UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO – Departamento de Engenharia Elétrica
Sistemas Digitais – ELE 08752 - 2ª Prova Parcial 2022/1- 10/08/2022

Aluno1: _____

Aluno2: _____

1ª QUESTÃO (5,0 pontos): Controlador de vídeo modo texto

Projete um circuito **vga_mult_result** gerador de texto que instancie a ROM de fontes e a utilize para escrever caracteres alfanuméricos no monitor VGA.

O circuito deve receber o número **a** com três dígitos em BCD **a_bcd2 (3 downto 0)**, **a_bcd1 (3 downto 0)** e **a_bcd0 (3 downto 0)** e o número **b** também com três dígitos em BCD **b_bcd2 (3 downto 0)**, **b_bcd1 (3 downto 0)** e **b_bcd0 (3 downto 0)** e deve escrever os valores no monitor VGA.

Além disto, o circuito deve receber o número **p** com cinco dígitos BCD, a saber **p_bcd4 (3 downto 0)**, **p_bcd3 (3 downto 0), p_bcd2 (3 downto 0)**, **p_bcd1 (3 downto 0)** e **p_bcd0 (3 downto 0)**, correspondendo ao produto de **a** e **b**, e escrever seu valor no monitor.

O circuito deve receber como entrada as informações do circuito de sincronismo de vídeo (**pixel_x**, **pixel_y** e **video_on**) e gerar como saída os sinais **text_on** (igual a '1' indica que está na área do caractere na tela) e **text_rgb** com 12 bits para determinar a cor a pintar os caracteres e a tela VGA de acordo com os valores a serem apresentados e a ROM de fontes que deve ser instanciada neste circuito.

Por exemplo, considerando **a_bcd2** = 2 **a_bcd1** = 4, **a_bcd0** = 9, **b_bcd2** = 1, **b_bcd1** = 3, **b_bcd0** = 7, **p_bcd4** = 3, **p_bcd3** = 4, **p_bcd2** = 1, **p_bcd1** = 1 e **p_bcd0** = 3, na tela do monitor VGA deve aparecer:
A = 249 B = 137 P=A * B = 34113 com fontes em maiúscula em tamanho 16x32 (ou seja, escalados por 2)

Crie um circuito **vga_ctrl** controlador de VGA que instancie:
- o circuito gerador de sincronismo **vga_sync** que gera como saída **pixel_x**, **pixel_y, video_on**, **pixel_tick, hsync** e **vsync**.
- o circuito gerador de texto **vga_mult_result** projetado para escrever os números no monitor que gera **text_on** e **text_rgb**.

O circuito **vga_ctrl** deve gerar o sinal **rgb** passando por um registrador para que ele tenha o mesmo atraso gerado para os sinais **hsync** e **vsync** no circuito **vga_sync**. O sinal **rgb** deve ser armazenado em um **rgb_reg** atualizado sincronamente com o clock de 100MHz usando **pixel_tick** como sinal de habilitação. O sinal **pixel_tick** é '1' a cada ciclo do clock de 25MHz que é o tempo de atualização do pixel no monitor no padrão VGA.
Embora os valores de **a**, **b** e **p** possam mudar a qualquer instante, lembre-se de alterar a informação que será apresentada na tela em sincronismo com a varredura usando o sinal **refresh_tick** que deve ser '1' apenas uma vez por varredura fora da área útil (por exemplo quando **pixel_x** é 0 e **pixel_y** é 480).

Use os códigos do vídeo VGA do Capítulo 13 modificados para comportar 12 bits de cores porque os circuitos serão testados na FPGA da placa Nexys A7.

2ª QUESTÃO (5,0 pontos): *PicoBlaze* - Interface de entrada e saída.
Modifique o circuito **pico_btn** do Capítulo 16 de forma a considerar que os valores de **a** e **b** sejam lidos das chaves **sw(11 downto 0)** quando o botão (**btn**(1)) é pressionado. A leitura das chaves se dará em duas etapas onde serão lidos os 8 bits LSB e os 4 bits MSB já que as instruções do *PicoBlaze* são de 8 bits. Os valores lidos serão tratados como 3 dígitos BCD, representando valores entre 000 e 255, ou seja, valores BCD fora dessa faixa devem ser descartados e um erro sinalizado no display de 7 segmentos.
A interface de entrada para leitura de **a** e **b** deve ser a mesma do circuito **pico_btn**, limpando a RAM de dados quando **btn**(0) é pressionado e lendo alternadamente os valores de **a** e **b** das chaves quando o **btn**(1) é pressionado. Se o valor na entrada não for válido (não for BCD ou for maior que 255) o sistema deve enviar uma mensagem de erro para a porta de saída, mantendo o valor atual de **a** (ou **b**), sem alterar **switch_a_b**, considerando que a leitura ainda não foi feita. Para valor válido uma mensagem deve ser enviada para apagar a mensagem de erro. Os botões devem passar por circuito de *debounce* e a saída *tick* usada para ativa o *flip-flop* de *flag* da interface de entrada.
Para realizar a multiplicação utilize a rotina **mult_hard** que faz a multiplicação de maneira combinacional externamente ao Picoblaze.

```
;=====================================================
; routine: mult_hard
;   function: 8-bit unsigned multiplication using
;             external combinational multiplier;
;   input register:
;      s3: multiplicand
;      s4: multiplier
;   output register:
;      s5: upper byte of product
;      s6: lower byte of product
;   temp register:
;=====================================================
mult_hard:
    output s3, mult_src0_port
    output s4, mult_src1_port
    input s5, mult_prod1_port
    input s6, mult_prod0_port
    return
```

A rotina **mult_hard** espera números binários, portanto os valores de **a** e **b** em BCD devem ser convertidos no *PicoBlaze* para binário gerando **a_bin** e **b_bin** antes da multiplicação ser efetuada. O resultado da multiplicação **p_bin** em 16 bits obtido pela rotina **mult_hard** deverá ser convertido para BCD e poderá ter 5 dígitos neste formato (**a** = **b** = 255, **p** = **a*b** = 65025, considerando os maiores valores de **a** e **b**).
Os valores de **a**, **b** e do produto **p** em BCD devem ser enviados para a porta de saída do *PicoBlaze* e armazenados em *buffers* para serem mostrados no monitor VGA usando o circuito da primeira questão. A interface de entrada precisa mudar para disponibilizar os valores de **sw**(11 downto 8) em um endereço diferente de **sw**(7 downto 0) e dos flip-flops de flag dos botões. A interface de saída deve ser alterada para realizar o armazenamento e instanciar o circuito da primeira questão. Além disso, deve também receber o valor de **a** e **b** em binário escrito na saída pela rotina **mult_hard**, realizar a multiplicação de maneira combinacional, gerando os sinais que serão lidos pelo *PicoBlaze*. A rotina **mult_hard** e o circuito de multiplicação combinacional foram usados no Capítulo 16 no projeto **pico_uart**.
A interface de saída deve receber a mensagem de erro, escrevendo no display de 7 segmentos

“Erro” se a saída ‘1’ ou apagando o display se a saída for ‘0’.
A rotina de conversão de binário para BCD deve seguir o algoritmo discutido na seção 6.3.3. O algoritmo deve ser usado de forma invertida para transformar de BCD para binário.

Conversão de binário para BCD: verifica-se se cada um dos dígitos BCD é maior que 4. Em caso afirmativo soma-se 3 ao dígito antes de deslocar à esquerda. Em cada deslocamento um bit do número binário é inserido à direita no número BCD. São necessários 8 passos para converter um número binário de 8 bits para BCD.

Passo	BCD2	BCD1	BCD0	Binário
Início	0000	0000	0000	1011 0100
1	0000	0000	0001	0110 1000
2	0000	0000	0010	1101 0000
3	0000	0000	0101+ 0011 1000	1010 0000
4	0000	0001	0001	0100 0000
5	0000	0010	0010	1000 0000
6	0000	0100	0101+ 0011 1000	0000 0000
7	0000	1001+ 0011 1100	0000	0000 0000
8	0001	1000	0000	0000 0000
	1	8	0	

Conversão de BCD para binário: faz-se o deslocamento para a direita, verifica-se se o valor em BCD é maior que 7 e se for subtrai-se 3. A cada passo um novo bit é deslocado à direita do número binário. São necessários 8 passos para obter de volta o número binário de 8 bits, a partir dos três dígitos BCD.

Passo	BCD2	BCD1	BCD0	Binário
Início	0001	1000	0000	0000 0000
1	0000	1100- 0011 1001	0000	0000 0000
2	0000	0100	1000- 0011 0101	0000 0000
3	0000	0010	0010	1000 0000
4	0000	0001	0001	0100 0000
5	0000	0000	1000- 0011 0101	1010 0000
6	0000	0000	0010	1101 0000
7	0000	0000	0001	0110 1000
8	0000	0000	0000	1011 0100

Obs: o algoritmo de conversão costuma ser chamado Double Dabble.

Bom trabalho! *Profa. Eliete Caldeira*