



**UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

GIOVANNA MENDES GARBÁCIO

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

**BOA VISTA-RR
2022**

GIOVANNA MENDES GARBÁCIO

LABORATÓRIO DE CIRCUITOS – CODIFICAÇÃO E SIMULAÇÕES

Relatório Científico apresentado ao Prof. Dr. Herbert Oliveira Rocha, com objetivo de obtenção de nota parcial para aprovação na disciplina DCC 301 - Arquitetura e Organização de computadores, do Departamento de Ciência da Computação da Universidade Federal de Roraima.

**BOA VISTA-RR
2022**

Lista de figuras

Figura - 1	Componentes do circuito integrado do Flip-Flop do tipo D.....	1
Figura - 2	Circuito integrado do Flip-Flop do tipo D.....	1
Figura - 3	Circuito integrado do Flip-Flop do tipo D implementado no Logisim.....	2
Figura - 4	Registrador Flip-Flop do tipo D de 8 bits implementado no Logisim.....	3
Figura - 5	Circuito e componentes do circuito integrado do Flip-Flop do tipo JK.....	3
Figura - 6	Circuitos do Flip-Flop do tipo JK implementado no Logisim.....	4
Figura - 7	Representações de um multiplexador 4x1.....	5
Figura - 8	Implementação de um multiplexador 4x1 no Logisim.....	5
Figura - 9	Multiplexador 4x1 no Logisim, Caso 00.....	6
Figura - 10	Multiplexador 4x1 no Logisim, Caso 01.....	6
Figura - 11	Multiplexador 4x1 no Logisim, Caso 10.....	7
Figura - 12	Multiplexador 4x1 no Logisim, Caso 11.....	7
Figura - 13	Circuito somador de um bit.....	8
Figura - 14	Circuito somador de quatro bits.....	11
Figura - 15	Circuito somador de oito bits.....	11
Figura - 16	Circuito somador de oito bits com quatro.....	13
Figura - 17	Porta lógica XOR.....	15
Figura - 18	Extensor de sinal de quatro bits para oito bits.....	15
Figura - 19	ULA encapsulada.....	16
Figura - 20	ULA implementada no Logisim.....	17
Figura - 21	Circuito integrado de um Banco de Registradores.....	22
Figura - 22	Circuito de um Banco de Registradores implementados no Logisim: Teste 1.....	22
Figura - 23	Circuito de um Banco de Registradores implementados no Logisim: Teste 2.....	23

Figura - 24	Circuito encapsulado de uma CPU MIPS de 32 bits com a unidade de controle destacada.....	24
Figura - 25	Definições das flags da UC.....	24
Figura - 26	Circuito UC implementado no Logisim.....	26
Figura - 27	Esquema de máquina de estados a ser implementado.....	27
Figura - 28	Memória ROM implementada no Logisim.....	32
Figura - 29	Memória ROM simplificada implementada no Logisim.....	33
Figura - 30	Memória RAM implementada no Logisim.....	34
Figura - 31	Memória RAM implementada no Logisim: Clear.....	34
Figura - 32	Circuito de Memória RAM implementada no Logisim.....	35

Lista de tabelas

Tabela - 1	Testes do circuito integrado do Flip-Flop do tipo D implementado no Logisim.....	2
Tabela - 2	Tabela verdade e de testes do Flip-Flop JK onde cada linha apresenta uma subida de clock.....	4
Tabela - 3	Código do seletor e saída do multiplexador com todas as entradas com energia.....	5
Tabela - 4	Esquema da soma de dígitos binários.....	8
Tabela - 5	Tabela verdade do funcionamento do somador de um bit.....	9
Tabela - 6	Funcionamento do somador de um bit no Logisim.....	9
Tabela - 7	Testes do circuito somador de oito bits.....	12
Tabela - 8	Testes do circuito somador de oito bits com quatro.....	13
Tabela - 9	Tabela verdade da porta XOR.....	14
Tabela - 10	XOR implementado no Logisim.....	14
Tabela - 11	Testes extensor de sinal.....	16
Tabela - 12	Testes da ULA.....	17
Tabela - 13	Testes unidade de controle.....	25
Tabela - 14	Estados da máquina de estados.....	28
Tabela - 15	Contador síncrono.....	30

Sumário

1.	Registrador Flip-Flop do tipo D e do tipo JK.....	1
1.1.	Registrador Flip-Flop do tipo D.....	1
1.2.	Registrador Flip-Flop do tipo JK.....	3
2.	Multiplexador de quatro opções de entrada.....	4
2.1.	CASO 00.....	6
2.2.	CASP 01.....	6
2.3.	CASO 10.....	7
2.4.	CASO 11.....	7
3.	Somador de 8 bits.....	8
3.1.	Somador de 1 bit.....	8
3.2.	Somador de 4 bits.....	10
3.3.	Somador de 8 bits.....	11
4.	Somador de 8 bits que recebe um valor inteiro e soma com o valor 4.....	13
5.	Porta lógica XOR usando port map com os componentes: AND, NOT, e OR.	14
6.	Extensor de sinal de 4 bits para 8 bits.....	15
7.	ULA de 8 bits, utilizando port map, com as seguintes operações: AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de bits à direita, soma e subtração.....	16
8.	Banco de registradores.....	21
9.	Unidade de controle unicycle do MIPS de 16 bits.....	24
10.	Máquina de estados.....	27
11.	Contador síncrono.....	30
12.	Memória ROM.....	32
13.	Memória RAM.....	33

1. Registrador Flip-Flop do tipo D e do tipo JK

1.1. Registrador Flip-Flop do tipo D

No geral, Flip-Flops são componentes eletrônicos com dois estados estáveis que são usados para armazenar dados em binário. O registrador Flip-Flop do tipo D, em particular, atua de modo a quando houver a presença da borda de subida do clock, o valor guardado no flip-flop será o valor na entrada D naquele instante, ou seja, o registrador “lembra” do sinal de entrada enquanto o clock não apresentar uma nova subida de sinal.

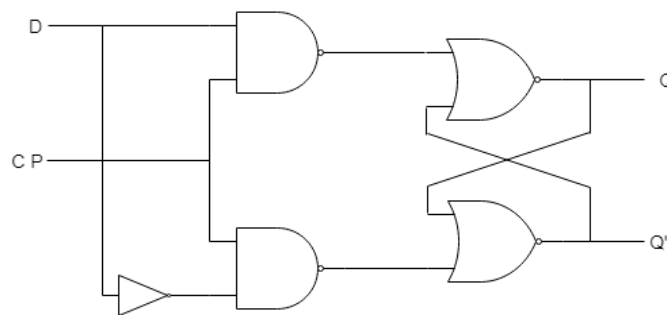


Figura 1 - Componentes do circuito integrado do Flip-Flop do tipo D

O circuito integrado do Flip-Flop D é composto pelas portas NAND, NOR e NOT. Ele possui uma entrada para o input do dado a ser guardado e outra para a conexão com o clock. A saída Q deve armazenar o dado da entrada D quando houver a subida da borda de clock e Q^* (Q barrado/ NOT Q) o inverso de Q.

A aplicação Logisim já possui um circuito Flip-Flop D pronto que possui outras duas entradas, R (Reset) que representa clear (quando 1, o estado do pino será 0 assíncronamente, que reseta a saída para 0) e outra S (Set) de enable (quando em 0, os gatilhos de clock não serão efetivados).

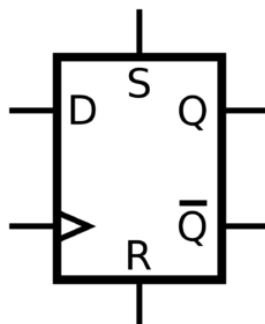


Figura 2 - Circuito integrado do Flip-Flop do tipo D

Esse circuito, por si só, já é um registrador de um bit. Implementando-o na Aplicação Logisim:

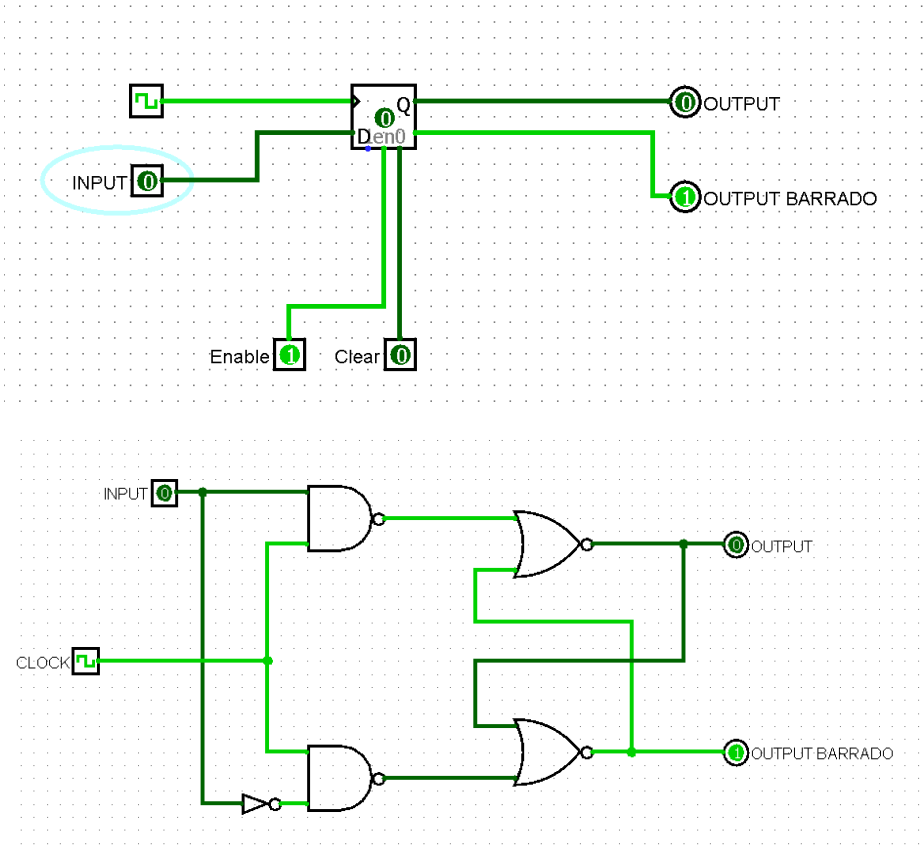


Figura 3 - Circuitos integrado do Flip-Flop do tipo D implementado no Logisim

Foram realizados os seguintes testes com esse circuito:

Tabela 1 - Testes do circuito integrado do Flip-Flop do tipo D implementado no Logisim

TESTE	CLOCK	INPUT	OUTPUT	DESCRIÇÃO DO TESTE
1	0	0	0	O clock sobe de 0 para 1, e o output é o igual ao input do momento de subida, 0. Mesmo mudando o input para 1, o output continua 0 pois o clock não subiu novamente.
	1	1	0	
	0	1	0	
2	0	1	0	No momento da subida de clock o output copia o input de 1. O input é trocado para 0 na descida do clock mas o output não muda. Mudado o input para 1, e o clock subindo, o output é atualizado e mantém o 1. Ele é estável mesmo com a troca do input para 0 pois não há subida de clock.
	1	0	1	
	0	1	1	
	1	0	1	

É interessante construir um registrador de oito bits para que ele seja usado como componente em um banco de registradores de oito bits posteriormente.

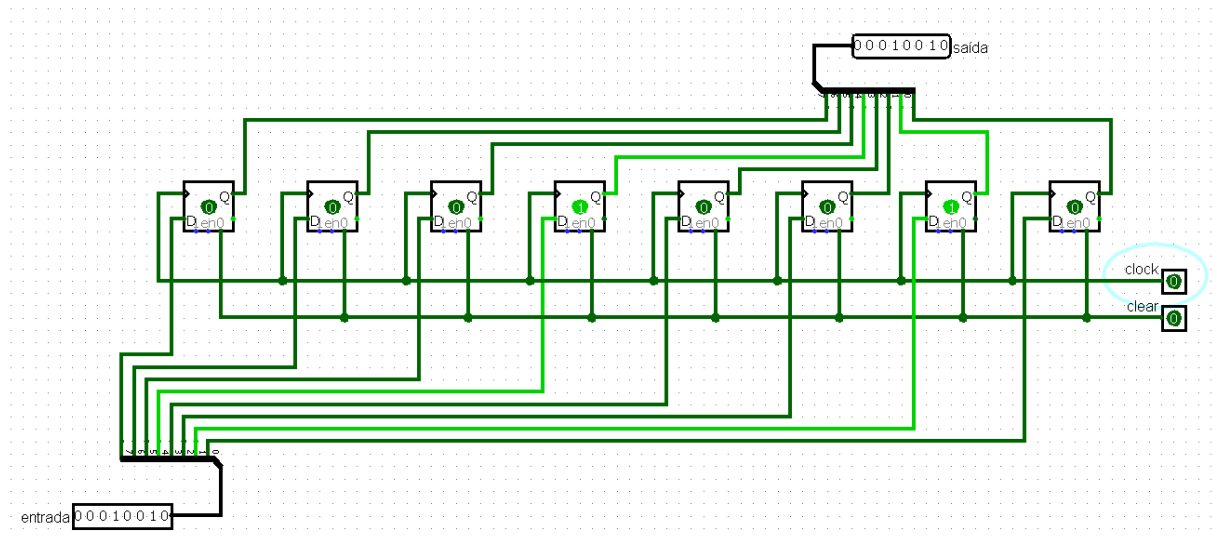


Figura 4 - Registrador Flip-Flop do tipo D de 8 bits implementado no Logisim

As entradas D de cada Flip-Flop estão ligadas em um input de oito bits, o clock é compartilhado por todos os Flips-Flops, assim como o Reset (clear). As saídas são recebidas em um distribuidor, que as organiza em um output de oito bits. A mesma lógica é aplicada, porém em vez de um bit de entrada, há oito.

1.2. Registrador Flip-Flop do tipo JK

O registrador JK é constituído por quatro portas NAND e uma entrada J (que representa Set) e K (que funciona como Reset). Ele funciona de modo que quando é aplicado 1 nas entradas J e K ocorre uma inversão dos níveis lógicos presentes das saídas anteriores à subida do clock.

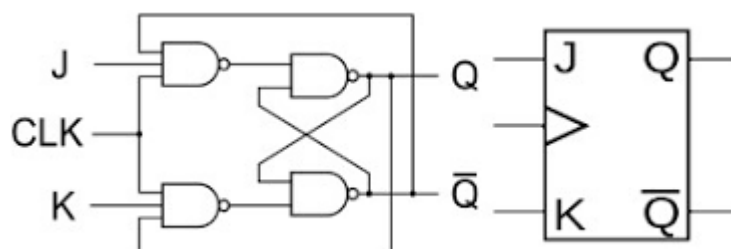


Figura 5 - Circuito e componentes do circuito integrado do Flip-Flop do tipo JK

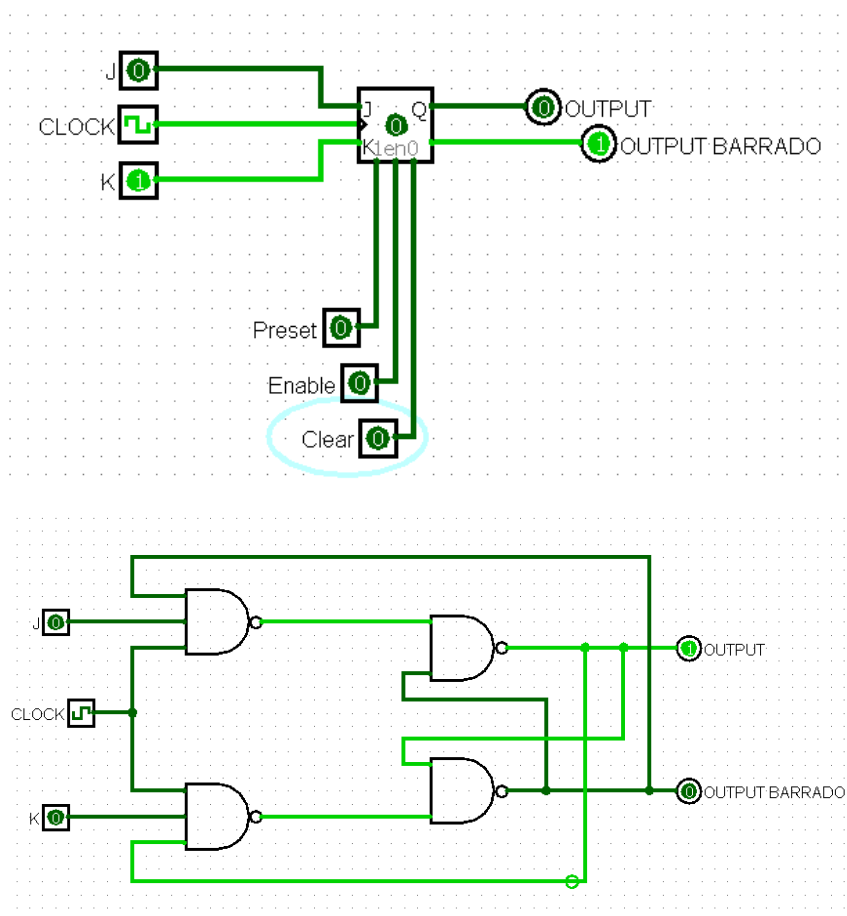


Figura 6 - Circuitos do Flip-Flop do tipo JK implementado no Logisim

Tabela 2 - Tabela verdade e de testes do Flip-Flop JK, cada linha apresenta uma subida de clock

J	K	OUTPUT
0	0	Não muda
1	0	1
0	1	0
1	1	Inverte

2. Multiplexador de quatro opções de entrada

Um multiplexador é um componente de circuito que recebe duas ou mais entradas e repassa uma delas como output a partir de um seletor. A quantidade de bits do seletor varia dependendo da quantidade de inputs, $2^{\text{quantidade de bits para seleção}}$ deve ser um número igual ou maior do número de entradas. No caso, com quatro entradas, teremos $2^2 = 4$, ou seja, dois bits de seleção.

Ele é formado por quatro portas AND, uma OR e duas NOT. Seu circuito integrado é representado dessa forma:

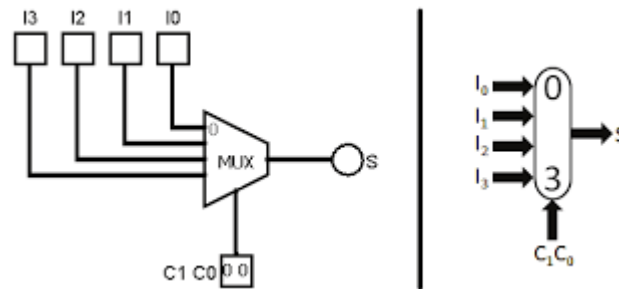


Figura 7 - Representações de um multiplexador 4x1

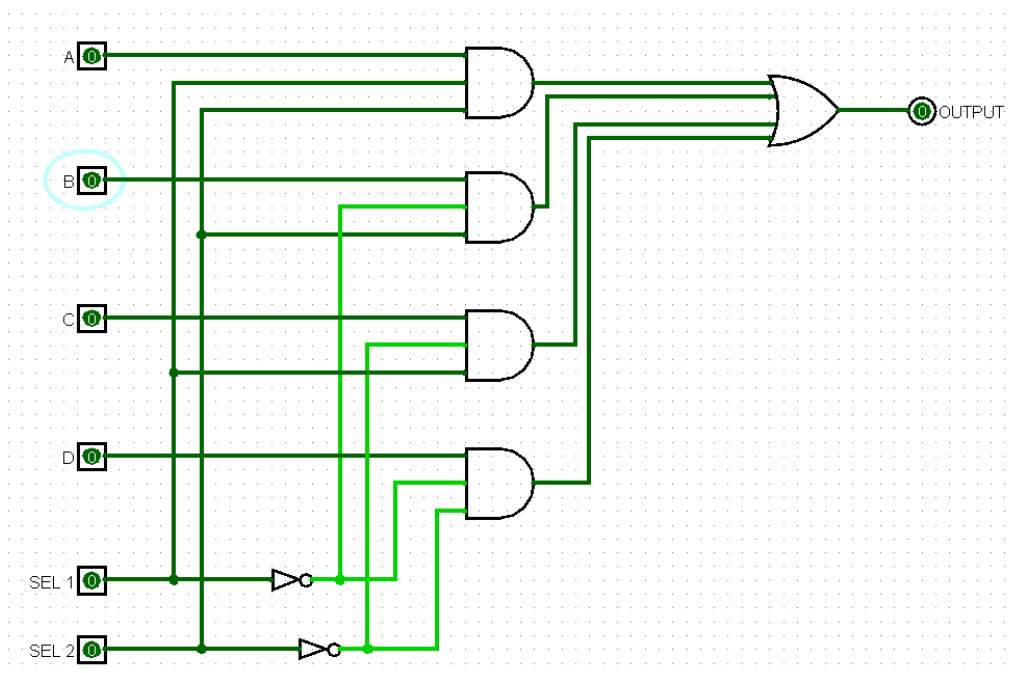


Figura 8 - Implementação de um multiplexador 4x1 no Logisim

Nessa forma de implementação, o código de seleção funciona de modo a ter a saída referente a entra:

Tabela 3 - Código do seletor e saída do multiplexador com todas as entradas com energia

SEL 1 e 2	OUTPUT
00	D
01	B
10	C
11	A

Analisando caso por caso, pode-se observar quais fios são alimentados para acionar a porta AND que repassa a opção escolhida pelo código seletor.

2.1. CASO 00

Nesta opção, o quarto AND é completamente alimentado pela negação das duas entradas seletoras 0, e pela entrada D, o output é D.

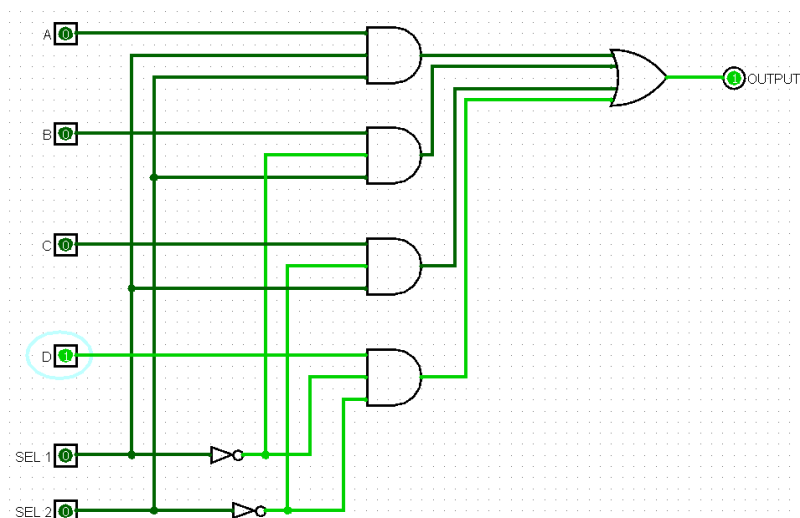


Figura 9 - Multiplexador 4x1 no Logisim, Caso 00

2.2. CASO 01

Para que o output seja B, o segundo AND tem as três entradas ativadas, o próprio B, a negação do seletor 1 que está como 0, o seletor 2.

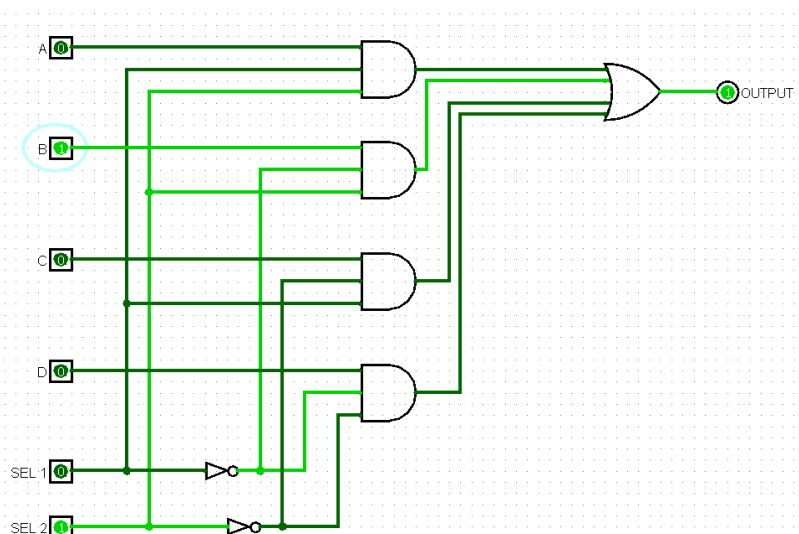


Figura 10 - Multiplexador 4x1 no Logisim, Caso 01

2.3. CASO 10

Alimentando a entrada C e o seletor 1, em conjunto com a negação do 0 do seletor 2, há a ativação do terceiro AND, que retorna o output igual a entrada C.

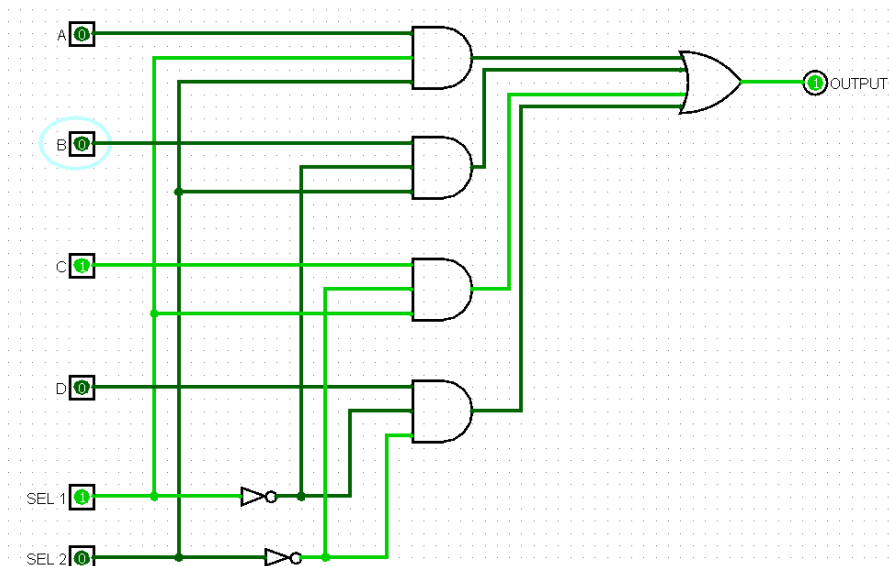


Figura 11 - Multiplexador 4x1 no Logisim, Caso 10

2.4. CASO 11

Neste último caso, a entrada A ativa a primeira porta AND com a alimentação dos dois seletores ativada. A é o output.

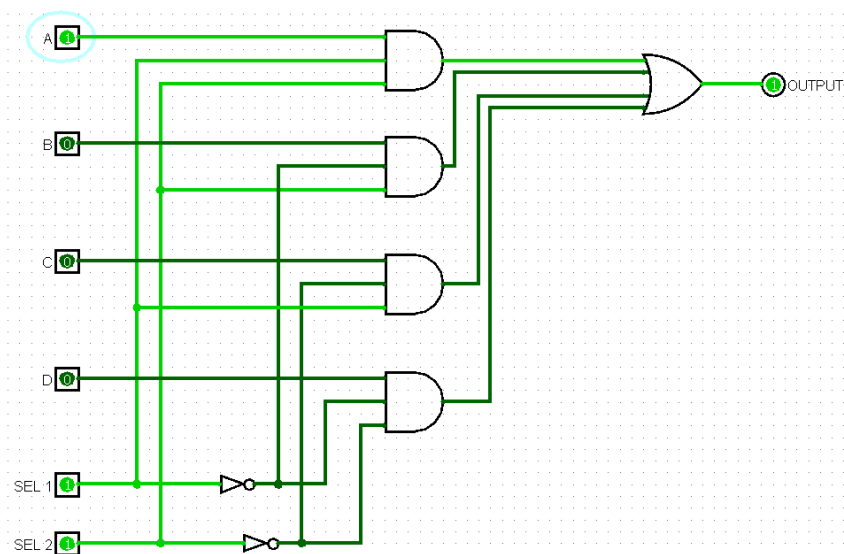


Figura 12 - Multiplexador 4x1 no Logisim, Caso 11

3. Somador de 8 bits

Para a construção de um somador é necessário compreender como funciona a operação de adição entre números binários.

Tabela 4 - Esquema da soma de dígitos binários

SOMA	RESULTADO	RESTO
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

3.1. Somador de 1 bit

A partir do esquema acima percebe-se a necessidade de entradas A e B para os números a serem somados, uma saída para o resto (Carry-out), uma entrada para o resto na casa seguinte (Carry-in), e uma saída para o resultado. Com essas informações, pode-se montar um somador de um bit com três portas ANDs, duas XORs e uma OR:

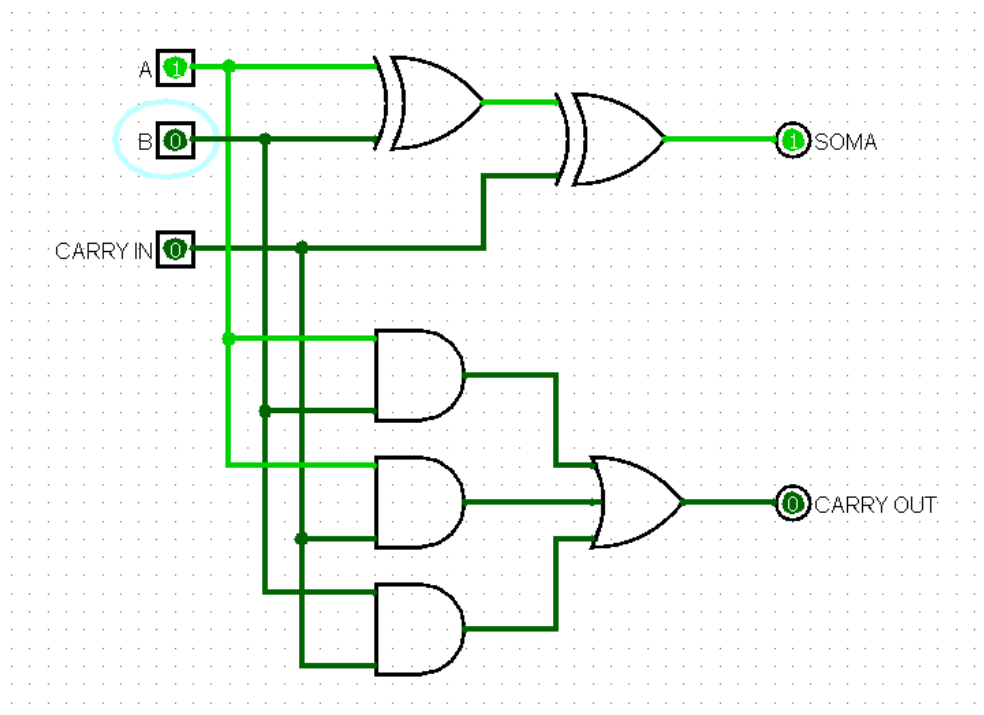


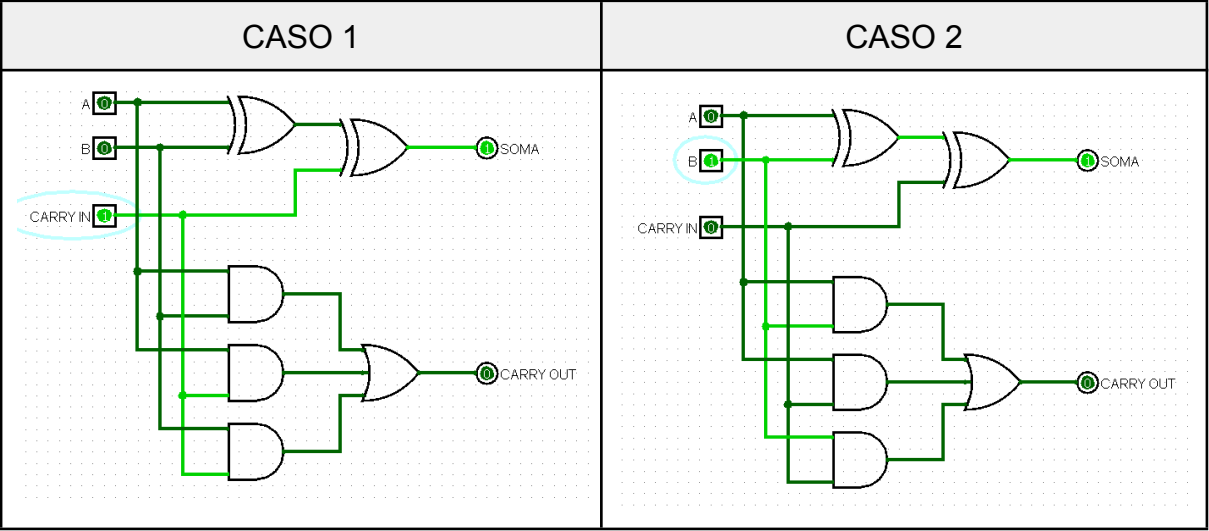
Figura 13 - Circuito somador de um bit

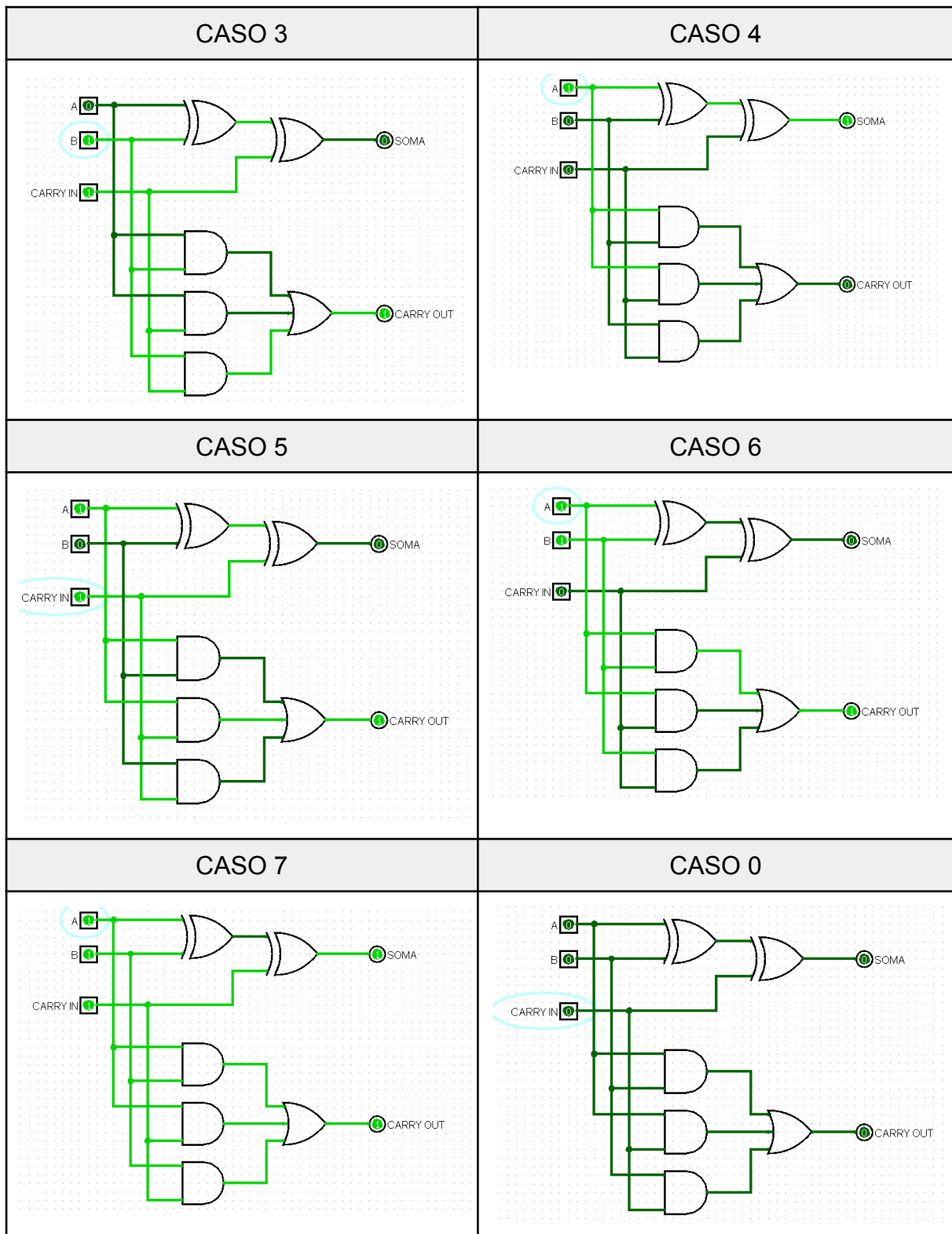
Em suma, toda vez que ocorre a soma de A, B e Carry-In, em qualquer ordem, representada por 1+0+0, SOMA é energizado com 1. Nessa mesma lógica, se ocorrer 1+1+0, Carry-out é a saída energizada, simbolizando o resto. No caso de 1+1+1, ambos Carry-out e SOMA tem saída 1. Isso está organizado e demonstrado nas tabelas a seguir:

Tabela 5 - Tabela verdade do funcionamento do somador de um bit

CASO	A	B	CARRY IN	SOMA	CARRY OUT
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Tabela 6 - Funcionamento do somador de um bit no Logisim





3.2. Somador de 4 bits

A partir do circuito apresentado, pode-se criar um circuito maior para soma de quatro bits utilizando quatro circuitos de soma de um bit. O primeiro bit a ser somado possui o seu carry-in ligado a um fio terra pois ele é sempre 0, os carry-ins seguintes

são ligados ao carry-outs do circuito anterior. O último carry-out vai ao final do resultado da soma como uma casa de bit extra/overflow.

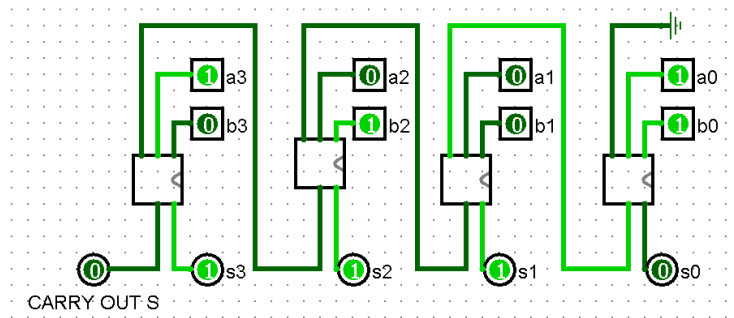


Figura 14 - Circuito somador de quatro bits

3.3. Somador de 8 bits

Conectados dois circuitos somadores de 4 bits, forma-se um de 8:

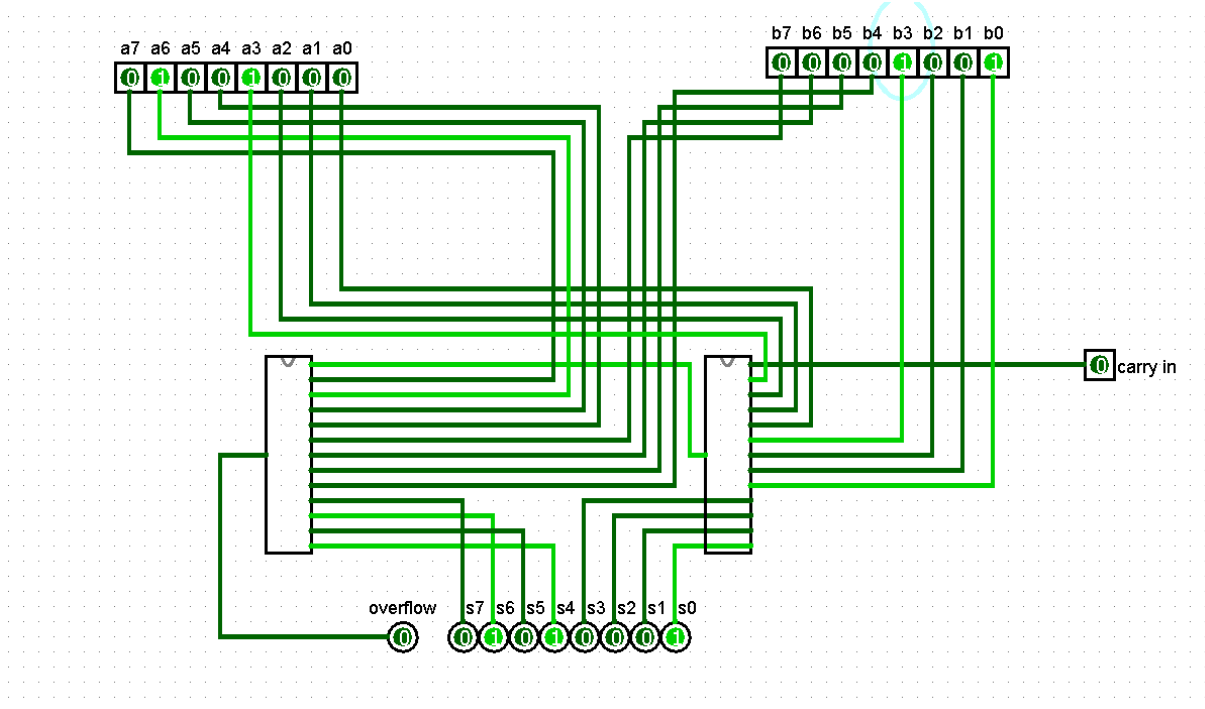
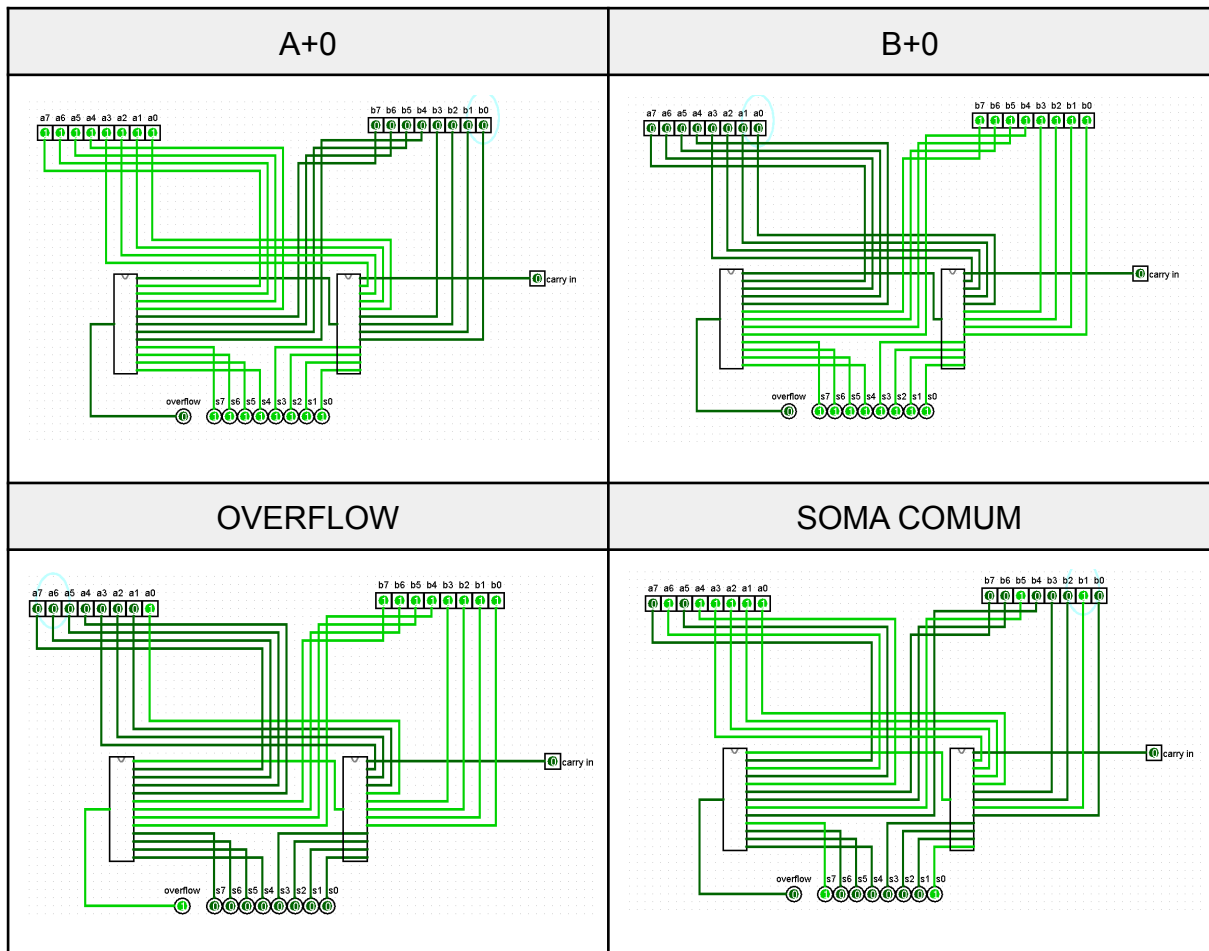


Figura 15 - Circuito somador de oito bits

O primeiro somador, da direita para a esquerda, recebe os quatro primeiros dígitos do número A (a0, a1, a2 e a3) e os de B (b0, b1, b2 e b3), enquanto o segundo recebe os 4 últimos de cada um (a4, a5, a6, a7, b4, b5 e b6). O carry-out do segundo somador indica um overflow na soma, quando o resultado é um número maior do que oito dígitos.

Como a tabela verdade gerado por este circuito é muito extensa, realizou-se alguns testes para averiguar a validade do sistema, destacam-se, a soma de A com 0, de B com 0, um caso de Overflow, e uma soma comum, respectivamente, na tabela:

Tabela 7 - Testes do circuito somador de oito bits



- A+0: O resultado é exatamente igual a A, pois soma-se cada dígito de A com 0 de B. Não há ocorrências de overflow nem carry-in/carry-out.
- B+0: Análogo a A+0.
- Overflow: Nesse exemplo, o resultado estoura o limite do resultado de 8 bits e o último carry-out ativa o output overflow.
- Soma comum: As somas de cada bit ocorrem normalmente, fazendo uso de carry-in/carry-out, sem acusar overflow.

4. Somador de 8 bits que recebe um valor inteiro e soma com o valor 4

Para somar um número inteiro de oito bits com quatro decimal (100 em binário), usa-se o circuito somador de oito bits feito anteriormente. Como input tem-se um número de 8 bits, e substitui-se o outro input por uma constante de 8 bits em binário que se traduz para quatro decimal.

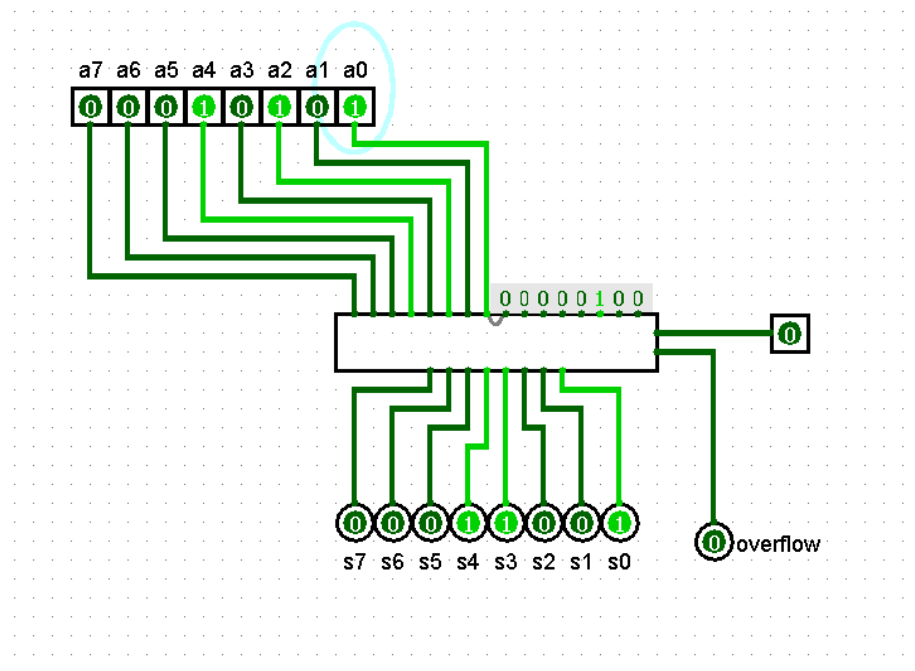


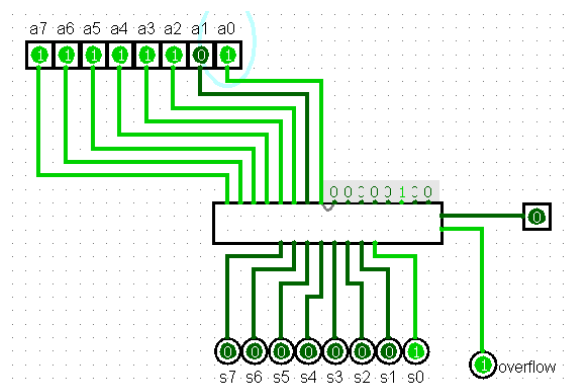
Figura 16 - Circuito somador de oito bits com quatro

Tabela 8 - Testes do circuito somador de oito bits com quatro

TESTE 1	
DESCRIÇÃO	<div><p>O input do número inteiro igual a 00000000, ao ser somado com 0100, obteve-se o resultado esperado de 00000100. O overflow não é acionado e o carry-in se mantém sempre em 0.</p><div><div>00000000 00000100+ 00000100</div><div><p>Diagrama de um somador de 8 bits. O input superior (a7 a0) contém os bits 0, 0, 0, 0, 0, 0, 0, 0. O input inferior (s7 s6 s5 s4 s3 s2 s1 s0) contém os bits 0, 0, 0, 0, 0, 1, 0, 0. O resultado (s7 s6 s5 s4 s3 s2 s1 s0) é 0, 0, 0, 0, 0, 1, 0, 0. O overflow é 0. O carry-in é 0.</p></div></div></div>
TESTE 2	

Nesse teste ocorre um overflow, isso verifica a validade das operações com carry-out e carry-in. Obteve-se o resultado esperado.

111111 Carry-in/out
 11111101
 00000100+
 100000101 Overflow



5. Porta lógica XOR usando port map com os componentes: AND, NOT, e OR

XOR é uma porta lógica formada por duas portas ANDs, uma porta OR e um NOT. Sua tabela verdade e lógica por trás do circuito implementado no Logisim estão representados nas duas tabelas a seguir:

Tabela 9 - Tabela verdade da porta XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 10 - XOR implementado no Logisim

OBSERVAÇÕES	XOR NO LOGISIM
A e B ambos possuem baixo nível de energia, a única parte energizada do circuito é a saída negada do primeiro AND, que não é suficiente para ativar o segundo AND com saída 1.	
B está energizado, ele entra no OR, que é ativado e no AND que não é ativado pois sua outra entrada A é 0, a negação desse AND de saída 0 com a saída 1 do OR entram em um AND, que permite que A XOR B = 1.	

<p>A está energizado, ele entra no OR, que é ativado e no AND que não é ativado pois sua outra entrada B é 0, a negação desse AND de saída 0 com a saída 1 do OR entram em um AND, que permite que $A \text{ XOR } B = 1$.</p>	
<p>Ambos A e B estão energizados, resultando em uma saída 1 no OR e no AND, no entanto, a saída do AND é negada, se tornando 0. O AND final recebe 0 e 1, resultando em uma saída 0.</p>	

Seu circuito encapsulado é representado conforme a imagem:

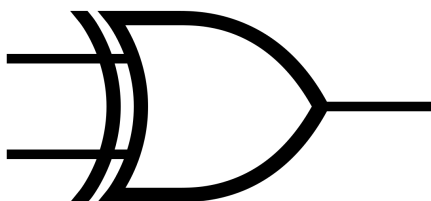


Figura 17 - Porta lógica XOR

6. Extensor de sinal de 4 bits para 8 bits

O extensor de sinal serve para estender as casas de um número à sua esquerda. O extensor de quatro bits para oito bits recebe um número binário de quatro dígitos em um divisor que redireciona cada um deles para as quatro primeiras casas no novo número por outro divisor, esse com oito espaços. Os quatro espaços restantes são completados com constantes 0.

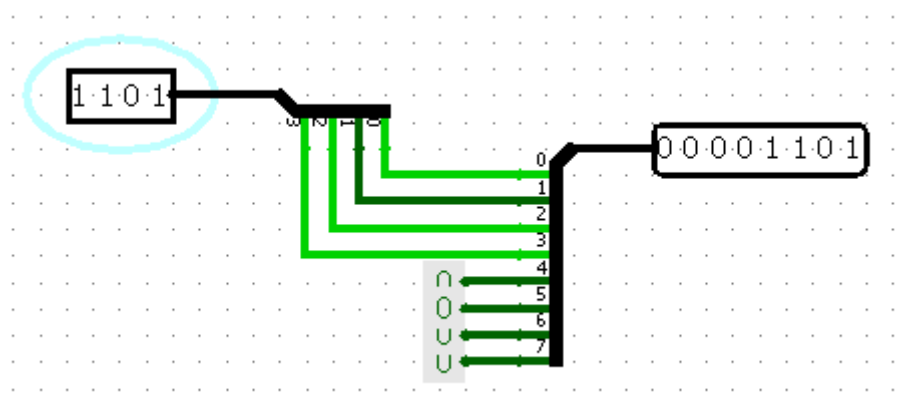
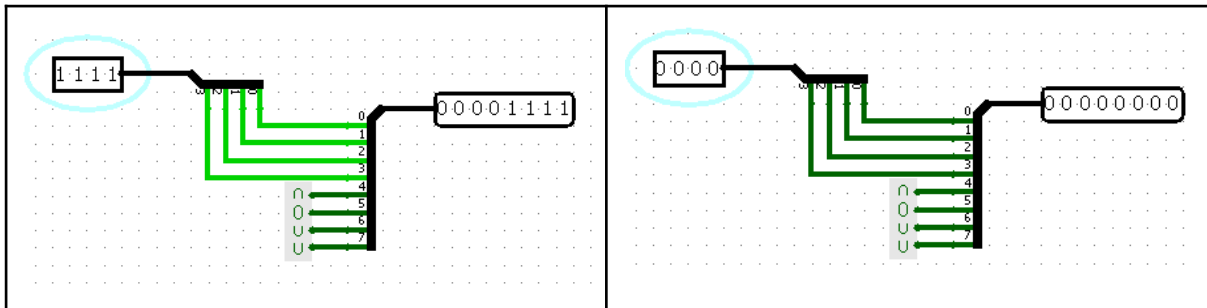


Figura 18 - Extensor de sinal de quatro bits para oito bits

Tabela 11 - Testes extensor de sinal



7. ULA de 8 bits, utilizando port map, com as seguintes operações: AND, OR, NOT, NOR, NAND, XOR, SHIFT de 2 bits à esquerda, SHIFT de bits à direita, soma e subtração

ULA é um acrônimo para Unidade Lógica e Aritmética, ela é um componente que realiza esses tipos de operações com inputs e retorna o seu resultado. A seleção de qual operação deve ser realizada é feita por meio de um código seletor no multiplexador que identifica a saída de cada componente para determinada ação. Ela é simbolizada como na figura abaixo onde A e B são os inputs, Opcode é o código seletor que controla o multiplexador, Out é a saída do resultado, Carry significa o resto de uma soma ou subtração ou caso de overflow, e a flag Zero sinaliza quando o output é igual a 0.

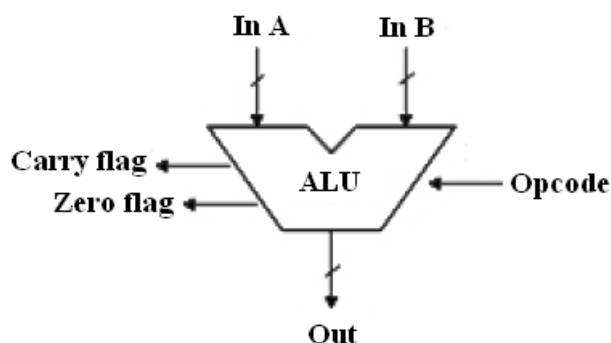


Figura 19 - ULA encapsulada

Aplicando as operações definidas no componente no Logisim, cria-se o circuito a seguir.

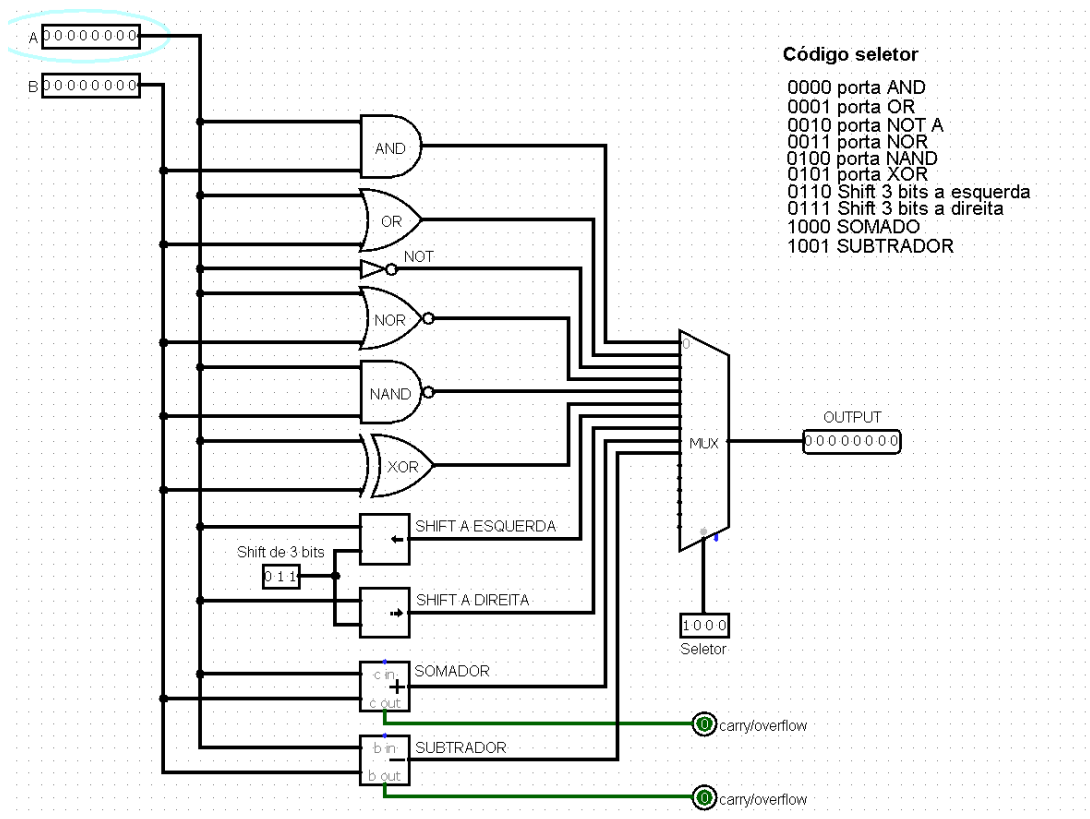
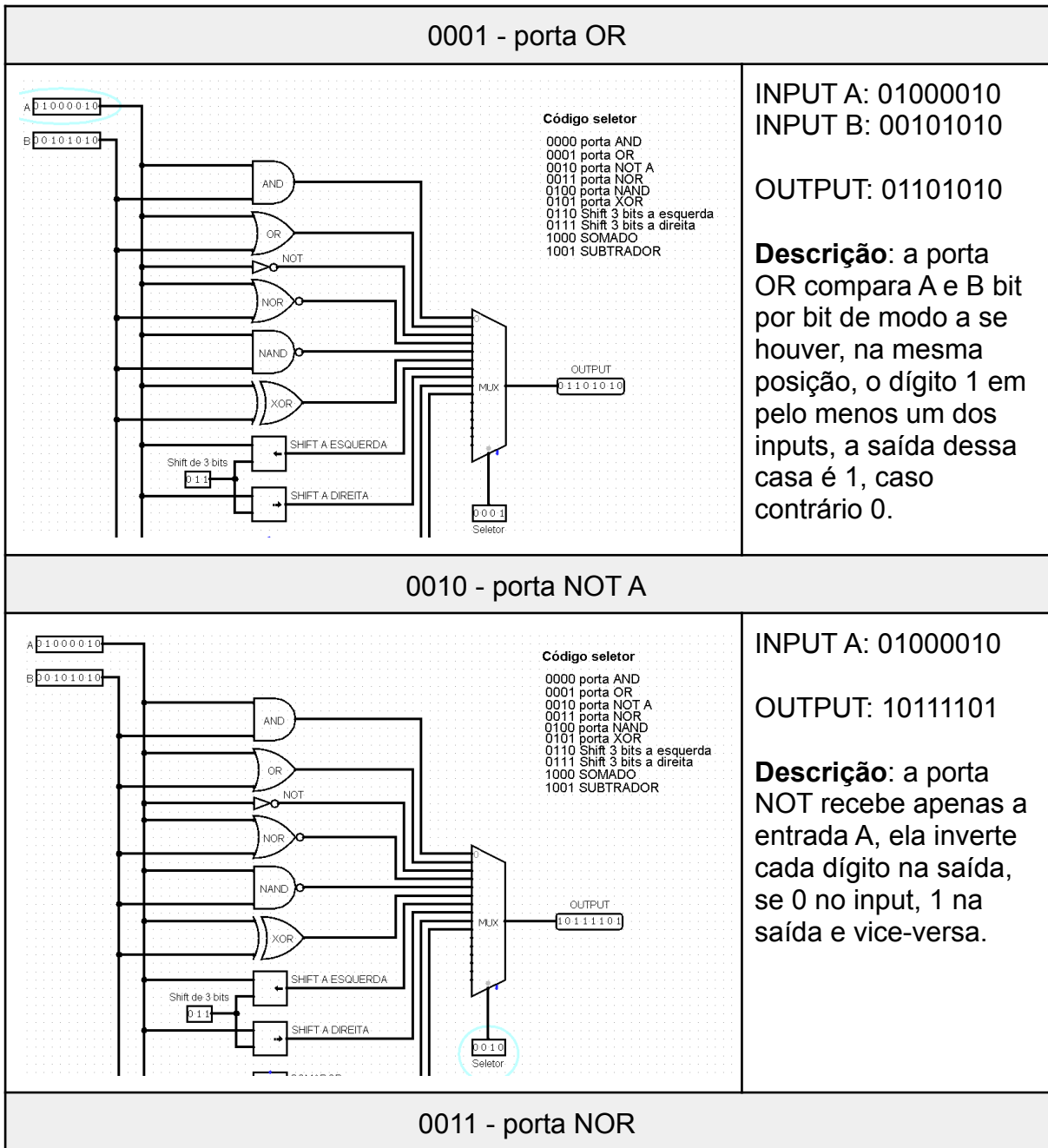


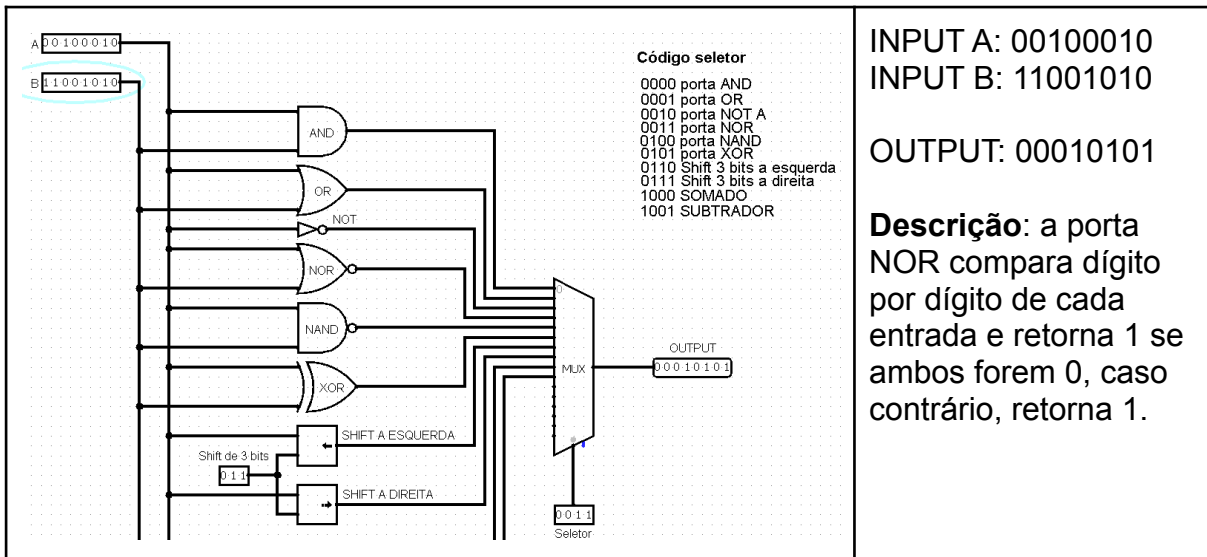
Figura 20 - ULA implementada no Logisim

A partir da observação e testes de cada operação foi formulada a tabela:

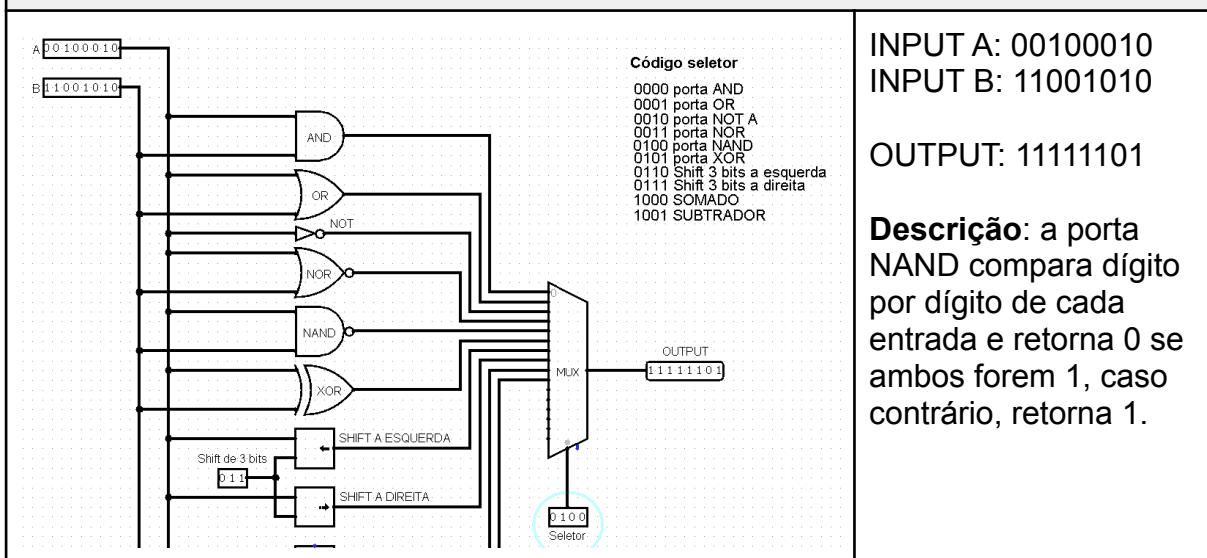
Tabela 12 - Testes da ULA

0000 - porta AND		
	<p>Código seletor</p> <ul style="list-style-type: none"> 0000 porta AND 0001 porta OR 0010 porta NOT A 0011 porta NOT B 0100 porta NAND 0101 porta XOR 0110 Shift 3 bits a esquerda 0111 Shift 3 bits a direita 1000 SOMADO 1001 SUBTRADOR 	<p>INPUT A: 00011110 INPUT B: 00101010</p> <p>OUTPUT: 00001010</p> <p>Descrição: A porta AND compara A e B bit por bit e formula uma saída formada por 0 (se bits na mesma posição forem diferentes ou ambos 0) e 1 (se ambos bits forem 1).</p>

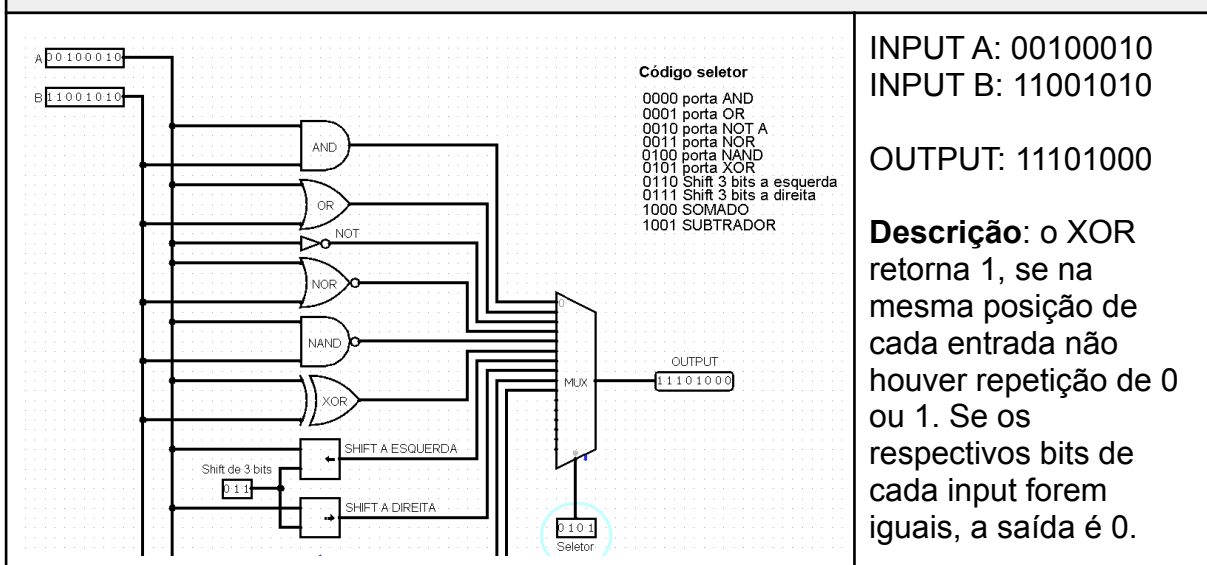




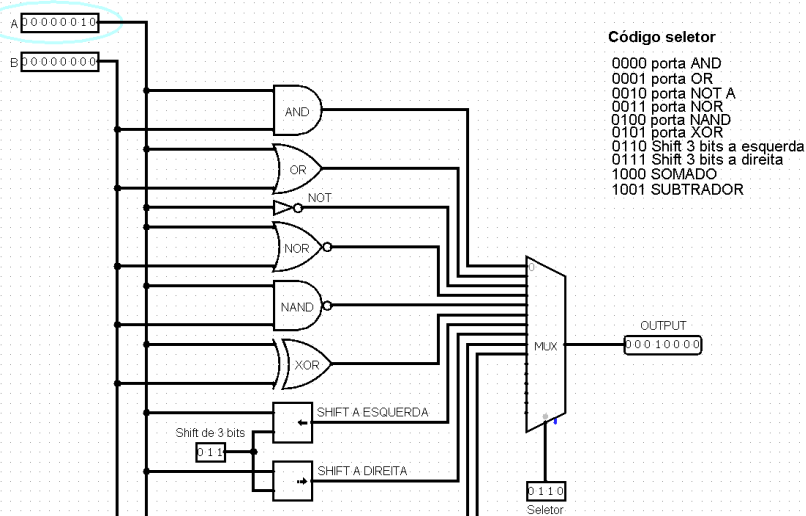
0100 - porta NAND



0101 - porta XOR



0110 - shift 3 bits a esquerda

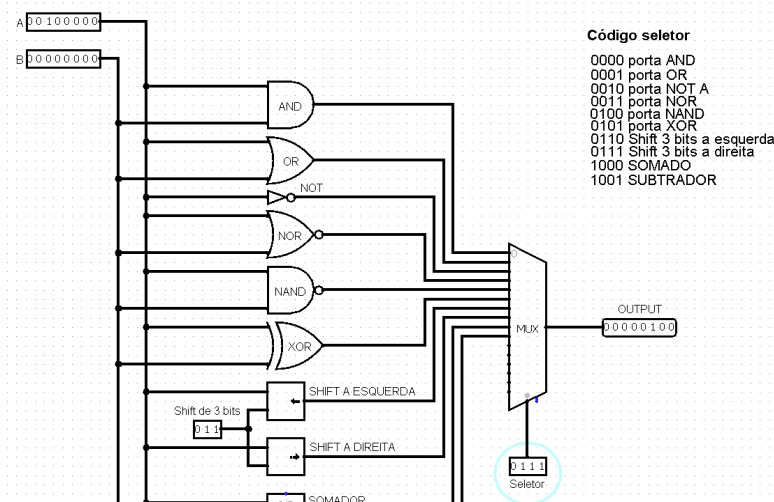


INPUT A: 00000010

OUTPUT: 00010000

Descrição: O shift adiciona três casas à direita da entrada com 0 e retira as últimas três à esquerda, deslocando o input inicial. Qualquer número nessas últimas três posições são perdidos.

0111 - shift 3 bits a direita

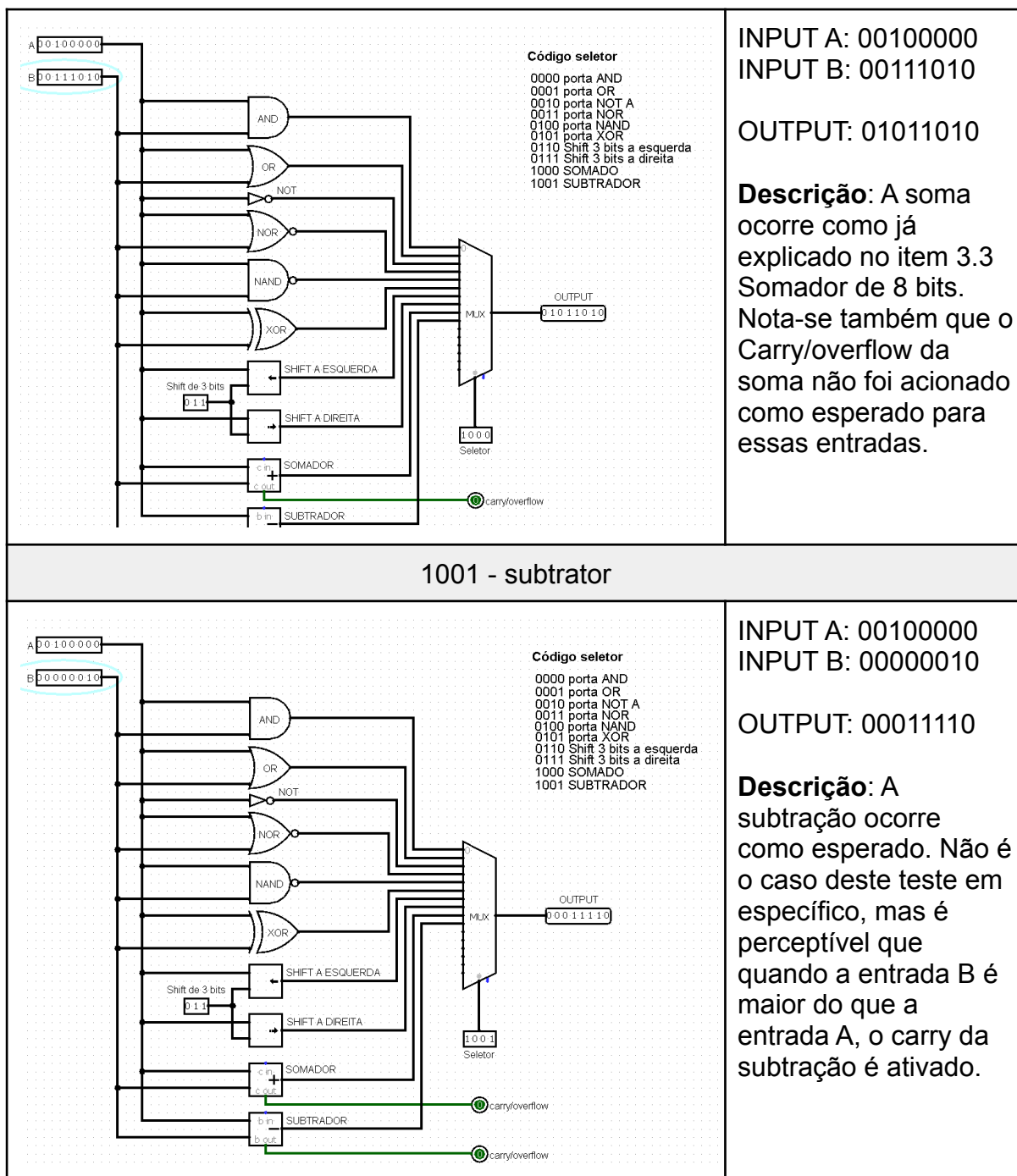


INPUT A: 00100000

OUTPUT: 00010000

Descrição: O shift adiciona três casas à esquerda da entrada com 0 e retira as primeiras três à direita, deslocando o input inicial. Qualquer número nessas primeiras três posições são perdidos.

1000 - somador



8. Banco de registradores

Usando um registrador de oito bits como o abordado no tópico 1.1 Registrador flip-flop do tipo D, pode-se criar um aglomerado de registradores dentro de um banco para facilitar o acesso, escrita e leitura de dados e endereços.

Nele, há uma entrada para escrita em um registrador, duas entradas e saídas para leitura de dados e um input para um retorno de informações do banco de dados.

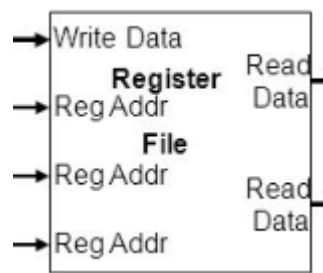


Figura 21 - Circuito integrado de um Banco de Registradores

Foram enfileirados oito registradores de oito bits, todos estão conectados ao mesmo clock, a mesma entrada de oito bits para ser usada no ato da escrita de um registrador e o mesmo controle de Clear (Reset). A saída de cada registrador é passada para dois multiplexadores para ser lida. A seleção de qual registrador será escrito é feita por um código de três bits referente a cada registrador que entra em um decodificador. Essa escrita é permitida enquanto o Enable desse decodificador é diferente de 0.

No modelo construído é possível escrever um registrador de cada vez e ler dois ao mesmo tempo devido a presença de dois multiplexadores ao final. A escolha de qual registrador é lido por qual multiplexador também é feita por um código de três bits.

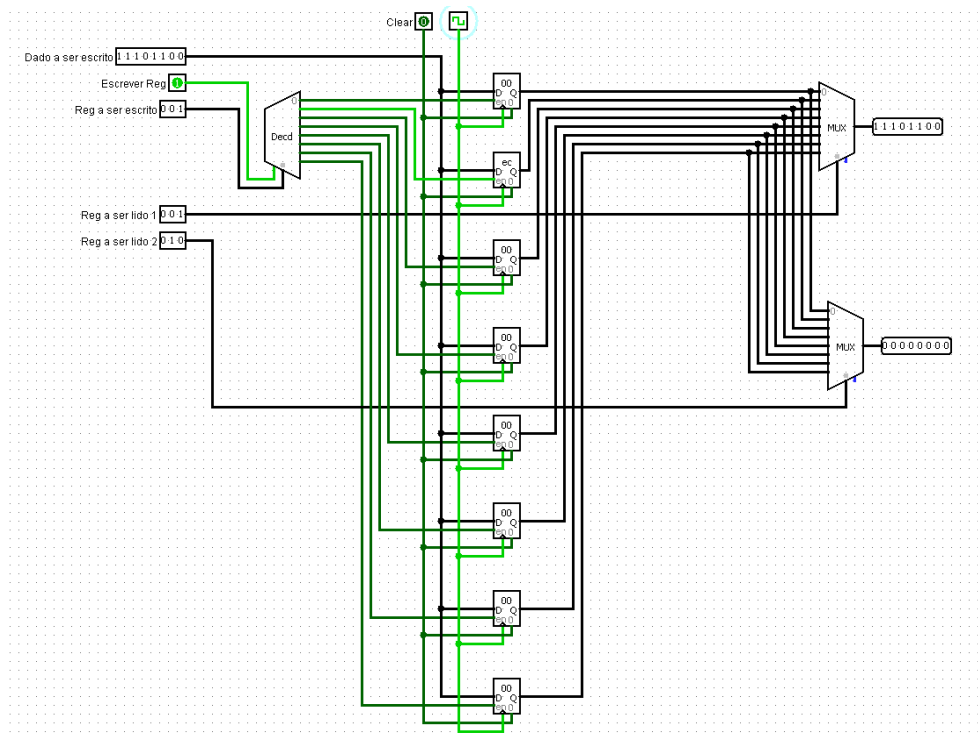


Figura 22 - Circuito de um Banco de Registradores implementados no Logisim: Teste 1

Na imagem acima, o dado a ser escrito foi definido com 11101100, a permissão para escrever no registrador está ligada e o clock passou pela sua subida de sinal permitindo que o dado fosse escrito no registrador 001 (segundo de cima para baixo) conforme definido no seletor do decodificador. Os outputs estão selecionados de modo a apresentar os dados dos registradores de códigos 001 (registrador que acaba de ser escrito) e 010 (registrador que não foi escrito após o Reset), consequentemente, as saídas são 11101100 e 00000000.

Logo após este teste, o clock foi desligado, o input de dados mudado para 00000001 e o registrador a ser escrito definido com o 110, assim como o segundo registrador a ser lido (o primeiro continua com o 001). Após a mudança do clock de baixa energia para alta energia, o circuito mudou a sua segunda saída para 00000001 e a primeira se manteve 11101100, como mostra a figura abaixo.

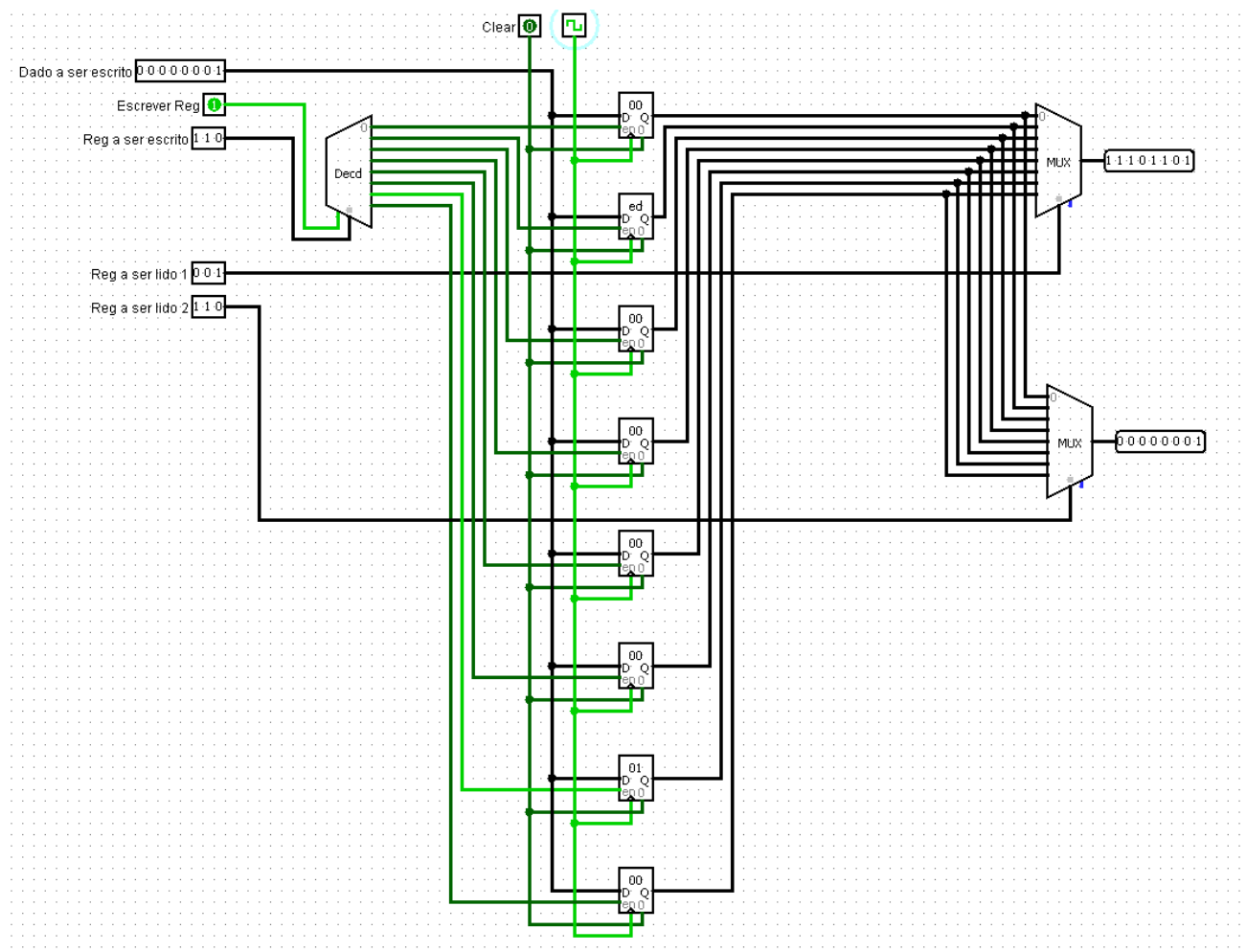


Figura 23 - Circuito de um Banco de Registradores implementados no Logisim: Teste 2

9. Unidade de controle uniciclo do MIPS de 16 bits

A unidade de controle é um componente essencial da CPU, ela é responsável por acionar e controlar os sinais para que cada instrução seja executada de forma correta. Ela funciona com um multiplexador controlado por um código para cada instrução ou grupo de instruções que seleciona quais flags serão ativadas.

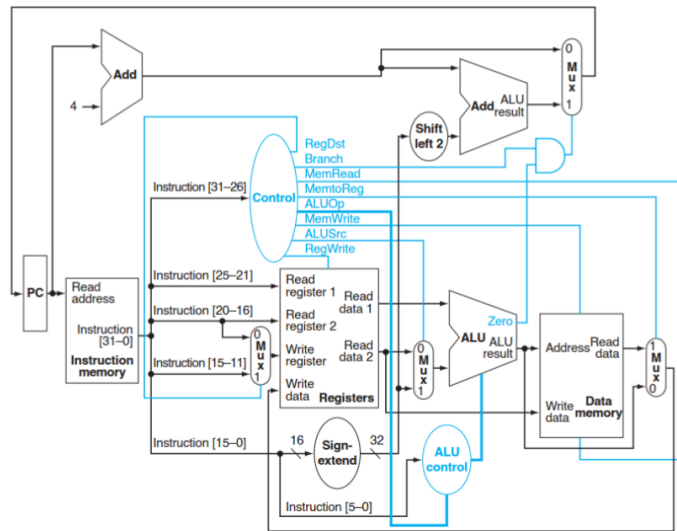


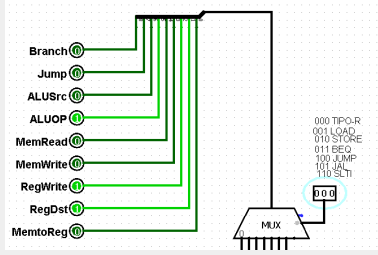
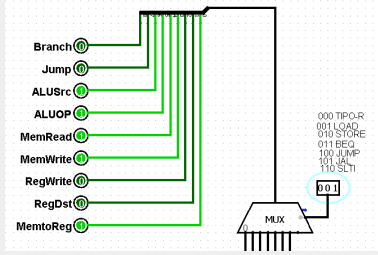
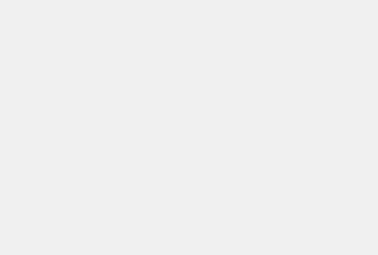
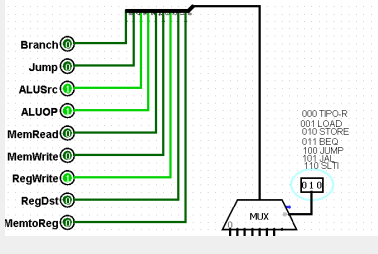
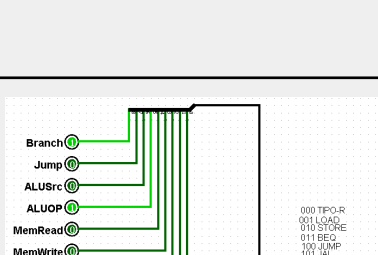
Figura 24 - Circuito encapsulado de uma CPU MIPS de 32 bits com a unidade de controle destacada

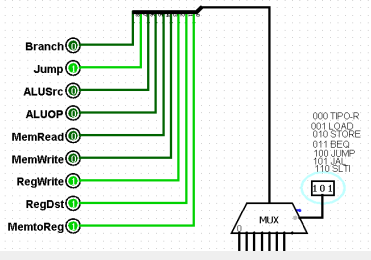
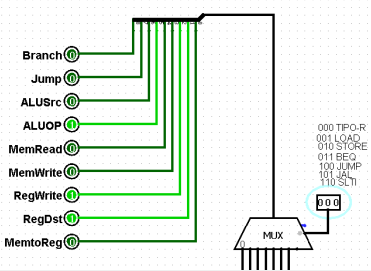
Primeiramente, é necessário definir qual instrução/grupo de instruções são utilizadas na UC (Unidade de Controle) do MIPS 16 bits e quais flags cada uma ativa.

Activity	Signal	Purpose
PC Update	Branch	Combined with a condition test boolean to enable loading the branch target address into the PC.
	Jump	Enables loading the jump target address into the PC (only appears in Figure 4.24 in Patterson and Hennessey).
Source Operand Fetch	ALUSrc	Selects the second source operand for the ALU (rt or sign-extended immediate field in Patterson and Hennessey).
ALU Operation	ALUOp	Either specifies the ALU operation to be performed or specifies that the operation should be determined from the function bits.
Memory Access	MemRead	Enables a memory read for load instructions.
	MemWrite	Enables a memory write for store instructions.
Register Write	RegWrite	Enables a write to one of the registers.
	RegDst	Determines how the destination register is specified (rt or rd in Patterson and Hennessey).
	MemtoReg	Determines where the value to be written comes from (ALU result or memory in Patterson and Hennessey).

Figura 25 - Definições das flags da UC

Tabela 13 - Testes unidade de controle

INSTRUÇÕES DO TIPO R	
<p>RegDst - Flag indica o registrador de destino; RegWrite - ativa o enable que controla a escrita de registradores; AluOp - especifica a operação a ser realizada pela Unidade Lógica e Aritmética.</p>	
LOAD	
<p>AluSrc - indica ao multiplexador próximo ao banco de registradores para selecionar o endereço vindo do extensor de sinal e não a saída de um registrador lido; MemtoReg - determina se o valor a ser escrito no registrador vem da memória e não do resultado da ULA RegWrite - ativa o enable que controla a escrita de registradores; MemRead - ativa o enable que controla a leitura de registradores; AluOp - especifica a operação a ser realizada pela Unidade Lógica e Aritmética.</p>	
STORE	
<p>AluSrc - indica ao multiplexador próximo ao banco de registradores para selecionar o endereço vindo do extensor de sinal e não a saída de um registrador lido; RegWrite - ativa o enable que controla a escrita de registradores; AluOp - especifica a operação a ser realizada pela Unidade Lógica e Aritmética.</p>	
BEQ	
<p>AluOp - especifica a operação a ser realizada pela Unidade Lógica e Aritmética; Branch - Combinado com um teste booleano, ativa o carregamento do desvio de endereço alvo para o PC.</p>	
JUMP	
<p>Jump - ativa o carregamento do desvio de endereço alvo para o PC.</p>	

JAL	
SLTI	

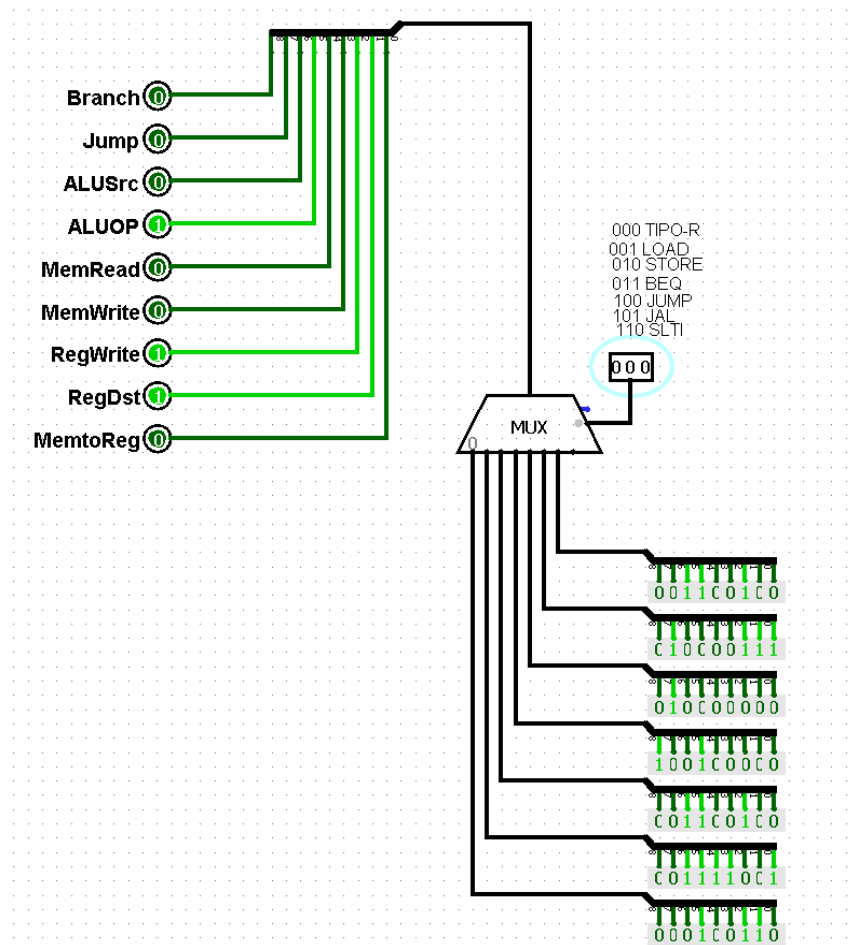


Figura 26 - Circuito UC implementado no Logisim

10. Máquina de estados

Uma máquina de estados é uma sistematização de um programa/sistema sequencial. Nesse tipo de programa, o output é determinado por portas e inputs internos que mudam com a subida de clock por meio de componentes flip-flops. Para implementar o sistema abaixo, é preciso antes de tudo compreendê-lo.

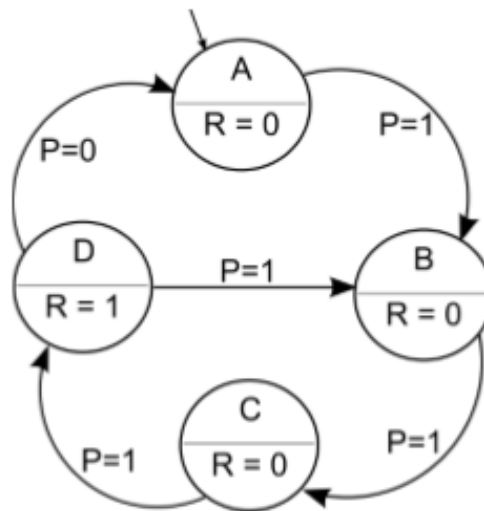
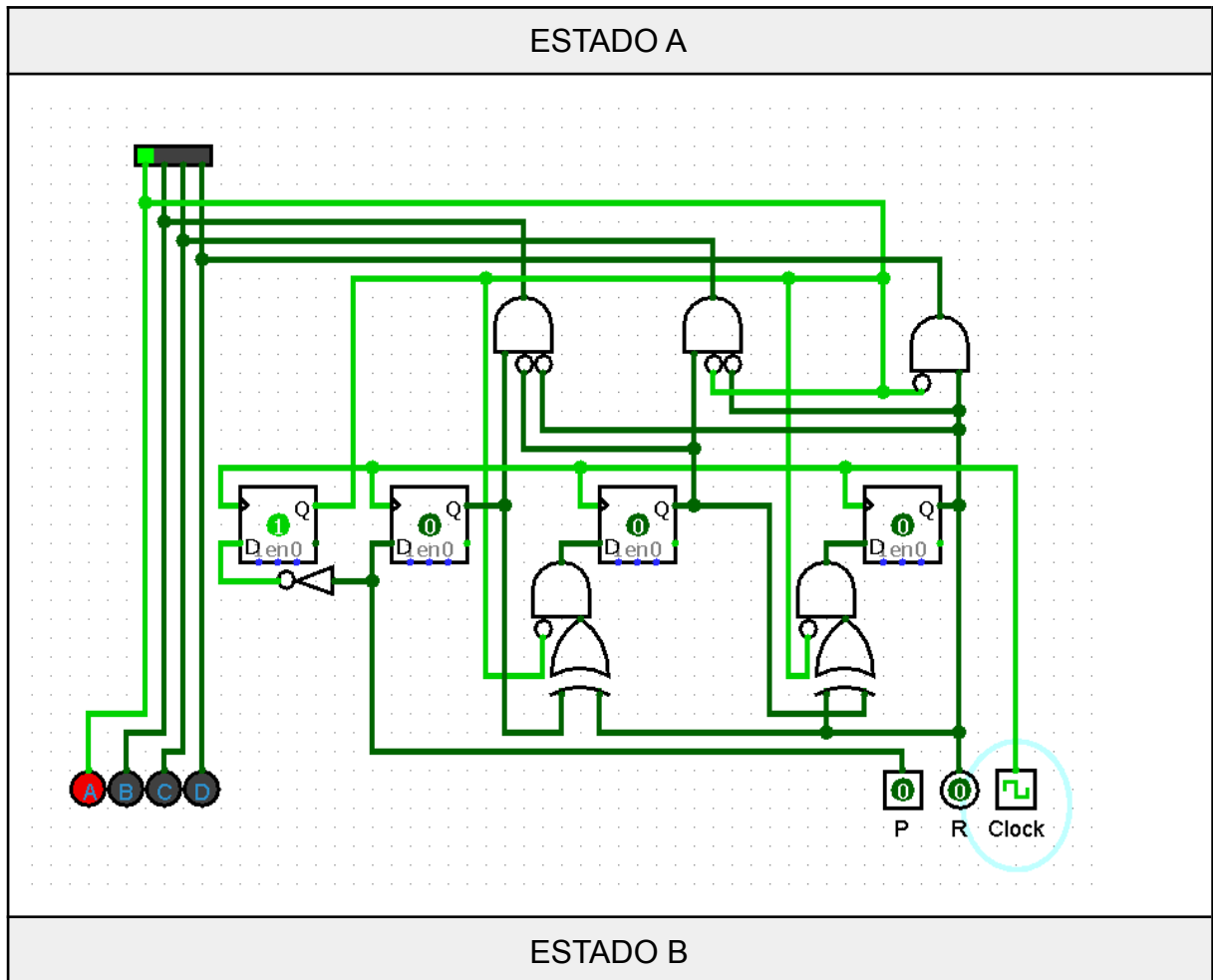


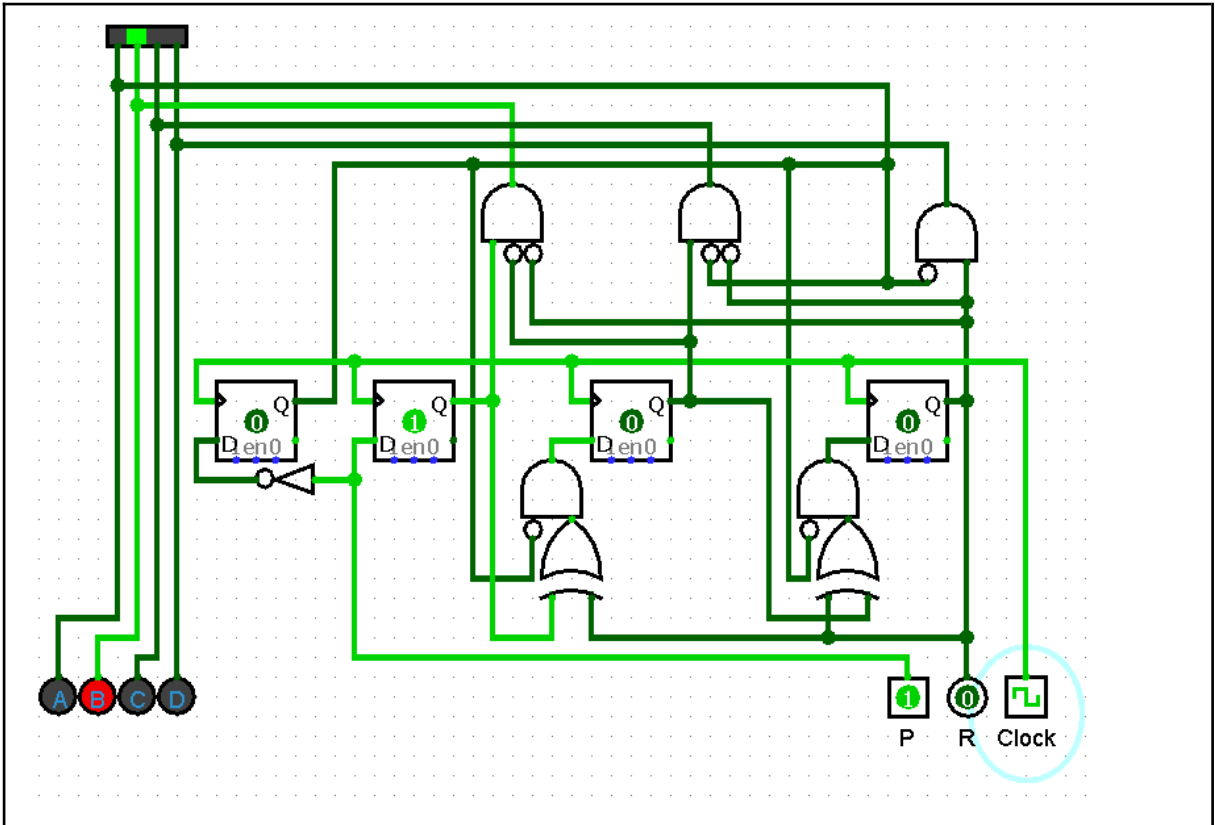
Figura 27 - Esquema de máquina de estados a ser implementado

São definidos quatro estados, A, B, C e D que se alternam com a mudança das variáveis P e R. Enquanto P=0, o estado sempre será A, no momento em que P=1, a máquina muda de A para B para C para D a cada subida de sinal do clock. Quando o estado é D, R=1 e o estado retorna para B se P continuar P=0.

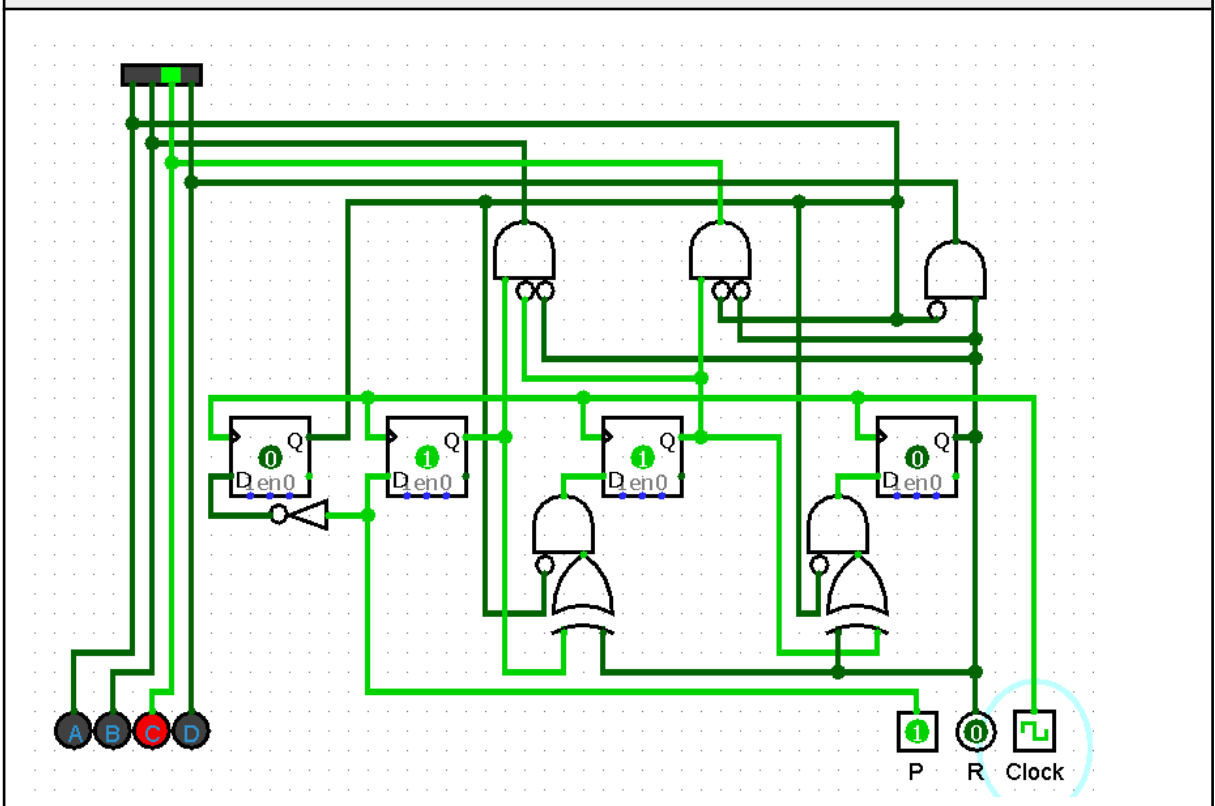
A seguir a representação dos quatro estados deste sistema após a subida do clock.

Tabela 14 - Estados da máquina de estados

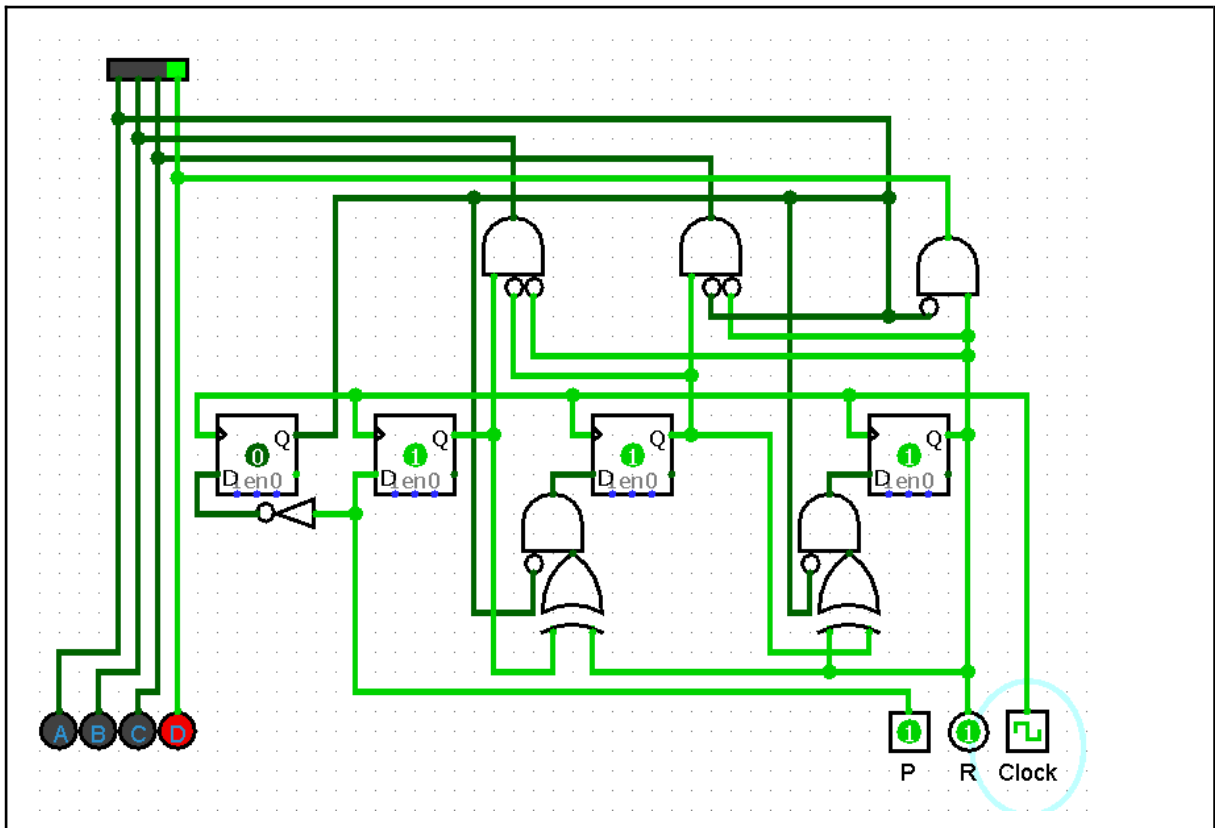




ESTADO C



ESTADO D



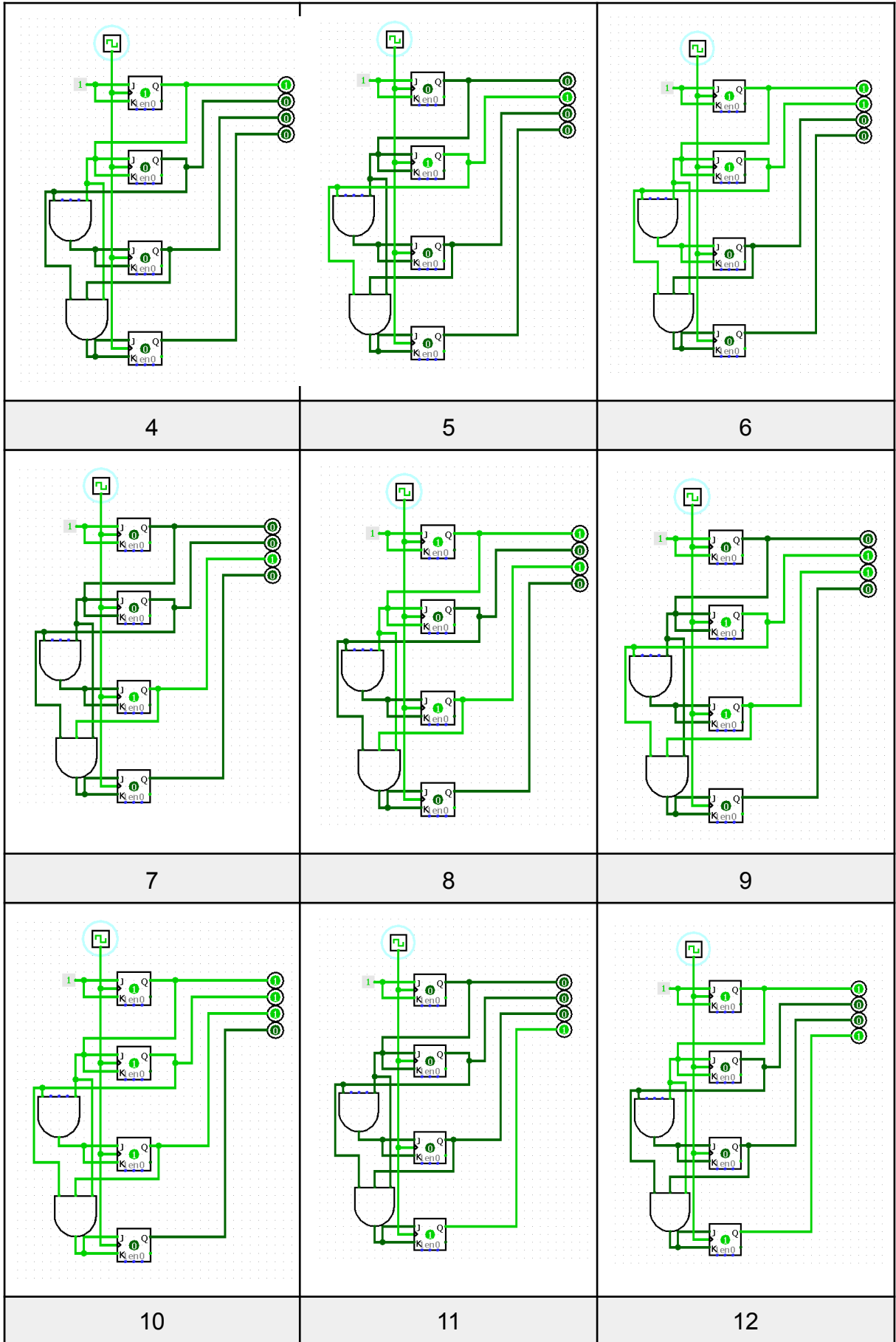
Se P não for desabilitado, os estados B, C, D ficam se alternando.

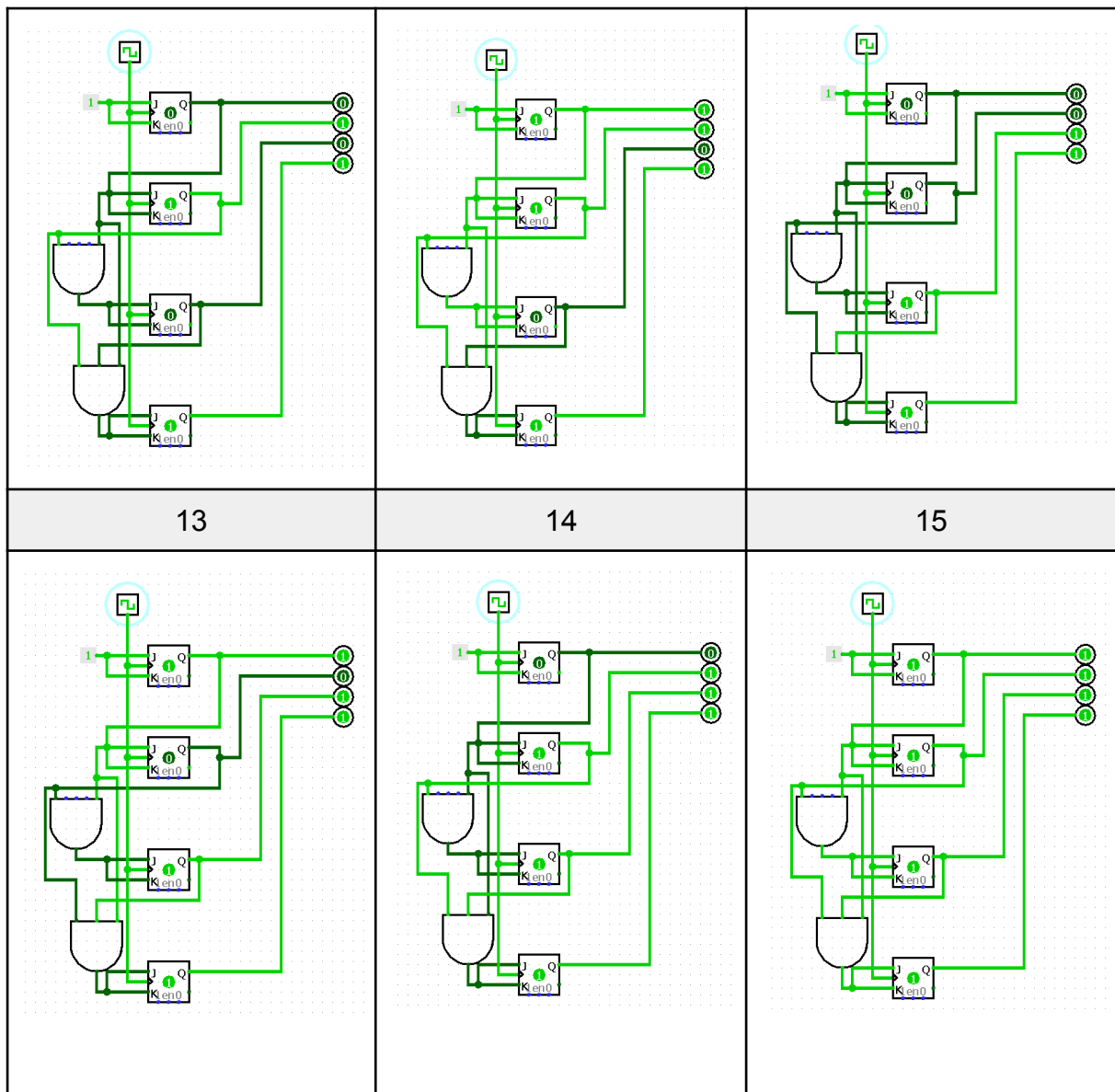
11. Contador síncrono

Um contador síncrono, diferente do assíncrono que verifica a onda de clock pelos flip-flops, tem a função de ser um contador que segue a tabela verdade. Ele é formado por Flip-Flops do tipo JK, portas AND, o clock e uma forma de output. Abaixo uma tabela mostrando um contador síncrono de quatro bits implementado no Logisim a cada subida de nível de clock.

Tabela 15 - Contador síncrono

1	2	3
---	---	---





12. Memória ROM

ROM é um acrônimo para Read Only Memory, que significa Memória apenas para leitura. Esse tipo de memória é não-volátil, ou seja, ela não perde os dados com o desligamento da energia. A seguir como ela foi implementada no Logisim.

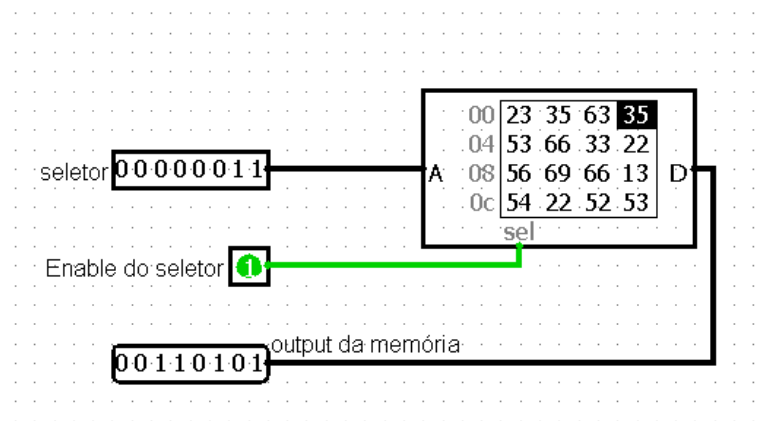


Figura 28 - Memória ROM implementada no Logisim

O input chamado de seletor é o que define qual célula de memória deve ser lida, na imagem definido como 11, ou seja, o quarto dado deve ser lido. O enable do seletor controla o ativamento da busca, como ele está com 1, o componente permanece ativado. Por último, a saída mostra o conteúdo da localidade de memória especificada no seletor.

A aplicação Logisim escreve a memória no sistema hexadecimal, por isso a localidade apresenta o dado 35 (hexadecimal) que correspondente a 110101 no sistema de numeração binário, explicando o porquê do output ser diferente do número apresentado dentro do componente da memória.

Olhando mais profundamente este circuito, pode-se construir uma memória simplificada a partir de células de memória.

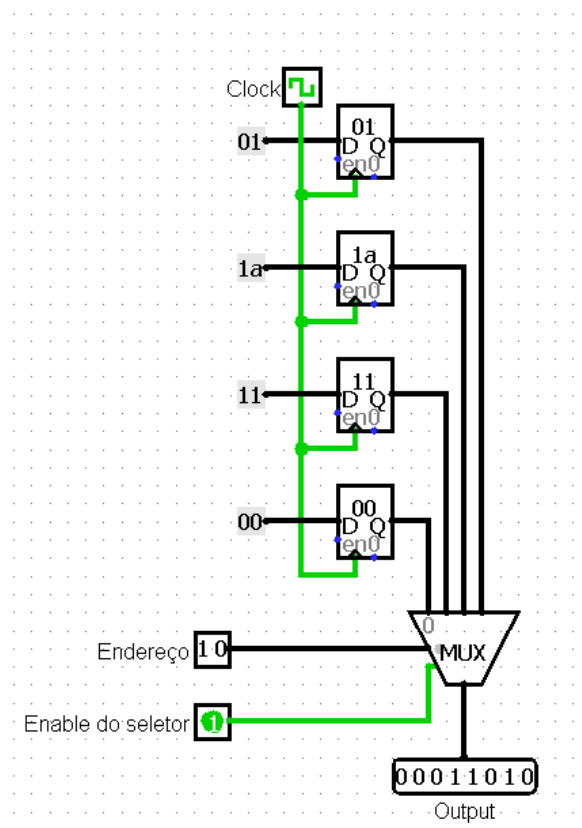


Figura 29 - Memória ROM simplificada implementada no Logisim

A mesma lógica é aplicada no sistema acima, o endereço é selecionado por uma entrada que alimenta o multiplexador que lê o conteúdo do endereço e retorna esse dado no output. Ressalta-se que para representar que a memória ROM é somente para leitura, os registradores têm constantes em hexadecimal como input.

13. Memória RAM

A memória RAM é uma memória de acesso aleatório, Random Access Memory, em inglês, que permite a leitura como a memória ROM e a escrita de arquivos. Ela é essencial para o bom funcionamento do computador por possibilitar contato imediato da CPU ao dados. Diferentemente da memória apenas de leitura, a RAM é volátil e tem suas informações perdidas com o desligamento da energia.

Dadas estas características, já é possível perceber algumas mudanças do circuito da memória apresentada no item 12. Acrescenta-se um input do dado a ser escrito, o chamado load que permite a escrita das informações fornecidas no endereço selecionado e o clear, que reseta os dados para 00.

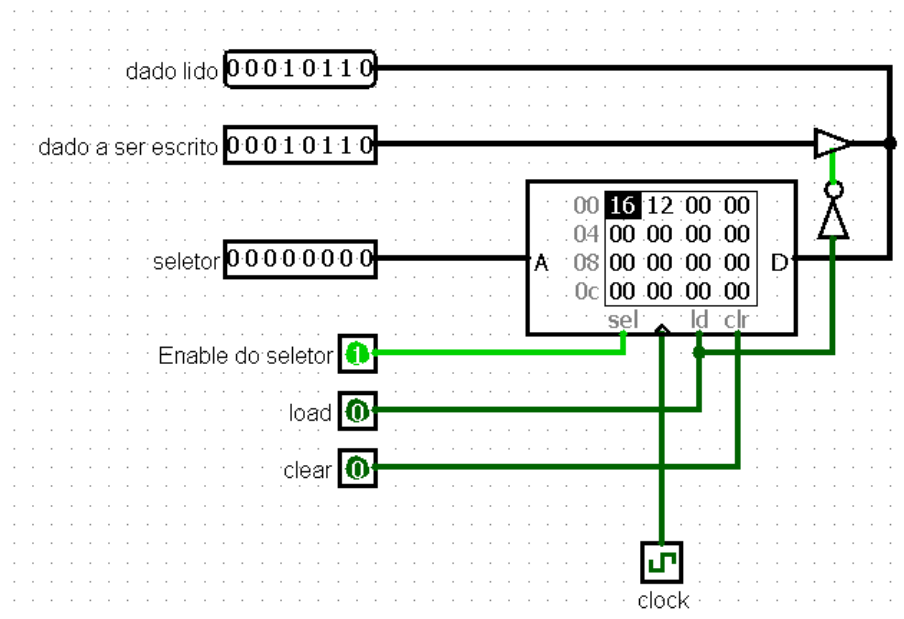


Figura 30 - Memória RAM implementada no Logisim

No teste da ilustração acima, o dado a ser escrito foi registrado no local 00 de acordo com o seletor (enable do seletor ativado, e com a subida do clock). Esse mesmo dado aparece como sendo lido pois o load foi ativado e desativado logo em seguida sem mudança no seletor. Após esse teste, o botão clear foi acionado, ele limpou os dados da memória, mas o output continuou pois não houve nova subida de clock.

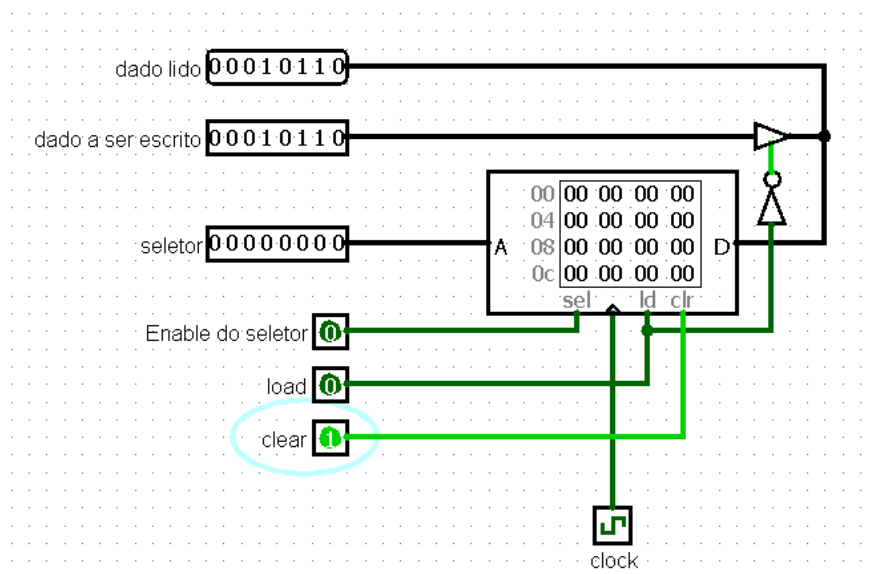


Figura 31 - Memória RAM implementada no Logisim: Clear

Assim como realizado com a memória ROM, foi feito um circuito simplificado da memória RAM. No teste mostrado abaixo o dado 10100 em binário foi registrado na célula de memória 01 dada pelo endereço como 14 (hexadecimal), logo com a elevação do sinal do clock.

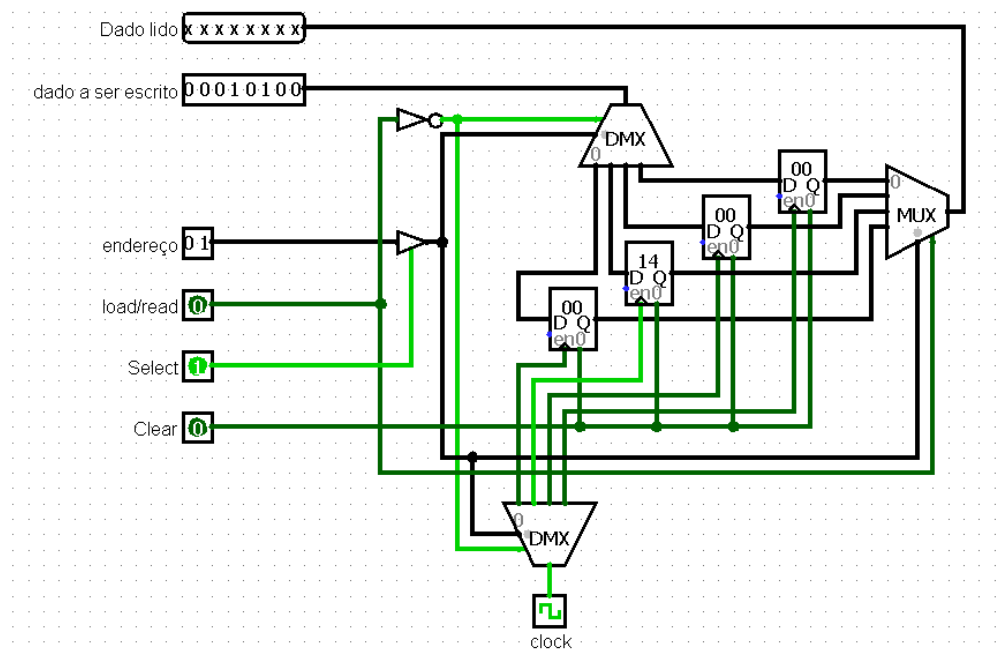


Figura 32 - Circuito de Memória RAM implementada no Logisim