Relatório PCS3225 - Aula 02

Grupo 06

Integrantes:

Marcelo Takayama Russo - 13680164

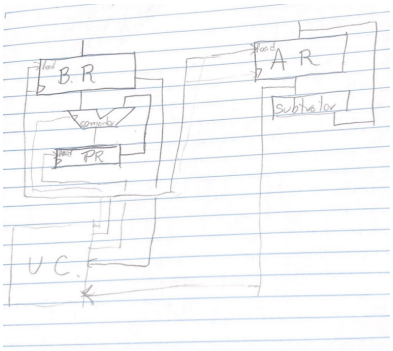
Thainara de Assis Goulart - 13874413

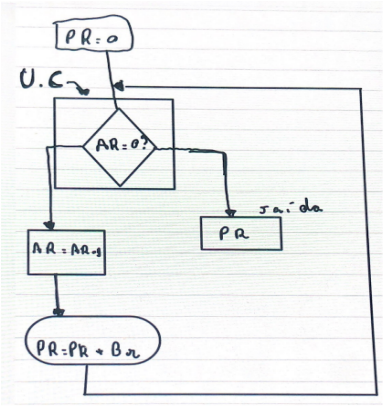
Victor Pedreira Santos de Pepe - 13679565

Análise e implementação da entrega do Grupo 21

obs: O grupo sorteado para nós foi o 11, mas eles não possuíam entrega. Dessa forma, foi pedido para que o nosso grupo fizesse a implementação do grupo 21.

- **Análise do Circuito e da ASM**





No datapath elaborado, os registradores BR e o AR recebem, respectivamente, as entradas do sistema, sendo elas o multiplicando e o multiplicador. Nesse circuito, o resultado final será armazenado no registrador PR, o qual é inicializado como 0.

As operações aritméticas ocorrem da seguinte forma: o valor armazenado em AR passa por um subtrator e seu valor é reduzido em 1 a cada

ciclo. Assim, o valor de BR é acrescido ao valor de PR ‘AR’ vezes. Importante ressaltar que esse multiplicador não utiliza nenhum shifter, mas apenas somas sucessivas. Enquanto o valor de AR for diferente de 0, a soma continua ocorrendo. As entradas do somador são o valor de BR e de PR, e a saída é o próprio PR. Nesse sistema, são necessários 3 registradores: 1 para o PR, 1 para o AR, e outro para o BR.

Vale lembrar que, nesse circuito, não havia nenhuma informação sobre quantos bits entravam e saíam dos registradores. Pelo enunciado do exercício, adotamos que ambas as entradas eram de 8 bits e a saída, de 16. Outro problema foi que a ASM não possuía todos os nomes dos estados, o que dificultou a implementação.

- ## Implementação do Circuito

Para testar se o circuito criado funciona da maneira esperada, criamos uma testbench com alguns exemplos para testarmos o desempenho do circuito.

Foi difícil entender 100% a lógica usada pelo grupo 21, por conta da falta de informações e de nomes no desenho. Isso tornou a implementação mais difícil e confusa, o que nos obrigou a usar a intuição para desenvolver o circuito do projeto.

No desenho da ASM, está dizendo para quando AR = 0, o estado muda para saída e o resultado fica armazenado no PR. Quando implementamos isso, o resultado sempre saía 1x maior do que o esperado. Por exemplo, 7 x 5 resultava em 40. Para resolver isso, colocamos para o estado mudar quando AR = 1, corrigindo o resultado.

Apesar dessas confusões, o projeto do grupo 21 obteve êxito em desenvolver o projeto do Multiplicador, visto que - baseando-se no circuito deles - implementamos um circuito que exerce a função de multiplicação, a partir de somas repetidas. Segue imagem da testbench:

```
Multiplicador8bits uut
    (
    .load_AR(lAR_tb),.load_BR(lBR_tb),.load_PR(lPR_tb),
    .mult1(M1_tb),.mult2(M2_tb),
    .CLK(CLK_tb),
    .RESET(RESET_tb),
    .result(RES_tb)
);


initial begin

    $monitor("M1 = %d || M2 = %d || RES = %d",M1_tb,M2_tb,RES_tb);
    CLK_tb = 1;
    lAR_tb = 1;
    lBR_tb = 1;
    lPR_tb = 1;
    RESET_tb = 1;

    #10
    RESET_tb = 0;

    #10
    M1_tb = 8'd7;
    M2_tb = 8'd5;



end
always #10 CLK_tb = ~CLK_tb;
endmodule
```

Nesse caso, iremos testar a multiplicação de 7 x 5, que deve resultar 35.

Resultado do Teste:

```
# //
# Loading work.Multiplicador8bits_tb(fast)
# M1 =   x || M2 =   x || RES =     x
# M1 =   7 || M2 =   5 || RES =     0
# M1 =   7 || M2 =   5 || RES =     5
# M1 =   7 || M2 =   5 || RES =    10
# M1 =   7 || M2 =   5 || RES =    15
# M1 =   7 || M2 =   5 || RES =    20
# M1 =   7 || M2 =   5 || RES =    25
# M1 =   7 || M2 =   5 || RES =    30
# M1 =   7 || M2 =   5 || RES =    35
# M1 =   7 || M2 =   5 || RES =     0
# End time: 18:32:26 on Mar 23,2023, Elapsed time: 0:00:39
# Errors: 0, Warnings: 0
```

- ## Conclusão e nota

Como visto na testbench, a implementação do grupo 21 funcionou. No entanto, houve muita falta de informação e muita desorganização no trabalho. Isso atrapalhou a implementação do circuito. Por conta dessa desorganização e dessa falta de informação, vamos dar 7/10 para o grupo 21, pois, apesar de todos os problemas, o circuito funciona.