Universidade Federal de São Carlos

Departamento de Computação

Disciplina:

Laboratório de Arquitetura e Organização de Computadores 1

Relatório 5

Unidade Lógica Aritmética (ULA)

Daniel Chaves Macedo RA: 280844

Pedro Gabriel Artiga RA: 351180

Felipe Fiori Campos Martins RA: 316660

São Carlos, 2011.

1. Introdução

A Unidade Lógica Aritmética (ULA) é responsável por realizar as operações matemáticas e de lógica binária em um processador moderno. O resto do circuito presente em um processador é geralmente utilizado para armazenamento de resultados, como por exemplo os registradores, ou para a decodificação das instruções recebidas.

2. Implementação

A ULA implementada para este relatório foi uma ULA que realiza operações em números de 64 bits. Os valores de entrada nesta ULa serão chamados de A e B. As operações que está ULA realiza no primeiro momento são:

- Soma
- Subtração
- “E” lógico
- “Ou” lógico
- Incremento
- Decremento
- Propagação do valor A ou do valor B

```
module ULA(A,B,opt,clock,Y,out);
    input [64:1] A;
    input [64:1] B;
    input [5:1] opt;
    input clock;

    output [64:1] out;
    reg [64:1] out;
    output [64:1] Y;
    reg [64:1] Y;


    always @ (posedge clock)
    begin
        case(opt[5:3])
            0: out = A;
            1: out = A+B;
            2: out = A-B;
            3: out = A&B;
            4: out = A|B;
            5: out = A+1;
            6: out = A-1;
            7: out = B;
        endcase
        case(opt[2:1])
            0: Y = out;
            1: Y = out<<1;
            2: Y = out>>1;
            3: Y = 0;
        endcase
    end
endmodule
```

Figura 1. Código da implementação de uma ULA de 64 bits.

A partir do resultado da operação escolhida acima a ULA ainda pode realizar uma das seguintes operações, alem de propagar o resultado sem qualquer outra alteração:

- Deslocar o resultado um bit a direita

- Deslocar o resultado um bit a esquerda
- Não propagar o resultado, colocando zero na saída

Na Figura 1 é possível observar o código escrito em Verilog que gera o diagrama de circuito equivalente a ULA descrita acima. Como o circuito resultante tem mais de trezentas células lógicas seria impraticável colocar o resultado da compilação neste documento.

3. Análise de Tempo

Utilizando a ferramenta de análise de tempo já incluída na ferramenta Altera Quartus II obtive os seguintes resultados:

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	15.601 ns	B[8]	Y[62]~reg0	--	clock	0
2	Worst-case tco	N/A	None	9.836 ns	out[55]~reg0	out[55]	clock	--	0
3	Worst-case th	N/A	None	-0.165 ns	A[28]	out[28]~reg0	--	clock	0
4	Total number of failed paths								0

Figura 2. Análise de tempo gerada para o circuito otimizado em área.

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	14.608 ns	opt[3]	Y[56]~reg0	--	clock	0
2	Worst-case tco	N/A	None	9.849 ns	out[14]~reg0	out[14]	clock	--	0
3	Worst-case th	N/A	None	-0.464 ns	A[50]	out[50]~reg0	--	clock	0
4	Total number of failed paths								0

Figura 3. Análise de tempo gerada para o circuito otimizado em tempo.

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	16.058 ns	opt[5]	Y[59]~reg0	--	clock	0
2	Worst-case tco	N/A	None	9.764 ns	out[29]~reg0	out[29]	clock	--	0
3	Worst-case th	N/A	None	-0.164 ns	A[6]	out[6]~reg0	--	clock	0
4	Total number of failed paths								0

Figura 4. Análise de tempo gerada para o circuito balanceado.

O valor de Tsu (clock setup time) representa o tempo necessário para que uma informação chegue a um registrador antes que o clock indique que o registrador deve armazenar aquela informação. O Tco (clock to output delay) é o atraso entre um registrador e um pino de saída. E finalmente o Th (clock hold time) é o tempo que uma informação deve ficar estável na entrada de um registrador após o clock ser recebido por aquele registrador. As Figuras 2, 3 e 4 apresentam os piores casos em cada uma das situações de geração de circuito possível no Altera Quartus II.

A ocupação dos blocos lógicos na FPGA para cada modo de otimização é a seguinte:

- Otimização por área: 384 blocos

- Otimização por velocidade: 455 blocos
- Otimização balanceada:: 322 blocos

É interessante notar que a otimização balanceada obtêm resultados melhores em quantidade de blocos consumidos que a otimização por área, pelo menos em um projeto pequeno como este.

4. Conclusão

Através deste experimento é possível notar como as linguagens de descrição de hardware auxiliam na criação de componentes complexos, como por exemplo a ULA apresentada.
A ferramenta de análise de tempo é essencial para se determinar a frequência na qual o projeto funcionará.