Desenvolvimento e Implementação de uma ULA de 8 Bits em Verilog: Arquitetura Modular e Validação

Eduardo de Melo Flam, Giovana de Oliveira Machado, Heloísa Tavares Nunes
Curso de Ciência da Computação
Universidade Católica de Santos
Santos, Brasil

***Resumo*—Este artigo apresenta o projeto e a implementação de uma Unidade Lógica e Aritmética (ULA) de 8 bits utilizando a linguagem de descrição de hardware Verilog. O trabalho detalha a arquitetura modular adotada para a execução de um conjunto otimizado de oito operações: aritmética (soma, subtração), lógica (inversão) e comparação (magnitude e igualdade). A metodologia abrange desde a concepção dos diagramas de blocos e síntese RTL até a validação funcional via simulação de *testbench*. Os resultados confirmam a precisão do circuito no processamento de vetores de 8 bits e a eficácia do gerenciamento de barramento via *buffers tristate*.**

I. Introdução

A Unidade Lógica e Aritmética (ULA) é um dos principais blocos funcionais em circuitos digitais, sendo responsável por executar operações lógicas e aritméticas sobre dados binários. Segundo Tavares e Couvre [1], uma ULA recebe dois operandos como entrada e, através de um sinal de controle, realiza uma das operações disponíveis, retornando o resultado e os *flags* que indicam o estado da operação.

A importância da ULA está no fato de ela ser o núcleo responsável pelas operações matemáticas e decisões lógicas de um sistema computacional. Em processadores modernos, sua estrutura é projetada para aumentar o desempenho, reduzir consumo energético e ocupar menos área em silício. Estudos recentes, como os de Oliveira et al. [3], mostram a evolução do conceito para versões de alta complexidade, reforçando a importância da ULA como base de arquiteturas modernas.

O objetivo deste projeto é estudar e projetar uma ULA de 8 bits implementada em Verilog, com foco nas operações: subtração, adição, inversão (A e B), igualdade, maior, menor e diferente.

II. Metodologia

A implementação seguiu a metodologia *top-down*, onde o módulo principal (`ula.v`) atua como integrador de sub-blocos especializados. A sincronização do sistema é garantida por um sinal de *clock* global (`ck`), assegurando que as transições de estado ocorram de maneira controlada.

A. Armazenamento e Controle de Fluxo

A estabilidade dos sinais é crítica em circuitos digitais. Para isso, as entradas A e B, bem como a saída S, são isoladas por registradores de 8 bits, conforme ilustrado na Fig. 1.

Como observado na Fig. 1, a estrutura interna utiliza um banco de *Flip-Flops* tipo D em paralelo. Isso garante que

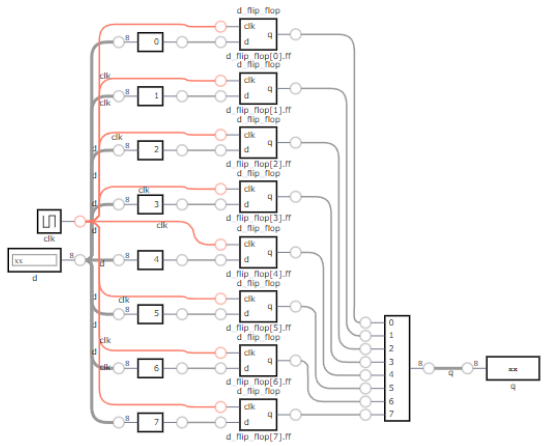

Figura 1. Esquemático do Registrador composto por Flip-Flops tipo D.

o vetor de dados de 8 bits seja capturado e armazenado simultaneamente na borda do *clock*, prevenindo que oscilações espúrias nas entradas afetem o processamento interno da ULA durante o ciclo de operação.

Para o gerenciamento do barramento de saída, utilizou-se uma estratégia baseada em decodificação e estados de alta impedância. O módulo `Decoder` (Fig. 2) interpreta o código de operação (`opcode`).

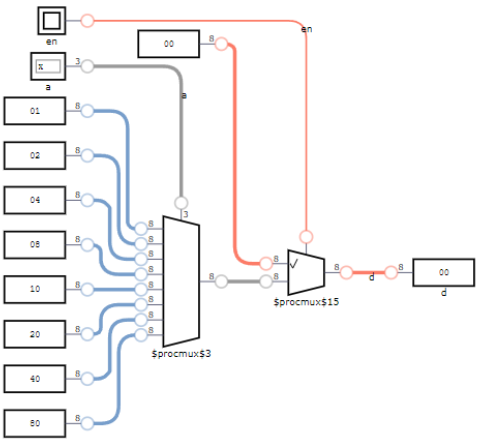

Figura 2. Decodificador 3 para 8 com saídas de habilitação.

A lógica apresentada na Fig. 2 converte a entrada de 3 bits

em 8 linhas de controle mutuamente exclusivas. Cada linha ativa um único bloco funcional por vez. Isso é essencial para o funcionamento dos *buffers tristate* (Fig. 3), que conectam os módulos ao barramento comum.

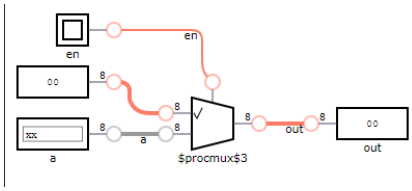

Figura 3. Buffer Tristate para isolamento de barramento.

O componente da Fig. 3 atua como uma chave eletrônica: quando desabilitado, sua saída assume estado de alta impedância ("Z"), isolando eletricamente o módulo do restante do circuito e evitando curto-circuitos lógicos no barramento compartilhado.

B. Operações Aritméticas

A operação de adição (opcode 000) foi implementada utilizando a arquitetura *Ripple Carry Adder*, visualizada na Fig. 4.

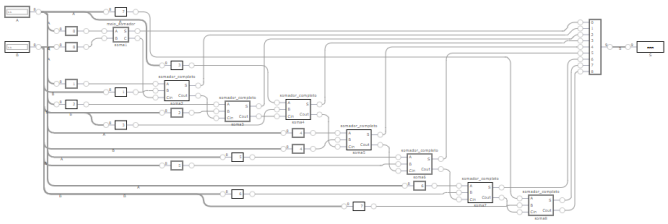

Figura 4. Somador de 8 bits com propagação de carry.

A análise da Fig. 4 revela o encadeamento de oito somadores completos (*Full Adders*). O bit de "vai-um"(C_{out}) de cada estágio alimenta o C_{in} do estágio seguinte. Embora introduza um atraso de propagação proporcional ao número de bits, essa topologia oferece simplicidade de implementação e baixo consumo de área para a largura de 8 bits proposta.

Para a subtração (opcode 001), utilizou-se a lógica de complemento, conforme o esquemático da Fig. 5.

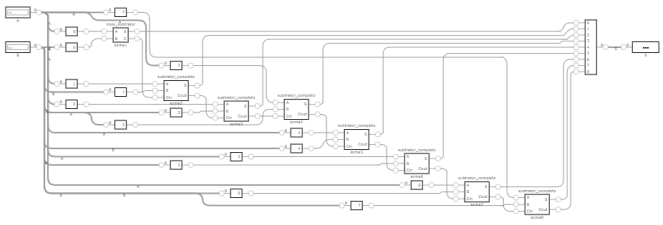

Figura 5. Subtrator de 8 bits utilizando lógica de empréstimo.

O circuito da Fig. 5 opera calculando a diferença $A - B$. A estrutura modular permite que o bit de empréstimo (*borrow*) se propague do bit menos significativo para o mais significativo, garantindo a correção aritmética da operação em representação binária.

C. Lógica e Comparação

As operações lógicas implementadas correspondem à inversão bit a bit (*bitwise NOT*) dos operandos. As Figuras 6 apresentam os módulos responsáveis por esta operação para as entradas A e B, respectivamente.

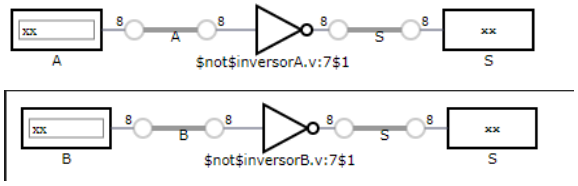

Figura 6. Módulos de inversão lógica para as entradas A (topo) e B (fundo).

Esses módulos aplicam a porta lógica NOT independentemente a cada um dos 8 bits do barramento, permitindo a obtenção do complemento de um do valor de entrada.

Para as operações de comparação de magnitude, o módulo "Maior"é detalhado na Fig. 7.

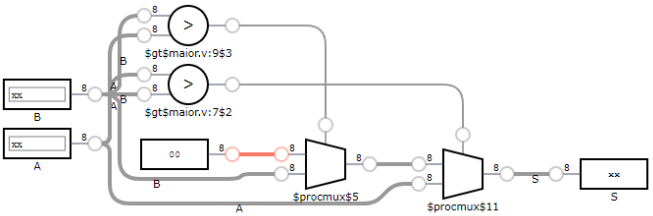

Figura 7. Lógica do comparador de magnitude com multiplexação.

Diferente de comparadores simples que retornam apenas um *bit* de status, a implementação mostrada na Fig. 7 utiliza multiplexadores. A lógica compara os vetores A e B; se $A > B$, o multiplexador direciona o próprio barramento de dados de A para a saída; caso contrário, direciona B ou zero. Isso permite que a ULA retorne diretamente o maior valor numérico.

A verificação de igualdade (Fig. 8) segue uma lógica de coincidência de bits.

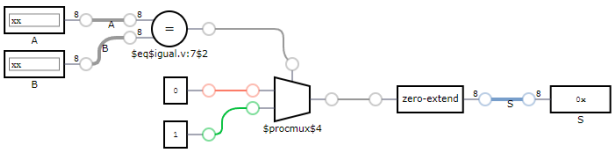

Figura 8. Comparador de Igualdade.

O circuito da Fig. 8 compara bit a bit os vetores de entrada. Caso todos os bits sejam idênticos, a lógica combinacional

resulta no vetor de saída `00000001` (Verdadeiro), facilitando o uso do resultado em desvios condicionais de processadores.

III. Resultados

A validação do projeto foi realizada através de simulação de bancada (*testbench*), submetendo a ULA a vetores de teste que cobrem todas as funcionalidades. A Fig. 9 apresenta o cronograma de sinais.

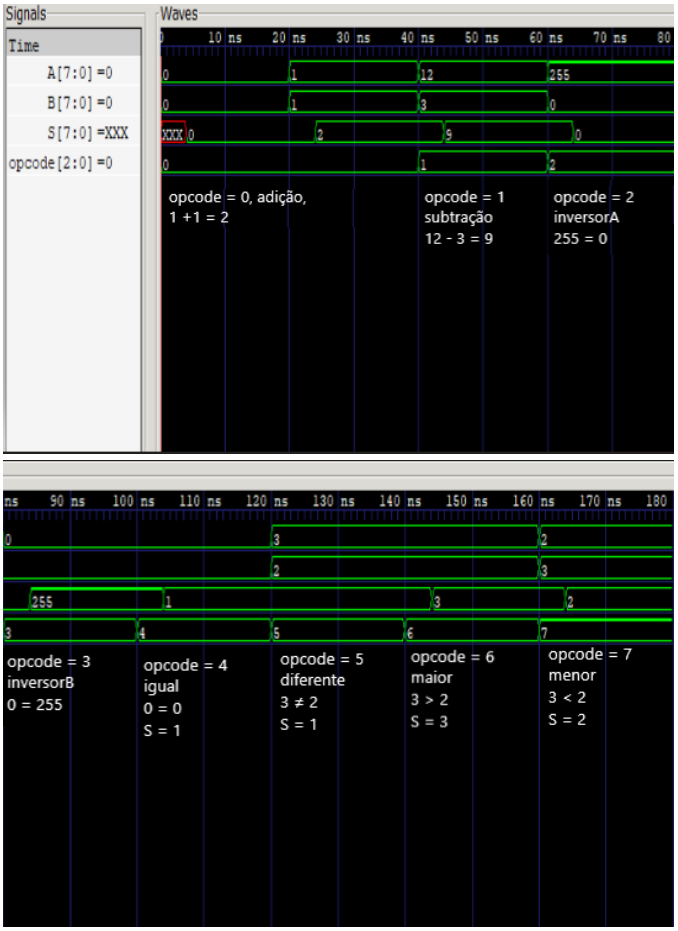

Figura 9. Waveform da simulação. Topo: Operações Aritméticas e Inversão A. Fundo: Inversão B e Comparação.

A análise do cronograma da Fig. 9 comprova a integridade funcional do projeto:

- **0 a 40ns (Aritmética):** Inicialmente, com `opcode=0`, a soma de $1 + 1$ resulta em 2. Ao alterar para `opcode=1` e entradas 12 e 3, o resultado estabiliza em 9, validando o subtrator.
- **40 a 80ns (Lógica):** A operação de inversão transforma a entrada 255 (todos bits em 1) para 0, confirmando a atuação das portas NOT em paralelo.
- **80 a 180ns (Comparação):** O sistema identifica corretamente desigualdades e magnitudes. Ao comparar 3 e 2, os módulos de seleção retornam corretamente os valores sem conflitos no barramento, provando a eficácia do decodificador.

IV. Considerações Finais

O desenvolvimento deste projeto permitiu consolidar os conceitos de circuitos digitais através da prática em Verilog. A análise dos esquemáticos sintetizados demonstra que a abordagem modular não apenas organiza o código, mas resulta em hardware estruturado e eficiente.

A simulação confirmou que a arquitetura proposta atende aos requisitos da Pesquisa Curricularizada, executando as 8 operações básicas com precisão. O uso de *buffers tristate* mostrou-se indispensável para a multiplexação econômica do barramento de saída. Conclui-se que a ULA projetada é funcional e apta a integrar sistemas de processamento elementares.

Referências

[1] TAVARES, Tiago; COUVRE, Marcos. *Unidade Lógica e Aritmética (ULA)*. EA773, 2015.

[2] BADWAN, Mohamad. *Projeto de uma Unidade Lógica Aritmética*. UNIPAMPA, 2011.

[3] OLIVEIRA, A. F. et al. *Implementação e Simulação de uma ULA de 16 Bits*. 2018.