LAB6 - VGA, rato e teclado na DE2

1. VGA

Em geral, um computador é ligado ao monitor através de uma ficha VGA (Video Graphics Adaptor) de 15 pinos. No entanto, só 5 desses pinos contêm sinais activos importantes para este trabalho. A figura seguinte mostra uma ficha VGA e os pinos correspondentes.

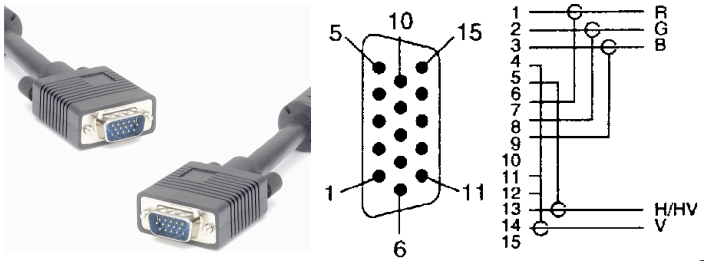

Figura 1 – Conector VGA

Os sinais que nos dizem respeito são o RED, GREEN e BLUE, assim como o sinal de sincronismo horizontal (H_SYNC) e o de sincronismo vertical (V_SYNC). Um monitor "olha" continuamente para esses sinais, a fim de desenhar a imagem que está a ser enviada pelo computador. Iremos de seguida ver como o faz.

No formato VGA standard, uma imagem é constituída por 640 pontos horizontais e por 480 pontos verticais. Um monitor tradicional contém um canhão de electrões que varre constantemente cada linha do ecrã, disparando electrões. O ecrã é revestido por três tipos diferentes de grãos de fósforo, que se iluminam quando atingidos por electrões: um tipo que se ilumina de vermelho; outro tipo que se ilumina de verde; e um terceiro tipo que se ilumina de azul.

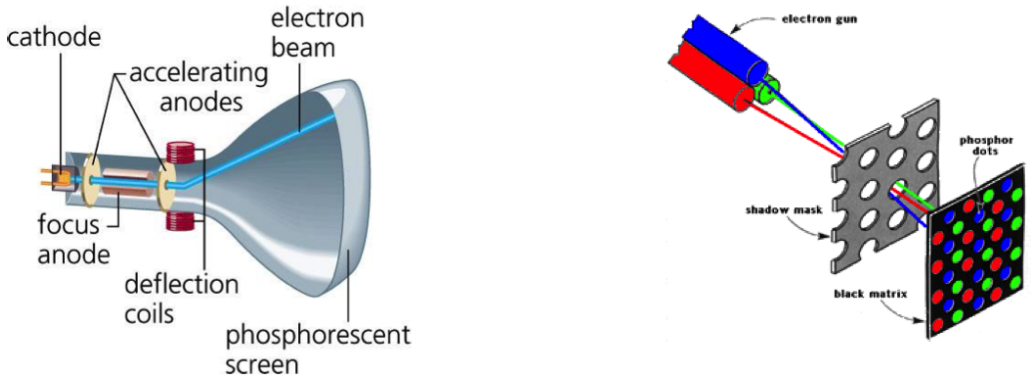

Figura 2 – Funcionamento de um ecrã VGA

Quando o canhão de electrões incide sobre um determinado pixel, emite três feixes de electrões, que irão excitar (ou não) cada um dos três tipos de fósforo presentes nesse pixel. Consoante cada tipo de fósforo é mais ou menos excitado, assim se ilumina mais ou menos. A junção das três cores produz a cor final, que é uma mistura de verde, vermelho e azul. A figura acima ilustra a ideia.

O canhão de electrões incide nos pixeis por ordem, linha a linha, desde o canto superior esquerdo ao canto inferior direito, sendo cada pixel refrescado um certo número de vezes por segundo. No caso da placa da Altera, irá ser produzida uma frequência fixa de refrescamento de 60Hz. A cor de cada pixel é determinada pelos valores presentes nas entradas RED, GREEN e BLUE. No modo de 640x480, com um refrescamento de 60Hz, isso corresponde a aproximadamente 40ns por pixel. Como sabe, placa DE2 Altera possui um gerador de relógio a 50Mhz. Este sinal de relógio é internamente convertido por um dos componentes a usar no trabalho num sinal de 25MHz, já que o período de um sinal de 25Mhz é 40ns (1/25e06).

A forma como o monitor sabe quais as frequências que estão a ser utilizadas é através dos sinais H_SYNC e V_SYNC. Assim, sempre que o computador coloca a linha H_SYNC a 0, isso instrui o monitor a mudar de linha. Sempre que o computador termina um ecrã, coloca o sinal V_SYNC a zero, ordenando o canhão de electrões a regressar ao topo esquerdo do ecrã. O esquema seguinte ilustra o princípio.

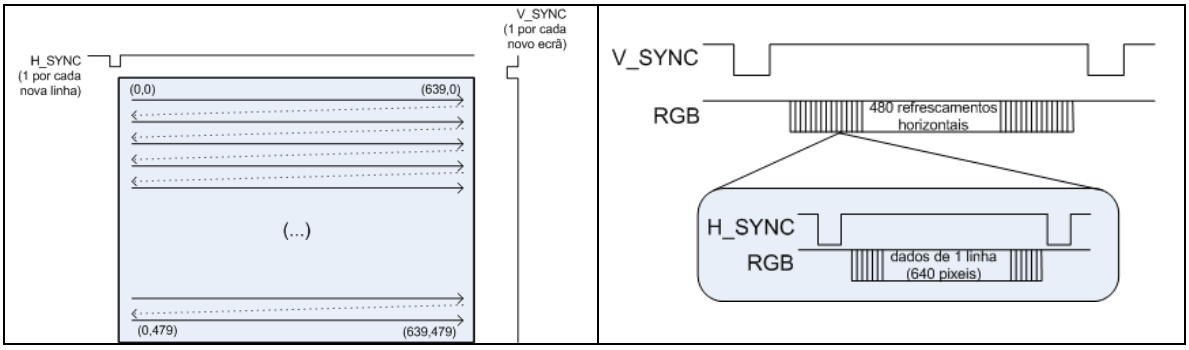

Figura 3 – Sinais de Sincronização Horizontal e Vertical

2. PS/2

Existe uma variedade de dispositivos que comunicam entre si transmitindo dados de forma série. Ou seja, existem dois dispositivos interligados pelo menos através de dois fios, sendo os *bits* enviados um atrás do outro, até que toda a informação tenha sido transmitida.

Alguns exemplos deste tipo de dispositivos são o rato, o teclado e todos os periféricos USB (*Universal Serial Bus*).

Vejamos um exemplo prático. Um teclado PS/2 é ligado a um computador utilizando uma ficha MINI-DIN-6, contendo 6 pinos. Existem dois pinos de alimentação do teclado (GND e VCC), um pino de relógio (KB_CLK) e um pino de dados (KB_DATA). Dois dos pinos do conector não são utilizados (N.C.: *"not connected"*) – ver figura seguinte.

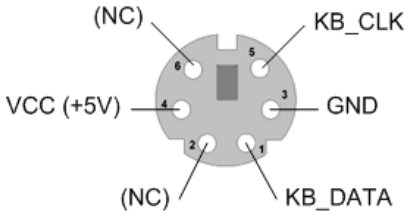

Fig. 1. `Ligações de um teclado PS/2.`

O computador fornece alimentação ao teclado através dos pinos VCC e GND. O teclado controla as linhas KB_CLK (*keyboard clock*) e KB_DATA (*keyboard data*). Sempre que uma tecla é carregada ou largada, o teclado envia para o computador um código correspondente a essa tecla. Para tal, procede da seguinte forma:

- Normalmente, a linha de KB_CLK está a 1. O teclado certifica-se que esta linha está efectivamente a 1, a fim de poder transmitir os dados[1].
- Para transmitir os dados, o teclado começa por colocar a linha KB_CLK a 0 durante cerca de 35μs.
- Após a linha KB_CLK ter ido a 0, o teclado envia 10 *bits* em série através da linha KB_DATA, um por cada vertente activa da linha KB_CLK. O teclado encarrega-se de gerar a onda de relógio em KB_CLK.
- Após os 10 *bits* terem sido transmitidos, a linha de KB_CLK volta a 1, assim como a linha de dados.

O conteúdo dos *bits*, pela ordem que são enviados, é o seguinte:

- ***0*: *"start bit"***, sempre 0, indicando o início da transmissão de dados.
- ***1*: "*bit de dados 0*"**, o primeiro *bit* do código da tecla.
- ***2*: "*bit de dados 1*"**, o segundo *bit* do código da tecla.
- ***3*: "*bit de dados 2*"**, o terceiro *bit* do código da tecla.
- ***4*: "*bit de dados 3*"**, o quarto *bit* do código da tecla.
- ***5*: "*bit de dados 4*"**, o quinto *bit* do código da tecla.
- ***6*: "*bit de dados 5*"**, o sexto *bit* do código da tecla.
- ***7*: "*bit de dados 6*"**, o sétimo *bit* do código da tecla.
- ***8*: "*bit de dados 7*"**, o oitavo *bit* do código da tecla.
- ***9*: *"bit de paridade (ímpar)"***, utilizado para detectar erros. Indica que o número total de *bits* a um (1), **incluindo o *bit* de paridade**, é um número ímpar. Caso não seja, houve um erro de transmissão.
- ***10*: "*stop bit*"**, sempre 1, indica o fim da transmissão de dados.

A figura seguinte ilustra o processo.

1 O computador, **que tem sempre precedência**, também pode enviar comandos para o teclado, nomeadamente para controlar os LEDs do mesmo (e nessa altura, obriga KB_CLK a ser 0). O procedimento a partir daí é análogo, mas em sentido inverso.

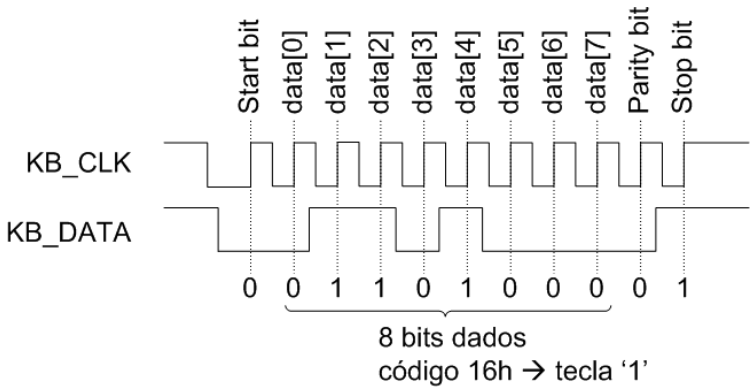

Fig. 2. **`Transmissão de dados correspondente ao pressionar na tecla '1'.`**

Para trabalhar com um teclado ou rato PS/2 os dados recebidos em série podem ser colocados num registo permitindo uma leitura simultânea (paralela) de todos os bits, recorrendo a um shift-register.

3. Módulos da biblioteca UP-Core da Altera

Para trabalhar com algumas das funcionalidades das placas DE2 e UP1/2 da Altera, como VGA, teclado e rato, existe uma biblioteca de módulos UP-Core da Altera. Estes módulos estão disponíveis na página da cadeira no inforestudante **de2.zip**. Os respectivos ficheiros devem ficar nas directorias **C:\altera\quartus\de2**, devendo no projecto indicar a localização da biblioteca para a placa pretendida em **Project->Assign Directories** e **Files->Libraries**.

3.1 CLK_DIV

As placas da Altera geram sinais de relógio internamente, mas têm uma frequência demasiado elevada para muitas aplicações, sendo necessário colocar divisores de frequência para gerar os valores pretendidos. O módulo **`CKL_DIV`** implementa contadores para efectuar essa divisão, disponibilizando um conjunto de sinais de relógio com frequências em potências de 10. Na Fig. 1 é representado o símbolo deste módulo, que tem como pino de entrada um *clock* de 50 MHz (pino PIN_N2 da DE2 [3]).

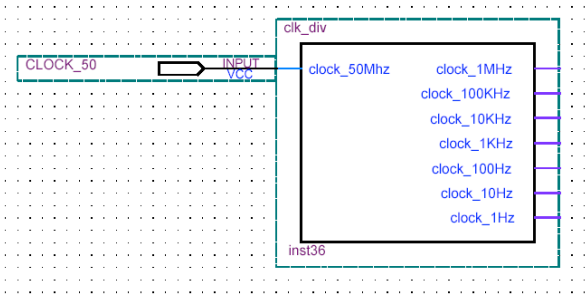

Fig. 3. **`CLK_DIV`: divisor de relógio para DE2.**

3.2 DEC_7SEG

Um outro módulo disponível é o descodificador hexadecimal para 7 segmentos **`DEC_7SEG`** (Fig. 2), que tem uma entrada em BUS para o número binário (**`hex_digit[3..0]`**) e 7 saídas para os segmentos. No Quartus II, as ligações de BUS são feitas por nome e não por ligação gráfica. Na Fig. 2, temos uma entrada **`IN_BYTE[7..0]`** a indicar o valor de dois dígitos enviados para dois visores de 7 segmentos.

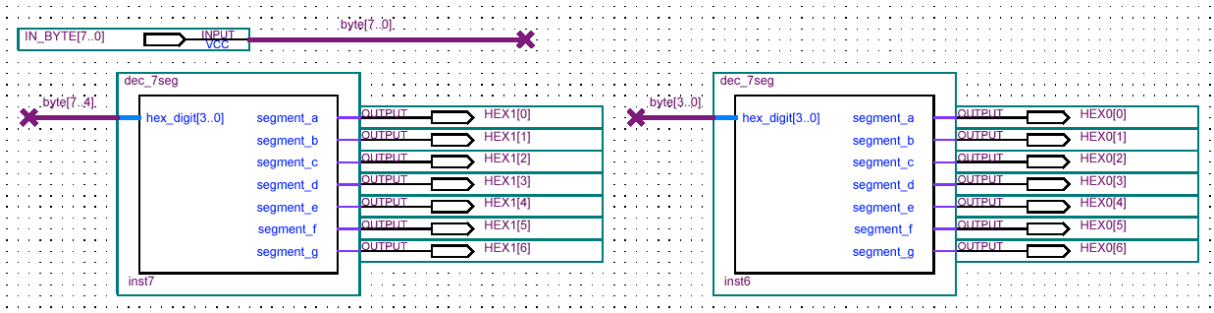

Fig. 4. `DEC_7SEG`: descodificador de hexadecimal para 7 segmentos.

Na página da cadeira na WoC (**`Material de apoio > LABs: dec_7_seg_b`**) está disponível um ficheiro VHDL (**`dec_7seg_b.vhd`**) com a implementação deste descodificador considerando uma entrada em BUS para o número binário (**`hex_digit[3..0]`**) e uma saída em BUS para os 7 segmentos (**`HEX0[6..0]`**).

3.3 VGA_SYNC

Para controlar a saída VGA é necessário enviar sinais analógicos de cada componente de cor RGB (*red, green, blue*), e ainda sinais digitais de sincronismo. Estes sinais correspondem ao varrimento linha a linha feito pelos monitores CRT (*cathode ray tube*) para gerar a imagem. As placas gráficas VGA geram os sinais apropriados a partir de uma memória dedicada que armazena a cor de cada *pixel* da imagem. Na placa UP1/2 não temos a possibilidade de gerar sinais analógicos, mas podemos gerar sinais mais simples, que apenas ligam ou desligam as componentes de cor RGB, ficando limitado a 2^3 cores distintas. No entanto seria necessário muita memória mesmo para um modo VGA simples de 640 por 480, não estando disponível na FPGA da UP1. O módulo VGA_SYNC recorre a um esquema alternativo que não requer uma memória integral da imagem, derivando os sinais de sincronismo a partir do relógio da placa, e disponibilizando sinais digitais que indicam o endereço do *pixel* controlado pelas entradas RGB a cada instante. Colocando lógica combinacional entre as saídas do endereço do pixel e as entradas RGB podemos criar a imagem pretendia. O ecrã é "refrescado" a cerca de 60 Hz, ou seja em cada segundo cada um dos *pixeis* do ecrã é endereçado 60 vezes.

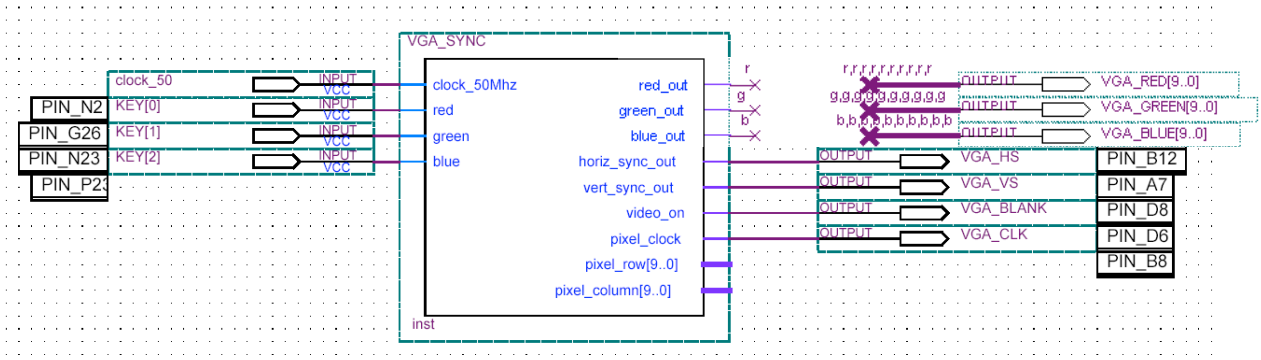

Fig. 5. VGA_Sync: Gerador de sincronismo para VGA.

Exemplo para DE2 com KEY[0..2] a controlar RGB global.

Na figura 3 temos um circuito simples que permite controlar a cor de todo o ecrã com 3 botões que controlam directamente as entradas RGB sem depender do varrimento dos pixeis. A figura 3 mostra a versão do módulo VGA_SYNC para a placa DE2. Ao incluir este módulo deve seguir os nomes indicados na figura para ter os pinos atribuídos automaticamente pelos ficheiros DE2_pin_assignments.cvs . Na figura 4 temos um exemplo de como podemos escrever caracteres no ecrã VGA. A forma de utilizar o componente é a seguinte: o relógio interno da placa liga-se ao CLOCK_50. As saídas RGB e de sincronismo, ligam-se directamente aos portos de saída VGA da placa. Como na DE2 podemos ter valores de RGB com 10bits, optou-se por replicar os bits RGB para ter maior intensidade de cor que pondo apenas, por exemplo, o bit mais significativo.

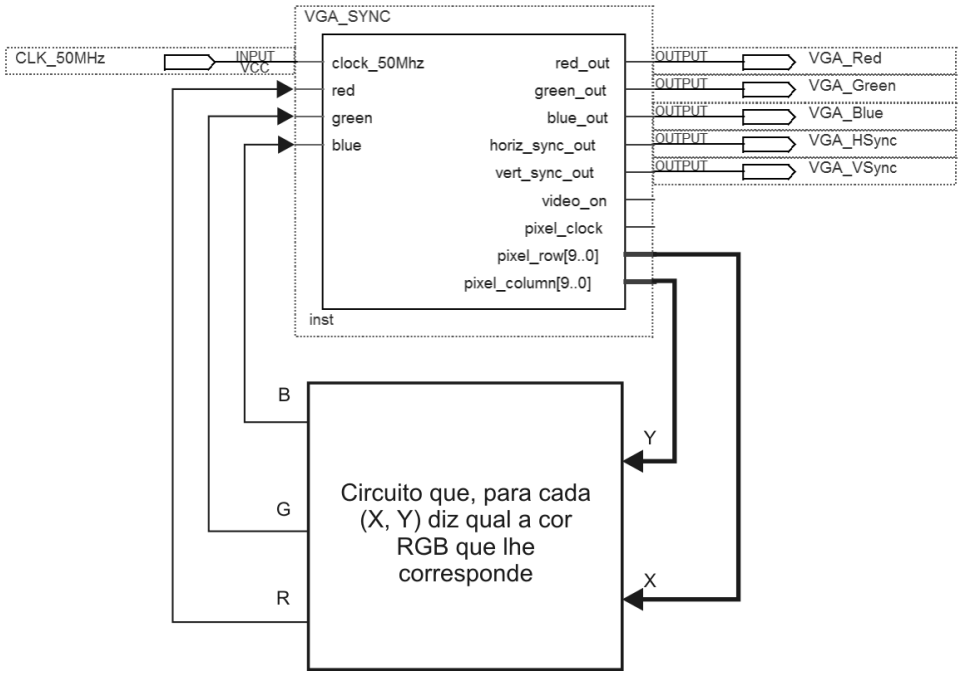

Fig. 6. Utilização do módulo VGA_SYNC.

A partir do momento em que o relógio está activo, este componente coloca em pixel_row e em pixel_column a coordenada (X,Y) do pixel que está a ser desenhado nesse instante. O utilizador

só tem de, para essa coordenada, colocar os valores RGB que deseja para esse pixel nas entradas red, green e blue. O esquema da Fig. 5 ilustra a ideia.

3.4 CHAR_ROM

Para ter caracteres disponíveis para gerar texto no ecrã VGA, temos o módulo CHAR_ROM. Cada caracter é representado por uma matriz de *pixeis* 8x8. A entrada **character_address** indica qual o caracter pretendido (temos 2^6=64 disponíveis, ver fig. 5), e as restantes entradas permitem fazer um varrimento da matriz 8x8 para ter na saída o *pixel* correspondente. A figura 4 exemplifica a utilização combinada dos módulos VGA_SYNC e CHAR_ROM para escrever caracteres no ecrã VGA.

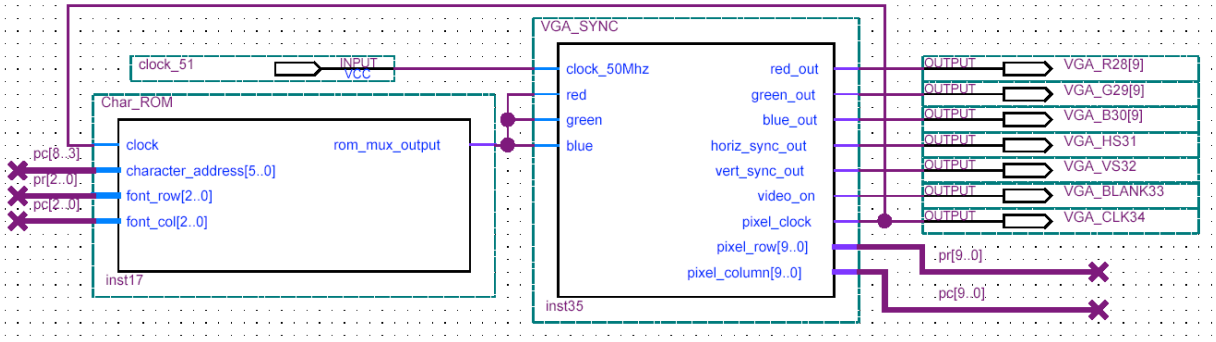

Fig. 7. Utilização combinada dos módulos VGA_SYNC e CHAR_ROM para escrever todos os caracteres disponíveis.

O módulo VGA_SYNC faz um varrimento dos *pixeis* da imagem, e podemos recorrer a lógica combinacional para a partir das saídas de linha e coluna de *pixel* gerar as 3 entradas RGB. Neste caso recorre-se ao módulo CHAR_ROM para indicar os valores. Alterando as linhas de entrada podemos ter caracteres maiores ou menores. Não esquecer que no Quartus II as ligações de BUS são feitas por nome e não por ligação gráfica. A versão CHAR_ROM para UP1/2 não tem a entrada de **clock**.

CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS	CHAR	ADDRESS
@	00	P	20	Space	40	0	60
A	01	Q	21	!	41	1	61
B	02	R	22	"	42	2	62
C	03	S	23	#	43	3	63
D	04	T	24	$	44	4	64
E	05	U	25	%	45	5	65
F	06	V	26	&	46	6	66
G	07	W	27	'	47	7	67
H	10	X	30	(	50	8	70
I	11	Y	31	)	51	9	71
J	12	Z	32	*	52	A	72
K	13	[	33	+	53	B	73
L	14	Dn Arrow	34	,	54	C	74
M	15	]	35	-	55	D	75
N	16	Up Arrow	36	.	56	E	76
O	17	Lft Arrow	37	/	57	F	77

Fig. 8. Tabela de caracteres de CHAR_ROM (endereços em octal).

3.5 KEYBOARD

Para ler os códigos das teclas (*scan codes*) enviados pelo teclado PS/2 temos o módulo KEYBOARD, indicado na figura 6. Tem duas entradas que recebem os dados do teclado PS/2 **keyboard_clk** (*PS2_CLK*) e **keyboard_data** (*PS2_DAT*), e a entrada de relógio. No BUS de saída **scan_code[7..0]** fica o código binário da tecla carregada ou libertada no teclado. A entrada **reset** limpa os registos internos. A saída **scan_ready** vai a 1 quando chega uma novo código do teclado, e só volta a zero quando **read** tiver uma vertente ascendente, podendo assim a lógica adicional detectar a chegada de novas teclas, implementando um *handshake* para ler apenas uma vez a cada novo código.

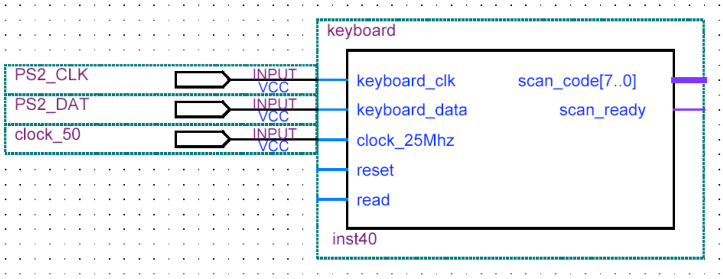

Fig. 9. KEYBOARD: recebe *scan codes* do teclado PS/2.

3.6 MOUSE

Para ler e processar a informação do rato PS/2 temos o módulo MOUSE, indicado na figura 7. Tem duas linhas de dados bidireccionais que ligam ao rato PS/2 **mouse_clk** (*PS2_CLK*) e **mouse_data** (*PS2_DAT*), e a entrada de relógio. Os BUSs de saída **mouse_cursor_row[9..0]** e **mouse_cursor_column[9..0]** indicam a posição do rato no ecrã 640x480, calculada por este módulo a partir os dados incrementais do movimento do rato PS/2. Temos ainda as saídas do estado dos botões. A entrada **reset** limpa os registos internos.

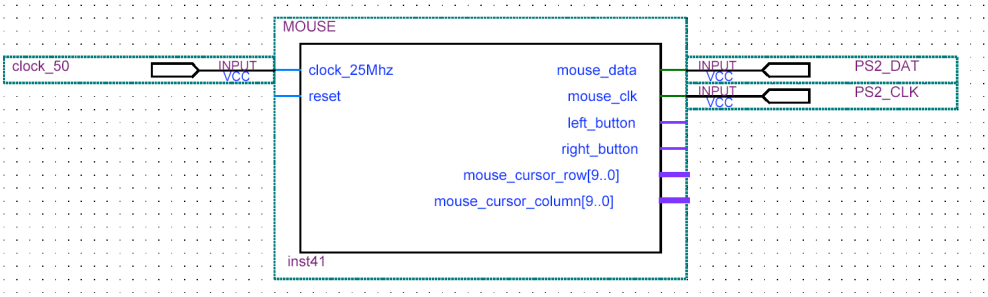

Fig. 10. MOUSE: recebe dados incrementais do movimento do rato PS/2 e do estado dos botões, disponibilizando a posição calculada do cursor e estado dos botões.

4. Exemplo de escrita no ecrã VGA

No trabalho anterior foi dado um exemplo de utilização de VGA e CHAR_ROM, na seguinte figura podemos ver um exemplo mais simples ilustrativo.

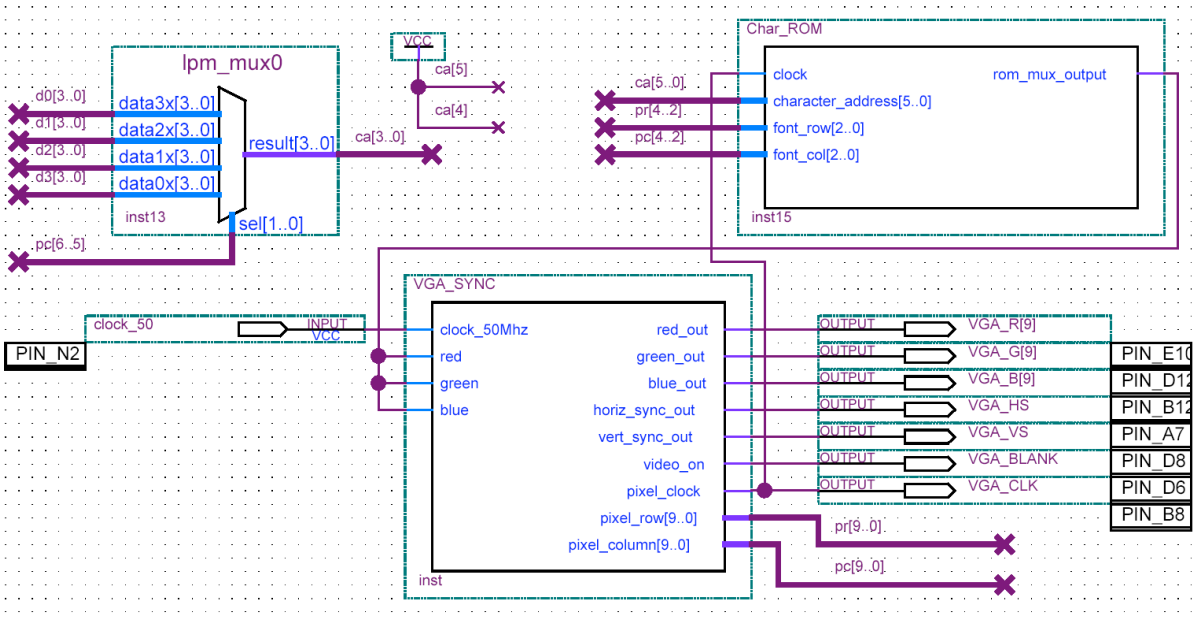

Fig. 11. Utilização combinada dos módulos VGA_SYNC e CHAR_ROM para escrever 4 dígitos no ecrã, recorrendo a um multiplexer.

A figura 4 exemplifica a utilização combinada dos módulos VGA_SYNC e CHAR_ROM para escrever caracteres no ecrã VGA. A figura 15 mostra como podemos controlar uma linha de caracteres para visualizar dígitos. Não esquecer que no Quartus II as ligações de BUS são feitas por nome e não por ligação gráfica. De acordo com a tabela de caracteres de CHAR_ROM, o caracter '0' tem o endereço $60_8=110000_2$, seguido dos restantes algarismos hexadecimais, pelo que dado um número de 4 bits basta juntar mais dois bits a 1 à esquerda para ter o caracter correspondente. Para ter os dígitos pretendidos seguidos temos que os mutiplexar, seleccionando com bits mais significativos das colunas.

O multiplexer pode ser especificado com o **MegaWizard Plug-In Manager**. Ao inserir um componente (symbol) no desenho esquemático, seleccionar **megafunctions/gates/lpm_mux**. Na janela seguinte escolher para gerar VHDL, depois em **Parameter Settings** o número e tamanho das entradas, e **finish** para terminar e colocar o componente no desenho esquemático.

5. Projecto

Neste trabalho tem que implementar as seguintes funcionalidades:

- Gerador de padrão vertical colorido no ecrã VGA
- Gerador de padrão em xadrez colorido no ecrã VGA
- Desenho de uma linha oblíqua, por exemplo x=y, no ecrã VGA
- Escrever no ecrã VGA o seu nome recorrendo à CHAR_ROM
- Indicar nos visores de 7 segmentos a posição XY do rato PS/2
- Indicar nos visores de 7 segmentos a tecla premida no teclado PS/2

Como mini-projecto mais avançado pretende-se que implemente:

- Emulação de um terminal de texto com editor de linha no ecrã VGA
- Desenho livre com o rato no ecrã VGA

6. Agradecimentos

Partes deste enunciado foram adaptadas dos enunciados de Laboratório de Sistemas Digitais do mesmo autor, tendo a apresentação do VGA e PS/2 sido adaptadas de enunciados de Tecnologia dos Computadores do Doutor Paulo Marques.

