

# MATA48/MATC90 UFBA

## Projeto - Logisim Lab

### Objetivos

- Pratique projetar e depurar circuitos lógicos digitais básicos no Logisim
- Ganhe mais experiência projetando e depurando circuitos com lógica combinacional e elementos com estado
- Ganhe experiência projetando FSMs e implementando-os como lógica digital

### Configurar

Extraia os arquivos do laboratório do repositório inicial do laboratório usando:

<https://www.dropbox.com/s/iu42tg9jkd0ak1l/proj-logisim.zip>

Todo o trabalho neste laboratório será feito a partir do programa de simulação de lógica digital **Logisim Evolution**, que está incluído nos arquivos iniciais do laboratório.

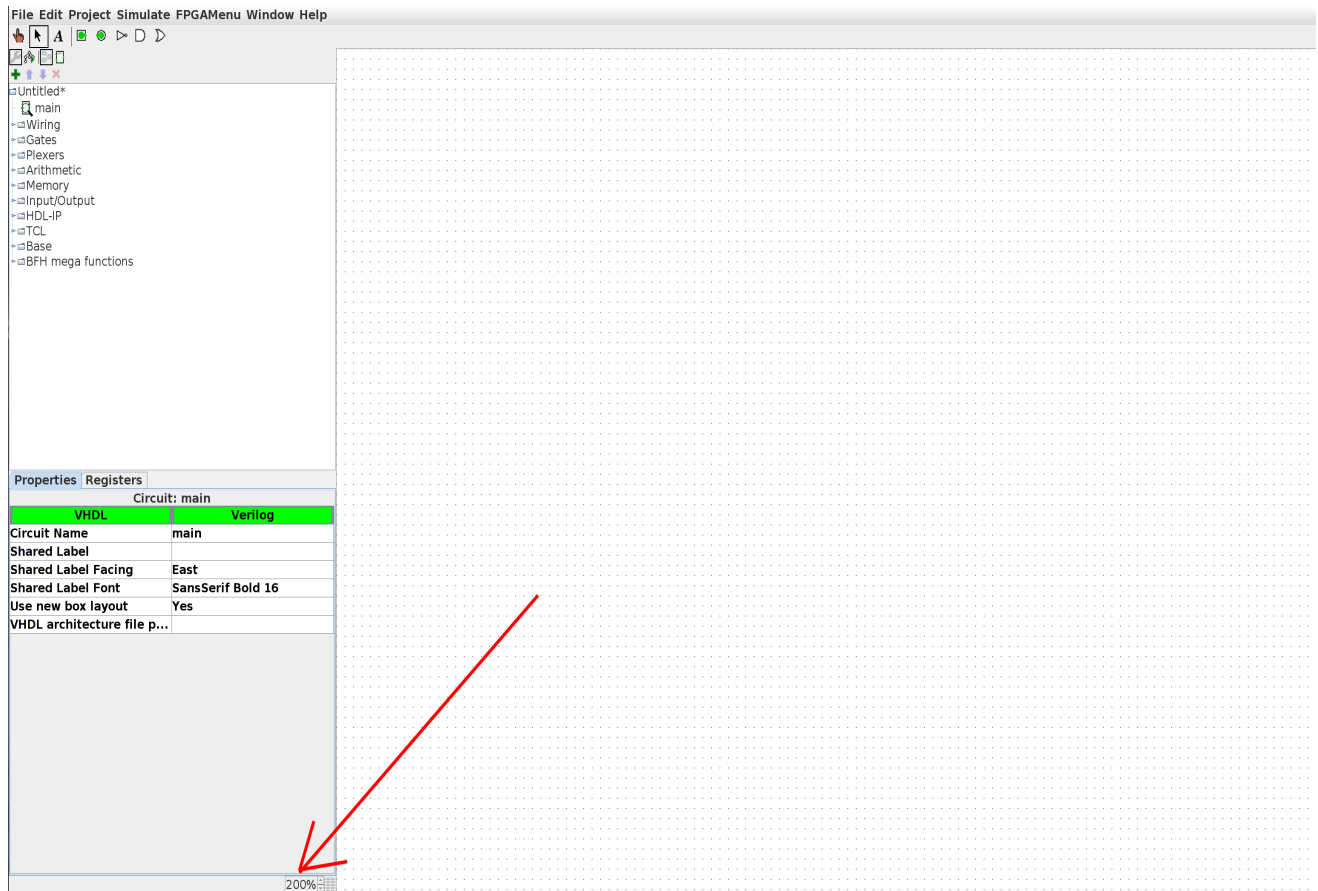
Você pode abrir o Logisim via:




```
java -jar logisim-evolution.jar
```

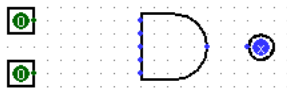
### Introdução


#### Parte 0: Aquecimento

Começaremos criando um circuito muito simples apenas para sentir como colocar portas e fios. Antes de começar, observe um recurso útil: a função de zoom! Está no canto esquerdo inferior e tornará sua vida muito mais fácil nas próximas semanas.

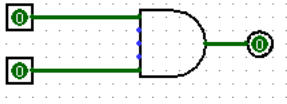


1.  Comece clicando no botão AND gate. Isso fará com que a sombra de uma porta AND siga o cursor. Clique uma vez na janela esquemática principal para colocar uma porta AND.
2.  Clique no botão Input Pin. Agora, coloque dois pinos de entrada em algum lugar à esquerda de sua porta AND.
3.  Clique no botão Output Pin. Em seguida, coloque um pino de saída em algum lugar à direita de sua porta AND. Seu esquema deve ser parecido com este neste ponto:



4.  Clique no botão da ferramenta Selecionar. Clique e arraste para conectar os pinos de entrada ao lado esquerdo da porta AND. Isso levará várias etapas, já que você só pode desenhar fios verticais e horizontais. Simplesmente desenhe um fio horizontalmente, solte o botão do mouse, clique e arraste começando do final do fio para continuar verticalmente. Você pode conectar o fio a qualquer pino na

porta AND no lado esquerdo. Repita o mesmo procedimento para conectar a saída da porta AND (lado direito) ao pino de saída. Depois de concluir essas etapas, seu esquema deve ser mais ou menos assim:



5. 🖱️ Finalmente, clique na ferramenta Poke e tente clicar nos pinos de entrada em seu esquema. Observe o que acontece. Isso corresponde ao que você acha que um AND Gate deve fazer? Observe que cutucar os próprios fios informa o valor da corrente naquele fio; isso será muito útil mais tarde, quando você construir circuitos mais complexos.

## Parte 1: Sub-circuitos

Assim como os programas C podem conter funções auxiliares, um esquema pode conter subcircuitos. Nesta parte do laboratório, criaremos vários subcircuitos para demonstrar seu uso.

**NOTA IMPORTANTE :** As diretrizes do Logisim Evolution dizem que você não pode nomear um subcircuito após uma palavra-chave (por exemplo, NAND ), também os nomes dos circuitos devem começar com “A-Za-z”, portanto, sem números.

Agora siga os passos abaixo:

1. Abra o esquema do Exercício 1 ( Arquivo-> Abrir-> ex1.circ ).
2. Abra o subcircuito vazio NAND1 clicando duas vezes no nome NAND1 no seletor de circuito no menu à esquerda. (observe o 1 no final; como há um componente chamado NAND , não podemos chamá-lo de NAND ).
1. Na nova janela esquemática que você vê, crie um circuito NAND simples com os 2 pinos de entrada no lado esquerdo e o pino de saída no lado direito. Faça isso sem usar a porta NAND integrada da pasta Gates (ou seja, use apenas as portas AND , OR e NOT fornecidas ao lado do ícone da ferramenta de seleção). Você pode alterar os rótulos das entradas e saídas selecionando a entrada / saída usando a ferramenta de seleção e alterando a propriedade Label no canto inferior esquerdo da janela.
1. Repetir estes passos para criar vários mais sub-circuitos: NOR , XOR , 2-a-1 MUX , e 4-a-1 MUX nos esqueletos dadas. Por favor, não mude os

nomes dos subcircuitos nem crie novos; você trabalha no circuito nomeado respectivamente ou então o autograder não funcionará corretamente. Não use nenhuma porta embutida diferente de AND , OR e NOT . Depois de construir um subcircuito, você pode (e é encorajado a) usá-lo para construir outros. Você pode fazer isso clicando e posicionando o subcircuito criado como faria com qualquer outro componente. Observação: para o MUX 4 para 1 , Sel0 e Sel1 correspondem aos bits 0 e 1 do seletor de 2 bits, respectivamente. Dica: tente escrever uma tabela verdade. Você também pode achar os slides de aula úteis para uma atualização sobre como construí-los. *Você pode querer considerar o uso de alguns de seus subcircuitos personalizados ao projetar os outros.*

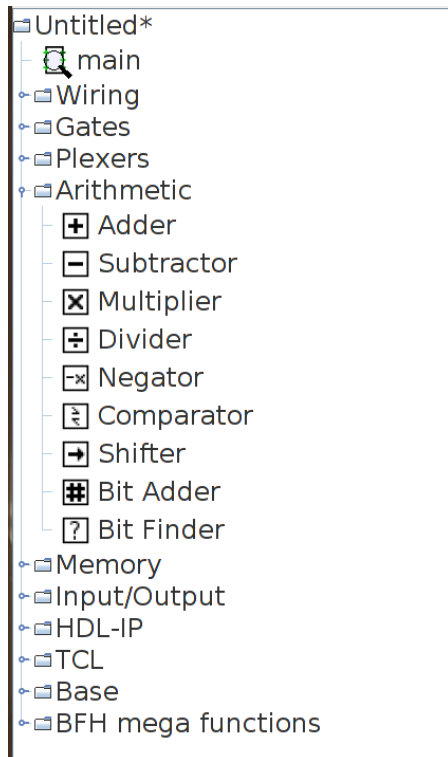
Neste ponto, certifique-se de estar confortável com o ambiente Logisim, criando subcircuitos e reutilizando tais circuitos em outros circuitos.

## Parte 2: Armazenamento de estado

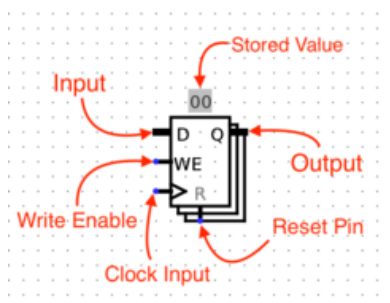
Vamos implementar um circuito que incrementa um valor ad infinitum. A diferença entre este circuito e os circuitos que você construiu para o laboratório até agora é que ele **armazenará** esse valor no **estado** de um **registrador** .

As etapas a seguir mostrarão como adicionar registradores ao seu circuito. Conclua as etapas:

1. Abra o esquema do Exercício 2 ( Arquivo-> Abrir-> ex2.circ ) e vá para o circuito AddMachine vazio.
2. Carga na Aritmética Biblioteca se ele já não está carregado (ir para Projeto-> Carga Library-> Construído em Biblioteca e selecione Aritmética ). Esta biblioteca contém elementos que realizarão operações matemáticas básicas. Ao carregar a biblioteca, o navegador de circuitos à esquerda terá uma nova pasta Aritmética .

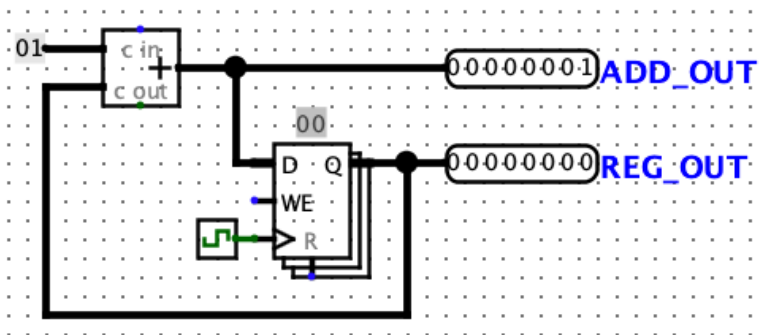


3. Selecione o subcircuito adicionador da biblioteca Aritmética e coloque o adicionador em seu subcircuito AddMachine .
4. Carga na memória Biblioteca se ele já não está carregado (ir para Projeto-> Carga Library-> Construído em Biblioteca e selecione Memória ). Esta biblioteca contém elementos de memória usados para manter o estado em um circuito. Uma nova pasta de memória aparecerá no navegador de circuitos.
5. Selecione o registro da pasta Memória e coloque um registro em seu subcircuito. Abaixo está uma imagem diagramando as partes de um registro.



6. Conecte um relógio ao seu registro. Você pode encontrar o elemento do circuito do relógio na pasta Wiring do navegador de circuitos.

7. Conecte a saída do somador à entrada do registrador e a saída do registro à entrada do somador.
  - Você pode obter um erro de “larguras incompatíveis” ao tentar conectar componentes. Isso significa que seu fio está tentando conectar dois pinos com larguras de bits diferentes. Se você clicar no somador com a ferramenta Seleção , notará que há uma propriedade Largura de bits de dados no campo inferior esquerdo da janela. Este valor determina o número de bits de cada entrada e saída que o somador possui. Altere este campo para 8 e o erro “larguras incompatíveis” deve ser resolvido.
8. Conecte uma constante 1 de 8 bits à segunda entrada do somador. Você pode encontrar o elemento de circuito Constante na biblioteca de fiação .
9. Conecte os dois pinos de saída ao seu circuito para que você possa monitorar o que sai do somador e do registrador. A saída do somador deve ser conectada a ADD\_OUT e a saída do registro a REG\_OUT . Assim, no final, seu circuito deve se parecer com o seguinte:



1. Agora comece a operar seu circuito indo para Simular→ Carrapatos Habilitados (ou Comando / Controle + K ). Seu circuito agora deve estar gerando um contador na forma binária.
2. Se você deseja executar seu circuito mais rápido, você pode alterar a frequência do tique em Simular→ Frequência do tique .

Neste ponto, certifique-se de que você está confortável com o projeto e a simulação de circuitos lógicos digitais simples no ambiente Logisim que usam uma combinação de *lógica combinacional* e *elementos de estado* (registradores).

### Parte 3: FSMs para lógica digital

Agora estamos prontos para fazer algo muito legal: traduzir um FSM em um circuito lógico digital.

Para aqueles que precisam de um lembrete, FSM significa Finite State Machine. Os FSMs rastreiam as entradas fornecidas, se movem entre os estados com base nessas entradas e produzem algo sempre que algo é inserido.

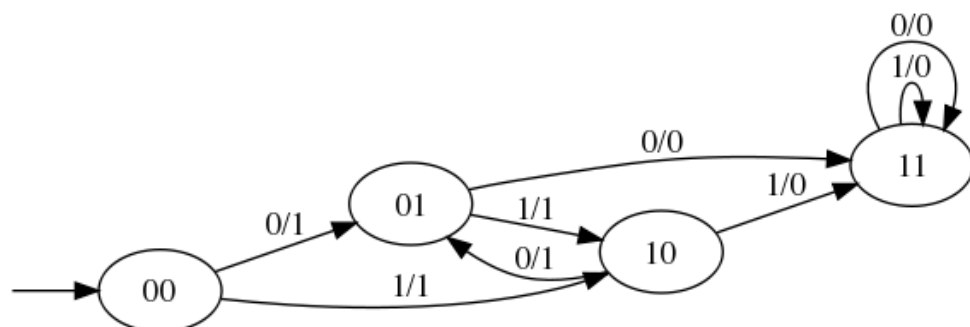
Usamos um registro para armazenar o estado do FSM em que estamos atualmente e uma lógica combinacional para mapear a entrada FSM e o estado do registro atual para a saída FSM e o próximo estado do registro.

Item de ação

Carregue o arquivo inicial `ex3.circ` fornecido no Logism. Modifique os subcircuitos `StateBitZero` e `StateBitOne` deste circuito para implementar o seguinte FSM:

**Se dois um em uma linha ou dois zeros em uma linha já foram vistos, produza zeros para sempre. Caso contrário, imprima um.**

1. Observe que o FSM é implementado pelo diagrama a seguir (os quatro nomes de estado `00`, `01`, `10`, `11` são apenas nomes escritos em binário - eles não têm relação direta com os zeros reais e uns da entrada / saída do FSM). Dedique algum tempo para entender como este diagrama implementa o FSM:



2. Observe que o seguinte é uma tabela verdade para o FSM (convença-se disso):

st1	st0	entrada	próximo st1	próximo st0	saída
0	0	0	0	1	1
0	0	1	1	0	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	0

3. Fornecemos um circuito inicial Logisim para começar em `ex3.circ` .
4. Observe que o nível superior do circuito parece quase exatamente igual ao nosso circuito adicionador anterior, mas agora há um bloco `FSMLogic` em vez de um bloco adicionador. `FSMLogic` é o bloco lógico combinacional para este FSM. Cuidamos do bit de saída para você, pois é o mais complicado de simplificar e implementar. Você deve completar o circuito através do preenchimento do `StateBitOne` e `StateBitZero` subcircuits, que produz os próximos bits de estado.

Neste ponto, você deve estar mais familiarizado com o projeto e implementação de FSMs e com a relação próxima entre FSMs e lógica digital.

## Logisim Avançado

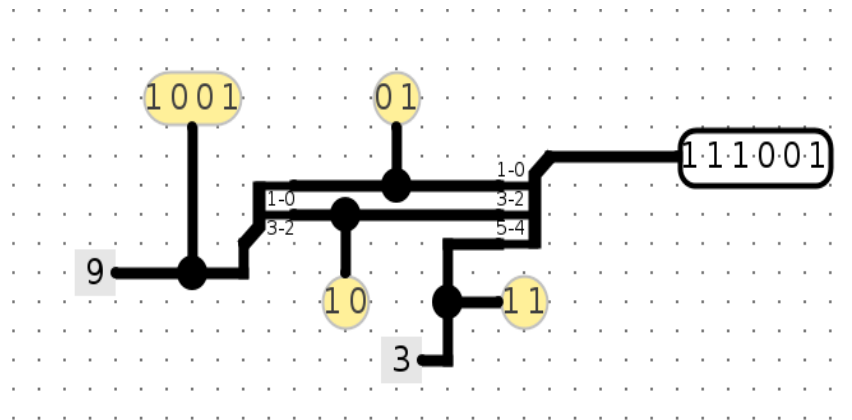
Sinta-se à vontade para fazer cada parte como subcircuitos separados no mesmo arquivo Logisim. As partes a seguir apresentarão a você técnicas / conceitos mais avançados no Logisim.

Aqui estão três recursos do Logisim que devem economizar muito tempo e deixar seus circuitos muito mais limpos.



## Divisores (“Splitters”)

Divisores permitem que você pegue um valor de vários bits e o divida em partes menores ou (apesar do nome) combine vários valores que são um ou mais bits em um único valor. Aqui, dividimos o número binário de 4 bits 1001 em 10 e 01 e , em seguida, o recombina com 11 no número final de 5 bits 111001 :



Clique em um divisor para obter seu menu na barra lateral. Você pode usar este menu para determinar o número de braços em seu divisor e quantos bits devem ser colocados em cada braço. Para o circuito acima, o menu do divisor esquerdo se parece com este:

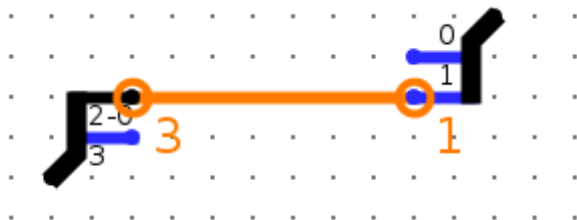
Properties Registers	
Selection: Splitter	
VHDL	Verilog
Facing	East
Fan Out	2
Bit Width In	4
Appearance	Left-handed
Bit 0	0 (Top)
Bit 1	0 (Top)
Bit 2	1 (Bottom)
Bit 3	1 (Bottom)

Enquanto o menu do divisor direito se parece com este:

Properties	Registers
Selection: Splitter	
VHDL	Verilog
Facing	West
Fan Out	3
Bit Width In	6
Appearance	Left-handed
Bit 0	0 (Top)
Bit 1	0 (Top)
Bit 2	1
Bit 3	1
Bit 4	2 (Bottom)
Bit 5	2 (Bottom)

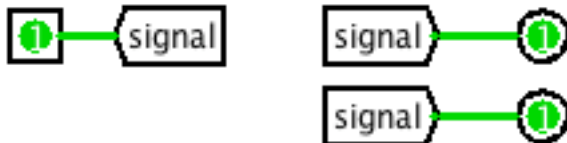
**Observe que existe uma opção chamada de facing** (frente) . Você pode usar isso para girar seu divisor. Acima, veja que o divisor à direita está voltado para o oeste, enquanto o divisor à esquerda está voltado para o leste.

Se você vir um fio de erro laranja, isso significa que a largura de entrada não corresponde à largura de saída. Certifique-se de que, se estiver conectando dois componentes com um fio, tenha definido corretamente a largura de bits no menu desse componente.

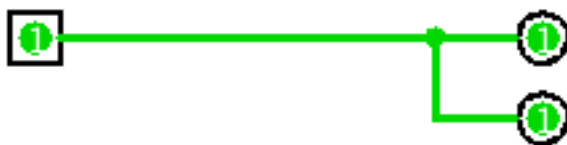


## Túneis

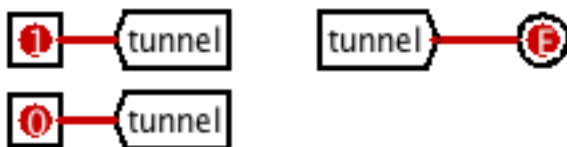
Um túnel permite que você desenhe um “fio invisível” para unir dois pontos. Os túneis são agrupados por rótulos com distinção entre maiúsculas e minúsculas fornecidos a um fio. Eles são usados para conectar fios como:



Que tem um efeito como o seguinte:



Alguns cuidados devem ser tomados em relação a quais fios estão conectados com túneis aos quais outros fios, como neste caso:



O que, por sua vez, tem o seguinte efeito:



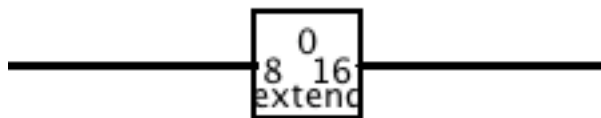
Recomendamos *fortemente* que você use túneis com Logisim, porque eles tornam seus circuitos muito mais limpos e, portanto, mais fáceis de depurar.

## Extensores

Ao alterar a largura de um fio, você deve usar um extensor de bits para maior clareza. Por exemplo, considere a seguinte implementação de extensão de um fio de 8 bits em um fio de 16 bits:



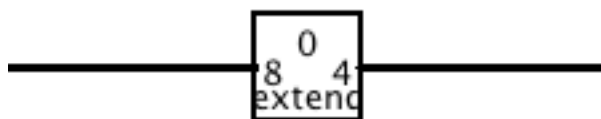
Considerando que o seguinte é muito mais simples, fácil de ler e menos sujeito a erros:



Além disso, considere o caso de jogar fora os bits. Neste exemplo, um fio de 8 bits está sendo convertido em um fio de 4 bits descartando os outros bits:



Apesar das implicações de seu nome, um extensor de bits também pode fazer a mesma operação:



## Parte 4: prática com divisores

Vamos construir um circuito que manipula um número de 8 bits.

Conclua as etapas a seguir para criar o circuito divisor. Quando você tiver concluído o circuito, responda à pergunta na etapa 11.

1. Abra o esquema do Exercício 4 ( Arquivo-> Abrir-> ex4.circ ) e vá para o circuito Split vazio.
1. Vá para a pasta Wiring e selecione o circuito Divisor . Este circuito pegará um fio e o dividirá em um conjunto de fios de largura menor. Por outro lado, ele também pode pegar muitos conjuntos de fios e combiná-los em um.
2. Altere a propriedade Bit Width In (largura do barramento) para 8 e a propriedade Fan Out (número de ramificações) para 3 . Seu divisor agora deve ter a seguinte aparência:



3. Agora, selecione quais bits enviar para qual parte do seu ventilador. O bit menos significativo é o bit 0 e o bit mais significativo é o bit 7 . O bit 0 deve sair no braço do ventilador 0 , os bits 1 , 2 , 3 , 4 , 5 e 6 devem sair no braço do ventilador 1 e o bit 7 deve sair no braço do ventilador 2 . Para sua informação: a opção Nenhum significa que o bit selecionado não sairá em NENHUM dos braços do ventilador.
4. Direcione In1 para o divisor. Anexe uma porta AND de 2 entradas aos braços do ventilador 0 e 2 e direcione a saída da porta AND para Out1 .
5. Agora, interprete a entrada como um número de “sinal e magnitude”. Coloque portas lógicas e outros circuitos para preparar Out2 para ser o valor negativo de “sinal e magnitude” da entrada. Sinal e magnitude são uma forma alternativa de representar valores com sinais - como o complemento de 2, mas mais simples! A lógica combinatória deve ser direta.
6. Precisaremos de outro divisor para recombina os ventiladores em um único barramento de 8 bits. Coloque outro divisor com as

propriedades adequadas (largura de bits In: 8, Fan Out: 3, larguras de leque corretas). Brinque com as propriedades Facing e Appearance para tornar seu circuito final o mais limpo possível. Neste ponto, Out2 deve ser a negação da entrada (interpretando a entrada como um valor de “sinal e magnitude”).

7. Responda a seguinte questão:

Se decidirmos pegar a entrada e interpretá-la como um número de complemento de 2, quais entradas produzirão  $\text{Out1} = 1$  ? *Dica: o que o primeiro e o último bits de um número de complemento de 2 sendo 1 dizem a você sobre o número?*

#### Parte 5: girar para a direita

Com seu conhecimento de divisores e seu conhecimento e experiência com multiplexadores, você está pronto para implementar um bloco lógico combinacional não trivial: `rotr`, que significa “Girar para a direita”. A ideia é que `rotr A, B` irá “girar” o padrão de bits da entrada `A` para a direita por `B` bits. Portanto, se `A` fosse `0b1011010101110011` e `B` fosse `0b0101` (5 em decimal), a saída do bloco seria `0b1001110110101011`. Observe que os 5 bits mais à direita foram girados para fora da extremidade direita do valor e de volta para a extremidade esquerda. Em RTL, a operação seria algo como  $R = A \gg B \mid A \ll (16 - B)$ .

Agora implemente um subcircuito denominado `rotr` com as seguintes entradas:

- `A` (16 bits), a entrada de 16 bits a ser girada
- `B` (4 bits), a quantidade de rotação (por que 4 bits?) Você pode encontrar o subcircuito inicial em `ex5.circ`.

A saída deve ser girada `A` para a direita pelas posições dos bits `B`, conforme descrito acima. Você **NÃO tem** permissão para usar shifters Logisim em sua solução, embora todas as outras lógicas combinacionais (MUXes, constantes, portas, somadores, etc.) sejam permitidas. Os MUXes integrados do Logisim (localize-os no menu `Plexers`) podem ser especialmente úteis. Sua solução não deve envolver um relógio ou qualquer elemento com clock, como registradores.

**Dica 1** : antes de iniciar a fiação, você deve pensar com muito cuidado sobre como pode decompor esse problema em problemas menores e juntá-los. Você deve se

sentir à vontade para usar subcircuitos ao implementar `rotr` . Se não o fizer, espere se arrepender.

**Dica 2** : Só porque fornecemos uma representação RTL, não significa que é a melhor maneira de olhar para este problema. Pense nos bits de entrada de `B` e pense em como usar divisores com eficácia! Você pode fazer algo com a forma binária? Lembre-se de por que o binário é bom para uso em computadores: um `1` é fácil de representar como um sinal LIGADO e um `0` é fácil de representar um sinal DESLIGADO . Digamos que queremos girar 9 vezes. 9 é `1001` em binário ou  $1 * 8 + 0 * 4 + 0 * 2 + 1 * 1$  . Você pode usar isso para fazer um circuito mais limpo? Usar os circuitos de podridão \* que fornecemos é uma boa ideia para manter as coisas limpas!

## Testando

Para testar, execute o script de teste por meio de:

```
./test.sh
```

Como o Logisim já estará em execução em uma janela de terminal, certifique-se de abrir uma nova janela para executar o script de teste. Se disser que você não tem permissão, execute:

```
chmod +x test.sh
```

Este script copiará seus circuitos em um chicote de teste, executará seus circuitos em entradas diferentes e comparará seus resultados aos nossos. Portanto, não toque em nada na pasta de teste . No entanto, você é mais que bem-vindo para verificar os circuitos envolvidos no teste de seu código.

## Submissão

Envie seus circuitos modificados (apenas os arquivos `ex{1,2,3,4,5}.circ`) usando o Autograder do Gradescope, contendo os circuitos `MUX 2` para `1` , `MUX 4` para `1` , `AddMachine` , `StateBitZero` , `StateBitOne` , `Split` e `rotr` .