Unidade Lógica Aritmética do MIPS

INE5406 - Sistemas Digitais

1st Carlos Henrique Zanon
Engenharia ELetrônica
UFSC
Florianópolis, Brasil
carlos.zanon@grad.ufsc.br

2nd Heloísa Jonck Hammes
Ciências da Computação
UFSC
Florianópolis, Brasil
heloisa.hammes@grad.ufsc.br

3rd Lucas Ryan Carneiro
Dept. de engenharia Elétrica e Eletrônica
UFSC
Florianópolis, Brasil
lucasryan9999@gmail.com

4th Leonardo Fonseca Franchini
Ciências da Computação
UFSC
Florianópolis, Brasil
leoffrano@gmail.com

5th Viccenzo Escarrone
Ciências da Computação
UFSC
Florianópolis, Brasil
viccenzoescarrone@gmail.com

6th Rebeca Ayumi Komatsu
dept. de engenharia elétrica e eletrônica
UFSC
Florianópolis, Brasil
komatsurebeca9@gmail.com

***Abstract*—Este documento tem por objetivo registrar o trabalho final da disciplina de Sistemas Digitais. O projeto desenvolvido consiste na Unidade Lógica Aritmética (ULA) do MIPS, com o adicional de um processo de redundância.**

***Index Terms*—ULA, MIPS, Sistemas Digitais**

I. Introdução

Uma versão simplificada da Unidade Lógica Aritmética do MIPS foi estudada durante as aulas teóricas da disciplina e dentre suas operações estão OR, AND, soma e subtração. Além da estrutura estudada, um processo de redundância foi implementado para aumentar a complexidade do projeto. Todas essa operações serão explicitadas mais detalhadamente no decorrer deste relatório.

II. Organização do Projeto

A. ULA

A Unidade Lógica Aritmética (ULA) foi dividida em uma parte lógica e outra aritmética, além de um módulo para o calculo da operação Set On Less Then e dois multiplexadores, a seguir explicamos melhor cada módulo:

SOLT: Neste módulo é feito o processamento da operação Set On Less Then, na qual comparamos os valores das entradas A e B, e caso o valor de A seja menor do que B, a saída deste módulo é colocada em nível lógico alta.

aritmetica: Responsável por fazer as operações aritméticas da ULA, sendo elas as somas e as subtrações a partir dos valores das entradas A e B e do controle c2.

logica: Este módulo processa as operações lógicas de seu bloco principal, sendo elas as operações AND e OR, também processadas a partir dos valores de entrada de A e B, os quais sofrerem as operações lógicas bit a bit.

Multiplexadores: Ambos multiplexadores de 2 entradas e uma saída, além do controle, de ambos e sinal de overflow de um deles. O primeiro multiplexador é utilizado para fazer a escolha do tipo de operação se ela será do tipo lógica ou aritmética, e o segundo usado para fazer a escolha entre a saída deste primeiro multiplexador ou a saída do módulo SOLT, ambos são determinados através de sinal de controle.

O diagrama a nível RT dos sinais pode ser vista na "Fig. 1":

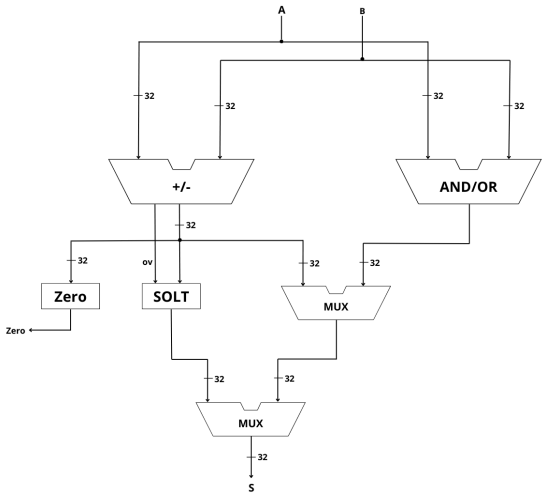

Fig. 1. Esquemático da ULA

B. Bloco Operativo

O bloco operativo foi dividido basicamente em duas partes: a ULA, que consiste nos operadores assíncronos descritos no tópico anterior e as barreiras de registradores, que permitem com que o circuito opere de maneira síncrona.

Foram inseridos três registradores de 32 bits, sendo dois deles para as entradas "A" e "B" e um para a saída "Resultado", além de um registrador de 3 bits para os sinais de controle "C". O único sinal do bloco operativo que não passa por um registrador é o sinal indicador de zero. O esquemático do bloco operativo a nível RT pode ser visualizado na "Fig. 2":

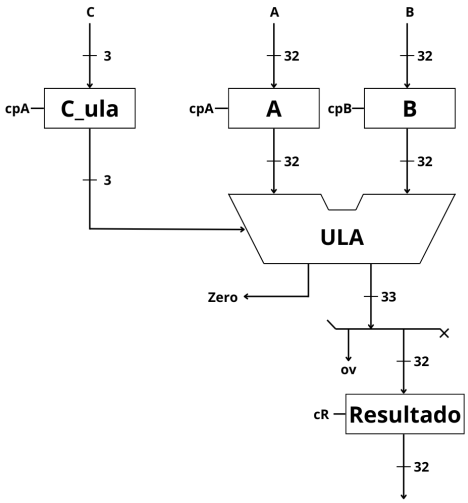

Fig. 2. Esquemático do bloco operativo da ULA

C. Bloco de Controle

O Bloco de controle contém os estados S0, S1, S2 e S3, cada um desempenhando funções específicas no processo de execução das operações lógicas e aritméticas. No estado S0, o sistema aguarda o sinal de iniciar para começar um novo ciclo, e o finaliza com a ativação do sinal pronto. No estado S1, o registrador que recebe o sinal de controle da ULA é ativado pelo sinal cULA, passando-o para a ULA, para que esta possa realizar uma operação específica definida pelo controle C. Em seguida, no estado S2, os valores de A e B são carregados nos registradores correspondentes usando os sinais cpA e cpB. No estado S3, o sinal c-red ativa o processo de redundância e depois o resultado da operação da ULA é armazenado no registrador de resultado pelo sinal cR.

O esquemático do bloco de controle pode ser visto na "Fig. 3":

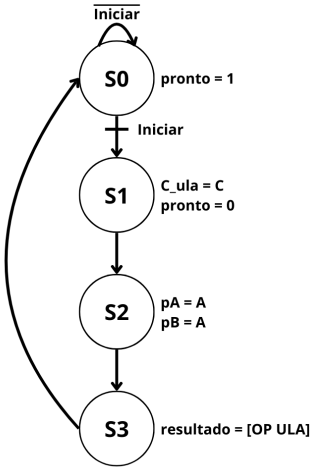

Fig. 3. Esquemático do bloco de controle da ULA

D. Processo de redundância

O processo de redundância se baseia na repetição do processamento, desta forma em ambientes com altas taxas de radiação eletromagnética, como no espaço, caso ocorra problemas de bitflip em um dos processamentos temos que a saída de determinado módulo pode ser mais confiável se forem feitos mais vezes, logo, o que foi desenvolvido no módulo de redundância da ULA, foi o processamento repetitivo das instruções abrangidas pela ULA. Após o processamento das ULAs é feito uma analise para ser determinado o valor correto, há uma comparação entre os valores e a determinação do valor real do processo, juntamente com a desconsideração da resposta alterada.

O esquemático do processo de redundância pode ser visto na "Fig. 4":

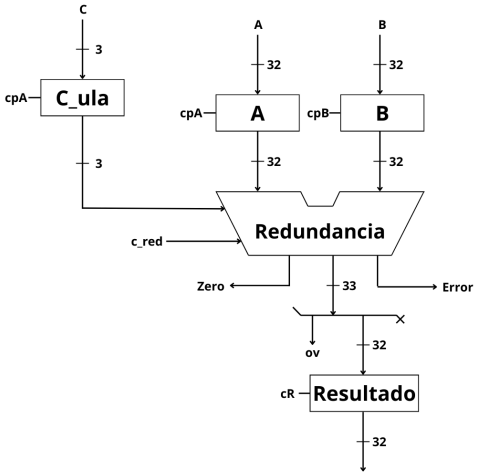

Fig. 4. Esquemático do processo de redundância da ULA

III. Testes

Para a realização de testes de verificação do projeto, utilizaram-se dois recursos: golden model, roteiro escrito em linguagem python para a geração do arquivo de estímulos, e o testbench, descrição em vhdl dentro do projeto a qual realiza a leitura dos estímulos fornecidos e verifica as respostas esperadas (de acordo com o golden model), comparando-as com o obtido pelo sistema descrito.

A. Golden Model

O código do golden model divide-se em duas funções principais: 'calculo-ula', a qual recebe os operandos A e B, e o sinal que indicará a operação a ser realizada (c-ula), de acordo com o dígito escolhido, a função realizará o cálculo de AND/OR bit a bit, soma, subtração ou 'solt' (set on less than), retornando o resultado e o sinal-zero, o qual levantará caso o resultado anterior zere; Além disso, há a função para a geração dos estímulos que recebe o arquivo de escrita, o número de valores de teste e a largura dos bits dos valores de entrada.

A função anterior também gera os operandos de maneira aleatória e recebe os resultados de 'calculo-ula', transformando, por fim, todos os valores de teste para o formato de vetores de bits (considerando casos de overflow) e escrevendo-os no arquivo de estímulos.

B. Testbench

Como informado anteriormente, o segundo recurso utilizado para a verificação dos resultados do projeto desenvolvido é o testbench, descrito em vhdl e compilado juntamente do projeto. Para o desenvolvimento do mesmo, foram descritas, inicialmente, a geração do clock e a instanciação do DUV (o arquivo topo do projeto) garantindo a associação das portas com os sinais definidos para teste.

Em seguida, tem-se o processo de estímulos, o qual define variáveis auxiliares para a leitura do arquivo e correta associação com os sinais definidos. Após isso, inicia-se o teste com o 'reset' e o sinal 'iniciar', para, em seguida, garantir a leitura e associação de todos os dados de estímulos através de um loop que irá, a cada borda de relógio e espera do sinal pronto ser igual a 1, verificar os resultados (saída 'S', saída 'Zero' e saída 'Erro', responsável por garantir que não houve erro durante o processo de redundância).

Por fim, após o sinal 'finished' ser definido como 1, finaliza-se a descrição do teste com uma mensagem de 'Teste Concluído', garantindo que todos os valores foram lidos e associados corretamente.

C. Análise de Resultados

De acordo com a análise dos resultados obtidos durante a etapa de testes e considerando a utilização de um escopo de números gerados aleatoriamente, pode-se comprovar a eficácia do projeto desenvolvido para qualquer caso de teste, já que não foram vistos erros significativos que pudessem prejudicar a concretização dos resultados esperados pelo roteiro de estímulos. Além disso, a garantia obtida pelo processo de redundância do projeto enfatiza robustez e consistência do resultado, evitando que qualquer possível erro (de sintaxe, lógica ou do próprio processamento) prejudique-o.

Os resultados (em onda) podem ser vistos na "Fig. 5":

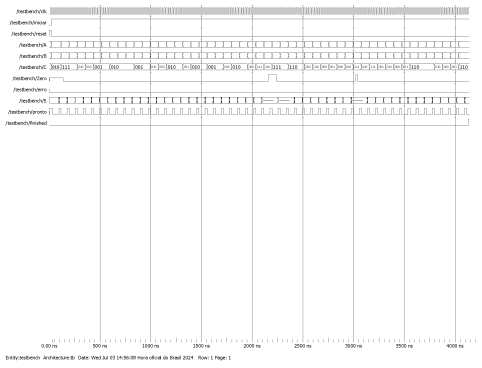

Fig. 5. Resultados obtidos

IV. Resultados de Síntese

Quartus:
OS: "Windows 11
Versão: 13.1.0 Build 162 10/23/2013 SJ Web Edition
Fpga:
- Família: Cyclone III
- Dispositivo: EP3C5F256C6
Utilização:
Total combinational functions: 230
Dedicated Logic Registers: 103
Total Pins: 105
Atraso:
Fmax: 100.01
Unidade: MHZ

V. Conclusão

Em suma, pode-se concluir que o objetivo central deste projeto, além de reproduzir uma versão simplificada da ULA do MIPS (com as operações básicas AND, OR, soma, subtração e 'set on less than'), é enfatizar a importância do processo de redundância em qualquer projeto que lide com entradas de dados e, portanto, esteja sujeito a riscos de processamento do hardware. Nesse sentido, a redundância para este sistema funciona captando os possíveis erros de processamento da ULA e, ainda, permite apresentar os valores corretos processados, realizando as operações citadas com maior consistência, trazendo, dessa forma, uma menor sucessão a erros não previstos.