VI Simpósio de Iniciação Científica 2023
Ciência como ferramenta de transformação da sociedade

IMPLEMENTAÇÃO DE CIRCUITOS ASSÍNCRONOS EM PLATAFORMAS FPGA: UM ESTUDO SOBRE A IMPLEMENTAÇÃO DE CÉLULAS DE MEMÓRIA

João Pedro Pereira Magalhães[1] (IC), Diogo Leonardo Ferreira da Silva (PQ)[1]
[1] *Universidade Federal de Itajubá*

Palavras-chave: Eletrônica Digital. *Quartus*. *Waveform*. *RTL Viewer*.

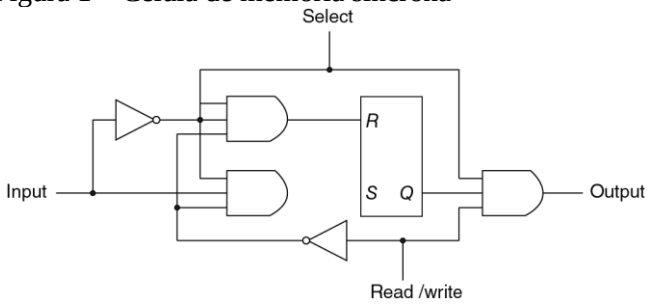
Introdução

No início da década de 60, Von Neumann apresentou a arquitetura de Von Neumann, a qual consistia na descrição arquitetural para a construção de um computador. Dentre os elementos pertencentes à arquitetura, Von Neumann descreveu o elemento responsável pelo armazenamento das instruções a serem executadas pelo computador, a memória (citar o artigo de Von Neumann). No início, as memórias eram construídas com válvulas, porém, com o advento da tecnologia semicondutora, as memórias, assim como os outros componentes de um computador, começaram a ser produzidos via transistores e apresentaram significativas melhorias de desempenho. Apesar dos constantes avanços nos elementos que compõem as memórias, a metodologia de implementação continua a ser um grande desafio, pois são necessários atender alguns requisitos: (1) as memórias precisam ocupar o menor espaço possível, (2) deve-se apresentar a relação custo por armazenamento como a menor possível e (3) precisam ser rápidas o suficiente para que não haja problemas de gargalo com o resto do sistema (Mano, 2016). Ademais, como visto na Lei de Moore, a cada dois anos, o número de transistores em circuitos digitais dobra, alcançando quantidades exorbitantes em um único chip. As memórias, assim como a maioria dos circuitos digitais clássicos, são construídas a partir da lógica síncrona, ou seja, existe um sinal global de *clock* responsável por sincronizar o funcionamento entre os elementos do circuito. Esse tipo de lógica vem sendo abordado pela indústria ao longo dos anos, porém, com a quantidade de transistores aumentando, problemas de sincronização, consumo e ruídos tornam-se cada vez mais presentes, demonstrando a necessidade de novos meios para construção de circuitos digitais. Tendo em vista as dificuldades no projeto de memórias, atrelado aos problemas com a lógica síncrona, o presente trabalho visa o estudo das células de memória, que são os componentes elementares da estrutura de uma memória e os responsáveis pelo armazenamento de informações a partir da lógica assíncrona. Por fim, será apresentada a construção de uma célula síncrona e outra assíncrona, demonstrando a partir da simulação o resultado de ambas e as possíveis vantagens e desvantagens no uso desse tipo de circuito para células de memória.

Metodologia

Para o desenvolvimento desta pesquisa, abordamos a metodologia baseada na construção dos circuitos e nas respectivas simulações, com intuito de analisar se o comportamento apresentado a partir dos estímulos nas entradas estava de acordo com o esperado. Em um primeiro momento, foi projetada a célula de memória síncrona, cujo circuito base está representado na figura 1.

Figura 1 – Célula de memória síncrona



Fonte: Mano, 2016

A figura 1 descreve uma célula de memória estática e volátil, ou seja, como o circuito não apresenta capacitores em sua construção, não ocorre perda de informação por correntes de fuga, assim, não é necessário um circuito de *refresh* para as células manterem os bits armazenados. Além disso, os bits são armazenados enquanto houver alimentação na célula (Tales, 2016).

Para chegar ao resultado da figura 1, será utilizado o *Quartus*, uma IDE capaz de compilar códigos desenvolvidos em linguagens de descrição de hardware. Na presente pesquisa, foi utilizada a linguagem *Verilog*, e abordaremos a construção do circuito de baixo para cima. Nesse tipo de abordagem, desenvolvem-se os elementos mais básicos primeiro para que, no final, esses sejam ligados a fim de chegar no circuito maior.

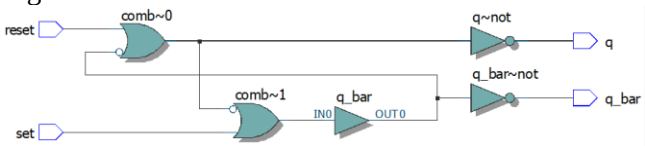
A figura 3 apresenta o resultado em *RTL Viewer* do

código apresentado na figura 2, que descreve em *netlist* um *latch* RS.

Figura 2 – Código *latch* RS síncrono

```
module RS_Latch(set, reset, q_bar, q);

    input set, reset;
    output q_bar, q;
    nor (q, q_bar, reset);
    nor (q_bar, set, q);
endmodule
```

Figura 3 – *RTL Viewer latch* RS síncrono

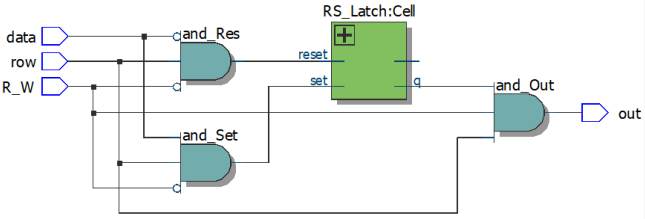


Com o *Latch* construído, a figura 4 representa o código da célula de memória baseada na figura 1, e na figura 5 temos o RTL *Viewer*.

Figura 4 – Código célula de memória síncrona

```
module Cell_RAM(data, row, R_W, out);


input data, row;
input R_W; //Read = 1, Write = 0

output out;

wire [6:0] conect;

    not Read_Write(conect[0], R_W);
    not Data(conect[1], data);
    and and_Res(conect[2], conect[1], row, conect[0]);
    and and_Set(conect[3], conect[0], row, data);
    
    RS_Latch Cell(conect[3], conect[2], conect[5], conect[6]);
    
    and and_Out(out, R_W, conect[6], row);

endmodule
```

Figura 5 – *RTL Viewer* célula de memória síncrona



Com o resultado apresentado na figura 5, nota-se que o circuito apresenta similaridade com o apresentado na figura 1.

Para a implementação de circuitos assíncronos, é possível utilizar dois tipos de células básicas: a célula Muller ou M de N. No caso desta pesquisa, foi utilizado o das células M de N, implementada com a lógica *Huffman* ilustrada por Oliveira et al., 2018.

Para o desenvolvimento dos códigos, será utilizado o algoritmo proposto por Mateus, 2022, o qual a partir da expressão booleana é capaz de desenvolver o circuito assíncrono NCL.

No primeiro momento, serão desenvolvidas as células básicas AND e NOT. As figuras 6 e 7 apresentam o código da célula AND e NOT, respectivamente.

Figura 6 – Código AND de três portas em *dual-rail*

```
module AND_Assinc_3(
    input A_t, A_f, B_t, B_f, C_t, C_f,
    output Out_t, Out_f);

    assign Out_f= (C_f) | (B_f) | (A_f);
    assign Out_t= (A_t & B_t & C_t) ;

endmodule
```

Figura 7 – Código NOT assíncrona

```
module Not_Assinc(
    input A_t, A_f,
    output Out_t, Out_f);

    wire hysteresis;
    assign hysteresis = A_t | A_f;

    assign Out_f= (A_t) | (hysteresis & Out_f);
    assign Out_t= (A_f) | (hysteresis & Out_t);

endmodule
```

Apesar de as células M de N apresentarem o circuito de histerese como forma de manter o resultado anterior quando a soma total dos níveis lógicos iguais a '1' nas entradas for menor do que N, neste caso não será utilizado, pois o circuito de histerese, tendo em vista o funcionamento de uma memória, possibilitará informações errôneas na saída. Assim, tanto a porta lógica AND quanto o *Latch*, nesse caso específico, serão construídos apenas em *dual-rail*, (ou seja, os valores '1' e '0' antes representados por um sinal, agora são representados por dois. '1' equivale a '10' e '0' a '01'), como apresentado nas figuras 6 e 8.

Figura 8 – Código *latch* RS em *dual-rail*

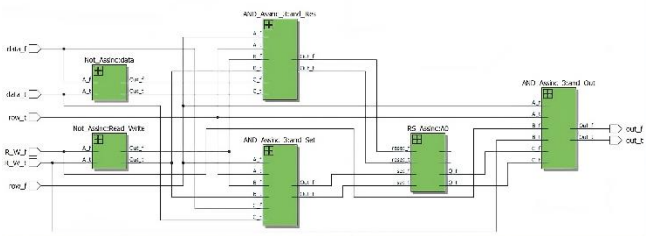
```
module RS_Assinc(
    input reset_t, reset_f, set_t, set_f,
    output Q_t, Q_f, Q_bar_t, Q_bar_f);

    assign Q_f= (Q_bar_t) | (reset_t) ;
    assign Q_t= (reset_f & Q_bar_f) ;
    assign Q_bar_f= (set_t) | (Q_t) ;
    assign Q_bar_t= (set_f & Q_f);

endmodule
```

Por fim, temos o código da célula de memória assíncrona e o RTL *Viewer* apresentados nas figuras 9 e 10, respectivamente

Figura 9 – Código da célula de memória assíncrona

```
module Cell_RAM_Assinc (data_t, data_f, row_t, row_f, R_W_t, R_W_f, out_t, out_f);

input data_t, data_f, row_t, row_f, R_W_t, R_W_f;
output out_t, out_f;

wire [5:0] conect_t;
wire [5:0] conect_f;

Not_Assinc Read_Write(R_W_t, R_W_f, conect_t[0], conect_f[0]);
Not_Assinc data(data_t, data_f, conect_t[1], conect_f[1]);

AND_Assinc_3 and_Res(row_t, row_f, conect_t[0], conect_f[0], conect_t[1], conect_f[1], conect_t[2], conect_f[2]);
AND_Assinc_3 and_Set(row_t, row_f, conect_t[0], conect_f[0], data_t, data_f, conect_t[3], conect_f[3]);

RS_Assinc A0(conect_t[2], conect_f[2], conect_t[3], conect_f[3], conect_t[4], conect_f[4]);

AND_Assinc_3 and_Out(row_t, row_f, R_W_t, R_W_f, conect_t[4], conect_f[4], out_t, out_f);

endmodule
```

Figura 10 – Visão em *RTL Viewer* da célula de memória assíncrona



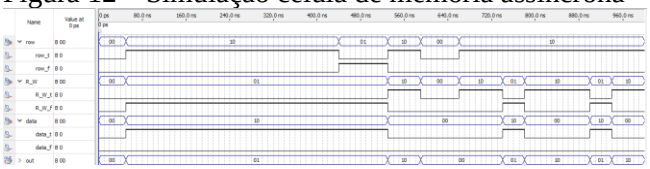
Resultados e discussão

A partir dos códigos desenvolvidos na seção anterior, serão simulados os circuitos a partir do *Waveform*, um simulador de ondas disponível no próprio *Quartus*. A figura 11 apresenta o resultado simulado da célula de memória síncrona, que quando apresenta nível lógico alto em ROW e nível lógico baixo em R/W, possibilita a escrita no *Latch* RS, que manterá os dados armazenados na memória até que sejam lidos (*row* nível lógico alto e R/W nível lógico alto), sobrescritos ou haja uma interrupção na alimentação do circuito.

Figura 11 – Simulação da célula de memória síncrona

Em seguida, temos a figura 12, que representa a resposta da simulação do código apresentado na figura 11. Nota-se que o resultado apresentado, mesmo que em *dual-rail*, está condizente com o da memória síncrona, demonstrando que ambos os circuitos funcionaram da maneira desejada.

Figura 12 – Simulação célula de memória assíncrona



Conclusões

A partir dos resultados apresentados na seção anterior, conclui-se que o desenvolvimento de circuitos assíncronos podem ser boas alternativas aos circuitos síncronos em determinadas situações. Afinal, como foi visto, os circuitos assíncronos apresentam um uso maior de portas lógicas e sinais de entrada e saída, dessa forma, aumentando a área do circuito, que, vai contra um dos requisitos expostos na introdução para o desenvolvimento de memórias. Porém, o menor consumo de energia, altas velocidades de operação e menores emissões de ruídos são vantagens interessantes para determinados sistemas que necessitam de memórias cada vez mais rápidas e confiáveis para execuções de tarefas críticas.

Agradecimentos

Gostaria de agradecer à Universidade Federal de Itajubá pela oportunidade e suporte durante toda a pesquisa, ao meu Orientador por acreditar no meu potencial e estar sempre me auxiliando e a CNPq pelo suporte financeiro ao longo de todo o processo.

Referências

KOWALTOWSKI, T.. Von Neumann: suas contribuições à Computação. Estudos Avançados, v. 10, n. 26, p. 237–260, jan. 1996.

LACERDA, M. E. de. Implementação de circuitos assíncronos em plataformas fpga. Itabira, Minas Gerais, 2022.

LACERDA, M. E. de. Método Automatizável para a Construção de Circuitos Assíncronos NCL. Monografia (Graduação) — Universidade Federal de Itajubá, Itabira, Minas Gerais, 2022.

MANO, M. M. Digital Logic And Computer Design. 1. ed. Estados Unidos da América.: Pearson, 2016.

OLIVEIRA, D. L. de; AL. et. Synthesis of qdi combinational circuits using null convention logic based on basic gates. Advances in Science, Technology and Engineering Systems Journal, v. 3, n. 4, p. 308–317, 2018. Disponível em: <https://www.astesj.com/publications/ASTESJ_030431.pdf>. Acesso em: 3.9.2023.

PIMENTA, T. C. Circuitos Digitais: Análise e Síntese Lógica - Aplicações em FPGA. 1. ed. Itajubá: Elsevier, 2016.