
	0	X	X	0	!X																																																																											
	!Q0		Q0		!Q0																																																																											
	T4		!T4																																																																													
C8	1	X	X	0	!X																																																																											
	1	X	X	0	X																																																																											
!C8	1	X	X	0																																																																												
	1	X	X	0	!X																																																																											
	!Q0		Q0		!Q0																																																																											
<div>J0 = X. !Q1 + T4.Q1</div>																																																																																

<div><div>!Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>x</td><td>1</td><td>1</td><td>x</td><td>!X</td></tr><tr><td>x</td><td>1</td><td>1</td><td>x</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>x</td><td>0</td><td>0</td><td>x</td><td></td></tr><tr><td>x</td><td>0</td><td>0</td><td>x</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	x	1	1	x	!X	x	1	1	x	X	!C8	x	0	0	x		x	0	0	x	!X		!Q0		Q0		!Q0	<div><div>Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>x</td><td>1</td><td>1</td><td>x</td><td>!X</td></tr><tr><td>x</td><td>0</td><td>0</td><td>x</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>x</td><td>0</td><td>0</td><td>x</td><td></td></tr><tr><td>x</td><td>1</td><td>1</td><td>x</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	x	1	1	x	!X	x	0	0	x	X	!C8	x	0	0	x		x	1	1	x	!X		!Q0		Q0		!Q0	K0
	T4		!T4																																																																													
C8	x	1	1	x	!X																																																																											
	x	1	1	x	X																																																																											
!C8	x	0	0	x																																																																												
	x	0	0	x	!X																																																																											
	!Q0		Q0		!Q0																																																																											
	T4		!T4																																																																													
C8	x	1	1	x	!X																																																																											
	x	0	0	x	X																																																																											
!C8	x	0	0	x																																																																												
	x	1	1	x	!X																																																																											
	!Q0		Q0		!Q0																																																																											
<div>K0 = C8. !Q1 + !X.Q1</div>																																																																																

<div><div>!Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>0</td><td>1</td><td>1</td><td>0</td><td>!X</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>0</td><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	0	1	1	0	!X	0	1	1	0	X	!C8	0	1	0	0		0	1	0	0	!X		!Q0		Q0		!Q0	<div><div>Q1</div><table><tr><td></td><td colspan="2">T4</td><td colspan="2">!T4</td><td></td></tr><tr><td rowspan="2">C8</td><td>x</td><td>x</td><td>x</td><td>x</td><td>!X</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>X</td></tr><tr><td rowspan="2">!C8</td><td>x</td><td>x</td><td>x</td><td>x</td><td></td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>!X</td></tr><tr><td></td><td colspan="2">!Q0</td><td colspan="2">Q0</td><td>!Q0</td></tr></table></div>							T4		!T4			C8	x	x	x	x	!X	x	x	x	x	X	!C8	x	x	x	x		x	x	x	x	!X		!Q0		Q0		!Q0	J1
	T4		!T4																																																																													
C8	0	1	1	0	!X																																																																											
	0	1	1	0	X																																																																											
!C8	0	1	0	0																																																																												
	0	1	0	0	!X																																																																											
	!Q0		Q0		!Q0																																																																											
	T4		!T4																																																																													
C8	x	x	x	x	!X																																																																											
	x	x	x	x	X																																																																											
!C8	x	x	x	x																																																																												
	x	x	x	x	!X																																																																											
	!Q0		Q0		!Q0																																																																											
<div>J1 = T4.Q0 + C8.Q0</div>																																																																																

!Q1						Q1						K1
C8	T4		!T4		!X	C8	T4		!T4		!X	
	x	x	x	x			0	1	1	0		
!C8	x	x	x	x	X	!C8	0	0	0	0	X	
	x	x	x	x			0	1	1	0		
!Q0		Q0		!Q0		!Q0		Q0		!Q0		
K1 = !X.Q0												

Figura 9 – Mapas de Karnaugh das funções J_0 , K_0 , J_1 e K_1

A concepção da máquina de estados é feita através da implementação das funções obtidas anteriormente, ligando-as de forma correcta às entradas dos Flip-Flops JK.

Recordemos então as funções de entrada nos Flip-Flops JK:

$$J_0 = X \cdot !Q_1 + T_4 \cdot Q_1$$

$$K_0 = C_8 \cdot !Q_1 + !X \cdot Q_1$$

$$J_1 = T_4 \cdot Q_0 + C_8 \cdot Q_0$$

$$K_1 = !X \cdot Q_0$$

Quanto aos outputs, obtivemos as seguintes funções, pelo processo anterior:

$$A1 = Q_0 \cdot !Q_1$$

$$L = !Q_0 \cdot Q_1$$

$$RST_4 = !Q_0 \cdot !Q_1 + !Q_1 \cdot T_4 \cdot C_8$$

$$RST_8 = !Q_0 \cdot Q_1$$

Através de todos estes processos é-nos possível construir o Sistema de Alimentação e Limpeza na totalidade, pois já temos todas as informações necessárias para o fazer.

Montagem

1 - Esquema

Para desenhar o esquema do Sistema de Alimentação e Limpeza recorreremos ao programa Xilinx que posteriormente vai servir para programar o CPLD. Para tornar o esquema mais simples, dividimo-lo em três partes distintas: a máquina de estados (controlador do sistema), a máquina de estados ligada aos periféricos, e todo o sistema montado, já com os outputs incluídos.

1.1 – Máquina de estados

Maquina_estados

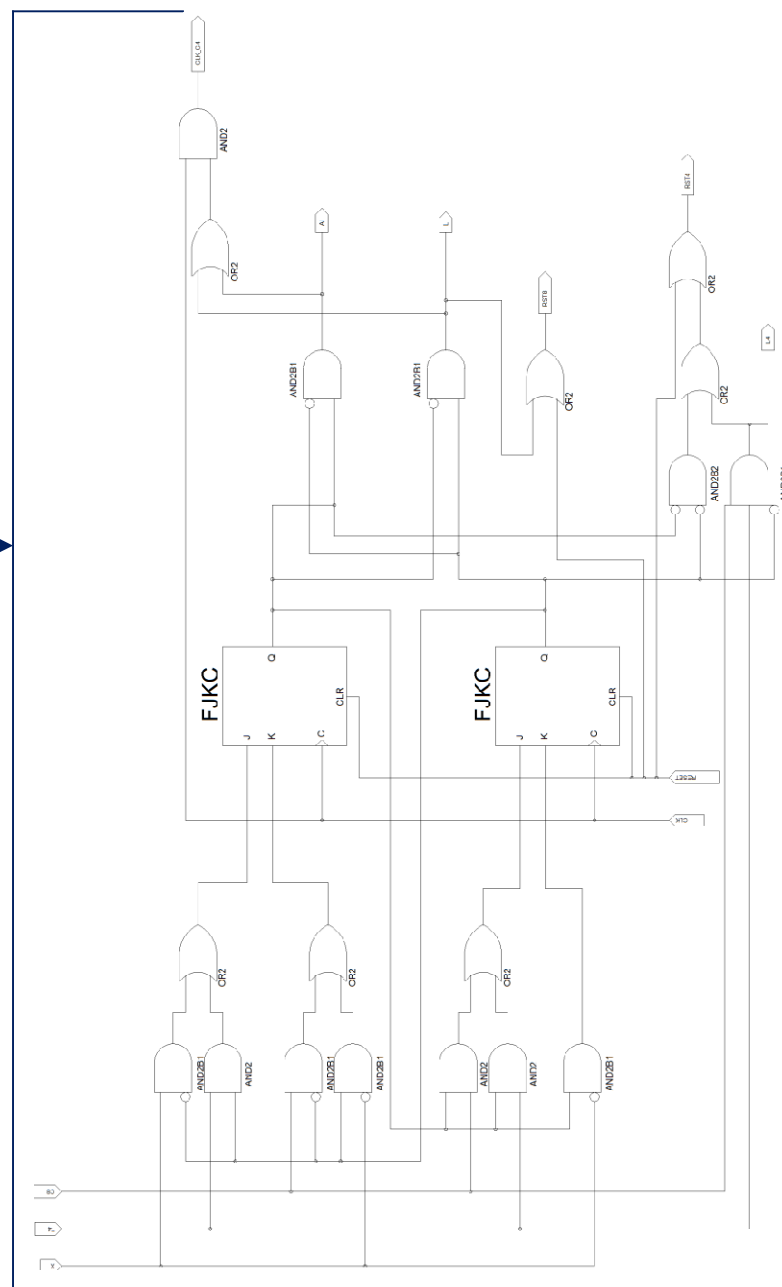
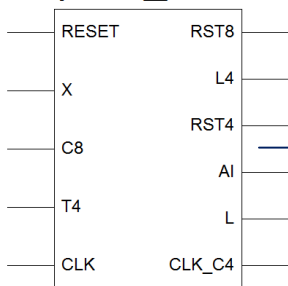


Figura 10 – Esquema da Máquina de Estados

1.1.1 – Máquina de estados através do diagrama de estados

Foi-nos pedido no enunciado que projectássemos a máquina de estados, não apenas com Flip-Flops, como mostra a página anterior, mas também utilizando um recurso do Xilinx, que nos permite desenhar um diagrama de estados, baseado na figura 7. Após criarmos esse diagrama, podemos convertê-lo num objecto (a dita máquina de estados), de forma a poder ligá-la aos restantes componentes do sistema.

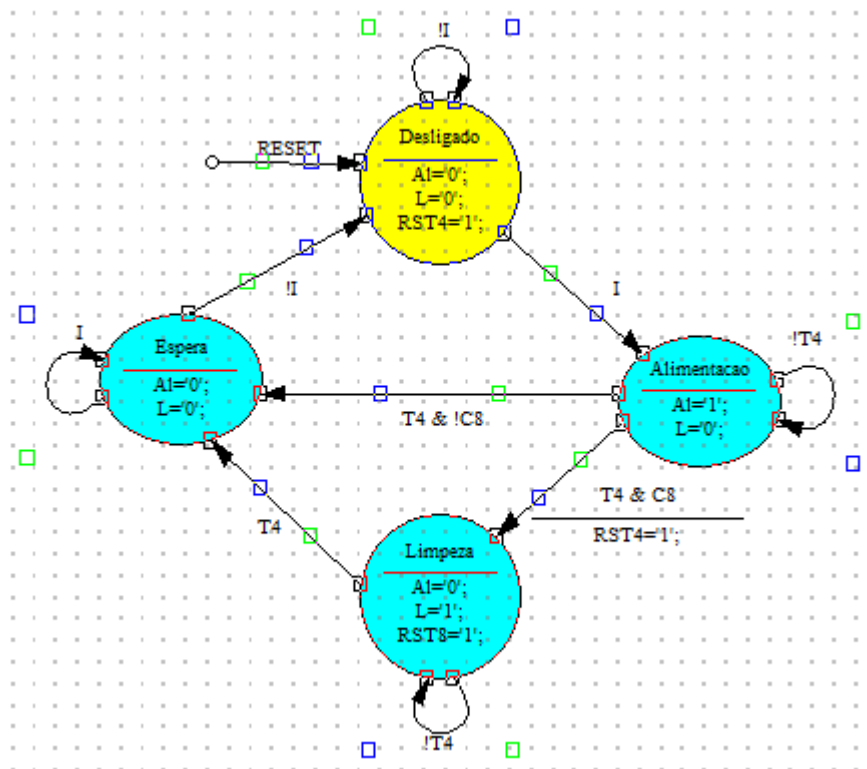


Figura 11 – Diagrama de estados

Após transformarmos o diagrama numa máquina de estados, esta fica semelhante à criada com os Flip-Flops (Figura 10) e o resto do processo – ligações e afins – é idêntico.

DIAGRAMA_ESTADOS

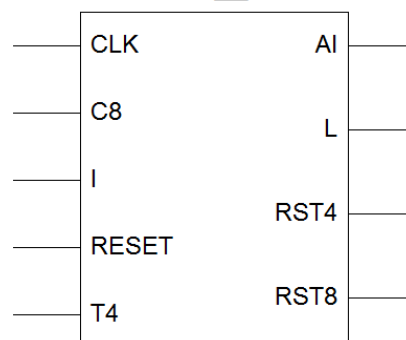


Figura 12 – Máquina de estados criada através do diagrama de estados

1.2 – Sistema completo

O sistema é constituído pela máquina de estados criada anteriormente e por dois contadores de 4 bits.

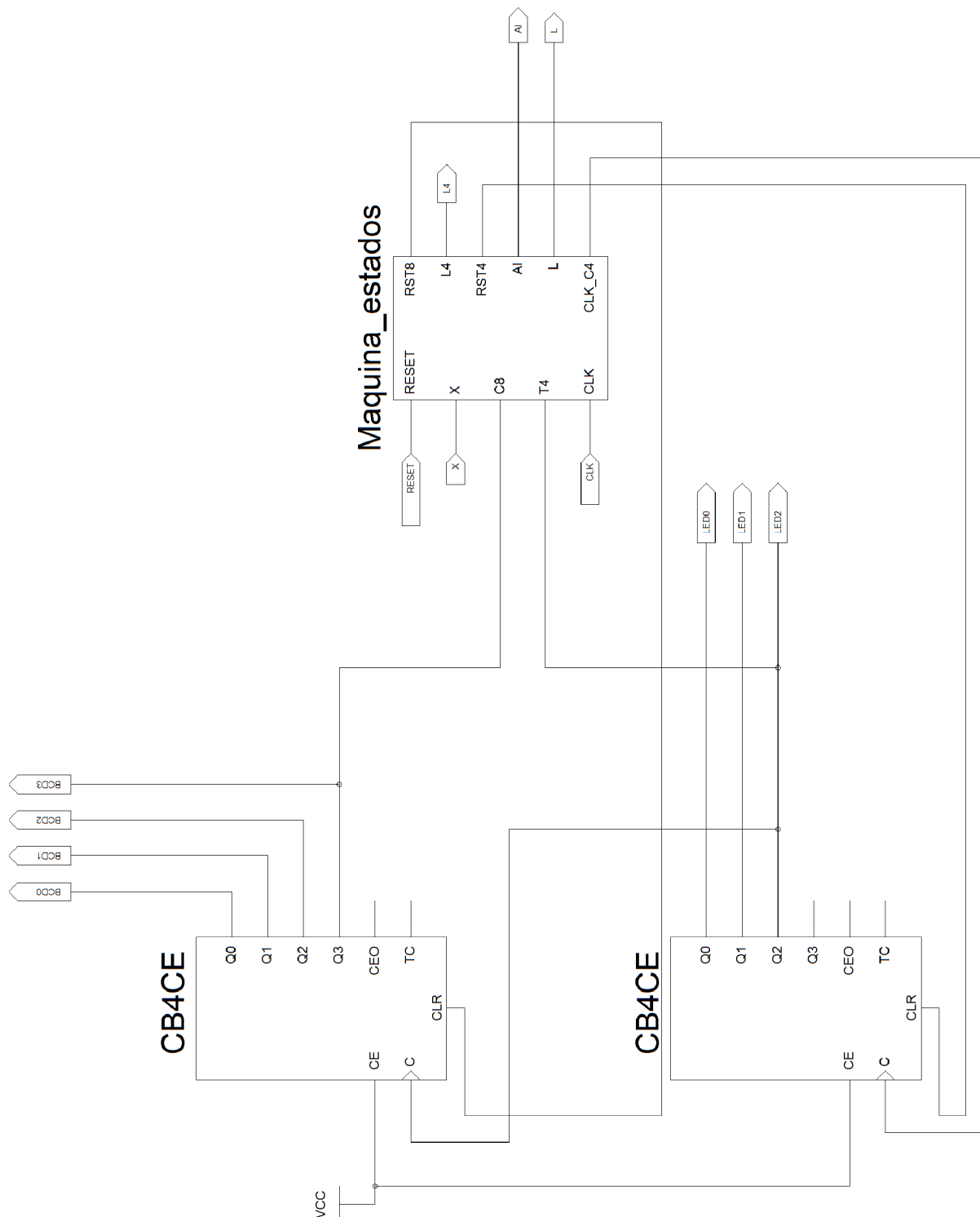


Figura 13 – Esquema do Sistema Completo

1.3 – Sistema final

Após completarmos o sistema, encapsulámo-lo, tal como tínhamos feito com a máquina de estados, e criámos aquilo a que chamamos o sistema final, com todos os inputs e outputs necessários ao seu correcto funcionamento.

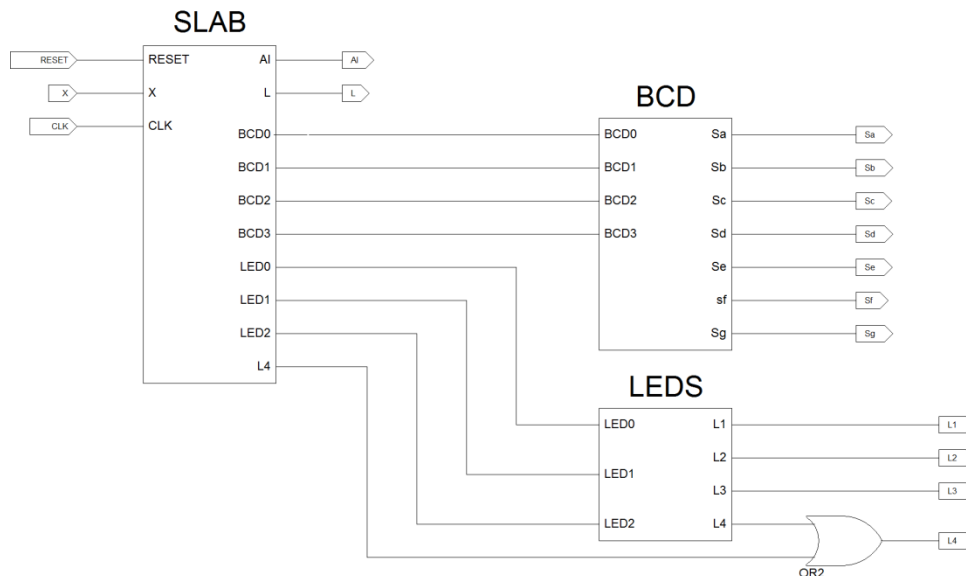


Figura 14 – Esquema do Sistema Final

BCD e LEDS são duas caixas que representam os outputs para o display BCD de sete segmentos, que conta o número de alimentações, e para quatro leds, que mostram o número de ciclos que dura a alimentação ou a limpeza. As funções para o BCD foram criadas através das funções simplificadas de cada segmento do display e as dos leds foram feitas tendo em conta o número do ciclo em que se encontra a alimentação ou a limpeza e as saídas do T4 correspondentes.

	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	1	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	x	x	x	x	x	x	x
10	1	0	1	0	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x

Figura 15 – Tabela de verdade do BCD de sete segmentos

2 – Explicação do Esquema

2.1 – Máquina de estados

Uma máquina de estados permite-nos definir vários estados, que transitam entre si, de acordo com determinadas condições. Como temos quatro estados, foi necessário utilizar dois Flip-Flops JK. A máquina de estados é aquilo a que chamamos o controlador do sistema, e é aqui que a evolução do sistema é controlada, tal como o que acontece aos periféricos e aos outputs. O controlador evolui com base nos inputs do sistema. Todas as funções lógicas presentes na máquina de estados foram obtidas a partir da tabela de verdade.

Apesar de não fazer parte da máquina de estados, foi também definido o CLOCK do T4, que será explicado mais à frente.

Outra hipótese foi criar a máquina de estados através de um diagrama de estados concebido a partir da figura 7. Apesar de ser um método diferente, os resultados são os mesmos, tal como as ligações dos periféricos à máquina.

2.2 – Sistema Completo

Após criarmos e encapsularmos a máquina de estados, chegou a altura de adicionar ao circuito os periféricos do sistema, neste caso os contadores - que vão controlar o ciclo das alimentações e das limpezas, e o número de alimentações da jaula - e ligá-los aos respectivos inputs (no caso das saídas dos contadores) e outputs (no caso dos resets dos contadores) da máquina de estados. É necessário referenciar que o RESET dos contadores foi ligado ao reset dos Flip-Flops, para que se for necessário efectuar um reset ao sistema, ele ser todo iniciado do zero, e não apenas a máquina de estados.

Quanto ao CLOCK dos periféricos, este deve estar desfasado do CLOCK da máquina de estados em um ciclo de relógio, e também entre si. Passamos a explicar: O CLOCK da máquina de estados está ligado directamente a um input, que neste caso vai ser um interruptor, que nos permite controlar o “andamento” do sistema. Como o CLOCK dos periféricos tem de estar desfasado do da máquina de estados, resolvemos criar uma função que activasse o CLOCK do contador até 4 apenas durante a alimentação e limpeza, pois é quando é necessário o contador trabalhar, e que tivesse início um CLOCK a seguir ao início alimentação/limpeza. Para isto, utilizámos uma porta lógica AND que apenas é activada um CLOCK após a alimentação/limpeza ter sido iniciada. O CLOCK do contador até oito (que tem como objectivo contar o número de alimentações) está ligado à saída do contador de 4, que neste caso indica que uma alimentação está concluída.

2.1 – Sistema Final

Tal como fizemos anteriormente, voltámos a encapsular o nosso sistema. Adicionámos duas “caixas” ao sistema. Uma que descodifica as funções para o display BCD de sete segmentos, que vai contar o número de alimentações, outra que determina qual dos quatro LEDS activar, de acordo com o período da alimentação ou da limpeza em que o sistema se encontra. Como inputs do sistema temos o interruptor que activa o mesmo, um botão de RESET e o CLOCK. Como outputs, além do display BCD de sete segmentos e dos LEDS anteriormente referidos, temos também um LED que indica se está a decorrer uma alimentação e outro que indica se a jaula está a ser limpa.

Posto isto, torna-se fácil perceber como funciona todo o sistema:

O Sistema de Alimentação e Limpeza tem início quando o interruptor ‘X’ é activado. Quando isto acontece, procede-se à alimentação do animal que dura quatro ciclos de relógio e é controlada por um contador que conta até quatro. Com o fim da alimentação, existe outro contador, que conta até oito, que é incrementado uma unidade. Se esse contador chegar a oito, dá-se início ao estado de limpeza. Caso contrário o sistema evolui para um estado de espera que tem como objectivo esperar que o interruptor ‘X’ seja desligado. Quando o sistema passa do estado de alimentação para o de limpeza, é feito um RESET ao contador de quatro pois é necessário este estar a zero para uma limpeza correcta, já que esta também dura quatro ciclos de relógio e depende do mesmo contador da alimentação. No estado de limpeza, a jaula é limpa durante quatro ciclos de relógio e é feito ao mesmo tempo um reset do contador de oito, para todo o processo de contagem de alimentações começar novamente. Depois de terminada a limpeza, o sistema avança para o estado de espera.

Ainda antes de o sistema voltar à alimentação existe outro estado de espera, onde é feito um reset ao contador de quatro, para que este seja novamente inicializado a zero, e que só evolui para o estado seguinte (alimentação) quando o interruptor ‘X’ for activado.

Para o caso de ocorrer algum problema com o Sistema de Alimentação e Limpeza existe um botão de RESET geral que reinicializa todo o sistema.

3 – Material Utilizado

- Fios de ligação;
- Fonte de alimentação, LEDS, mostradores BCD e switches;
- CPLD (Complex programmable logic device);
- Computador com o programa Xilinx para a programação do CPLD e respectivo cabo de conexão.

Conclusão

A realização deste trabalho correu bastante bem. Conseguimos obter os resultados pretendidos sem muita dificuldade e tudo funcionou correctamente, quer no simulador do Xilinx, quer no sistema físico. Todo o trabalho foi verificado minuciosamente, pois basta haver apenas uma falha numa ligação para este não funcionar correctamente.

Bibliografia

- Circuitos Digitais e Microprocessadores - Herbert Taub - McGraw-Hil
- Electrónica Digital – L. Cuesta – A. Gill Padilla – F. Remiro