Laboratório 2

Matheus Olivera de Almeida Marques da Cruz* Pedro Henrique dos Santos † Nicole de Oliveira Sena ‡
João Pedro Carvalho de Oliveira Rodrigues §

Universidade de Brasília, 4 de novembro de 2023

RESUMO

Nesse laboratório introduzimos os conhecimentos de Linguagem de Descrição de Hardware Verilog, com FPGA DE1-SoC e o software QUARTUS Prime da Intel, que desempenham um papel fundamental no desenvolvimento de sistemas digitais avançados. Vamos analisar, sintetizar e caracterizar sistemas digitais usando HDL (Hardware Description Language), explorar e compreender a estrutura e o funcionamento da Unidade Lógica Aritmética (ULA) e da Floating Point Unit (FPULA), componentes essenciais em processadores e sistemas embarcados.

Palavras-chave: Verilog · FPGA DE1-SoC · QUARTUS Prime · HDL · ULA · FPULA ·

1 UNIDADE LÓGICO ARITMÉTICA DE INTEIROS

1.1 ULA DE INTEIROS

1. **OPAND**:
   - Função: Realiza a operação de E lógico bit a bit entre iA e iB.
   - Código da Operação: 00000
2. **OPOR**:
   - Função: Realiza a operação de OU lógico bit a bit entre iA e iB.
   - Código da Operação: 00001
3. **OPXOR**:
   - Função: Realiza a operação de OU exclusivo (XOR) bit a bit entre iA e iB.
   - Código da Operação: 00010
4. **OPADD**:
   - Função: Realiza a adição entre iA e iB.
   - Código da Operação: 00011
5. **OPSUB**:
   - Função: Realiza a subtração entre iA e iB.
   - Código da Operação: 00100
6. **OPSLT**:
   - Função: Compara iA e iB e define oResult como 1 se iA for menor que iB, caso contrário, 0.
   - Código da Operação: 00101
7. **OPSLTU**:
   - Função: Compara iA e iB como valores não assinados.
   - Código da Operação: 00110
8. **OPSLL**:
   - Função: Realiza uma operação de deslocamento à esquerda em iA pelo número de posições especificado em iB[4:0].
   - Código da Operação: 00111
9. **OPSRL**:
   - Função: Realiza uma operação de deslocamento lógico à direita em iA pelo número de posições especificado em iB[4:0].
   - Código da Operação: 01000
10. **OPSRA**:
    - Função: Realiza uma operação de deslocamento aritmético à direita em iA pelo número de posições especificado em iB[4:0].
    - Código da Operação: 01001
11. **OPLUI**:
    - Função: Define oResult com o valor de iB.
    - Código da Operação: 01010
12. **OPMUL**:
    - Função: Calcula a multiplicação de iA por iB e armazena os 32 bits inferiores do resultado em oResult.
    - oResult recebe mul[31:0], onde mul é o resultado da multiplicação.
    - Código da Operação: 01011
13. **OPMULH**:
    - Calcula a multiplicação de iA por iB e armazena os 32 bits superiores do resultado em oResult. Result recebe mul[63:32], onde mul é o resultado da multiplicação.
    - Código da Operação: 01100
14. **OPMULHU**:
    - Função: Calcula a multiplicação sem sinal de iA por iB e armazena os 32 bits superiores do resultado em oResult.

*211055343@aluno.unb.br
†200026127@aluno.unb.br
‡190114860@aluno.unb.br
§221017032@aluno.unb.br

- oResult recebe mulu[63:32], onde mulu é o resultado da multiplicação sem sinal.
- Código da Operação: 01101

15. **OPMULHSU**:
    - Função: Calcula a multiplicação com sinal de iA por iB e armazena os 32 bits superiores do resultado em oResult.
    - oResult recebe mulu[63:32], onde mulu é o resultado da multiplicação sem sinal.
    - Código da Operação: 01110
16. **OPDIV**:
    - Função: Realiza a divisão de iA por iB e armazena o resultado em oResult.
    - oResult recebe o resultado da divisão.
    - Código da Operação: 01111
17. **OPDIVU**:
    - Função: Realiza a divisão sem sinal de iA por iB e armazena o resultado em oResult.
    - oResult recebe o resultado da divisão sem sinal.
    - Código da Operação: 10000
18. **OPREM**:
    - Função: Calcula o resto da divisão de iA por iB e armazena o resultado em oResult.
    - Código da Operação: 10001
19. **OPREMU**:
    - Função: Calcula o resto da divisão sem sinal de iA por iB e armazena o resultado em oResult.
    - Código da Operação: 10010
20. **OPNULL**:
    - Função: Define oResult como ZERO (presumivelmente um valor nulo ou de inicialização).
    - Código da Operação: 11111
21. **default**:
    - Função: Define oResult como ZERO em caso de valor de iControl não reconhecido.

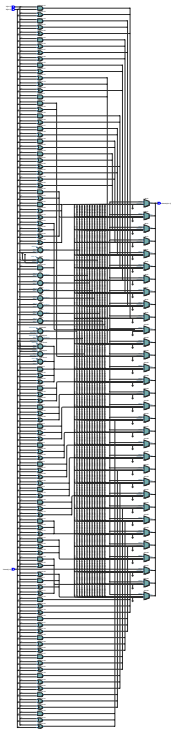
1.2 Arquivo de Forma de Onda

Para testar o comportamento da ULA com diferentes entradas de dados vamos executar a simulação em forma de onda. A ideia é testar tanto os comportamentos mais básicos quanto os mais problemáticos, como overflow e divisão por zero. Os resultados podem ser encontrados nas figuras 2 à 4.

Observa-se nos resultados que a ULA se comporta como o esperado para as mais diversas situações. Por exemplo: nas operações com "0000000F"e "00000002", obtemos pela divisão o número "7", e como resto "1"; nas operações com "FFFFFFFF"e "00000000", na divisão, obtemos "1"como resultado, pois não há divisão por zero; nas operações com "FFFFFFFF"e "FFFFFFFF", na adição ocorre overflow, resultando em "FFFFFFFE", na multiplicação o resultado seria "FFFF FFFE 0000 0001"e a ULA retorna "00000001".

Com esses resultados confirmamos a capacidade da ULA de lidar com os diferentes casos de borda.

Figura 1: *Circuito Sintetizado da ULA de Inteiros*



1.3 Requisitos Físicos

Os requisitos físicos foram medidos com o Time Analyser e o relatório que o Quartus gera. Fizemos as medições para a ULA completa e para cada uma das funções separadamente. O resultado se encontra na tabela 1.

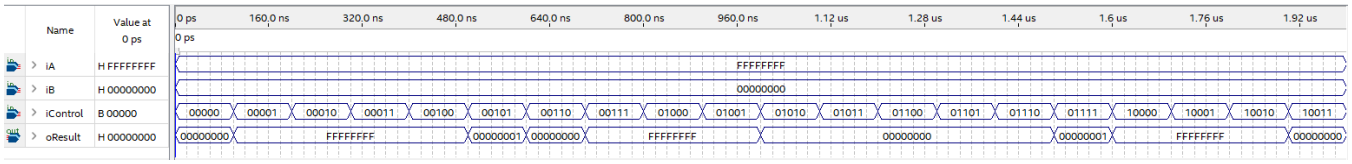


Figura 2: *Operações com "FFFFFFFF"e "00000000"*

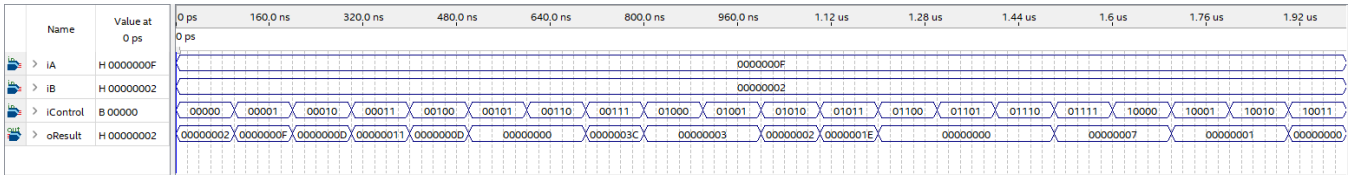


Figura 3: *Operações com "0000000F"e "00000002"*

OPERAÇÃO	TPD	ALMS
OPAND	2.421	65
OPOR	2.421	65
OPXOR	2.421	65
OPADD	3.127	65
OPSUB	3.282	65
OPSLT	4.490	79
OPSLTU	4.490	79
OPSLL	4.150	120
OPSRL	4.412	121
OPSRA	4.639	118
OPLUI	0.516	49
OPMUL	7.688	56
OPMULH	8.156	72
OPMULHU	8.684	72
OPMULHSU	10.803	88
OPDIV	94.892	704
OPDIVU	83.135	633
OPREM	90.438	731
OPREMU	88.790	652
OPNULL	NULL	49
ULA TOTAL	106.096	3029

Tabela 1: *Requisitos físicos da ULA completa e das suas funções individuais.*

1.4 ANÁLISE DE IMPACTOS

A tabela 1 mostra que as funções com maiores circuitos no tamanho da ULA são as operações de divisão (OPDIV, OPDIVU, OPREM e OPREMU).

As operações de divisão são complexas em termos de hardware. Elas envolvem a execução de uma série de suboperações, incluindo divisões sucessivas e seleção dos resultados apropriados. Essas operações requerem circuitos mais elaborados para realizar a divisão e o cálculo do resto.

A compilação da ULA completa utiliza "3029"ALM's, mas quando retiramos as operações de divisão número de ALM's necessários vão para "467", uma redução para "15,42%"dos ALM's totais. Esse resultado evidencia mais ainda o impacto das operações de divisão na ULA. Com isso sabemos que os principais pontos de otimização são as operações de divisão. O caminho de maior atraso é o "OPDIV", com tpd de "94,892".

1.5 SINTETIZAÇÃO NA DE1-SOC

O vídeo dos experimentos com a DE1-SoC podem ser encontrados no seguinte link. Foi feita a verificação de que a ULA funciona de acordo com os testes que fizemos, apesar de não ser possível reproduzir algo como um overflow, pois no DE1-SoC temos disponível 9 bits para utilizar como entrada.

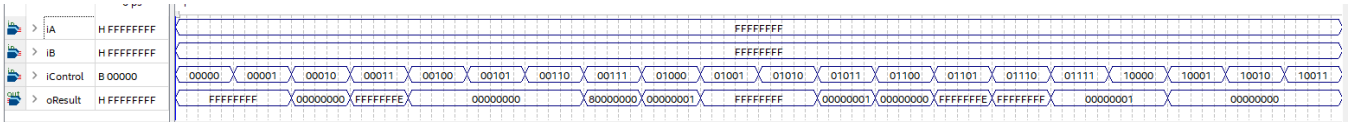


Figura 4: *Operações com "FFFFFFFF"e "FFFFFFFF"*