RELATÓRIO LABORATÓRIO 4: PICOBLAZE

Arthur Faria Campos 16/0024242

Programa de Engenharia Eletrônica
Faculdade Gama – Universidade de Brasília
St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
email: arthur-fc@hotmail.com

Felipe Lima Alcântara 16/0027918

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
email: lipelima0327@gmail.com

RESUMO

O documento apresenta o relatório técnico do quarto experimento, da matéria Prática de Eletrônica Digital 2. Este experimento denominado “PicoBlaze”. Esta atividade teve o objetivo de realizar implementações em VHDL de Sistema em Chip (SoC) utilizando PicoBlaze.

1. INTRODUÇÃO

O PicoBlaze é um microcontrolador de 8 bits, ele é similar a vários microcontroladores, porém ele foi projetado e otimizado para a utilização nas FPGAs da Xilinx.

O PicoBlaze necessita de 2 ciclos de clock para realizar um comando, além disso para a implementação é necessário definir os comandos no assembly do PicoBlaze e utilizar um compilador para transformá-lo em um código VHDL .

No experimento utilizamos o KCPSM6 que é o PicoBlaze otimizado para os modelos Spartan-6, Virtex-6 e 7-Series.

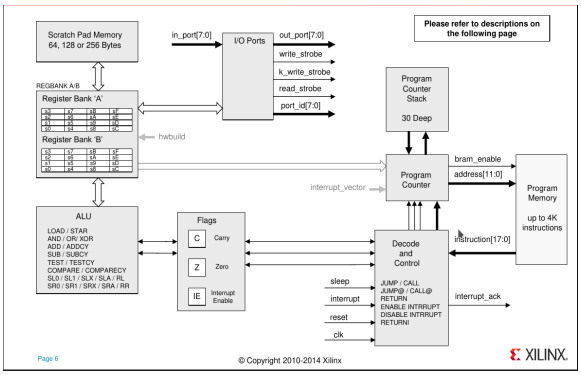

Figura 1: KCPSM6 Architecture

Para inserir KCPSM6 em um projeto, existem apenas dois arquivos que definem dois componentes.

Não surpreendentemente, 'kcpsm6' define o processador atual e suas portas. Isso também tem três valores genéricos, mas foram atribuídos valores padrão que podem ser usados até ter algum motivo para alterá-los.

O segundo componente define a memória que conterá o seu programa uma vez que tenha sido escrito e montado. Ele também possui três valores genéricos que precisam ser configurados adequadamente.

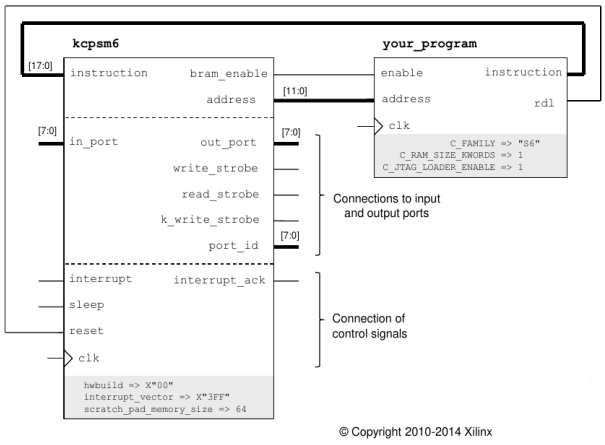

Figura 2: Conexões do KCPSM6

2. EXPERIMENTO

Para o Laboratório 04 foi pedido a realização de um SoC (Sistema em Chip), parar isso eram necessários três microcontroladores PicoBlaze para implementar a arquitetura, mostrada na figura abaixo.

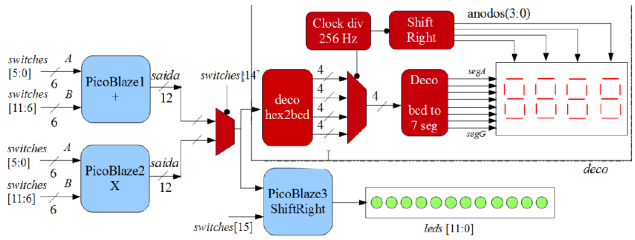

Figura 3: Sistema em Chip

- O primeiro PicoBlaze realiza a soma de duas entradas cada uma de 6 bits mapeadas nos switches da placa.
- O segundo PicoBlaze realiza a multiplicação das mesmas entradas.
- O Switch14 escolhe qual resultado será apresentado nos displays de 7 segmentos.
- O terceiro PicoBlaze realiza o deslocamento à direita do resultado da soma ou multiplicação e apresenta o valor nos leds da placa.

O deslocamento deve ser realizado uma vez quando o valor de Switch15 muda de zero para um.

Além disso foi necessário implementar, um divisor de clock, um multiplexador para alternar o valor de saída do display, um conversor de binário para bcd 7 segmentos. Sendo todos esses componentes previamente feitos e disponibilizados no Moodle, bastando apenas instanciá-los na TopMain do programa.

Deve-se ressaltar também a importância da configuração da ROM no momento de insancia-la no Port Map como mostrado abaixo.

```
246     ROM_B: mult                     --Name to match your PSM file
247     generic map( C_FAMILY => "7S",   --Family 'S6', 'V6' or '7S'
248                  C_RAM_SIZE_KWORDS => 2,      --Program size '1', '2' or '4'
249                  C_JTAG_LOADER_ENABLE => 0)      --Include JTAG Loader when set to '1'
```

Figura 4: Configurações da ROM

2.1. PicoBlaze Soma

O Pico Blaze da Soma é bem simples, utilizamos três PORT ID para sincronizar as entradas com os dados enviados IN PORT e as saídas OUT PORT.

No loop principal recebemos a entrada A e guardamos no registrador s0 depois pegamos a entrada B e guardamos no registrador s3, realizamos a soma pelo ADD e enviamos pela OUT PORT. Como temos apenas 2 entradas de 6 bits a maior soma será de 7 bits não sendo necessário 2 registradores de saída como na Multiplicação.

```
1  ; Este programa le 12 switches e soma  a(5:0) + a(11:6)
2
3  CONSTANT Source_A, 01 ; switchs no portID 01
4  CONSTANT Source_B, 02 ; switchs no portID 02
5  CONSTANT Out_A, 03; switches no portID 03
6
7  soma:    INPUT s0, Source_A
8           LOAD s2,s0
9           INPUT s1, Source_B
10          LOAD s3,s1 ; Copiei sw em
11          ADD s2, s3
12          OUTPUT s2,Out_A
13          JUMP soma
```

Figura 5: Assembly Soma

2.2. PicoBlaze Multiplicação

O Pico Blaze da multiplicaçao já foi mais complexo, utiliza-se de 4 PORT ID, sendo dois registradores de entrada e dois para a saida, alem de um registrador auxiliar.

No loop principal(mult_soft) recebemos a entrada A e guardamos no registrador s3 depois pegamos a entrada B e guardamos no registrador s4.

```
1  CONSTANT    Source_A,    00
2  CONSTANT    Source_B,    01
3  CONSTANT       Out_A,    02
4  CONSTANT       Out_B,    03
5
6  NAMEREG           sA,     i
7
8  mult_soft:          INPUT     s0, Source_A
9                      LOAD      s3, s0
10                     INPUT     s0, Source_B
11                     LOAD      s4, s0
12                     LOAD      s5, 00
13                     LOAD       i, 08
14
15 mult_loop:          SR0       s4
16                     JUMP      NC, shift_prod
17                     ADD       s5, s3
18
19 shift_prod:         SRA       s5
20                     SRA       s6
21                     SUB        i, 01
22                     JUMP      NZ, mult_loop
23
24 Saidas:             OUTPUT       s6, Out_A
25                     OUTPUT       s5, Out_B
26                     JUMP      mult_soft
```

Figura 6: Assembly Multiplicação

Zeramos um dos registradores de saída e adicionamos o valor 8 no auxiliar. Desloca-se o s4 para a direita e verifica-se se tem carry, caso afirmativo salta para o shift_loop e la se deslocam os registradores de saida adicionando o carry e tambem subtraindo 1 do auxiliar e retorna para o mult_loop.

Caso no deslocamento de s4 não houver carry e somado o s3 no s5 e depois deslocado todos.

Quando o auxiliar chegar a zero as saidas são liberadas com o resultado da multiplicação. Sendo o registrador s4 tendo os 4 bits mais significativos.

2.3. PicoBlaze Shift Right

Assembly para o deslocamento para direita (divisão por dois).

```
CONSTANT    Source_A,    00
CONSTANT    Source_B,    01
CONSTANT          Sw,    04
CONSTANT       Out_A,    02
CONSTANT       Out_B,    03

Compara:    INPUT     s0, Source_A
            LOAD      s1, s0
            INPUT     s0, Source_B
            LOAD      s2, s0
            COMPARE   sA, s1
            COMPARECY   sB, s2
            JUMP       Z, igual
            JUMP Atualiza


igual:      INPUT     s0, Sw
            LOAD      s9, s0
            COMPARE   sC, s9
            JUMP       C, Desloca
            JUMP       Z, Compara
            JUMP Carega

Desloca:       LOAD      sC, 01
               SR0       s4
               SRA       s3
            OUTPUT       s3, Out_A
            OUTPUT       s4, Out_B
            JUMP      Compara

Carega:        LOAD      sC, 00
               JUMP   Compara

Atualiza:   INPUT     s0, Source_A
            LOAD      sA, s0
            LOAD      s3, s0
            INPUT     s0, Source_B
            LOAD      sB, s0
            LOAD      s4, s0
            OUTPUT       s3, Out_A
            OUTPUT       s4, Out_B
            JUMP Compara
```

Figura 7: Assembly Shift Right

As instruçao são uma serie de loops que verificam os valores de entrada e verifica se os ultimos valores de entrada eram os mesmos, caso sejam iguais torna-se possível o deslocamento com um pulso no valor de 1 pela terceira entrada. Sendo possível que outro deslocamento ocorra apenas e o switch de deslocamento for para 0 e depois para 1 novamente.

Caso os valores de entrada sejam diferentes do anterior será salvo os novos valores nos registradores de comparação e os deslocamentos serão resetados.

3. RESULTADOS

Antes de ser implementado na FPGA, realizamos simulações por meio de Test Bench no Vivado e a simulação no pBlazeIDE.

3.1. Simulações

A simulação do Exercício busca verificar o funcionamento dos PicoBlazes na soma multiplicaçao de ShiftRight(Divisão). .

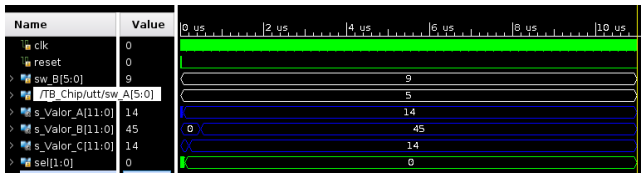

Figura 8: Simulação da soma e multiplicação

Os valores de entrada são 9 e 5 sendo possível verificar o resultado da soma no sinal s_Valor_A, o qual são as saídas do PicoBlaze de soma. No sinal s_Valor_B temos o resultado da multiplicação.

Aqui já podemos perceber que a multiplicação demora muito mais tempo que a soma.

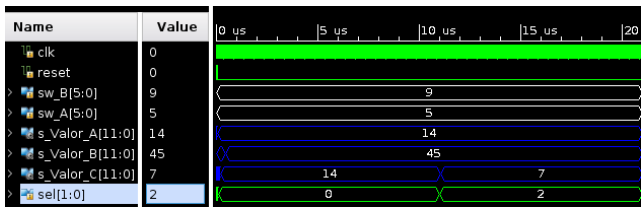

Figura 9: Simulação do Shift Right

Na figura acima vemos a simulação do ShiftRight (s_Valor_C), quando acionamos um pulso para o deslocamento, ocorrendo assim o deslocamento para a direita e dividindo o valor por 2. A açao e quase simultanea com o pulso.

Abaixo temos a simulação do arquivo psm da soma no PblazeIDE. Somando se 7 + 24 = 31.

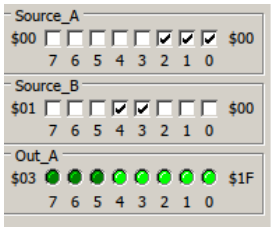

Figura 10: Soma no PblazeIDE

Abaixo temos a simulação do arquivo psm da multiplicação no PblazeIDE. Sendo 51 x 12 = 612, assim percebemos que os bits mais significativos se encontram no Out_B.

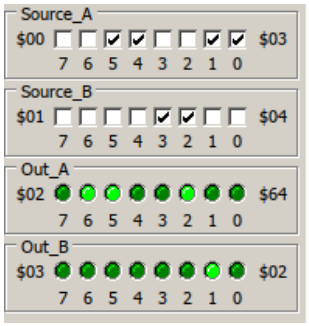

Figura 11: Multiplicação no PblazeIDE

Abaixo temos a simulação do arquivo psm do Shift Right no PblazeIDE. Neste vemos o número normalmente sem ser deslocado.

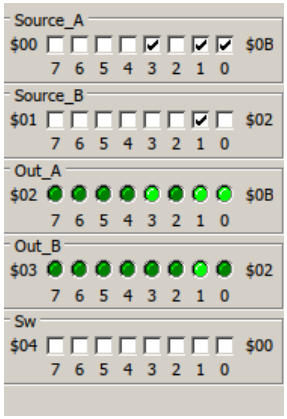

Figura 12: ShiftRight no PblazeIDE (1/2)

Já na figura abaixo vemos o valor 1 no sw e o ShiftRight, sendo a saída Out_B os números mais significativos.

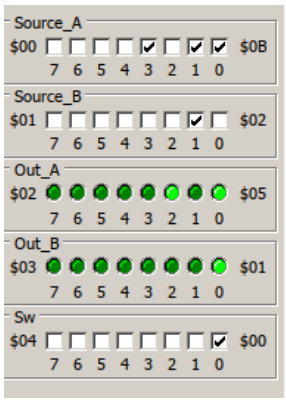

Figura 13: ShiftRight no PblazeIDE (2/2)

Com uma simples análise da Tabela abaixo percebemos que muito pouco da capacidade da FPGA fora utilizada para implementar três PicoBlazers, sendo a maior parte por conecçoes.

Resource	Utilization	Available	Utilization %
LUT	486	20800	2.34
LUTRAM	72	9600	0.75
FF	308	41600	0.74
BRAM	3	50	6.00
IO	40	106	37.74
BUFG	1	32	3.13

Tabela 1: Utilização dos componentes da FPGA

A Figura abaixo mostra a região da placa Basys 3 em que está implementado os PicoBlazers.

Em Amarelo eta a região ocupada pelo PicoBlaze da Soma, em verde-claro pela Multiplicação e em azul o ShiftRight.

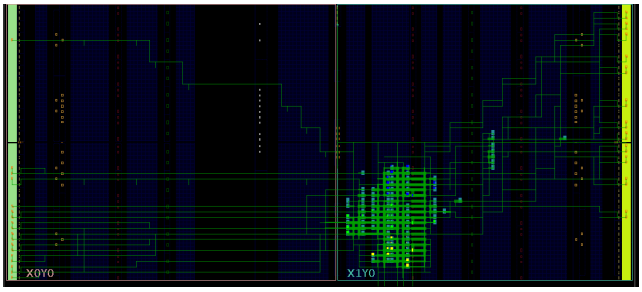

Figura 14: Região da Placa

4. DISCUSSÃO E CONCLUSÕES

Um fato interessante é o fator do Shift Right ter um funcionamento incorreto quando interligado ao clock, um dos motivos seja a questão do tempo de realização de comandos do PicoBlaze. O funcionamento correto desta função só sendo efetivo quando realizado sem o clock. O Shift por esse motivo, acaba sendo das funções implementadas a mais rápida.

Fazendo uma comparação com a implementação de uma ULA e a implementação do PicoBlaze, o PicoBlaze utiliza muito mais componentes da FPGA que a ULA, além de utilizar mais energia da placa:

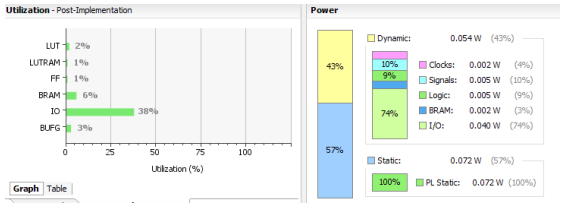

Figura 15: Uso de energia do PicoBlaze

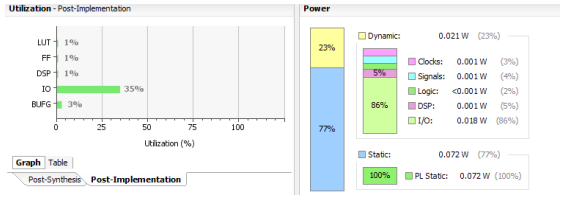

Figura 16: Uso de energia da Ula

Por isso que é muito necessário saber muito bem como será utilizado e o que você quer com sua implementação.

Assim, a conclusão a ser feita é que, a menos que o dispositivo que contenha KCPSM6 seja operado em um ambiente significativamente mais hostil, então o potencial para o KCPSM6 falhar durante a operação é quase insignificante quando comparado com quase tudo ao seu redor.

Na realidade, a confiabilidade operacional do sub-sistema KCPSM6 quase certamente depende mais da qualidade do design de hardware e do código PSM usados com ele.

Por isso, é muito melhor investir tempo e esforço garantindo que o seu código fundamental seja correto do que saber como melhorar o fundamental.

5. REFERÊNCIAS

[1] Chu, Pong P., FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version, Wiley, 2011. [EBRARY]

[2] Villanova, G. (2016). Uma arquitetura PWM em VHDL. [online] Embarcados. Disponivel em: https://www.embarcados.com.br/uma-arquitetura-pwm-em-vhdl/ [Acessado 7 Sep. 2017].

[3] Wakerly, John F., Digital Design: Principles and Practices, 4th ed., Prentice Hall, 2005.

[4] D'Amore, R., VHDL: Descrição e Síntese de Circuitos Digitais, 2 a . edição, LTC, 2012