Relatório 4: Projeto e Síntese de uma ULA para o MIPS

Vitor Praxedes Calegari
(22200379)
INE
UFSC
Florianópolis, Brasil

Rita Louro Barbosa
(22203157)
INE
UFSC
Florianópolis, Brasil

Antônio Escobar Torres
(22204119)
INE
UFSC
Florianópolis, Brasil

Gustavo Ribeiro Bôdi
(23100474)
INE
UFSC
Florianópolis, Brasil

Bianca Mazzuco Verzola
(22202621)
INE
UFSC
Florianópolis, Brasil

Pedro Henrique Taglialenha
(22203674)
INE
UFSC
Florianópolis, Brasil

***Resumo*—Este trabalho se propôs a fazer uma Unidade Lógica e Aritmética - ULA de um processador MIPS monociclo. Para isso, foram projetados os blocos: ULA, Ctrl-ULA(controle da ULA), registrador de 1 bit, registradores de 32 bits, Datapath (Bloco Operacional), uma máquina de estados (correspondente a um Bloco de Controle) e os Testbenchs necessários. Todos os blocos foram descritos em VDHL (VHSIC Hardware Description Language). Para o escopo deste trabalho, foi utilizada uma simplificação do restante do processador, assumindo a existência de três registradores de dados (dois de entrada de dados e um para a saída dos dados) que se ligam diretamente na ULA. Para controlar o fluxo de dados e operações, foi projetada uma máquina de estados, que será ligada ao datapath pelo TopLlevel do projeto. Este trabalho está dividido em 6 partes principais: Introdução do tema do projeto e dos conceitos básicos referentes ao tema; Descrição da função (responsabilidade) de cada bloco projetado; Explicação sobre a metodologia utilizada para a construção de cada bloco, fornecendo mais detalhes sobre seu funcionamento; Explanação a respeito do funcionamento dos Testbenchs projetados; Apresentação dos desafios superados ao longo da realização do trabalho e a conclusão. Este projeto conclui a parte prática da disciplina de Sistemas Digitais, implementando os conceitos teóricos e práticos apresentados e desenvolvidos ao longo do curso.**

***Keywords*—ULA, MIPS, Sistemas Digitais, VHDL**

I. Introdução

Este Relatório centra-se na arquitetura do microprocessador MIPS (Microprocessor without Interlocked Pipeline Stages), conceito fundamental nos estudo dos sistemas digitais, e a implementação prática de sua Unidade Lógica e Aritmética (ULA) em VHDL. Este trabalho prático tem como objetivo materializar conceitos complexos e permitir uma compreensão mais profunda do funcionamento de sistemas digitais.

O MIPS é uma arquitetura de microprocessador do tipo RISC (Reduced Instruction Set Computing) desenvolvida pela MIPS Technologies. Este tipo de arquitetura é caracterizado pela execução de um conjunto pequeno de instruções simples com alto desempenho. A arquitetura MIPS é de grande relevância no domínio dos sistemas embarcados, sendo amplamente aplicado em dispositivos como roteadores, televisores e consoles de videogame, por exemplo.

Durante nossas aulas teóricas, aprendemos sobre a simplicidade e eficiência dos processadores MIPS e sua ampla aplicabilidade. Neste trabalho prático, tivemos a oportunidade de aplicar esses conceitos à realidade com o foco na ULA do MIPS em VHDL.

A ULA é a parte do processador responsável pela realização de operações lógicas e aritméticas. No caso da arquitetura MIPS, essa unidade é essencial, pois executa as operações fundamentais que o processador necessita para funcionar, como adições, subtrações, e operações lógicas.

No decorrer deste relatório, descreveremos detalhadamente a implementação da ULA do MIPS em VHDL, permitindo assim uma percepção clara da interação entre os elementos teóricos e práticos no estudo dos sistemas digitais

II. Função de cada bloco

A. ULA

A ULA do MIPS dá suporte para as seguintes instruções do processador.: Load word (LW), Store word (SW), Salto condicional (beq), Adição (add), Subtração (sub), AND bit a bit (and), OR bit a bit (or) e Set on less Than (slt)

Para dar suporte a essas instruções, A ULA realiza as seguintes operações: adição, subtração, and bit a bit, or bit a bit e set on less than (que verifica se o valor de um registrador é menor que o valor do outro).

B. Ctrl ULA

O controle da ULA determina a operação que será realizada no bloco de operações ULA, a partir de dois sinais de entrada de dados: o sinal funct de 6 bits e o sinal ULAOp de 2 bits.

C. *Máquina de estados (BC)*

Correspondente ao Bloco de Controle de nossa versão simplificada do processador. Essa máquina de estados determina os sinais de enable dos registradores de entrada e de saída de dados, o sinal de pronto e lida com o sinal de reset. Ela possui 4 estados: RESET, S0, S1 e S2.

D. *Datapath*

O datapath é o bloco que conecta os registradores de entrada e saída de dados, a ULA e o controle da ULA.

E. Top-Level

Em relação à interação com o usuário, o Toplevel é responsável por receber os dados a serem utilizados para a realização da operação (dados do cálculo e dados de seleção do modo da ULA). Ele recebe também do usuário o sinal de reset.

Além disso, ele é responsável por fornecer ao usuário o resultado da operação e sinalizar o sinal de "pronto" que indica o final da operação.

A respeito do funcionamento interno do programa, o TopLevel é responsável por conectar o bloco de Datapath com o bloco da máquina de estados (Bloco de Controle). É responsável também por enviar para esses dois blocos os sinais fornecidos pelo usuário.

III. Metodologia de desenvolvimento de cada bloco

A. ULA

O bloco operativo ULA possui três entradas: "dataA" e "dataB" referentes aos dados que serão operados (vindos dos registradores RegA e RegB presentes no datapath) e o sinal de controle "sel". A função que será realizada sob os dados das entradas é definida pelos valores de sel.

Para a implementação da ULA, foram utilizadas as funções otimizadas do VHDL para as operações de soma e de subtração. Os sinais de entrada de dados são definidos como "unsigned". Para as operações de AND e OR bit a bit, também foram utilizadas as funções prontas and e or do VHDL.

Para lidar com os casos de overflow nas operações da ULA, adicionou-se um sinal de saída que informa essa ocorrência. Para implementar esse sinal, os dados das entradas da ULA dataA (vindo do registrador RegA) e dataB (vindo do registrador RegB) são transformados para 33 bits com a concatenação de um bit '0' como bit mais significativo. Esse incremento de 1 bit no tamanho dos sinais operados possibilita que o bit mais significativo, ao final da operação, receba o valor lógico '1' em caso de overflow e '0' nos demais casos. Esse bit é ligado na saída de overflow, como sinalizador do possível erro.

B. Ctrl ULA

O bloco Controle da ULA tem duas entradas (a entrada do sinal ULAOp (de 2 bits) e a entrada do sinal funct (de 6 bits)) e uma saída (sel (com 3 bits)). Os valores da saída sel é que definirão o comportamento da ULA.

Ambos os sinais ULAOp e funct vêm de entradas de dados do top-level do projeto. O sinal ULAOp é responsável por determinar o tipo de instrução que o MIPS está realizando. Para os casos de instruções do tipo I (lw, sw, beq), esse é o único sinal necessário para determinar o comportamento da ULA. O sinal de Controle "funct" delimita, dentre as instruções do tipo R, qual será a operação a ser realizada pela ULA. Esses sinais estão melhor explicitados na tabela de correspondência abaixo, que mostra a relação dos sinais de entrada com o valor da saída "sel":

TABELA I. Sinais de Controle da ULA

Instrução	ULAOp	Campo "funct"	Operação da ULA	sel
lw	00	XXXXXX	adição	010
sw	00	XXXXXX	adição	010
beq	01	XXXXXX	subtração	110
add	10	100000	adição	010
sub	10	100010	subtração	110
and	10	100100	and	000
or	10	100101	or	001
slt	10	101010	set on less than	111

Fonte: Elaborado pelos autores (2023)

Fig. 1. Correspondência entre "funct" & "ULAOp" com "controle da ULA".

Para descrição do VHDL, foi utilizado o formato "when - else" como forma de determinar os valores da saída. Foi definido o valor "011" como valor sinalizador de erro para os casos em que os valores de entrada fornecidos não correspondem a um valor válido dentre as opções.

C. Máquina de estados

O controle de estados do projeto possui os seguintes sinais de entrada: O sinal de Clock (clk), o sinal de ativação do reset (reset), e o sinal de controle iniciar. Como saída, ele possui os sinais de controle de enable dos registradores A, B e C

(EnRegA, EnRegB, EnRegC, respectivamente) do projeto simplificado do processador, o sinal de reset que será ligado com o datapath e o sinal de pronto.

O projeto da máquina de estados é de uma máquina de Moore, ou seja, o valor do sinal de saída depende unicamente do valor do estado atual.

Ao todo, são 4 estados:

- RESET: Estado com ativação pelo sinal de reset fornecido pelo usuário. Envia sinal de reset para o Datapath;
- S0 :Estado inicial/final;
- S1: Estado que atribui valor lógico alto para os sinais EnRegA e EnRegB;
- S2: Estado que atribui valor lógico alto para o sinal EnRegC .

O sinal de reset do usuário é assíncrono e a qualquer momento ele pode ser ligado, forçando a ida para o estado "R "da máquina. Neste estado, este bloco envia um sinal de reset para os registradores do datapath. Do estado reset, a máquina passa automaticamente para o estado "S0".
O sinal iniciar habilita a passagem do "S0" para o "S1".
O sinal de saída "pronto" indica que a máquina se encontra no "S0".

Sobre o controle de estados, é importante mencionar que o bloco principal do trabalho, a ULA, é um bloco puramente combinacional e independe de uma máquina de estados para funcionar. Entretanto, para o propósito deste projeto, verificou-se a necessidade de implementar uma forma de controlar o fluxo de dados nos registradores de entrada e de saída, controlar o valor de sinal pronto, além de habilitar a implementação de um controle de reset.

D. DATAPATH

O Datapath realiza a conexão entre os registradores RegA, RegB, Regc, a ULA e o controle da ULA. Essa conexão é feita da seguinte maneira:

Os registradores RegA e RegB recebem os dados de entrada vindos do TopLevel.
As entradas de 32 bits da ULA se conectam com as saídas dos registradores RegA e RegB. A saída de dados da ULA se conecta com a entrada do registrador RegC. A entrada do sinal de controle "sel" da ULA recebe o sinal de saída do controle da ULA (Ctrl ULA). O bloco Ctrl ULA recebe os seus sinais de entrada ULAOp e funct vindos do top-level. Para o registro do sinal de overflow da ULA, foi acrescentado um registrador de 1 bit ligado na saída de overflow do bloco da ULA.

É no datapath também que os registradores recebem os sinais de enable, clock e reset vindos do top-level (enRegA habilita a escrita do registrador RegA, enRegB habilita a escrita de RegB e enRegC habilita a escrita de RegC).

E. Toplevel

O Toplevel recebe em suas entradas os sinais de iniciar, reset, UlaOp, funct, a e b fornecidos pelo usuário para o controle de início e reset do processamento e para a escolha e realização da operação pela ULA. Ele recebe também o sinal de Clock.

Com esses recebimentos, ele envia os sinais iniciar, reset e clock para o Bloco de Controle (Máquina de estados) e os sinais UlaOp, funct, a e b para o Bloco Operativo (Datapath).

Além disso, o Toplevel realiza a conexão entre o BO e o BC, funcionando como interface para o fluxo de informações que esses blocos trocam entre si.

Ao final da operação do sistema, o Toplevel fornece ao usuário por meio de sua saída o resultado obtido vindo do Datapath.

IV. TESTBENCHS

1) Explicação geral dos Testbenchs:

Os testes são um componente integral de qualquer projeto de desenvolvimento de hardware, fornecendo uma maneira de verificar se o design atende às especificações e realiza as tarefas pretendidas corretamente. No contexto do nosso projeto, utilizamos testbenches VHDL para simular a ULA e o controle da ULA, alimentando-os com entradas pré definidas e verificando se as saídas eram como esperado.

2) Testbenchs individuais:

a) Testbench da ULA:

O primeiro testbench foi projetado para testar a Unidade Lógica Aritmética (ULA). Este teste focou principalmente na função da ULA como a principal executora de operações aritméticas e lógicas. No nosso testbench, aplicamos uma série de operações em diferentes valores de entrada e verificamos se a ULA gerava os resultados corretos. Os casos de teste cobriram todas as operações que a ULA deste projeto foi projetada para realizar (adição, subtração, operações lógicas bitwise (AND, OR) e set on less than (SLT)). Esta cobertura abrangente garante que a ULA esteja funcionando corretamente e seja capaz de realizar todas as operações necessárias.

Para as operações de and, or e soma, os testes utilizaram o mesmo padrão

de definição dos conjuntos de valores de entrada. Em um dos passos, os registradores RegA e RegB receberam ambos o valor máximo comportado para 32 bits (todos os bits em ‘1’). No segundo passo, testou-se a possibilidade extrema de todos os bits de RegA receberem valor lógico alto e todos os bits de RegB receberem valor lógico baixo. Para um terceiro caso de teste, ambos os registradores receberam todos os bits em valor lógico ‘0’. Testou-se também uma possibilidade de entrada com valores aleatoriamente definidos para os registradores.

Para a operação de subtração, foi feito um caso de teste em que ambas as entradas dos registradores estão com todos os bits em valor lógico ‘1’. Foram feitos também dois casos de teste em que as entradas de RegA e RegB estão opostamente carregadas (um dos registradores está com todos os bits em ‘1’ e o outro está com todos os bits em ‘0’). Para abarcar demais possibilidades, testou-se dois valores quaisquer de entrada para os registradores. Esses valores foram testados espelhadamente em dois casos de teste (em um, RegA recebeu o padrão de bits “x” e RegB recebeu o padrão de bits “y”. No outro caso de teste, RegA recebeu “y” e RegB recebeu “x”).

Finalmente, para testar a operação de set on less than , testou-se os seguintes casos (considere A para o valor da entrada de RegA e B para o valor da entrada de RegB): A>B, A<B, e A=B.

Em cada caso, após atribuir valores aos sinais de entrada e esperar um intervalo de tempo especificado pelo sinal passo, fizemos uma assertiva para verificar se a saída correspondia ao valor esperado. Se a saída não correspondesse ao esperado, uma mensagem de erro seria exibida, indicando a operação que havia falhado. Isso nos permite identificar rapidamente qualquer problema com uma operação específica.

O passo definido para os testes deste bloco foi de 20 ns.

b) Testbench do Ctrl ULA:

O segundo testbench foi criado para testar o controle da ULA. Este controle é fundamental para garantir que a ULA realize a operação correta, dependendo dos sinais de funct e ULAOp.

Neste testbench, os valores de entrada dos casos de teste foram definidos de forma a abarcar todas as possibilidades relevantes de seleção dos valores de ULAOp e funct. Isto é: Em um caso de teste, foi atribuído o valor “00” para ULAOp. Como nesse caso o valor do sinal funct não é relevante para a determinação do valor de saída, foi atribuído para ele o valor ‘0’ em todos os bits. Seguindo a mesma metodologia foi feito o caso de teste para o valor “01” de ULAOp. Para testar os casos em que ULAOp recebe “10”, foi feito um caso de teste para cada uma das possibilidades de valor de entrada do sinal “func” apresentadas na tabela I deste relatório.

Em todos esses casos de teste foi verificado se o sinal de saída “sel” correspondia ao esperado. Semelhante ao primeiro testbench, usamos assertivas para verificar se as saídas eram como esperado e exibir uma mensagem de erro se houvesse um problema.

O passo utilizado para os testes deste bloco foi de 10ns.

c) Testbench do Toplevel:

O testbench do toplevel constitui uma simulação de alto nível, responsável por englobar todas as entidades e funcionalidade de nossa Unidade Lógica e Aritmética (ULA). Isso proporciona uma avaliação abrangente do sistema em um contexto que se assemelha ao cenário real de aplicação.

No código do testbench para este módulo, configuramos sinais de entrada para representar diversas operações relevantes. Entre eles, temos o sinal de relógio clk, os sinais de início (iniciar) e reset (reset), que controlam o ciclo de vida das operações , e os sinais de operação e função da ULA (UlaOp, funct), que definem qual operação a ULA deve executar. Os operandos a serem manipulados na operação são representados pelos sinais a e b. O sinal de saída c nos fornece o resultado da operação processada pela ULA.

Nesses testes iniciamos com a configuração dos sinais de entrada, seguida

por um período de espera delimitado pelo sinal passo. Ao término desse intervalo, implementamos uma verificação do sinal de saída c e checamos se condiz com o valor esperado. Caso alguma verificação falhe, será exibido uma mensagem de erro apontando qual operação apresentou comportamento inesperado. Este procedimento é replicado para cada uma das variedades de operação que a ULA realiza: adição, subtração, operações lógicas bitwise (or e and)e operação de comparação (SLT). Para que essa abrangência de teste das operações seja garantida, os valores definidos para as entradas de controle e de dados variam seguindo a metodologia de definição utilizada para os casos de teste dos Testbenchs da ULA e do Ctrl ULA.

O passo definido para os testes deste bloco foi de 80 ns e o valor de período de clock estipulado foi de 20ns.

d) Conclusão:

No geral, acreditamos que nossos testbenches fornecem uma cobertura abrangente das funcionalidades que a ULA e seu controle devem fornecer. No entanto, estamos cientes que, mesmo com testes abrangentes, ainda é possível que existam casos marginais ou erros que não foram capturados.

V. Resultado da síntese

TABELA II. Síntese do projeto

Family	Cyclone III
Device	EP3C5F256C6
Timing Models	Final
Total logic elements	229/5136 (4%)
Total combinational functions	208/5136 (4%)
Dedicated logic registers	100/5136 (4%)
Total registers	100
Total pins	108/183 (59%)
Total virtual pins	0
Total memory bits	0/423936 (0%)
Embedded Multiplier 9-bit elements	0/46 (0%)
Total PLLs	0/2 (0%)

Fonte: Elaborado pelos autores (2023)

VI. Análise dos Atrasos Críticos

No módulo toplevel, observamos que o maior tempo de setup ocorre no bit 2 do sinal funct com o valor de 8.452 ns. Além disso, o maior atraso do relógio para a saída ocorre no bit 13 do sinal c sendo este de 7.336 ns

.

No módulo datapath, o maior tempo de setup é de 8.400 ns, que ocorre no bit 5 do sinal funct. O maior atraso do relógio para a saída ocorre no bit 2 do sinal c com o valor de 6.869 ns

Na ULA, o maior atraso de propagação ocorre entre o bit 1 do sinal de entrada dataA e o bit 28 do sinal de saída resultado, com um valor de 14.026 ns.

No módulo controlador de estados, o maior tempo de setup é de 2.175 ns, associado ao sinal iniciar. O maior atraso do relógio para a saída ocorre no sinal enReB, sendo este de 6.403 ns.

No registrador de 1 bit, o atraso é de 4.633 ns para o sinal q. No registrador de múltiplos bits, o maior atraso ocorre no bit 3 do sinal q, sendo este de 6.357 ns

VII. Desafios encontrados e superados

A implementação do projeto da ULA do MIPS apresentou diversos desafios que contribuíram para o crescimento do grupo e desenvolvimento de habilidades essenciais de gerenciamento de projetos e trabalho em equipe.

O principal desafio enfrentado foi a coordenação eficaz de uma equipe recém formada em um curto período de tempo. Inicialmente composto por três duplas independentes, a reconfiguração do grupo para uma equipe de seis pessoas exigiu uma rápida adaptação e estabelecimento de um ritmo de trabalho comum entre todos os envolvidos.

Os integrantes do grupo, não estando familiarizados uns com os outros, inicialmente enfrentaram dificuldades em se alinhar sobre a direção e ritmo do projeto. Esse desconhecimento inicial gerou obstáculos para a colaboração eficaz para o cumprimento das tarefas dentro do cronograma.

Quanto à estrutura interna do projeto, um ponto de dúvida inicial foi a forma que deveria-se lidar com os casos de overflow na ULA. A solução encontrada foi colocar um sinal de saída que informa a ocorrência ou não do overflow.

Outro desafio significativo foi o entendimento comum a respeito da organização dos blocos e seus níveis de abstração instituídos. Quanto a isso, um ponto que demandou mais atenção foi o projeto e o desenvolvimento do Top-Level do MIPS. O encontro de diferentes compreensões a respeito de qual deveria ser a função a a forma de implementação desse bloco foi um desafio a ser superado durante a fase de desenvolvimento , o que prolongou o tempo necessário para finalização desta parte do projeto.

No entanto, estes desafios possibilitaram aos indivíduos do grupo desenvolverem habilidades valiosas em gerenciamento de projetos, colaboração eficaz e resolução de conflitos. Além disso, esses obstáculos incentivaram o grupo a aprimorar a comunicação interna e estabelecer estratégias mais eficazes que abrangessem todos os membros.

VIII. Conclusão

Apesar dos desafios encontrados, a implementação da Unidade Lógica e Aritmética (ULA) do MIPS foi concluída com sucesso. Foi possível desenvolver e integrar todos os componentes essenciais da ULA, incluindo o controle da ULA, a máquina de estados, o datapath e o toplevel, todos fundamentais para o funcionamento eficiente do microprocessador. Além disso, os testbenchs realizados para cada módulo confirmaram o desempenho esperado de cada componente e da ULA como um todo.

O desenvolvimento deste projeto possibilitou um aprofundamento significativo de nossos conhecimentos acerca da arquitetura MIPS e dos sistemas digitais como um todo. A necessidade de materializar conceitos teóricos em um sistema funcional proporcionou uma compreensão mais rica e aplicada desses conceitos. Além disso, o trabalho em equipe exigido pelo projeto permitiu que desenvolvêssemos habilidades essenciais de colaboração e comunicação intrapessoal.

Referências

[1] J. L. Güntzel e I. Seidel ,“Aula 7P – Descrição de FSMs em VHDL & Relatório Prático II,” Departamento de Informática e Estatística (INE). Abril 2023.

[2] J. L. Güntzel e I. Seidel ,“Descrição de testbenches em VHDL & Relatório Prático III” Departamento de Informática e Estatística (INE). Abril 2023.

[3] J. L. Güntzel e I. Seidel ,“Aula 4P – VHDL (declarações, processos, generate)” Departamento de Informática e Estatística (INE). Abril 2023.

[4] J. L. Güntzel, R. Cancian e I. Seidel ,“Aula 4P – Aula 7-T O Processador MIPS: conjunto de instruções e exemplos de uso (noções de programação assembly)” Departamento de Informática e Estatística (INE). 2023.

[5] J. L. Güntzel, R. Cancian e I. Seidel ,“Aula 8T 3. O Processador MIPS monociclo: construção do bloco operativo (incluindo a ULA e seu controle).” Departamento de Informática e Estatística (INE). 2023.