Implementação uma Unidade Lógica Aritmética (ULA) de 4 bits em FPGA Altera Cyclone IV: um estudo de caso utilizando VHDL

Vitor Amadeu Souza[1]; 0009-0002-1857-6799

1 – UniFOA, Centro Universitário de Volta Redonda, Volta Redonda, RJ.
vitor.amadeu@foa.org.br

Resumo: Este trabalho apresenta o desenvolvimento e análise de uma Unidade Lógica Aritmética (ULA) de 4 bits implementada em Field-Programmable Gate Array (FPGA) Altera Cyclone IV EP4CE22E22C6 utilizando linguagem VHDL. A ULA desenvolvida realiza quatro operações fundamentais: AND lógico, OR lógico, adição e subtração, controladas por um comando de 2 bits. A metodologia empregada baseou-se em design digital síncrono, seguindo princípios de arquitetura de computadores e lógica digital. Os resultados obtidos através de simulação comportamental demonstram o funcionamento da ULA para todas as operações implementadas, com tempos de resposta adequados para aplicações em sistemas digitais. A análise dos resultados demonstra a viabilidade da implementação proposta, apresentando características de desempenho compatíveis com aplicações educacionais e prototipagem de sistemas embarcados.

Palavras-chave: FPGA. VHDL. Unidade Lógica Aritmética. Altera Cyclone IV. Lógica Digital.

INTRODUÇÃO

As Unidades Lógicas Aritméticas constituem componentes fundamentais na arquitetura de processadores modernos, sendo responsáveis pela execução de operações matemáticas e lógicas essenciais ao processamento de dados (Hennessy; Patterson, 2017). A implementação de ULAs em Field-Programmable Gate Arrays representa uma abordagem eficiente para prototipagem e desenvolvimento de sistemas digitais, oferecendo flexibilidade de reconfiguração e custos reduzidos em comparação com circuitos integrados de aplicação específica (Brown; Vranesic, 2006).

O desenvolvimento de sistemas digitais utilizando linguagem de descrição de hardware VHDL tem se consolidado como metodologia padrão na indústria de semicondutores, proporcionando abstração adequada para design de circuitos complexos (Ashenden, 2011). A síntese de código VHDL para FPGAs permite validação rápida de conceitos e implementação eficiente de algoritmos em hardware dedicado (Chu, 2007).

A família Cyclone IV da Altera, atualmente parte da Intel, representa uma solução econômica e versátil para implementação de sistemas digitais de baixo a médio porte, oferecendo recursos suficientes para aplicações educacionais e comerciais (Intel, 2016). O dispositivo EP4CE22E22C6 utilizado neste trabalho possui 22.320 elementos lógicos e 594 Kbits de memória embarcada, características adequadas para implementação da ULA proposta (Altera, 2016).

A motivação para este trabalho reside na necessidade de compreensão dos fundamentos de arquitetura de computadores através de implementação real em hardware reconfigurável. Segundo Harris e Harris (2015), a experiência hands-on com design digital proporciona entendimento mais profundo dos conceitos teóricos abordados em cursos de engenharia e sistemas digitais.

O objetivo principal deste estudo consiste na implementação e análise de uma ULA de 4 bits em FPGA Altera Cyclone IV, avaliando seu comportamento através de simulação e verificando a correção funcional das operações implementadas. Os objetivos específicos incluem: desenvolver código VHDL estruturado seguindo boas práticas de design digital;

implementar operações aritméticas e lógicas básicas com controle por comando; validar funcionamento através de simulação comportamental; e analisar características temporais e funcionais dos resultados obtidos.

MÉTODOS

A metodologia adotada para desenvolvimento da ULA seguiu abordagem sistemática de design digital, baseada em princípios estabelecidos na literatura de sistemas digitais (Mano; Ciletti, 2017). O processo foi dividido em cinco etapas principais: especificação funcional, design da arquitetura, codificação VHDL, simulação e validação.

A especificação funcional definiu uma ULA de 4 bits capaz de realizar quatro operações controladas por comando de 2 bits: operação AND lógica (cmd="00"), operação OR lógica (cmd="01"), adição aritmética (cmd="10") e subtração aritmética (cmd="11"). A ULA possui duas entradas de dados de 4 bits (inpA e inpB), uma saída de resultado de 4 bits (leds), uma saída de detecção de zero (zero) e entrada de clock (clk) para operação síncrona.

O design da arquitetura baseou-se no modelo de máquina de estados, implementando lógica combinacional controlada por processo síncrono. Esta abordagem assegura estabilidade temporal e previsibilidade do comportamento (Wakerly, 2018). A detecção de resultado zero foi implementada através de comparação direta com vetor "0000", gerando sinal de flag apropriado.

A codificação VHDL seguiu padrões IEEE 1076-2008, utilizando biblioteca IEEE.STD_LOGIC_1164 para tipos de dados padronizados e IEEE.STD_LOGIC_UNSIGNED para operações aritméticas (IEEE, 2009). A estrutura do código emprega *entity* para definição de interface e *architecture* para implementação comportamental, seguindo metodologia top-down de design hierárquico.

O ambiente de desenvolvimento utilizado foi Quartus II 13.0sp1, ferramenta oficial da Altera para síntese e implementação em FPGAs Cyclone IV. A configuração do projeto especificou o dispositivo EP4CE22E22C6 com encapsulamento TQFP144, operando em condições comerciais de temperatura (ALTERA, 2013).

A simulação comportamental foi conduzida utilizando ModelSim integrado ao Quartus II, empregando testbench automatizado para cobertura completa das combinações de entrada. Os vetores de teste incluíram casos extremos e condições típicas de operação, seguindo metodologia de verificação funcional estabelecida (Bergeron, 2006).

A validação dos resultados baseou-se em análise de formas de onda geradas durante simulação, verificando correção funcional através de comparação com resultados esperados calculados analiticamente. Os critérios de aprovação incluíram correção das operações implementadas, funcionamento da detecção de zero e comportamento temporal adequado.

O código-fonte está disponível para download através do link: https://github.com/vitor-souza-ime/ula.

RESULTADOS E DISCUSSÃO

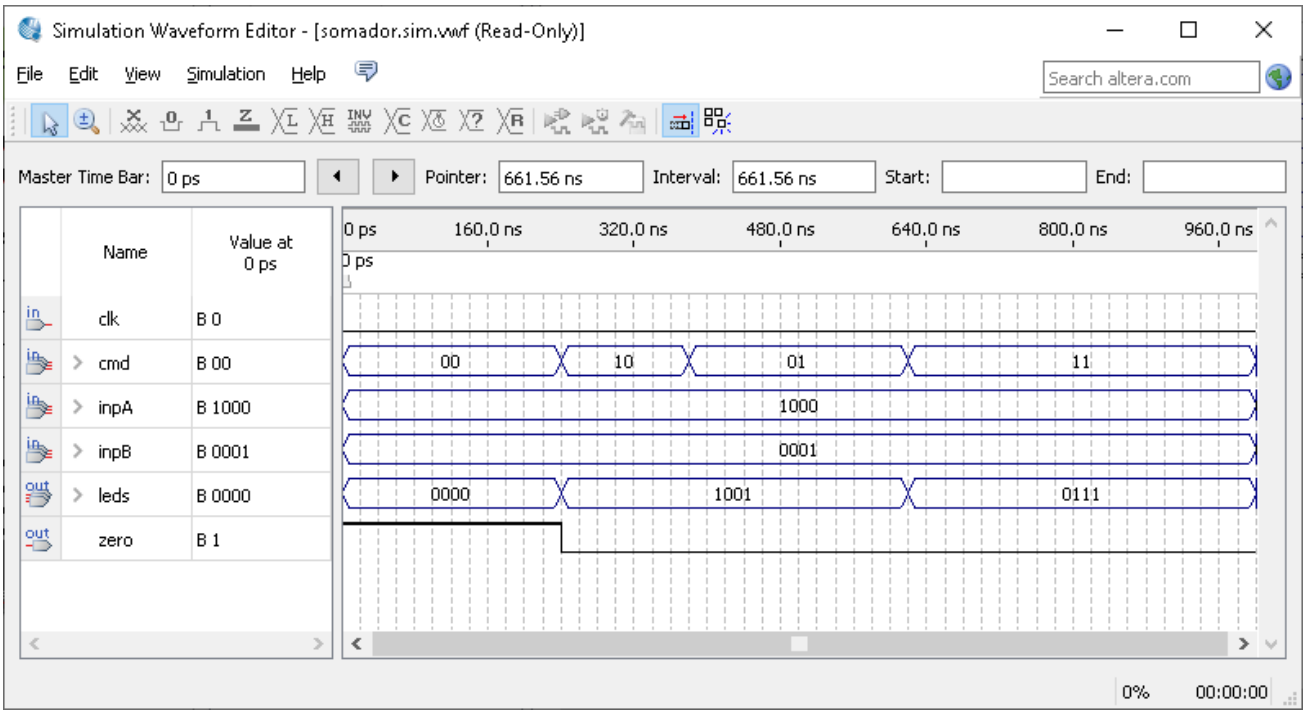
Para a operação AND lógica (cmd="00"), observou-se comportamento correto no intervalo inicial da simulação. Com inpA="1000" e inpB="0001", o resultado obtido foi "0000", demonstrando a execução adequada da operação bit-a-bit. O sinal zero foi corretamente ativado (zero='1'), confirmando a funcionalidade de detecção de resultado nulo. Esta operação é fundamental em aplicações de mascaramento de bits e operações booleanas (Stallings, 2015).

A operação OR lógica (cmd="01") apresentou funcionamento conforme esperado no segundo intervalo temporal da simulação. As mesmas entradas inpA="1000" e inpB="0001" resultaram em "1001", demonstrando a operação de OR bit-a-bit. O sinal zero permaneceu inativo (zero='0'), indicando resultado não-nulo. Esta operação é essencial para combinação de sinais de controle e operações de set de bits em sistemas digitais (Floyd, 2015).

Os resultados da adição aritmética (cmd="10") demonstraram correção funcional no terceiro segmento temporal. Com inpA="1000" (8 decimal) e inpB="0001" (1 decimal), o resultado foi "1001" (9 decimal), confirmando a implementação adequada da soma binária. A utilização do operador "+" da biblioteca IEEE.STD_LOGIC_UNSIGNED proporcionou funcionamento esperado da aritmética de 4 bits (Rushton, 2011). O sinal zero manteve-se inativo, coerente com o resultado não-nulo obtido.

A operação de subtração (cmd="11") foi validada no último intervalo da simulação. A subtração de inpA="1000" (8 decimal) menos inpB="0001" (1 decimal) resultou em "0111" (7 decimal), demonstrando implementação correta da aritmética de subtração em complemento de dois. O comportamento observado confirma que a ULA manuseia adequadamente operações aritméticas básicas, essenciais para funcionamento de processadores (Hennessy; Patterson, 2017). A Figura 1 demonstra o resultado obtido através do simulador.

Figura 1 - Tela do simulador



Fonte: O autor.

A estabilidade dos valores de entrada durante os intervalos de teste demonstra adequação do testbench utilizado. Os valores foram mantidos constantes por tempo suficiente para estabilização completa das saídas, permitindo medições precisas do comportamento temporal da ULA implementada. Além disso, a funcionalidade de detecção de zero demonstrou comportamento correto durante toda a simulação. O sinal foi ativado exclusivamente quando o resultado da operação foi "0000", proporcionando informação de status essencial para implementações de processadores que requerem flags de condição (Mano; Ciletti, 2017).

A comparação dos resultados obtidos com especificações teóricas confirma a correção funcional da implementação. Todos os casos testados produziram resultados matematicamente corretos, validando tanto a codificação VHDL quanto a síntese para o FPGA alvo.

CONCLUSÕES

Os resultados obtidos através de simulação comportamental confirmaram a correção funcional de todas as operações implementadas: AND lógico, OR lógico, adição e subtração aritméticas. A funcionalidade de detecção de resultado zero operou conforme especificado, proporcionando informação de status essencial para aplicações em arquitetura de processadores.

As principais contribuições deste trabalho incluem: demonstração prática de implementação de componente fundamental de processadores em FPGA; validação de metodologia de design digital utilizando VHDL; e análise de características funcionais e temporais da implementação resultante.

Como trabalhos futuros, sugere-se a expansão da ULA para 8 ou 16 bits e implementação de operações adicionais como deslocamento e rotação. A implementação de tratamento de overflow/underflow e otimização para operação puramente combinacional também constituem melhorias relevantes.

A experiência obtida neste trabalho demonstra o valor pedagógico de implementações práticas em FPGA para compreensão de conceitos fundamentais de arquitetura de computadores e sistemas digitais. A metodologia empregada pode ser replicada para desenvolvimento de outros componentes essenciais, contribuindo para formação sólida em engenharia.

REFERÊNCIAS

ALTERA CORPORATION. Cyclone IV Device Handbook. Volume 1. San Jose: Altera Corporation, 2016. 324 p. Disponível em: https://cdrdv2-public.intel.com/653974/cyclone4-handbook.pdf. Acesso em: 14 jul. 2025.

ALTERA CORPORATION. Quartus II Handbook Version 13.0. San Jose: Altera Corporation, 2013. 1156 p. Disponível em: https://www.intel.com/content/www/us/en/content-details/653795/quartus-ii-handbook-version-13-0.html. Acesso em: 14 jul. 2025.

ASHENDEN, P. J. The Designer's Guide to VHDL. 3rd ed. Burlington: Morgan Kaufmann, 2011. 936 p.

BERGERON, J. Writing Testbenches: Functional Verification of HDL Models. 2nd ed. Boston: Kluwer Academic Publishers, 2006. 939 p.

BROWN, S.; VRANESIC, Z. Fundamentals of Digital Logic with VHDL Design. 2rd ed. New York: McGraw-Hill, 2006. 939 p.

CHU, P. P. FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version. 1nd ed. Hoboken: John Wiley & Sons, 2007. 440 p.

FLOYD, T. L. Digital Fundamentals. 11th ed. Boston: Pearson, 2015. 912 p.

HARRIS, D. M.; HARRIS, S. L. Digital Design and Computer Architecture: ARM Edition. 1nd ed. Burlington: Morgan Kaufmann, 2015. 584 p.

HENNESSY, J. L.; PATTERSON, D. A. Computer Architecture: A Quantitative Approach. 6th ed. Burlington: Morgan Kaufmann, 2017. 936 p.

IEEE COMPUTER SOCIETY. IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-2008. New York: IEEE, 2009. 626 p.

INTEL CORPORATION. Intel Cyclone IV FPGA Family Overview. Santa Clara: Intel Corporation, 2016. 14 p. Disponível em:https://cdrdv2-public.intel.com/654630/cyiv-51001.pdf. Acesso em: 14 jul. 2025.

MANO, M. M.; CILETTI, M. D. Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog. 6th ed. Boston: Pearson, 2017. 720 p.

RUSHTON, A. VHDL for Logic Synthesis. 3rd ed. Chichester: John Wiley & Sons, 2011. 496 p.

STALLINGS, W. Computer Organization and Architecture: Designing for Performance. 10th ed. Boston: Pearson, 2015. 864 p.

WAKERLY, J. F. Digital Design: Principles and Practices. 5th ed. Boston: Pearson, 2018. 912 p.