V Simpósio de Iniciação Científica 2022
Ciência, Tecnologia e Inovação no Brasil

IMPLEMENTAÇÃO DE CIRCUITOS ASSÍNCRONOS NA PLATAFORMA FPGA

Mateus Eusébio de Lacerda[1] (IC), Diogo Leonardo Ferreira da Silva (PQ)[1]
[1]*Universidade Federal de Itajubá*

Palavras-chave: Eletrônica, Digital, NCL, Protocolo, Síntese.

Introdução

A arquitetura digital síncrona foi adotada por muitos anos como padrão na indústria de semicondutores, esse tipo de circuito é caracterizado por usar um único sinal para controlar a transação de dados entre seus blocos. Geralmente chamado de sinal de Relógio ou *clock*, ele é distribuído por todo o circuito de modo a sincronizar a transmissão dos dados, como pode ser observado na Figura 1a.

Já a arquitetura oposta, chamada de assíncrona não necessita de um único sinal de sincronia, e pode ser observada na Figura 1b. Nela, os blocos usam um protocolo de comunicação para trocar informações e coordenar suas ações entre si (MÔCHO, 2006).

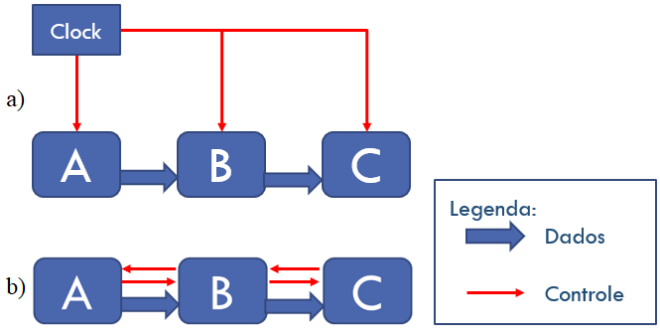

Figura 1 – Arquitetura. a)Síncrona. b)Assíncrona.

Porém, com circuitos cada vez maiores, hoje na casa de um trilhão de transistores, um sinal de *clock* global gera diferentes problemas, como a distribuição do sinal cada vez mais complexa, a agravação do atraso de *clock* para áreas distantes da fonte do sinal e alto consumo de energia devido à atualização desnecessária de áreas inativas do circuito.

Os circuitos assíncronos, por sua vez, não possuem esses problemas, já que sua transmissão de dados é feita de maneira descentralizada e, ainda tendem a ser mais econômicos (VAN BERKEL, 1994), menos suscetíveis a ruídos e tem uma construção modular (MULLER, 1963).

O objetivo desta pesquisa é investigar as bases teóricas que sustentam os circuitos assíncronos, implementar circuitos com várias funcionalidades e testar seu funcionamento. Essa pesquisa é justificada pelo fato de que hoje existe uma grande janela de oportunidade na indústria de semicondutores. Em primeiro lugar, pelo fato desses circuitos terem sido ignorados durante muitos anos, existe uma grande área a se explorar. Em segundo lugar, como surgem cada vez mais empecilhos na construção da arquitetura síncronas, há uma crescente busca no mercado por alternativas.

As ferramentas usadas foram: a linguagem de descrição de *hardware* Verilog HDL para descrever o circuito, o *software Quartus* para a compilação e visualização da lógica gerada e o *ModelSim* para simulação e *debug.*

Metodologia

A estrutura de um circuito assíncrono pode ser definida em três níveis de abstração, como pode ser observado na Figura 2. Para implementa-los, foi usada uma abordagem de baixo pra cima, onde, primeiro as células básicas são inicialmente descritas em detalhes, depois, essas células são associadas para formar um elemento maior chamado de estágio assíncrono, por último esses estágios são associados até completar o nível mais alto, o circuito completo.

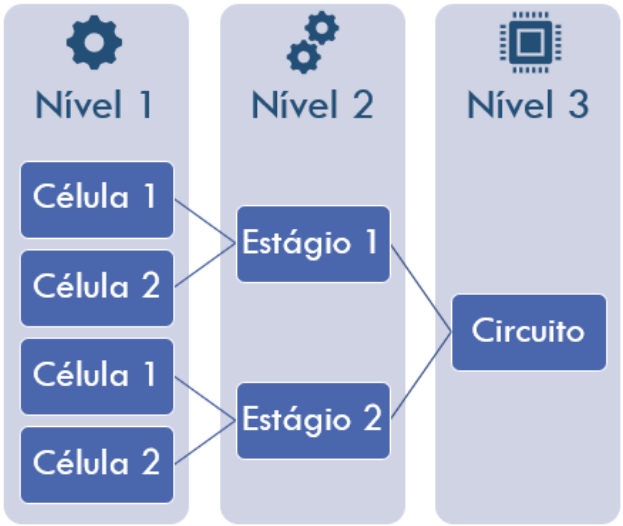

Figura 2 – Metodologia e Níveis de Abstração

A abordagem de baixo pra cima permite dar foco a testes iniciais, de forma que é possível testar cada um dos módulos individualmente antes que os sistemas mais complexos sejam construídos, em termos práticos, nessa pesquisa, cada uma das células básicas foram projetadas e testadas antes de serem conectadas aos estágios assíncronos, assim como cada um dos estágios foram projetados e testados antes de fomarem o circuito final. Isso facilita muito a achar erros de projeto antes da integração total do sistema.

Cada uma das células básicas possui uma atribuição específica, existem quatro delas (SPARSO, 2001):

- Célula M de N: Recebe dados de outros estágios assíncronos e realiza a lógica combinacional do circuito, isto é, o processamento de dados.
- Detector de Fim de Cálculo: É ligado na saída da lógica combinacional, e envia um sinal de controle no instante em que o processamento estiver completo.
- Célula Muller: Se conecta ao Detector de Fim de Cálculo e outros estágios assíncronos, é responsável por ativar o Registrador Assíncrono e realizar a lógica do protocolo de transmissão de dados.
- Registrador Assíncrono: Armazena os dados processados pelas células M de N. É controlado pela célula Muller e envia um barramento de dados a outros estágios assíncronos.

Na Figura 3 é possível observar a estrutura padrão de um Estágio Assíncrono, e como normalmente as células básicas são conectadas dentro do mesmo. Ele é definido como um bloco capaz de receber dados, realizar seu processamento, armazenar o resultado, e enviar esse resultado a outros blocos, tudo isso usando a lógica do protocolo de comunicação.

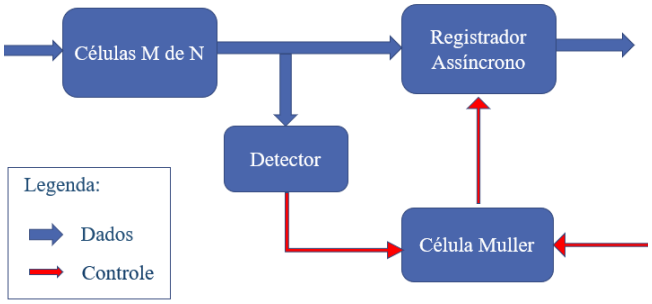

Figura 3 – Estágio Assíncrono.

As Células M de N são usadas na lógica *Null Convention Logic* (NCL) para realizar o processamento do circuito (FANT, 1996). Previamente só era conhecida a implementação dessas células em tecnologia CMOS, então, para implementar esse tipo de célula em FPGA foi usado o método de Huffman, descrito em DE OLIVEIRA (2018), SHOKIBA (2019) e NAKAHODO (2007). Essa implementação permite usar portas lógicas básicas como *AND, OR e NOT* para simular circuitos *NCL* e testá-los em FPGA, algo que a princípio só seriam implementados em tecnologia CMOS.

O método usado para construir cada um dos Estágios do processador foi:

1. Construir o circuito NCL usando o método de Huffman a partir da lógica desejada do estágio.
2. Construir o Detector de Fim de Cálculo. Ele deve ser baseado no número de *bits* do barramento de saída do circuito NCL, para cada bit é necessária uma porta lógica *OR* para detectar a validade dos dados
3. Construir o vetor de Registradores Assíncronos baseados no número de *bits* do barramento de saída do circuito NCL. Para cada *bit* é necessário um Registrador Assíncrono.
4. Contruir a Célula Muller. Uma de suas entradas é a saída do Detector de Fim de Cálculo, e as outras entradas são os sinais de *acknowledgement* dos Estágios posteriores, enquanto a sua saída é o sinal de Habilita dos Registradores Assíncrono e, ao mesmo tempo o sinal de *acknowledgement* que o estágio atual envia aos estágios anteriores.

Após dominar o projeto de variados estágios asssíncronos o projeto partiu para a aplicação dessas técnicas em uma prova de teste. E um processador pode ser considerado uma prova de teste muit boa, já que possui internamente os mais variados circuitos, desde circuitos combinacionais como a Unidade Lógica e Aritmética(ULA), contadores como o Contador de Programa, memórias como a RAM e a Cache e registradores gerais como o Registrador de Instrução.

Então antes de iniciar o projeto do Processador Assíncrono, buscou-se entender o funcionamento de um processador em geral, então, partindo de pesquisas, foi possível entender como geralmente funciona um processador e quais as instruções mais comuns. Por fim, foi implementado um processador síncrono de 8 bits com 10 instruções, que foi usado como base para a construção do circuito mais desafiador dessa pesquisa, o Processador Assíncrono de 8 bits.

Resultados e discussão

O processador construído possúi 6 estágios base.

- A Memória RAM é usada tanto para dados como para instruções. Ela armazena 16 posições de 8 bits cada.

- O Contador de Programa aponta o endereço de leitura da Memória RAM, cada vez que uma instrução é executada, o contador é incrementado para que possa apontar para a próxima instrução da RAM, é possível também forçar um endereço no contador de programa caso uma instrução de JUMP seja executada.
- O Registrador de Instruções recebe a instrução atual da RAM e pode enviá-la ao contador de programa caso seja uma instrução de JUMP ou ao controlador, caso contrário.
- O controlador é responsável por enviar sinais a Memória Cache quanto a execução da instrução atual.
- A memória Cache possui 4 posições de 8 *bits* e se conecta a memória RAM para executar as instruções de READ e LOAD, e se conecta a Unidade Lógica Aritmética(ULA) para executar as instruções de ADD, SUB, AND ou OR. A grande maioria das operações do processador são feitas usando os registradores da memória Cache.

A Figura 4 mostra a dinâmica desses blocos no processador, mas, é importante salientar que, devido à natureza do protocolo de comunicação, é impossível que um estágio envie dados e receba dados do mesmo estágio, como é o caso da conexão entre Memória Cache e a ULA, por exemplo, nessa conexão, foi necessário adicionar dois estágios assíncronos *buffer*, um para servir de intermediário quando a Cache enviar dados para a ULA e outro para quando a ULA enviar dados para a Cache. Por questões didáticas, isso não foi mostrado na Figura 4.

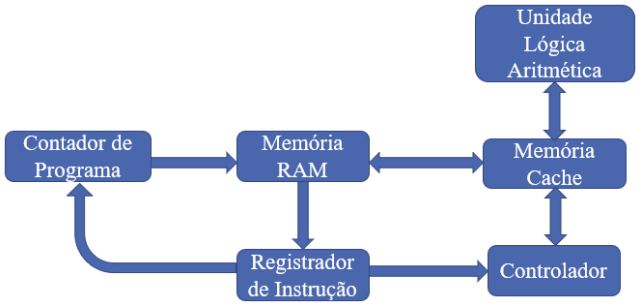

Figura 4 – Conexão dos Estágios do Processador

Por fim, 10 instruções foram implementadas, o processador parte do presuposto que todas a memória já foi carregada para a execução. O fluxo de operação pode ser o seguinte: a instrução LOAD é usada para carregar dados da RAM para a Cache, após isso, as instruções da ULA – ADD, SUB, AND, OR – podem fazer quaisquer operações com esses dados, para que seu resultado possa então ser carregado de volta na RAM. Existem também as instruções de Fluxo – NOP e HALT – que param a execução momentaneamente ou permanentemente, e as instruções de JUMP e JUMP_NEG, que permitem executar loops infinitios e loops condicionais.

É possível por exemplo executar uma multiplicação, usando um loop condicional com uma operação de soma. Na Tabela 1 é possível visualizar todas as instruções e usa sintaxe no processador.

Instrução	Opcode	Descrição
NOP	00 00 - - - -	Não faz nada por um ciclo
HALT	00 01 - - - -	Para a execução permanentemente
JUMP	00 10 xxxx	Pula o contador de programas para o endereço xxxx
JUMP_NEG	00 11 xxxx	Se a Flag Neg for 1, Pula o contador de programas para o endereço xxxx
LOAD	10 yy xxxx	Lê o endereço da RAM xxxx e carrega seus dados no endereço cache yy
READ	01 yy xxxx	Lê o endereço cache yy e carrega seus dados no endereço RAM xxxx
ADD	11 00 zzyy	Soma os registradores cache yy e zz, armazena o resultado em yy
SUB	11 01 zzyy	Subtrai o registrador cache yy do zz, armazena o resultado em yy
AND	11 10 zzyy	Realiza uma lógica E nos endereços yy e zz, armazena o resultado em yy
OR	11 11 zzyy	Realiza uma lógica OU nos endereços yy e zz, armazena o resultado em yy

Tabela 1 – Instruções e Sintaxe do Processador.

Conclusões

O processador é talvez a conquista mais chamativa dessa pesquisa, mas, para conseguir construí-lo várias dificuldades tiveram que ser superadas. A maioria dessas dificuldades vieram do fato de circuitos assíncronos terem sido pouco visados durante muitos anos, então existem poucas informações e poucos métodos de construção já criados, mas isso também pode ser visto como uma oportunidade, pois, a cada vez que um obstáculo era superado, uma nova solução era criada.

O primeiro desafio foi entender o funcionamento de cada uma das células básicas através da literatura existente, e também a dinâmica de interdependência delas dentro do Estágio Assíncrono. Paralelo a isso, as células M de N são um tópico muito difícil, já que, a princípio elas não são células básicas como as portas lógicas, e é preciso uma adaptação para simulá-las ou testá-las numa FPGA.

Também entender o trânsito de dados entre os estágios assíncronos foi um tópico complidado, a chave para entender esse tópico foi muito estudo a cerca do protocolo de comunicação e entender como interigar as Células Muller dos diferentes Estágios.

Como o processador foi escolhido como a prova de teste do método, foi necessário primeiro entender e projetar um processador síncrono de 8 bits. A parte síncrona foi escolhida primeiro porquê é mais fácil achar informações disponíveis, e, apesar de ser um tópico complexo, devido a ser um tópico mais bem estabelescido, foi um desafio mais fácil de superar.

Por último, a maior dificuldade foi sem dúvida conectar todos os estágios de modo a garantir o correto funcionamento do processador de 8 bits. Foi necessário inserir alguns estágios *buffer* entre certas trasmissões, haviam *muxes* e *demuxes* assíncronos para transferências em que os dados deviam ser divididos ou juntados, e terminar a conexão de todas as Células Muller de todos os estágios para que o protocolo funcionasse sem nenhuma perda de dados em nenhum ponto do circuito foi certamente uma dificuldade enorme, mas que também trouxe grande satisfação no momento de conclusão.

Olhando por uma perspectiva acadêmica, o processador não é o maior produto dessa pesquisa, e sim os métodos desenvolvidos para a superação dos obstáculos na construção do mesmo. Dessa forma os objetivos do projeto de pesquisa foram alcançados, já que foi documentada toda a a dinâmica entre os as células básicas dos circuitos assíncronos, foi referenciada e testada a meneira de projetar circuitos combinacionais NCL usando células M de N para simulação e prototipação em FPGA, e, também documentada e testada a maneira de conectar estágios assíncronos para uma correta sincronização dos dados usando o protocolo.

Assim, espera-se que o leitor dessa pesquisa tenha mais facilidade ao projetar quaisquer circuitos assíncronos que seja de sua vontade, contribuindo, assim, para a divulgação do conhecimento na área.

Agradecimento

Agradecemos ao programa PIBIC, a CNPq e a UNIFEI pela oportunidade de ajudar a divulgar conhecimento científico na área tecnológica no Brasil.

Referências

DE OLIVEIRA, Duarte Lopes; et al. **Synthesis of QDI Combinational Circuits Using Null Convention Logic Based on Basic Gates:** Advances in Science, Technology and Engineering Systems Journal, Vol 3, 2018.

FANT, K. M; BRANDT, S. A; **NULL Convention Logic:** A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. International Conference on Application Specific Systems, Architectures and Processors, 1996.

MÔCHO, Renato U. Reis. **Circuitos Assíncronos na Plataforma FPGA.** Programa de Pós-Graduação em Computação, 2006.

MULLER, David E. **Asynchronous logics and Application to Information Processing.** Stanford University press, 1963.

NAKAHODO, M; YAMADA, C; NAGATA. Y; **Threshold Gate with Hysteresis using neuron MOS.** Proceedings of the 7th WSEAS International Conference on Systems Theory and Scientific Computation. Athens – Greece, 2007.

SHOKIBA, C; et al. **A Novel Null Convention Logic (NCL) Gates Architecture Based on Basic Gates**. International Journal of Intellectual Advancements and Research in Engineering Computations., Vol 7, 2019, p. 1229-1237.

SPARSO, Jens; FURBER, Steve. **Principles of Asynchronous Circuit Design:** A Systems Perspective. Boston/Dordrecht/London: Kluwer Academic Publishers, 2001.

VAN BERKEL, Kees C. H. et al. **Asynchronous Circuits for low Power:** A DCC Error Corrector, Philips Research Laboratorie, Eindhoven: Eindhoven University of Technology, 1994.