UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT
CURSO DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

IAN SANTOS NASCIMENTO
LUCAS ANDERSON LADISLAU AGUIAR

LABORATÓRIO DE CIRCUITOS

BOA VISTA, RR
2022

IAN SANTOS NASCIMENTO
LUCAS ANDERSON LADISLAU AGUIAR

LABORATÓRIO DE CIRCUITOS

Trabalho apresentado, como requisito de obtenção de nota parcial da disciplina de Arquitetura e Organização de ComputadoresI - DCC 301

Orientador (a): Dr. Prof. Herbet Rocha

BOA VISTA, RR
2022

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

Este relatório tem por objetivo compreender melhor as unidades funcionais e o desenvolvimento de alguns componentes que fazem parte da estrutura de um processador, os componentes foram desenvolvidos no Logsim e no Intel Quartus(simulador de processador). Os Testes aqui encontrados foram feitos via logsim, com o teste em casos práticos, e WaveForm.

2. COMPONENTES

2.1. REGISTRADORES FLIP FLOP D E JK

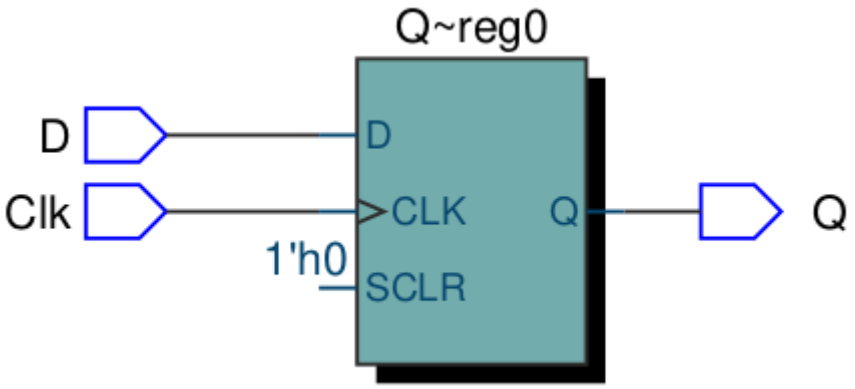
Registradores flip-flops(FF's) são basicamente componentes para armazenar 1 bits, funcionando como uma espécie de "memória" em circuitos sequenciais, pode se dizer também que os FF's lembram o estado anterior para poder calcular o próximo estado do circuito, possuem também um clock para fazer o seu controle. Um dos primeiros é o set-reset(FF's RS) que é ativado quando o set = 1 e o reset = 0 e desativado quando set=0 e reset =0, tendo como sua saida o Q, porém ele não faz o tratamento quando estas entradas são iguais, set = reset= 1.

O registrador JK surge para solucionar esse problema, passando a interpretar essas entradas iguais como inversão de sinal do FF e mantendo as interpretações anteriores de set e reset.

Temos também o FF D, ou comumente conhecido como FF de dado, pois este não muda o estado anterior, forçando basicamente o FF funcionar como uma memória de bit que é definido pela entrada D, possui uma entrada e o clock, o valor da saida sempre recebe o valor de D na mudança do clock.

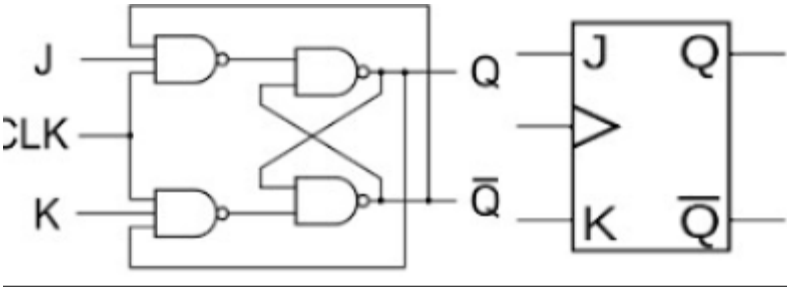
2.1.1. Descrição pinos e lógica

Figura 1 - Flip Flop D



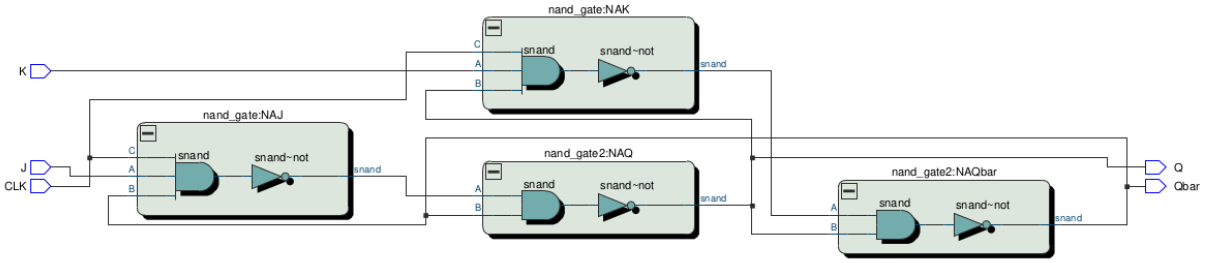
Esse é circuito que representa o FF D, possui 2 pinos de entrada, sendo um dos dados(D) e o clock, temos também uma saída (Q) que receberá o sinal de D a cada pulso do clock

Figura 2 - Circuito Equivalente JK



Para a implementação do FF JK utilizamos um circuito equivalente para a implementação desse componente, o circuito equivalente é o visto acima. Logo abaixo temos esse circuito equivalente gerado pelo rtl_viewer para o FF JK.

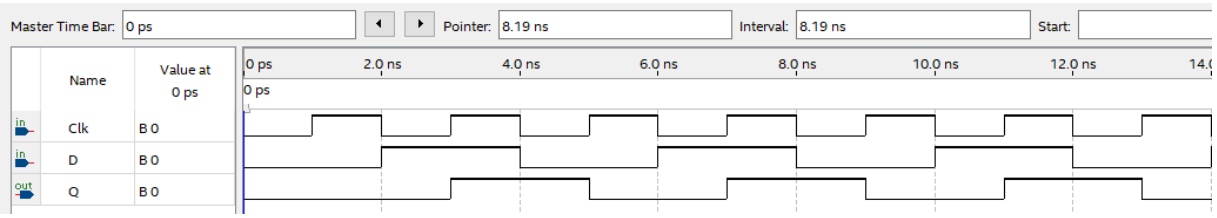
Figura 3 - Flip Flop JK



Como podemos ver na figura acima o FF JK faz uso de 4 portões nand e possui 3 pinos de entrada(Clock, Entrada de dados 1 - J, Entrada de dados 2 - K) e 2 saída(uma saída resultante do circuito - Q e a saída negada - Qbar)

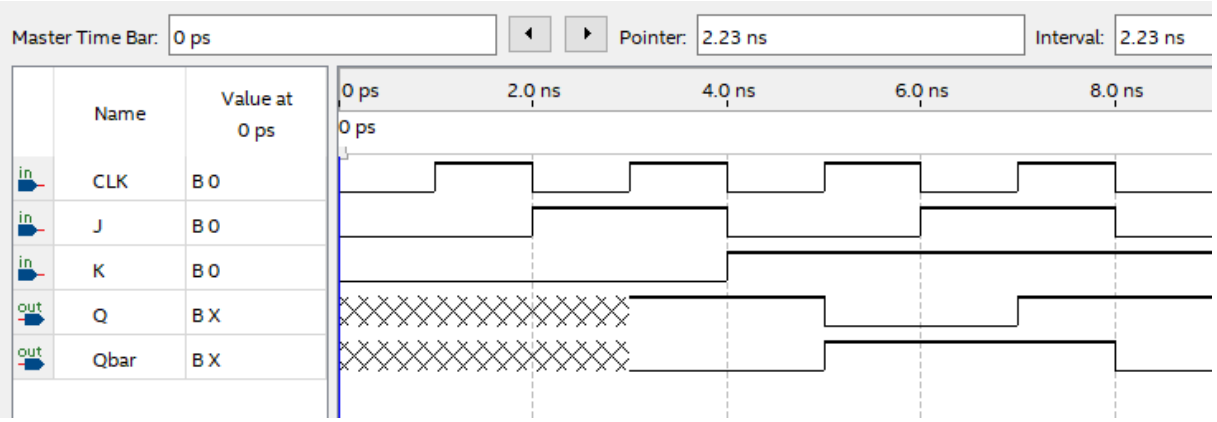
2.1.2. Testes do componente

Figura 4 - WaveForm Flip Flop D



Como dito antes o FF D funciona mais como um memória de 1 bit, como podemos observar a cada subida de clock, assumindo o valor 1, a saida(Q) receberá o valor de da entrada(D) até a próxima borda alta do clock.

Figura 5 - Wave Form Flip Flop JK



Temos então o waveform do Flip Flop JK quando ambas as entradas fora o clock são 0 então o valor de Q e Qbar não são alterados, já quando temos J = 1 e K=0, o valor que se espera na borda alta do clock para Q é Q=1 e já que Qbar é a negada de Q então Qbar =0. Na situação de J=0 e K=1, o valor esperado das saídas é Q=0 e Qbar = 1 e por no caso final em que J=K=1 a saída esperada é que Q = Qbar, o que acontece na waveform, logo o circuito foi executado corretamente.

2.2. MULTIPLEXADOR 4 OPÇÕES DE ENTRADA

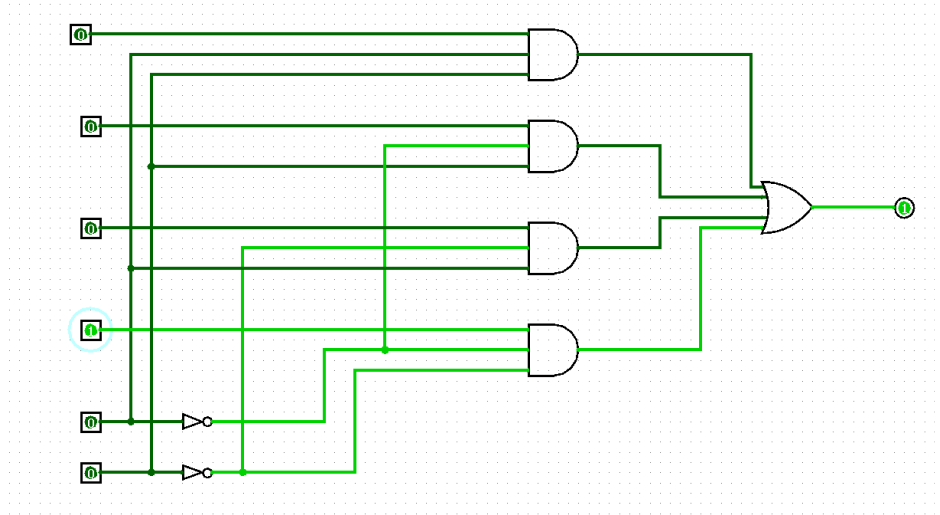
Um multiplexador é basicamente um switch em que existem entradas de valores que serão escolhidas e entradas que irão definir qual valor será escolhido

2.2.1. Descrição pinos e lógica

São 4 entradas em que cada uma entra em uma porta AND, juntamente com duas outras entradas que é a posição escolhida (selecionada). E o resultado das portas AND entrarão numa porta OR para que seja qual for escolhido o resultado, uma luz de LED acender resultando na escolha de uma das opções

2.2.2. Testes do componente

Figura 6 - Multiplexador 4 entradas



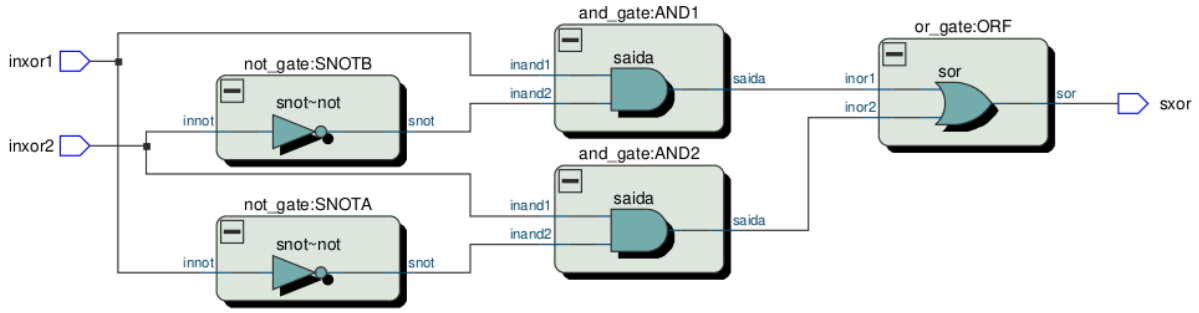
Teste mostrando a escolha default do circuito. Em que apenas o último valor é selecionado(ativado), as entradas de decisão não são acionadas.

2.3. PORTA LÓGICA DO XOR

O XOR é uma porta lógica dos circuitos digitais, representando o ou exclusivo, ou seja, sua saida será 1 quando os valores lógicos das entradas forem 1. Na questão é solicitado um portmap ou seja teremos de ver o circuito integrado(CI) equivalente que represente o XOR.

2.3.1. Descrição pinos e lógica

Figura 7- XOR com porMap

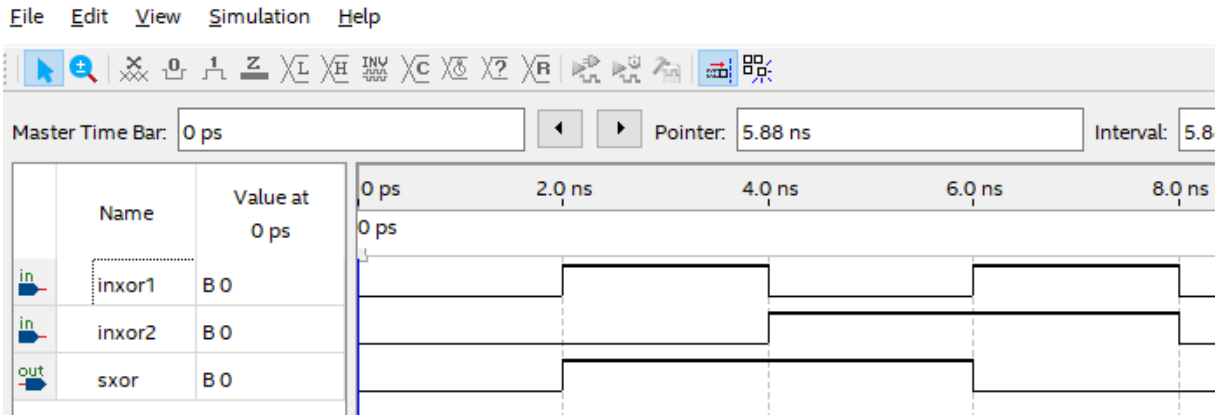


Como podemos ver o modelo gerado no RTL view do circuito xor equivalente seria esse, 2 entradas, portões lógicos not(2),and(2) e or(1). A lógica do circuito nos portões and seria uma das entradas com a negada da outra entrada que não foi

escolhida, o resultado vai das portas and vão para a porta or e logo temos a nossa saída da do circuito.

2.3.2. Testes do componente

Figura 8 - WaveForm XOR com porMap



A porta lógica xor pode ser entendida como um ou exclusivo, ou seja, só teremos 1(borda alta) como saída nos casos em que as entradas são diferentes (inxor1 =1 e inxor2=0; inxor1 =0 e inxor2=1) em caso contrários a esses a saída deve ser 0(borda baixa), demonstrado pela waveform gerada pelo circuito do xor.

2.4. SOMADOR 8 BITS + 4

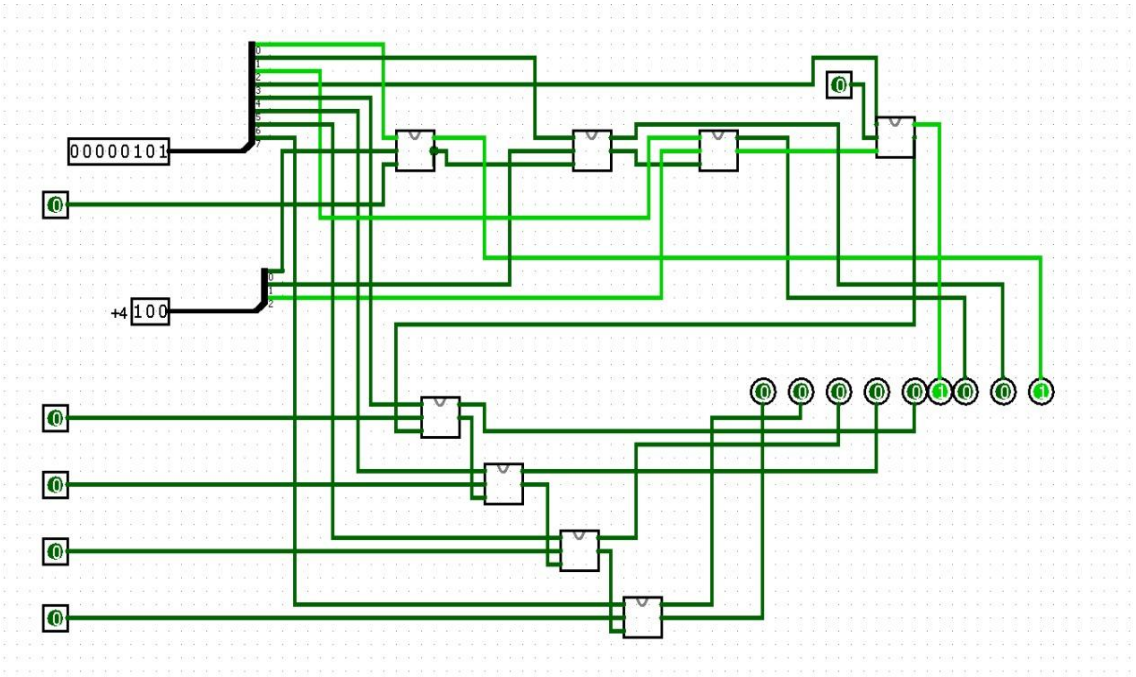
Somador de um valor de 8 bits com o valor 4(3 bits). A lógica foi usada para separar todos os bits dos valores e somar individualmente cada posição, começando pelo bit menos significativo, e possuindo um carry-out para o próximo somador e uma saída para representar o valor que “fica”. As portas isoladas na parte inferior são apenas para que existam entradas, podendo também ser substituídas por o valor 4 com 8 bits.

2.4.1. Descrição pinos e lógica

São uma entrada de 8 bits, uma entrada de 3 bits(valor 4) e 4 entradas “lixo” apenas para conexão. Os bits são separados para 1 bit cada e somados começando pelo bit menos significativo, até o mais significativo. Mostrando o resultado com os pinos no final.

2.4.2. Testes do componente

Figura 9- Somador com + 4



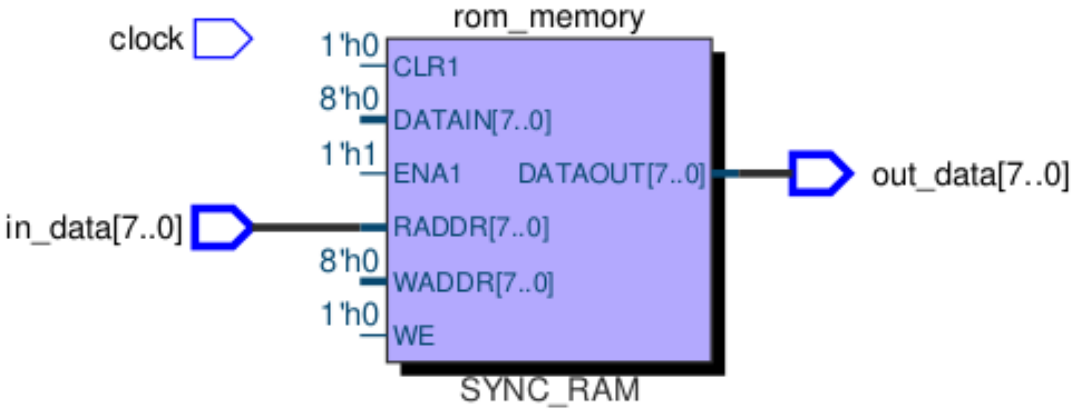
O teste é feito somando 00000101 com 100 dando assim a saída mostrada 000001001

2.5. MEMÓRIA ROM

A memória rom é basicamente uma memória de leitura de dados que estão armazenados, na situação de componente do CI é nela que vão estar as instruções do processador, conhecida como memória instrução do datapath de um MIPS.

2.5.1. Descrição pinos e lógica

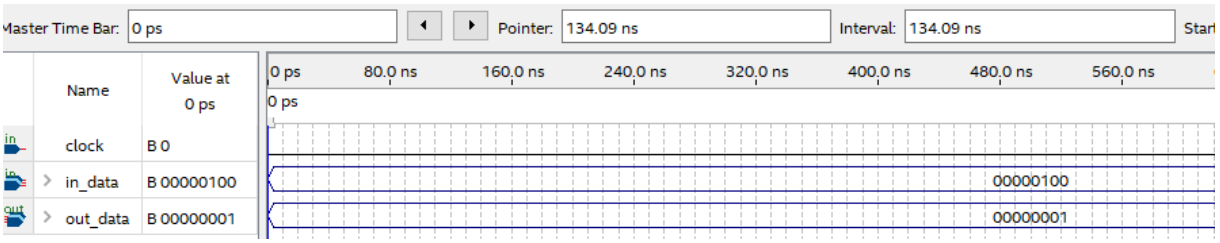
Figura 10- Memória ROM



Na memória rom temos como pinos entrada de dados(o valor do endereço na memória) com um vetor de 8 bits, junto com o clock e a saída da instrução, também um vetor de 8 bits, que será executada.

2.5.2. Testes do componente

Figura 11 - WaveForm Memória ROM



Na rom uma determinada posição na memória é dada como entrada e a saída esperada é a instrução armazenada naquela posição, como podemos ver na waveform estamos buscando a instrução armazenada na posição $4_{(dec)}$ = $00000100_{(bin)}$, como no código de teste a instrução armazenada nessa posição corresponde ao código binário de 00000001 a saída deve ser idêntica e é justamente o valor que obtemos

2.6. MEMÓRIA RAM

Um espaço disponível para armazenamento de valores temporários, facilitando e agilizando o acesso para o processador, conhecida nos processadores como memória de dados.

2.6.1. Descrição pinos e lógica

São 8 bits de entrada, 2 bits de entrada de endereçamento, 1 bit de entrada destinado a limpar a memória que foi preenchida do endereçamento, um clock e uma “saída” de 8 bits para mostrar o valor no endereçamento escolhido

Figura 12 - Memória RAM - celula 1x2

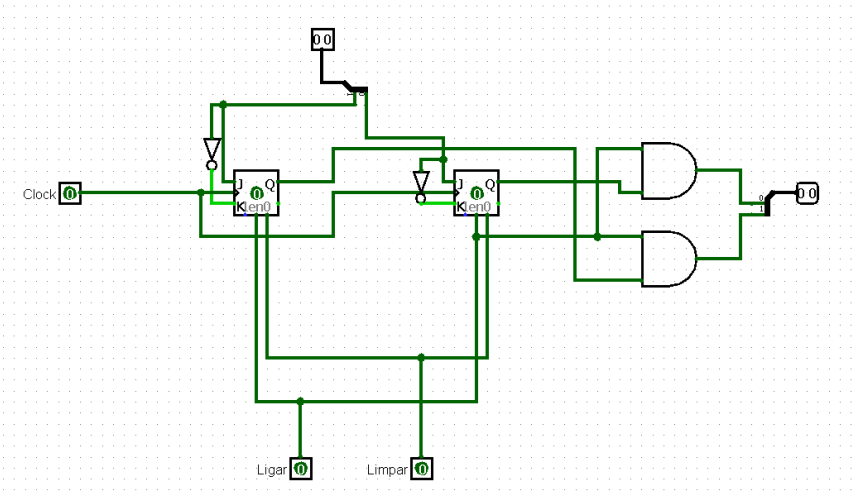


Imagem da célula que recebe 2 bits de entrada, onde cada um entra num Flip Flop JK com uma entrada normal e outra negada, depois a saída de cada um irá para uma porta AND diferente, onde a outra entrada é o bit “Ligado”.

Figura 13 - Memória RAM - celula 1x8

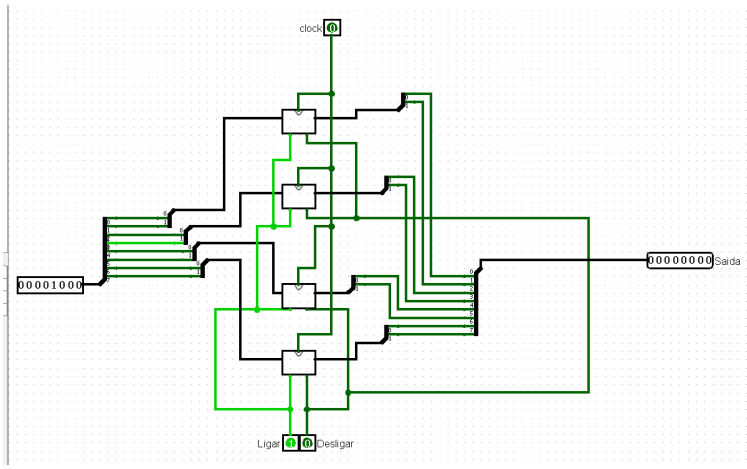


Imagem da célula de 8 bits com 4 células de 2 bits, onde os 8 bits de entrada são separadas em entradas de 2 bits em cada célula.

Figura 14 - Memória RAM - main

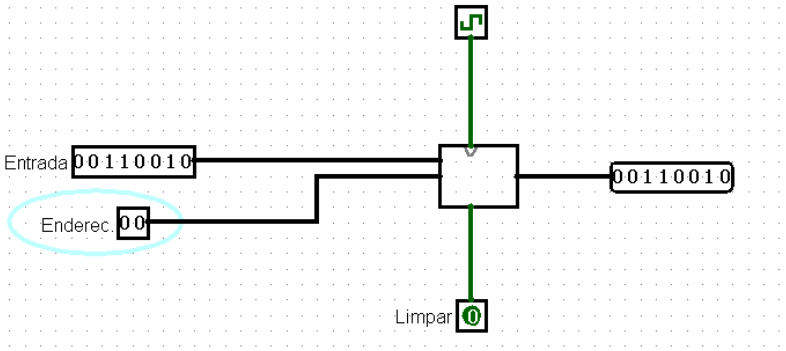


Imagem de uma entrada de 8 bits, com 2 bit de endereço, uma entrada de clock, uma saída de 8 bit e 1 bit para limpar

2.6.2. Testes do componente

Os testes foram o armazenamento de dados de 8bits nos 4 endereços de armazenamento disponíveis. Fazendo da seguinte forma: Definindo um valor de entrada, definindo para qual espaço de armazenamento irá e ativando o clock. Para verificar se foi armazenado o valor deve-se desligar o clock e definir o espaço da memória para qual o valor foi designado, mostrando assim o valor na saída.

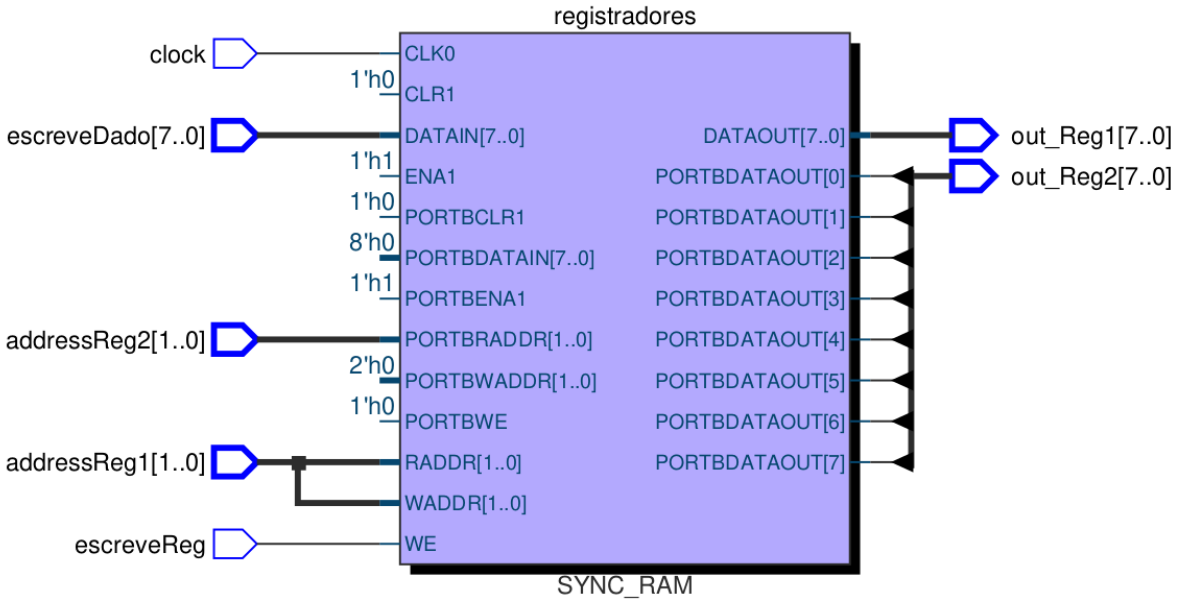
2.7. BANCO DE REGISTRADORES

Banco de registradores é um componente que como o próprio nome sugere é formado por um conjunto de registradores que são acessados de forma organizada, os quais podem ser lidos e escritos nessa unidade.

2.7.1. Descrição pinos e lógica

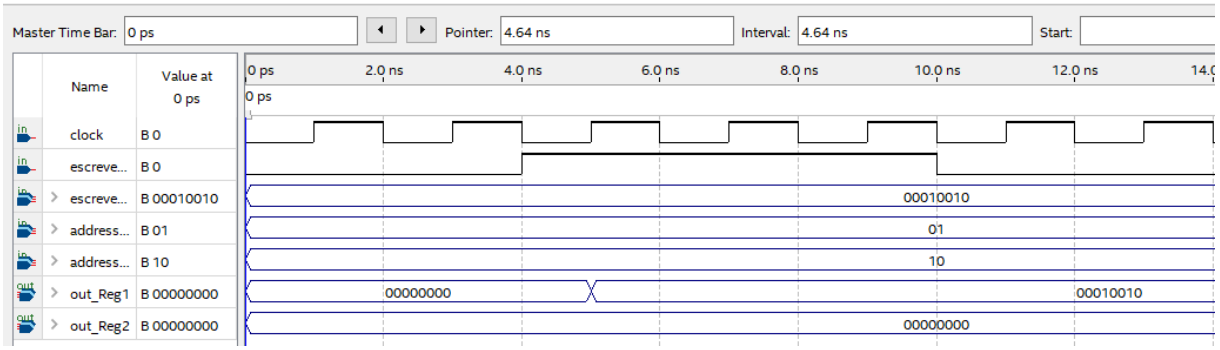
No banco de registradores teremos 5 pinos entradas(1 para o clock, 1 para o dado que esta sendo inserido - escreveDado, 1 para saber se o dado deve ser escrirto - escreveReg e 2 para o endereço do registrador) e 2 pinos de saída 1 que irá diretamente para a ULA e outro que será tratado posteriormente com um multiplexador para selecionar se ele será escolhido ou não.

Figura 15- Banco de Registradores



2.7.2. Testes do componente

Figura 16- WaveForm Banco de Registradores



Para a situação de teste do banco de registradores vamos simular a escrita de dados, podemos observar que enquanto a entrada escreveReg está desativada a saída do registrador se mantém inalterada, já que entendesse que seria uma leitura de registrador, quando ela assume o valor 1(borda alta o dado) a saída(out_reg1) automaticamente receberá o valor de entrada do escreveDado, que seria o dado a ser escrito no registrador(seria o passo de escrita de volta - WB)

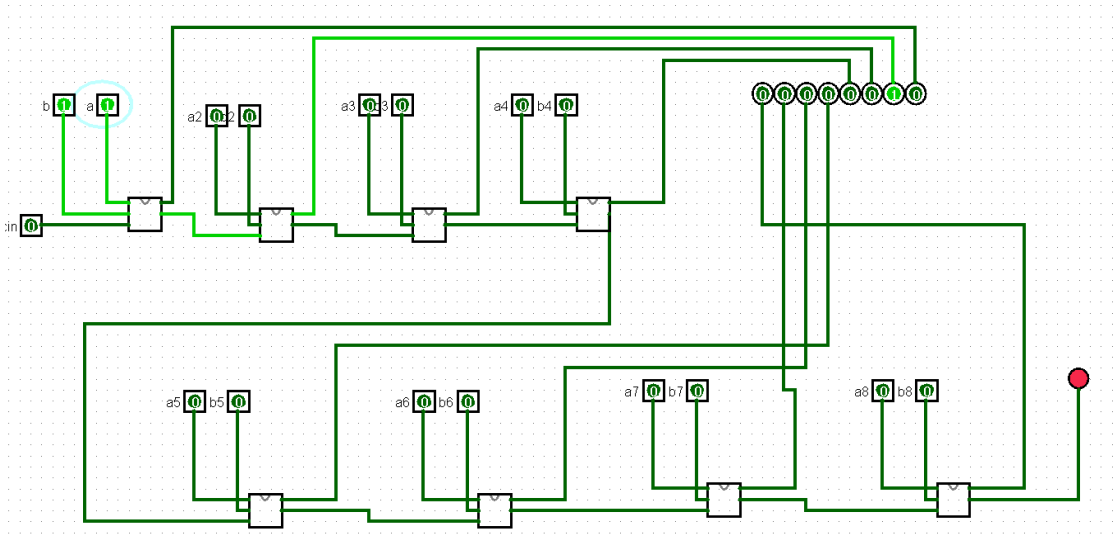
2.8. SOMADOR 8 BITS

Somador feito com os valores de 8 bits de cada, em que a representa o primeiro 8 bit e b representa o segundo bit. Sendo assim, é pego o primeiro bit de

cada um, somando as duas e tendo duas saídas em que uma é o resultado e o outro é o carry-in “mais um” que irá para o próximo somador. Poderia ser feito também duas entradas de 8 bits e um distribuidor com 8 bits isolados como saída, fazendo isso para as duas sequência de bits e somando individualmente

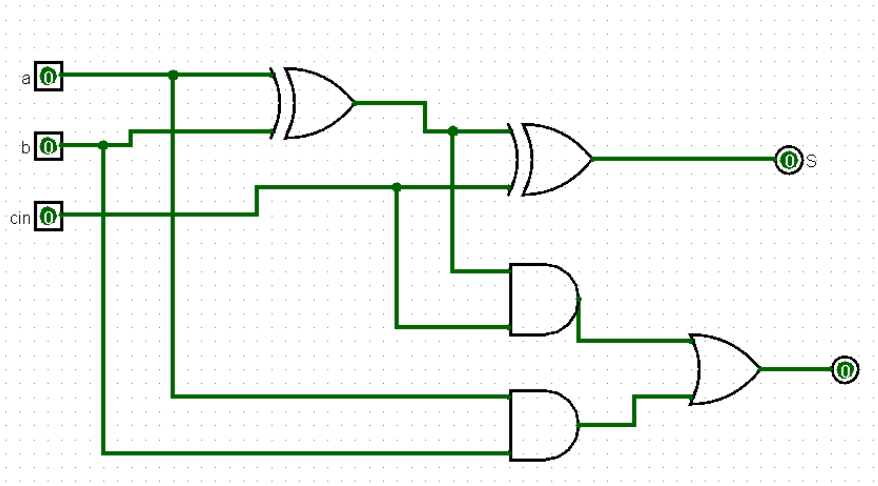
2.8.1. Descrição pinos e lógica

Figura 17- Somador 8 bits



Os pinos “a” representam a sequência de bits de A, e de “b” a sequência de bits de B. Somando os valores individuais e no final, mostrando o resultado é um overflow caso exista

Figura 18- Somador 8 bits(subcomponente)



Esta é a imagem e lógica do somador de 1 bit, usado como célula na Figura 17 e figura

2.8.2. Testes do componente

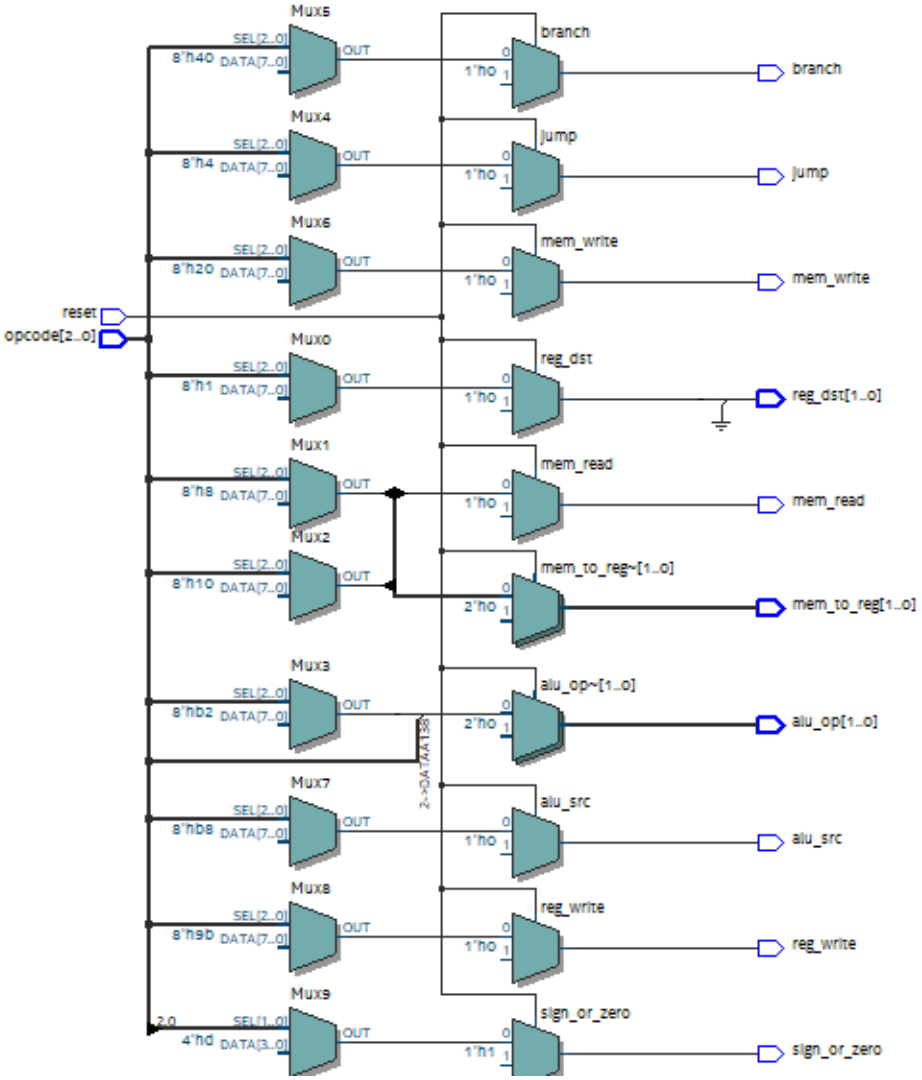
Na figura 15 pode-se ver a soma do valor 00000001 com 00000001, resultando numa saída de valor 00000010

2.9. UNIDADE DE CONTROLE 16 BITS

A unidade de controle é o componente encarregado de alimentar o circuito com as flags das operações que serão executadas de acordo com o opcode inserido nela, controlando assim os sinais da CPU para o seu correto funcionamento

2.9.1. Descrição pinos e lógica

Figura 19 - Unidade de Controle 16 bits

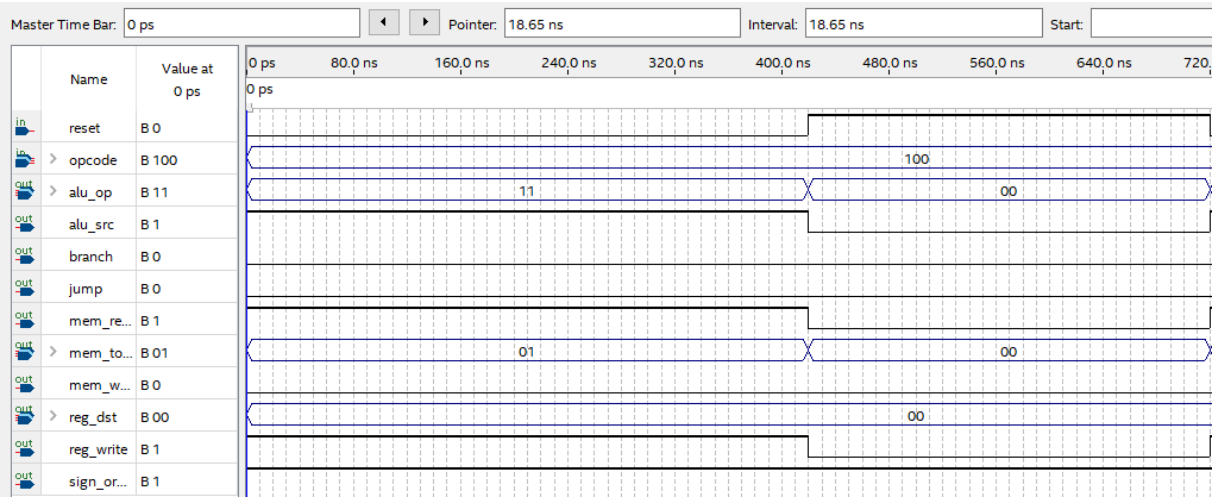


Para a unidade de controle de 16 bits nós temos um circuito com uma entrada com opcode de um vetor de 3 bits e um reset(que reiniciará os valores caso seja necessário), como saída temos as flags correspondentes aos componentes do

datapath que serão definidas de acordo com o opcode, para isso são utilizados diversos multiplexadores, já que estamos definindo uma saída fixa de acordo com as ocorrências do opcode. O reset está interligado diretamente com todas as saídas, pois caso seu valor seja 1, todos os valores devem ser zerados

2.9.2. Testes do componente

Figura 20- WaveForm Unidade de Controle 16 bits



Como podemos ver no teste o opcode de entrada foi 100, que é uma instrução de load(lw), ou seja, as trilhas de saída mem_read, alu_src, reg_write, sign_or_zero devem estar com borda alta e devemos ter uma saída de mem_to_reg(memória para registrador) = 01 e alu_op=11, o que evidentemente é a saída que ocorre com o nosso teste.

2.10. ULA 8 BITS

A ULA é um componente fundamental para a lógica de processamento, onde sua função é definir qual processo será realizado e as operações executadas.

2.10.1. Descrição pinos e lógica

São duas entradas de 8 bits, uma entrada de 4 bits que representa a operação que será ativada. E uma saída que representa o resultado da operação(Foto 1)

O Seletor usado na imagem 1, é definida por 2 bits de entradas de valores, 2 bits para SHIFT(esquerda e direita), entrada de 1 bit de carry-in ,4 bit de entrada para seleção, 1 bit de carry-out de saída e 1 bit de S(resultado) de saída. A operação de NOT do seletor está sendo escolhida por um multiplexador com a constante 1, sendo então sempre escolhida a opção de bit B

Figura 21- ULA 8 bits

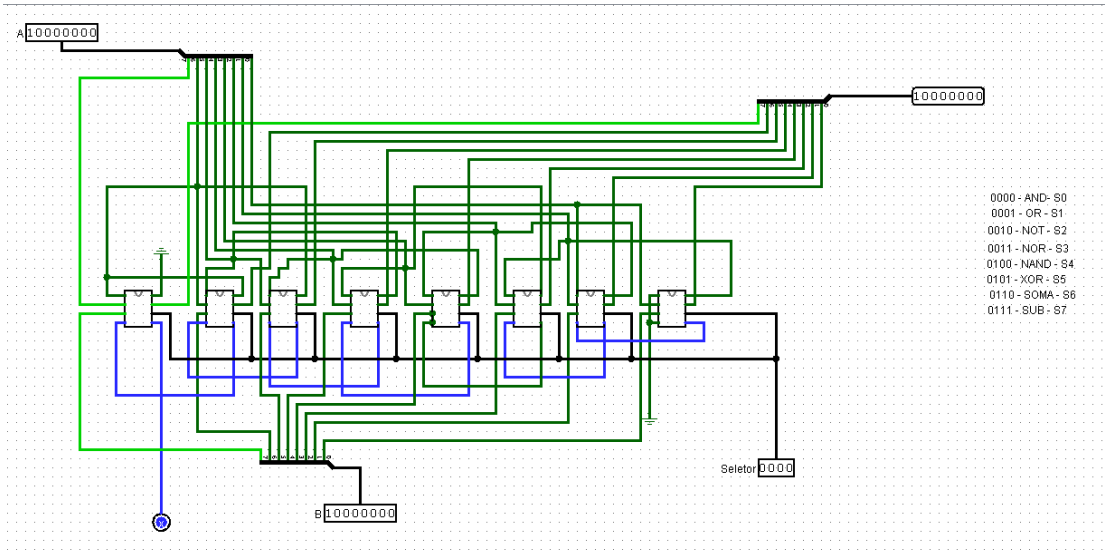
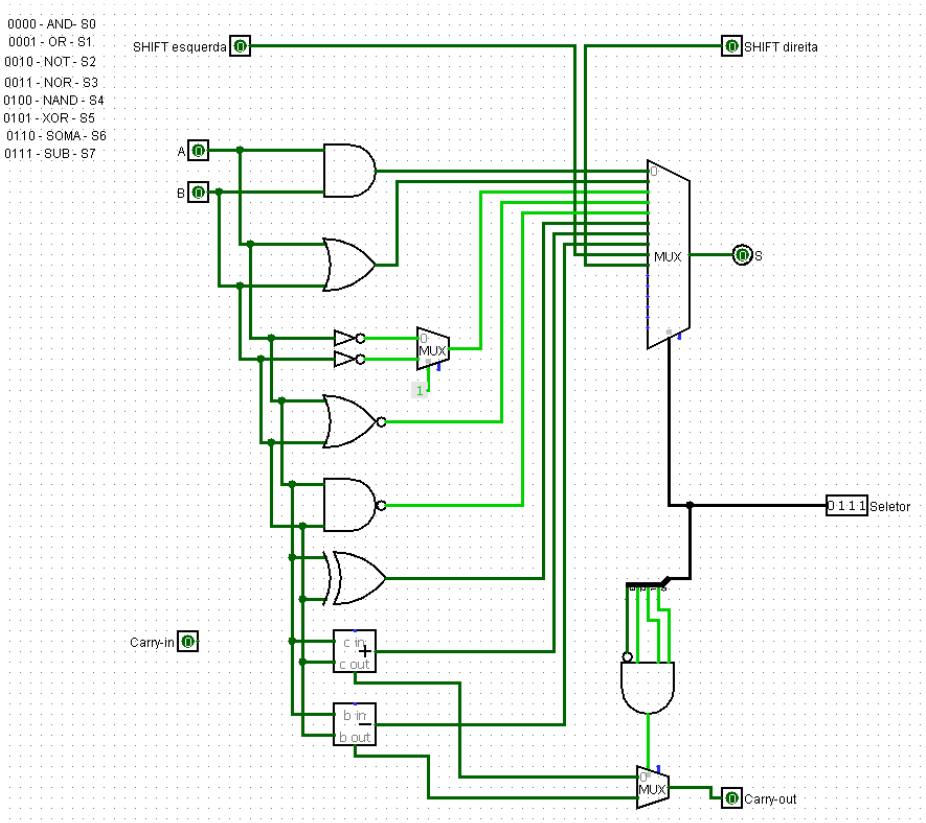


Figura 22- ULA 8 bits(seletor)



2.10.2. Testes do componente

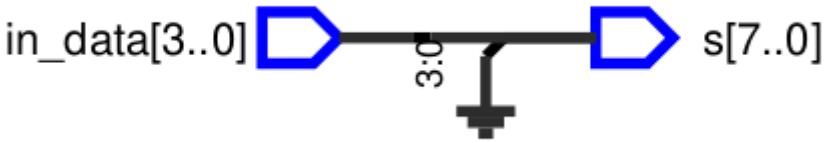
Na figura 19 pode-se ver um teste sendo realizado com os valores de A 10000000 e B 10000000 com operação 0000 (que representa a AND). Retornando na saída 10000000

2.11. EXTENSOR DE SINAL 4 PARA 8

Um extensor de sinal faz com que uma entrada de bits fique com uma largura maior que a anterior, na questão foi solicitada uma extensão de sinal de 4 par 8, logo se recebermos uma entrada de 4 bits 0011 a saída deverá ter 8 bits, no qual os novos bits são setados como 0, sendo igual a 00000011.

2.11.1. Descrição pinos e lógica

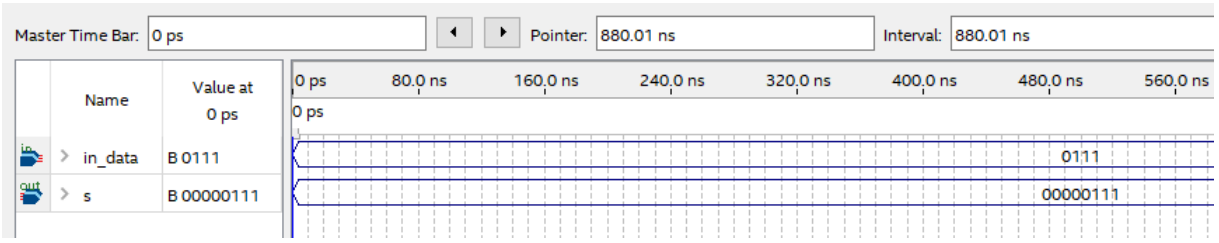
Figura 23- Extensor de sinal 4-8



De acordo com a definição do extensor de sinal temos um pino de entrada, com um vetor de 4 bits, e um pino de saída, com 8 bits. A extensão dos bits é feita com o acréscimo de bits menos significativos no vetor.

2.11.2. Testes do componente

Figura 24- WaveForm Extensor de sinal 4-8



Como podemos ver na waveform gerada temos o input do número $7_{(dec)}$ = $0111_{(bin)}$ e a saída esperada pela funcionalidade do extensor de sinal é 00000111, do mesmo modo como está ilustrado na figura é o resultado final.

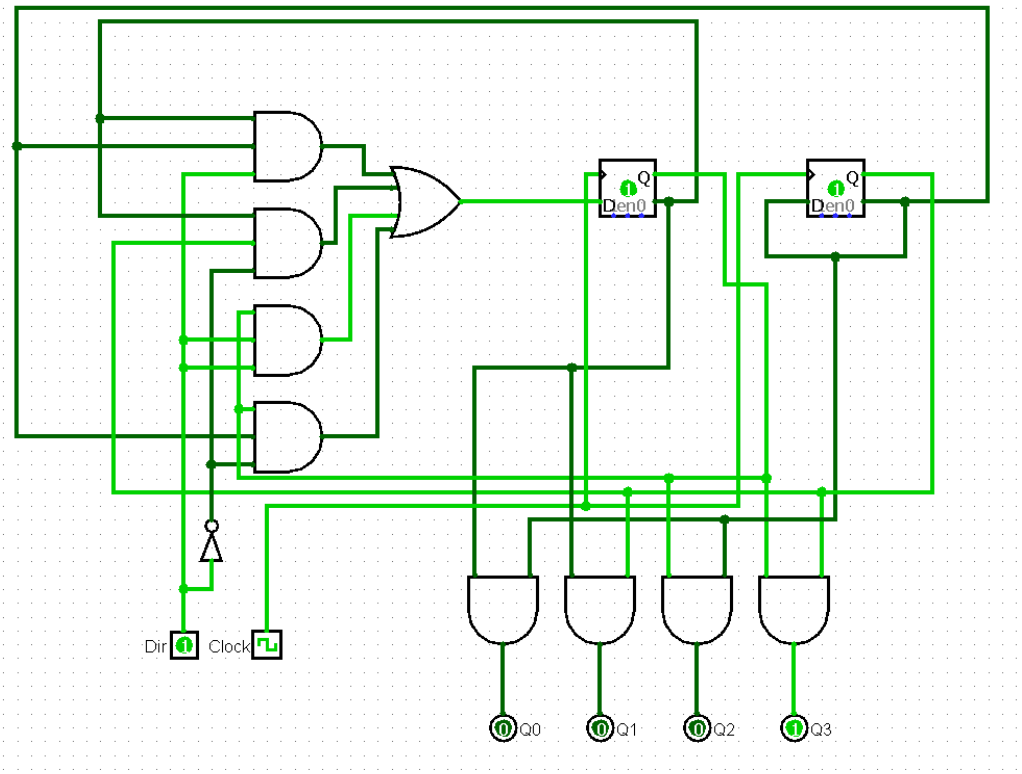
2.12. MÁQUINA DE ESTADO

Componente que itera e guarda estado de um objeto, mudando de comportamento quando o estado é alterado

2.12.1. Descrição pinos e lógica

No circuito temos uma entrada que seria P de acordo com a descrição da atividade e os resultados finais seriam o próprio R, estado é mudado de acordo com o cliclo de clock juntamente com o valor de P.

Figura 25- Máquina de estados



2.12.2. Testes do componente

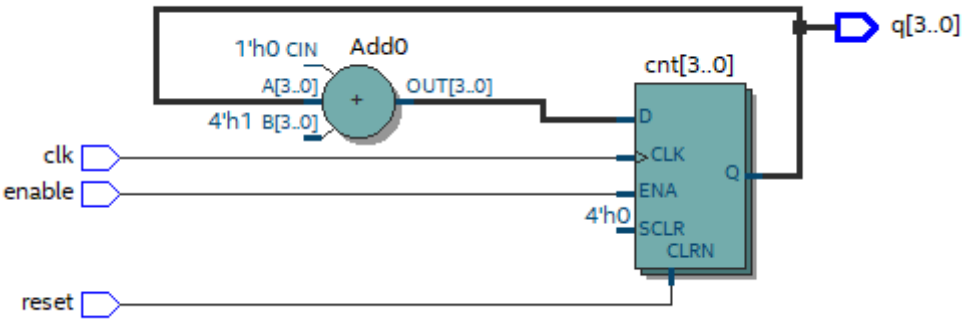
Como teste, o componente está no estado D que seria o Q3, onde P=1 e R=1, com o pulso de clock sendo dado recentemente o que alterou o estado para Q3 tendo como saída 1.

2.13. CONTADOR SÍNCRONO

Contador síncrono é um circuito digital formado por flip-flops em paralelos, tal que todas as entradas clocks estejam conectados na mesma fonte. Contadores síncronos são circuitos sequenciais capazes de executar contagens binárias de forma aleatória, pois possuem um circuito combinacional externo, que utiliza como entradas as saídas Q e Q' de cada flip-flop e suas saídas são conectadas às entradas dos flip-flops utilizados.

2.13.1. Descrição pinos e lógica

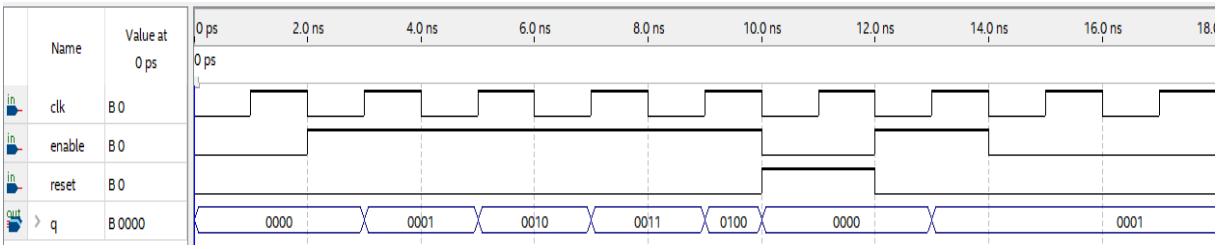
Figura 26- Contador Síncrono



Para o circuito do contador síncrono temos 3 pinos de entrada(clock, enable reset) e 1 pino de saída representado por um vetor de bits que representa o número do contador.

2.13.2. Testes do componente

Figura 27- WaveForm Contador Síncrono



No teste via waveform podemos observar o circuito clock normalmente, a entrada enable está ativa a partir do segundo ciclo de clock logo em seguida o contador é iniciado e só será reiniciado quando a entrada reset estiver ativa. Como podemos ver no teste o contador foi até $4_{(dec)}$ e depois teve um reset, que acabou por reiniciar o contador e este por sua vez está ativo logo em seguida com o enable ativo novamente fazendo com que o contador volte a ter um incremento no seu valor.

3. CONSIDERAÇÕES FINAIS

Podemos compreender que um processador é bem mais que somente um componente, ele seria mais um todo, há diversas partes e processos que o compõem. Como mostrado ao longo do relatório pudemos ver os circuitos integrados que dão o funcionamento ao processador, bem como os testes gerados em cima dos circuitos.

4. REFERÊNCIAS

Patterson, David A. Organização e projeto de computadores: a interface hardware/software/ David A Patterson e John L Hennessy; tradução de Daniel Vieria - Rio de Janeiro : Elsevier, 2005 - 2° reimpressão.

de la Vega, Alexandre Santos. Apostila com Códigos de Programas Demonstrativos usando a linguagem VHDL para Circuitos Digitais / Alexandre Santos de la Vega. – Niterói: UFF/TCE/TET, 2020.

MOL, Rian. Entendendo os Flip-FLop. **Flip e Flop, 2021**. Disponível em: https://www.filipeflop.com/blog/entendendo-o-flip-flops/. Acesso em: 20/11/2022.

Autor desconhecido. Flip-FLop. **Meca Web, 2019**. Disponível em: http://www.mecaweb.com.br/eletronica/content/e_flip_flop. Acesso em: 20/11/2022.

SILVEIRA, Daniel D. .Circuitos Lógicos Multiplexadores e Demultiplexadores, 2019. 17 slides. Disponível em: https://www.ufjf.br/daniel_silveira/files/2011/06/aula_101.pdf . Acesso em: 24/11/2022.

LIMA, Thiago. Somador Completo(Full adder). Embarcados, 2015. Disponível em: https://embarcados.com.br/tutorial-de-verilog-somador-completo/. Acesso em: 21/11/2022.

Como funciona um processador? (CPU, UC, ULA, Registradores). Canal TI, 2017. Disponível em: https://www.canalti.com.br/sistemas-operacionais/como-funciona-um-processador-cpu-uc-ula-registradores/ . Acesso em: 20/11/2022