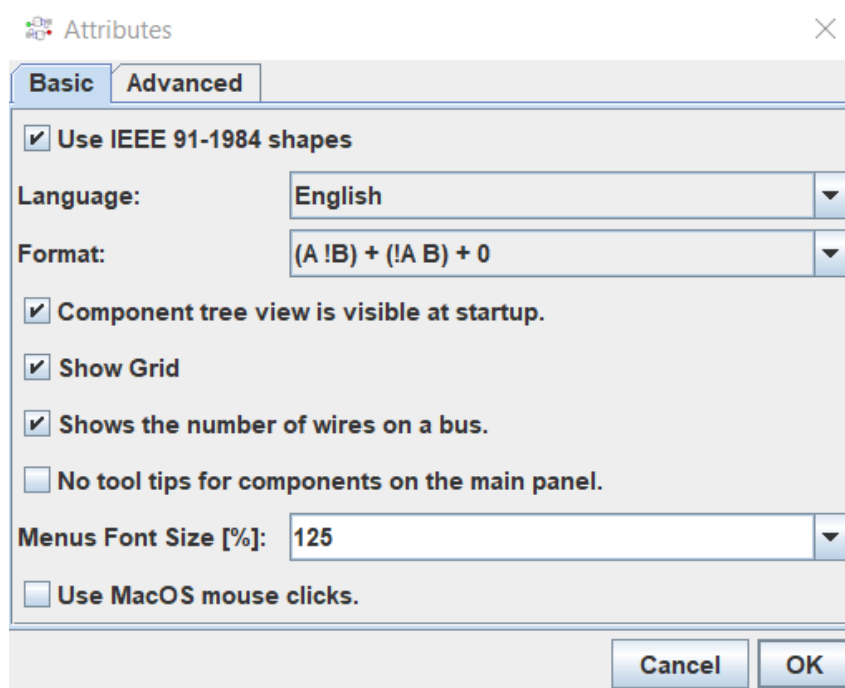


Simulação de Circuitos Lógicos com DIGITAL

O simulador DIGITAL é uma ferramenta *open-source* para simulação de circuitos digitais simples. O programa pode ser obtido em <https://github.com/hneemann/Digital> e usado em qualquer sistema que suporte a linguagem Java.

1 Configuração

Antes de iniciar o desenho de circuitos lógicos, configurar o sistema usando **Edit > Settings**. A configuração deve ficar como indicado a seguir.



Confirmar que a opção **View > Component Tree View** está selecionada.

2 Projeto de um circuito combinatório simples

Para começar, vamos implementar um circuito com 4 bits de entrada e uma saída. A saída vem a 1 apenas quando as entradas são todas iguais.

Designando as entradas por X_1, X_2, X_3, X_4 , a saída F é dada por

$$F(X_1, X_2, X_3, X_4) = X_1 \cdot X_2 \cdot X_3 \cdot X_4 + \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot \overline{X_4}.$$

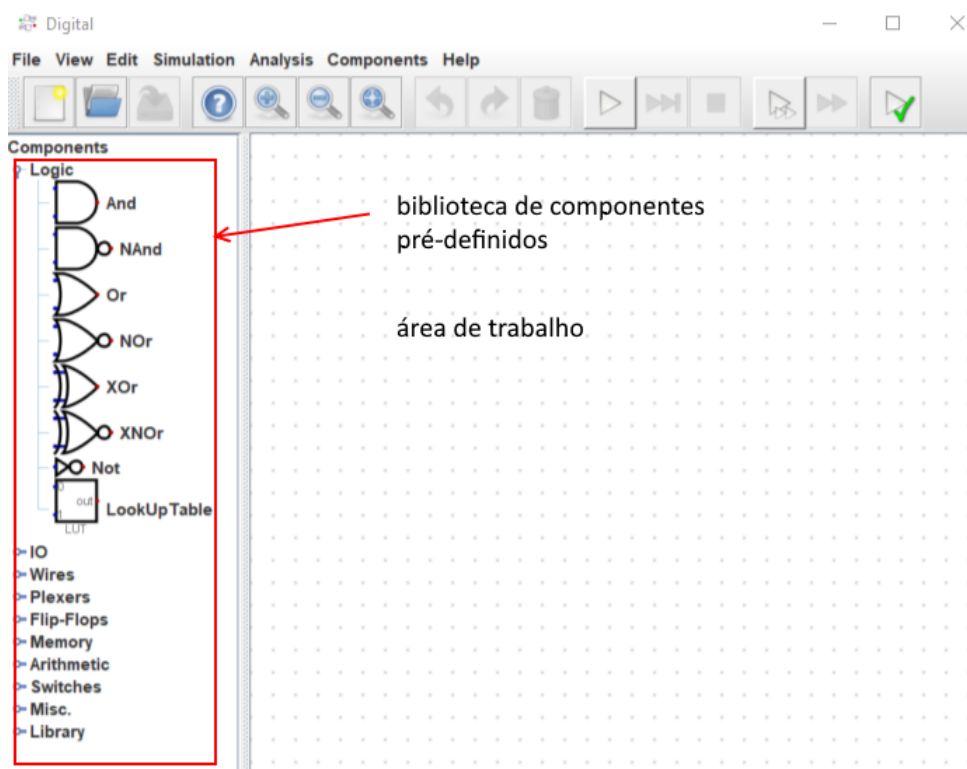
Justificar a equação de F .

A implementação direta da expressão obriga a negar todos os sinais de entrada. A seguinte versão equivalente permite evitar isso:

$$F(X_1, X_2, X_3, X_4) = X_1 \cdot X_2 \cdot X_3 \cdot X_4 + \overline{(X_1 + X_2 + X_3 + X_4)}.$$

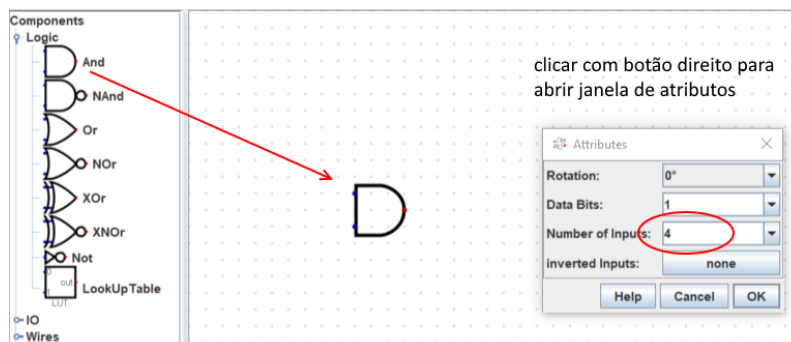
Justificar a transformação. Porque é que a transformação é vantajosa para a implementação?

Para criar um circuito, usar **File > New**. A janela apresentada é a seguinte

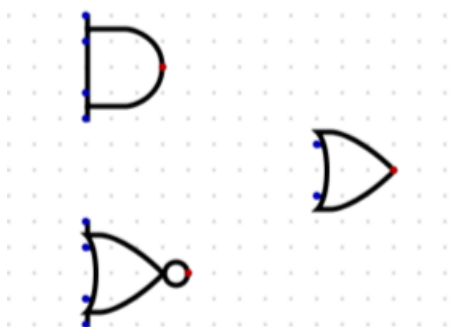


Para criar o circuito, procede-se da seguinte forma:

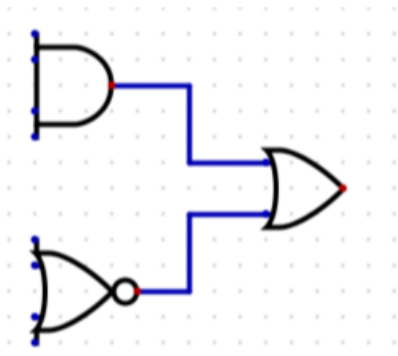
1. A figura seguinte mostra como colocar uma porta AND4 no circuito,



2. Acrescentar as restantes portas lógicas (NOR4 e OR2) conforme indicado na figura seguinte.

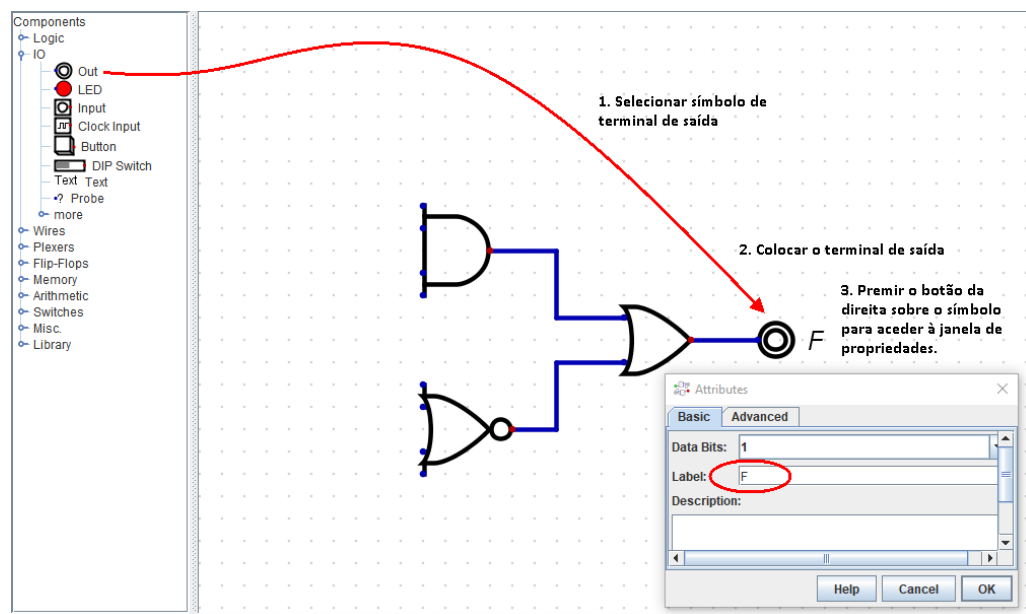


3. Para interligar duas portas, basta clicar e largar a saída de uma porta lógica (ponto vermelho), clicar em locais intermédios e terminar na entrada de outra porta lógica (ponto azul). Regra geral: um ponto vermelho pode apenas ser ligado a um ou vários pontos azuis.

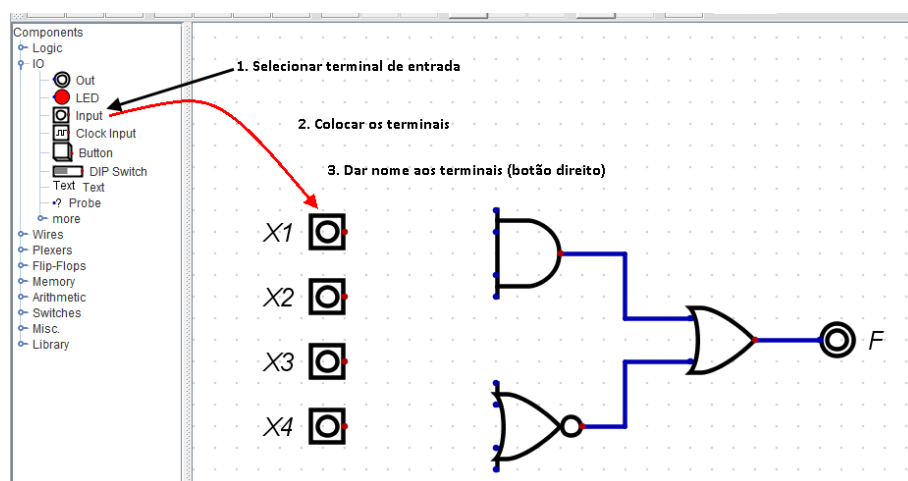


Alguns comandos interativos para modo de edição:

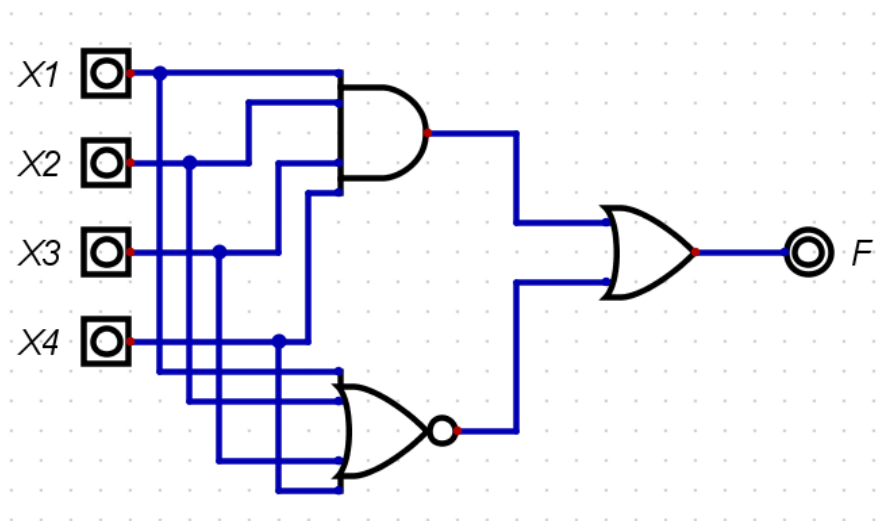
- Desenhar ligações: clicar com botão esquerdo nos pontos desejados; terminar: botão direito.
 - Selecionar uma ligação: **Ctrl**+botão esquerdo do rato.
 - A tecla **Esc** interrompe qualquer comando.
 - Rodar botão do rato para a frente: *zoom in*; para trás: *zoom out*.
 - Selecionar componentes: seleção "retangular"arrastando o cursor com botão esquerdo premido.
 - Movimentar o desenho: premir botão direito numa zona vazia e arrastar.
 - Aceder à janela de propriedades de um componente: clicar com o botão direito do rato sobre o componente.
 - Colocar nova instância do mesmo componente: tecla **L**.
 - Rodar o componente em colocação ou selecionado: tecla **R**.
4. O passo seguinte consiste em acrescentar *a terminal de saída* e ligá-lo conforme mostrado na figura da página seguinte.



5. Para colocar os terminais de entrada, procede-se de forma semelhante, conforme indicado na figura seguinte (usando terminais de entrada).

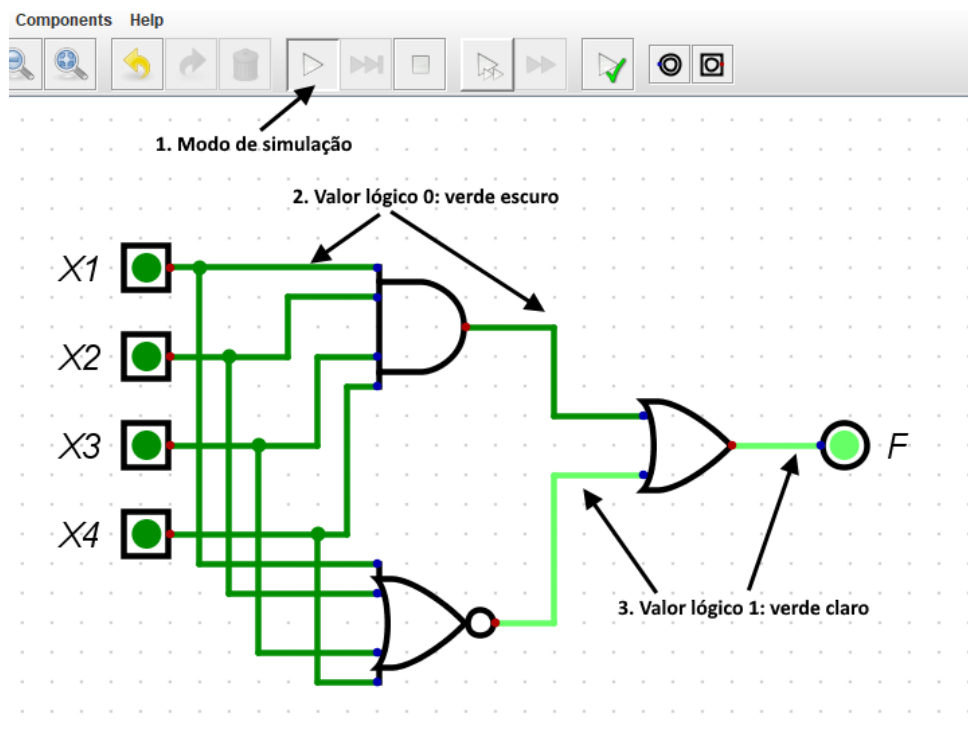


6. Para completar o circuito, basta ligar os terminais de entrada às entradas das portas lógicas, por forma a obter-se a função desejada (cf. figura seguinte). Gravar o circuitos usando o nome **detect** (ficheiro detect.dig).



3 Simulação e verificação

Para simular o funcionamento do circuito, é preciso ativar **o modo de simulação** usando o menu **Simulation** > **Start simulation**. Conforme se pode ver na figura, os valores das entradas estão, por omissão, a zero.

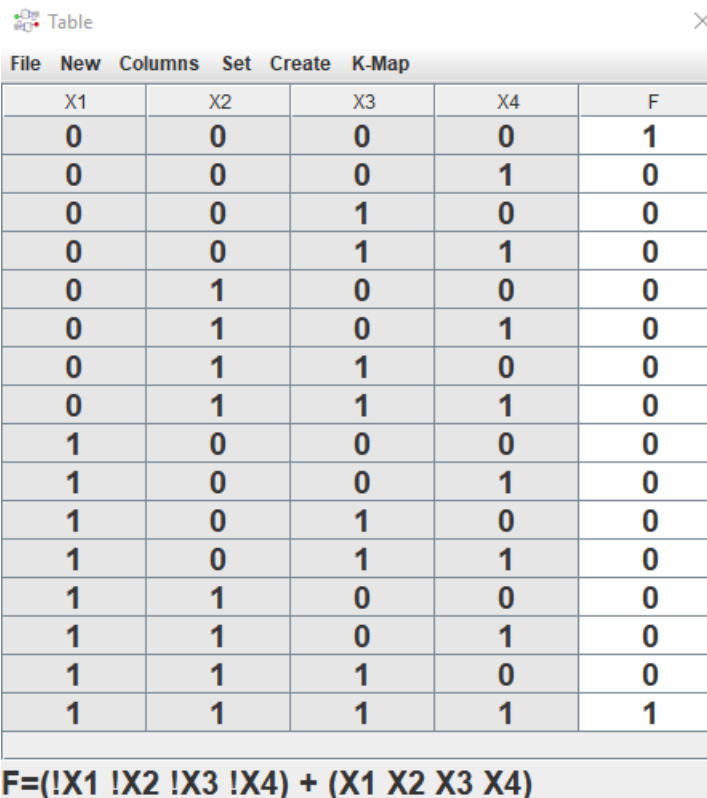


O valor de cada linha também é indicado quando se deixa repousar o cursor sobre ela.

Para alterar os valores de entrada, basta clicar sobre a entrada correspondente. O estado do circuito é alterado imediatamente para refletir a alteração de entrada.

Para verificar completamente o funcionamento correto do circuito é preciso analisar o seu comportamento para todos os padrões de entrada. Neste caso, existem $2^4 = 16$ padrões de entrada diferentes.

Uma forma alternativa de validar o circuito aproveita a capacidade que o sistema DIGITAL tem de analisar o circuito. Para proceder à análise, usar o menu **Analysis** **>** **Analysis**. A janela de análise tem o seguinte aspeto:



X1	X2	X3	X4	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

F = (!X1 !X2 !X3 !X4) + (X1 X2 X3 X4)

A tabela de verdade mostra claramente que a saída F apenas assume o valor 1 nas condições desejadas ($X_1 = X_2 = X_3 = X_4$).

Na parte inferior da janela, podemos ver que a função determinada pelo DIGITAL a partir do circuito é igual à que se pretendia implementar.

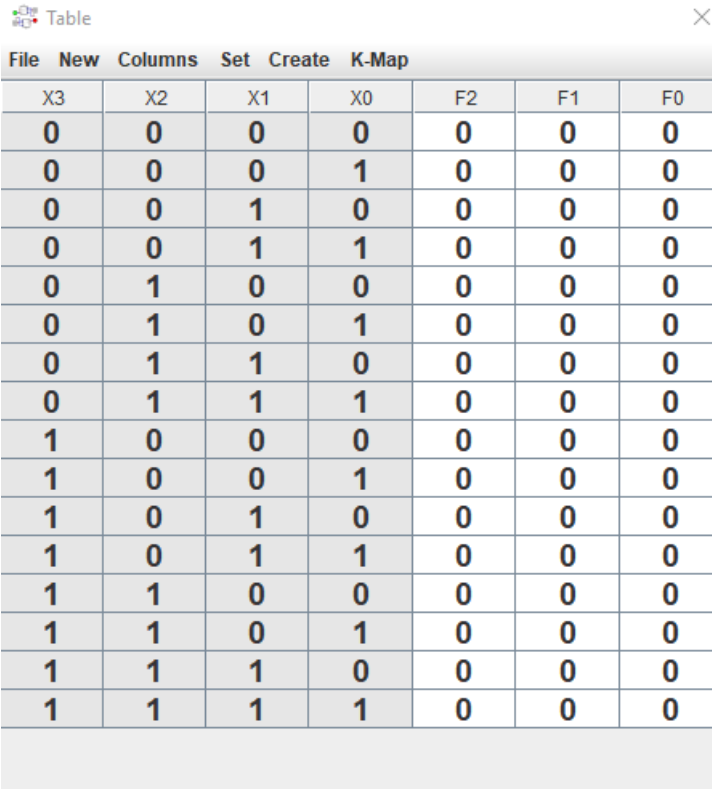
4 Outras formas de especificar circuitos

Pretende-se realizar um circuito que indique o número de bits iguais a 1 presentes à sua entrada. O número de entradas deverá ser 4 (X_0, X_1, X_2, X_3), representando o número binário $x_3x_2x_1x_0$. O número de saídas é 3 (F_2, F_1 e F_0), devendo ser interpretadas como representando o número binário $f_2f_1f_0$.

Em vez de especificar o circuito através de portas lógicas interligadas, vamos especificar a tabela de verdade correspondente.

1. Começar por criar um novo circuito usando **File** **>** **New**.
2. Usar o comando **Analysis** **>** **Synthesize** para abrir a ferramenta de análise/geração de circuitos combinatórios.

3. Na nova janela, usar o comando **Columns >> Add Input Variable** para criar a quarta variável de entrada. (Para já ignorar a janela intitulada "All possible solutions".)
4. Usar o comando **Columns >> Add Output Columns** duas vezes para criar mais duas saídas.
5. Clicar com o botão da direita sobre o cabeçalho de cada coluna para alterar a designação das entradas e das saídas. A tabela deve ter o seguinte aspeto:



X3	X2	X1	X0	F2	F1	F0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

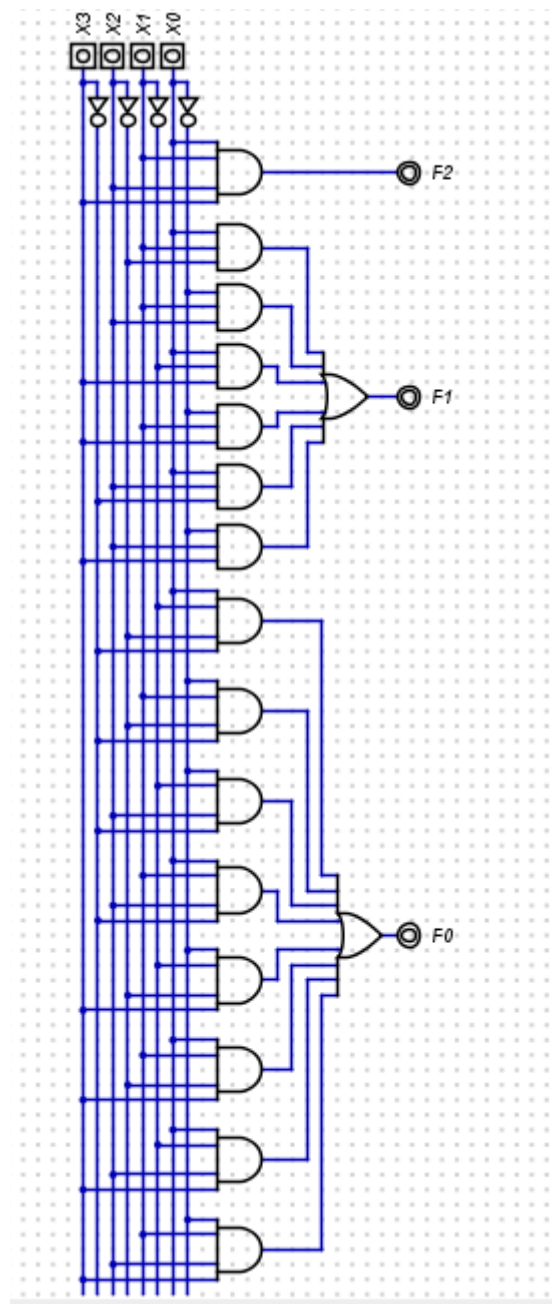
6. Especificar os valores de cada saída para as combinações de valores da entrada conforme indicado na figura seguinte. (Nota: Nenhum elemento da tabela deve ficar com X.)

Table						
File	New	Columns	Set	Create	K-Map	
X3	X2	X1	X0	F2	F1	F0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Justificar os valores atribuídos a F0, F1 e F2.

As expressões booleanas resultantes podem ser vistas na janela `All possible solution`. O mapa de Karnaugh correspondente a cada saída pode ser consultado usando o menu `K-Map`.

7. Sintetizar automaticamente o circuito usando o menu `Create > Circuit`. Este passo cria uma realização lógica da funcionalidade pretendida. O circuito pode ser visto no editor (cf. figura seguinte). Gravar o circuito com o nome **bitcount**.



8. Em vez de especificar a tabela de verdade, também é possível especificar as funções lógicas usando o menu **Analysis** **»** **Expression** e criar o circuito a partir daí.

5 Utilização de barramentos

Em muitas situações, as entradas ou saídas binárias podem ser agrupadas. Um grupo de sinais designa-se por *barramento*. A utilização de barramentos torna os diagramas lógicos mais claros.

Vamos criar uma versão de bitcount com barramentos. A nova versão vai chamar-se bitcount2.

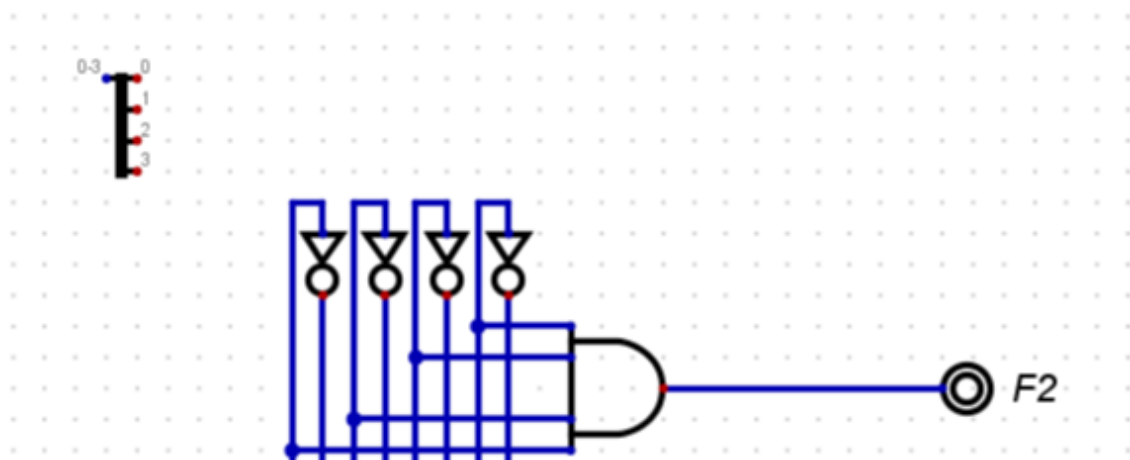
1. Copiar o ficheiro bitcount.dig para um novo ficheiro bitcount2.dig. Abrir este ficheiro com o

programa DIGITAL.

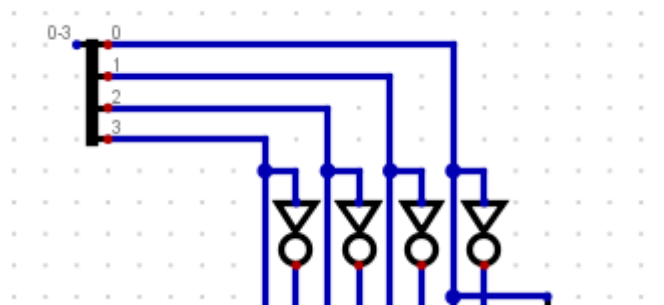
Vamos agora editar bitcount2 para usar barramentos.

2. Apagar os terminais de entrada (mas tomar nota da sua ordem). Colocar o elemento Splitter (da biblioteca Wiring) num local próximo daquele em que estavam os terminais de entrada. As propriedades do Splitter devem ser alteradas para:

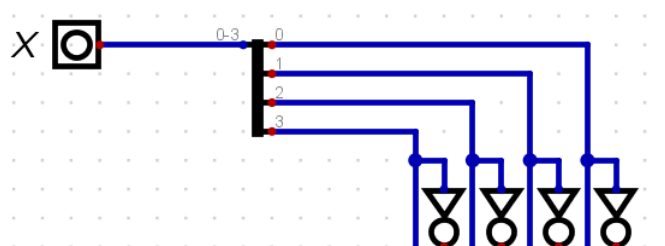
- Input Splitting: 4 (um barramento de 4 bits).
- Output Splitting: 1,1,1,1 (porque teremos 4 linhas a sair do Splitter).
- A posição dos 4 bits deve ser a indicada na figura. (Usar a tecla **L** para rodar um componente.)



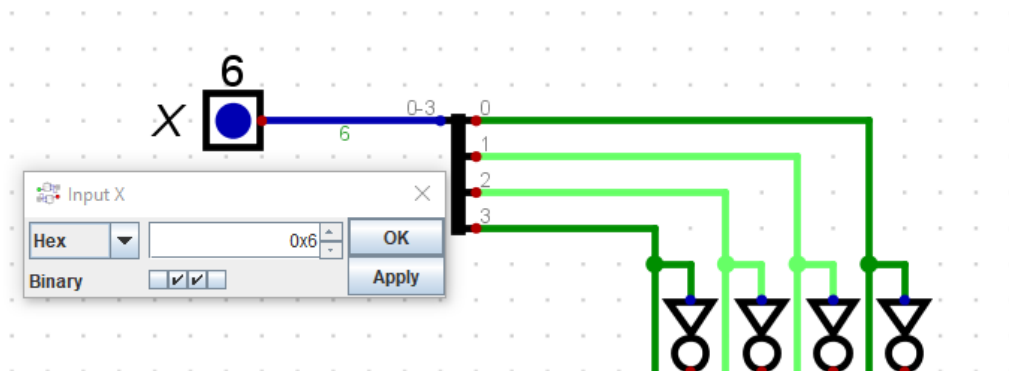
3. Ligar os terminais 0–3 aos fios que estavam ligados a X0–X3, *pela mesma ordem*.



4. Colocar um terminal de entrada e alterar a propriedade Data Bits para 4 e o nome para X. Ligar o terminal de entrada ao Splitter.

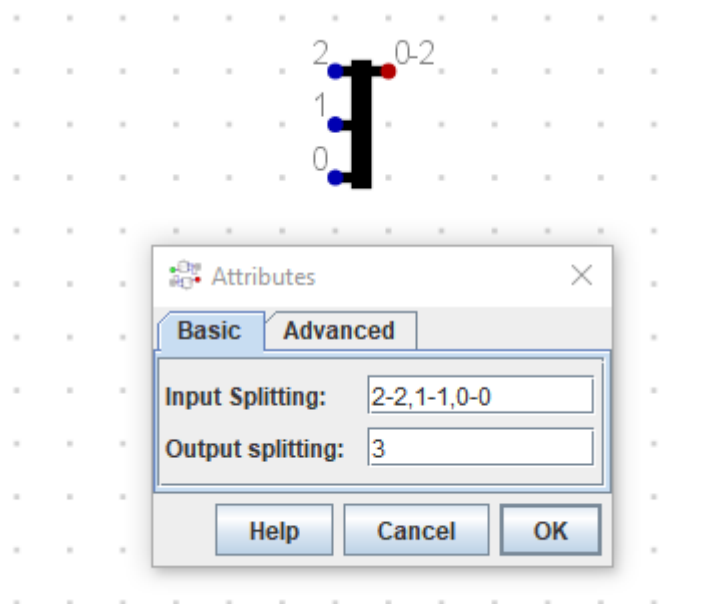


Em modo de simulação, clicar na entrada permite alterar os valores.

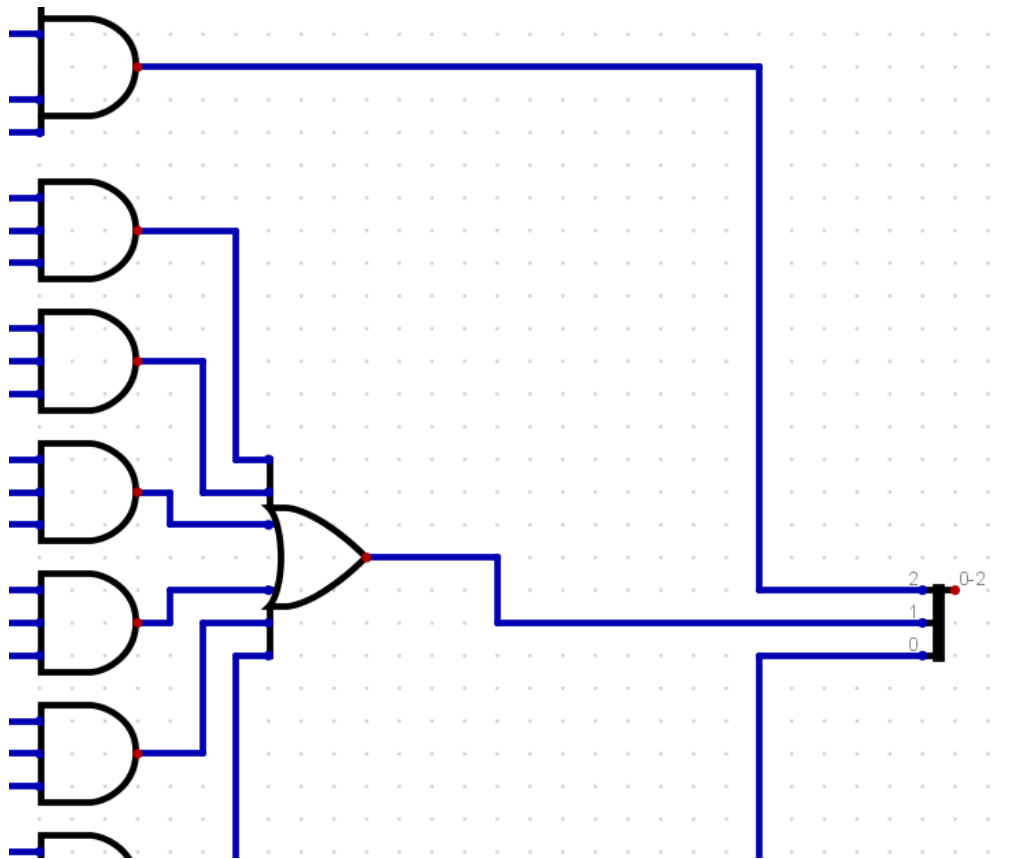


5. Um procedimento semelhante pode ser usado para definir o barramento de saída. Neste caso, são usados três bits. As características do componente Splitter da saída são:

- Input Splitting: 2-2, 1-1, 0-0
- Output Splitting: 3



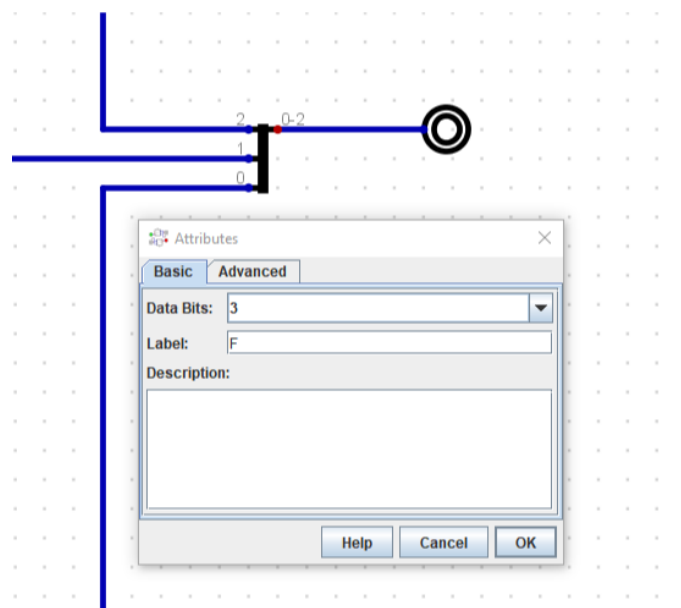
Apagar os terminais de saída e ligar as linhas correspondentes ao *Splitter* de saída (figura seguinte).



O terminal de saída deve ter as seguintes propriedades:

- Data Bits: 3
- Label: F

O resultado deve ser semelhante ao apresentado na figura a seguir.



6 Circuitos com hierarquia

6.1 Exemplo 1: sem barramentos

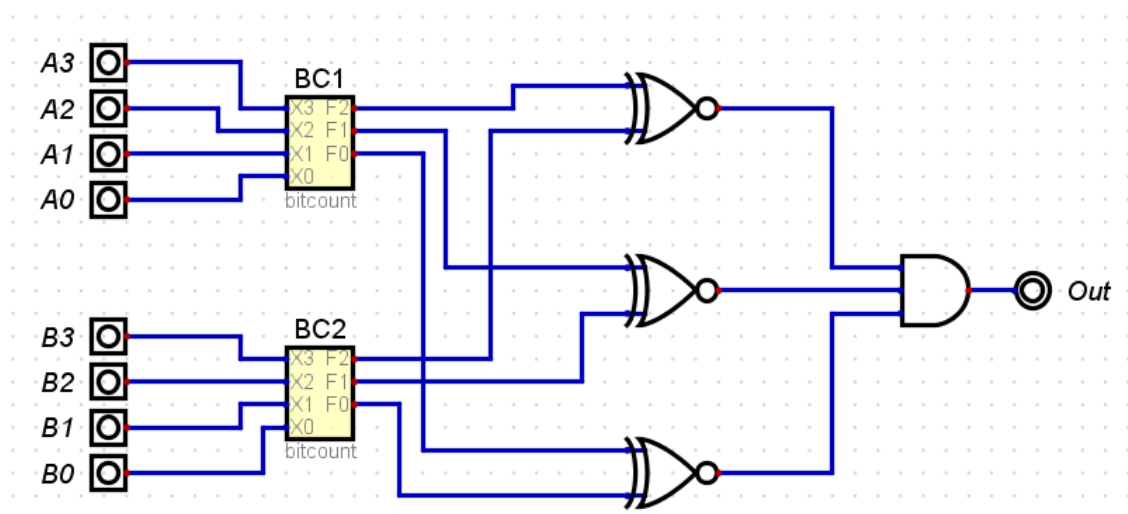
Os componentes desenvolvidos pelo utilizador podem ser usados na construção de outros componentes em conjunto com componentes já existentes. Nota: Todos os componentes devem estar na mesma pasta (ou em sub-pastas) e ter nomes diferentes.

Vamos mostrar como fazer um circuito que deteta se dois números (de 4 bits) têm o mesmo número de bits a 1. Para isso, vamos usar o componente `bitcount` da secção 4.

1. Começar por criar um novo circuito (vazio) e gravá-lo com o nome `Sys1`.
2. Colocar um componente do tipo `bitcount` no novo circuito (da biblioteca *Custom*). Atribuir-lhe o nome `BC1` usando a janela de propriedades.



3. Completar o circuito para obter o resultado indicado na figura. A saída `Out` vem a 1 sempre que os números representados por `A3–A0` e `B3–B0` tiverem o mesmo número de uns.



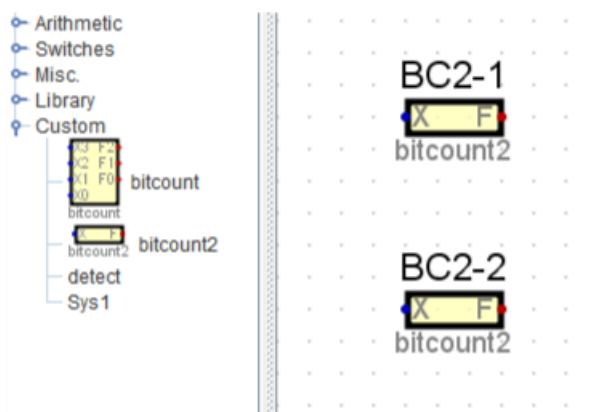
O circuito pode agora ser simulado como habitualmente.

Explicar porque é que o circuito realiza a função pretendida.

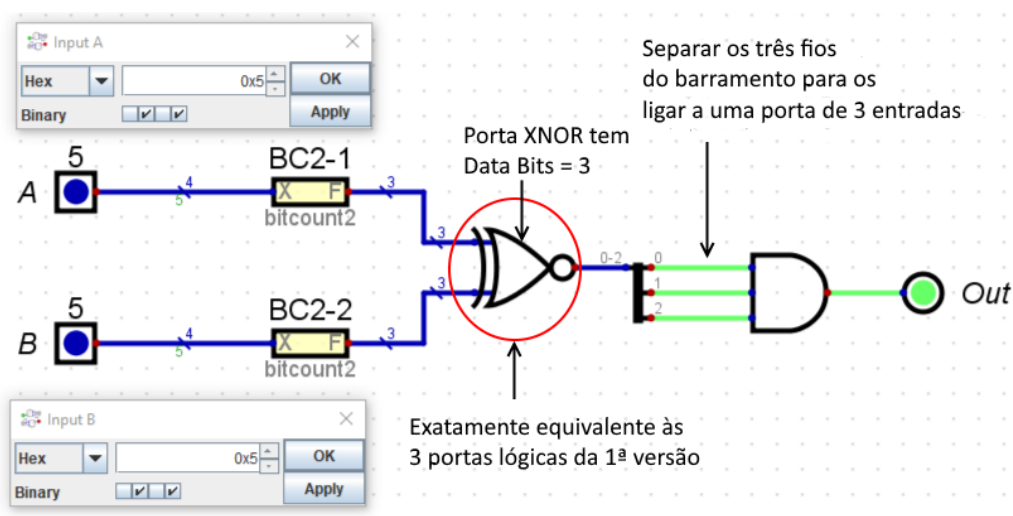
6.2 Versão 2: com barramentos

O circuito Sys1 não usa barramentos, o que teria facilitado a sua implementação. Vamos realizar uma nova versão (designada por Sys2), que aproveita barramentos para simplificar o diagrama da implementação.

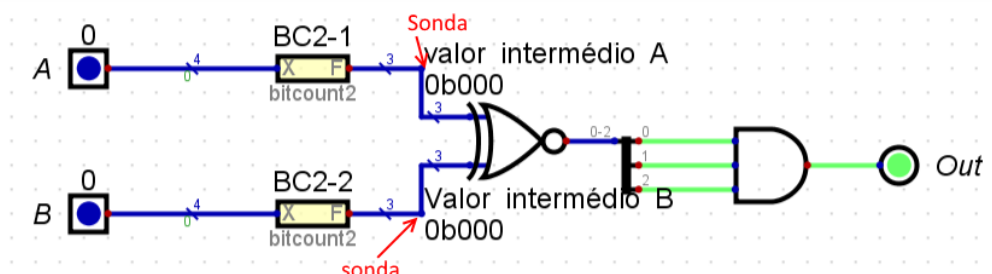
1. Começar por criar um novo componente Sys2 (vazio).
2. Colocar os componentes do tipo bitcount2 no novo circuito. Notar que cada um apenas tem um terminal de entrada e um terminal de saída. Cada terminal de entrada permite ligar 4 bits; cada terminal de saída permite ligar 3 bits (conforme especificado na construção do circuito na secção 5).



3. Completar o circuito para obter o resultado indicado na figura.



O circuito pode agora ser simulado como habitualmente. Notar que os barramentos são sempre desenhados a azul. Para exibir permanentemente os valores existentes num barramento pode ligar-se uma sonda ao barramento (componente Probe da biblioteca IO), conforme se mostra na figura seguinte.



A etiqueta da sonda e a base usada para representação do número são controladas pelas propriedades Label e Number Format, respetivamente.

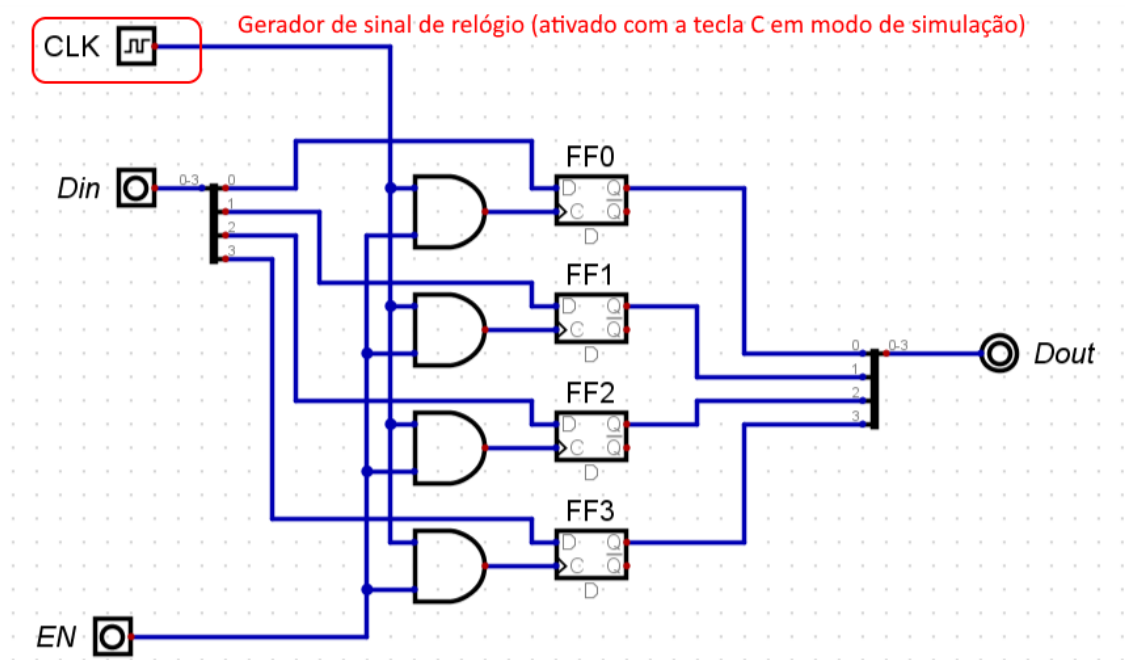
7 Circuitos sequenciais

Circuitos sequenciais síncronos têm um sinal especial de sincronismo, o sinal de relógio. A simulação deste tipo de circuitos requer o tratamento especial deste sinal.

7.1 Exemplo 1: registo de 8 bits

O *flip-flop* do tipo D permite armazenar um valor lógico. Frequentemente, queremos armazenar um grupo de vários bits. Para isso, pode-se construir um *registo* a partir de *flip-flops* individuais. Vamos construir um registo de 4 bits com sinal de habilitação (*enable*)

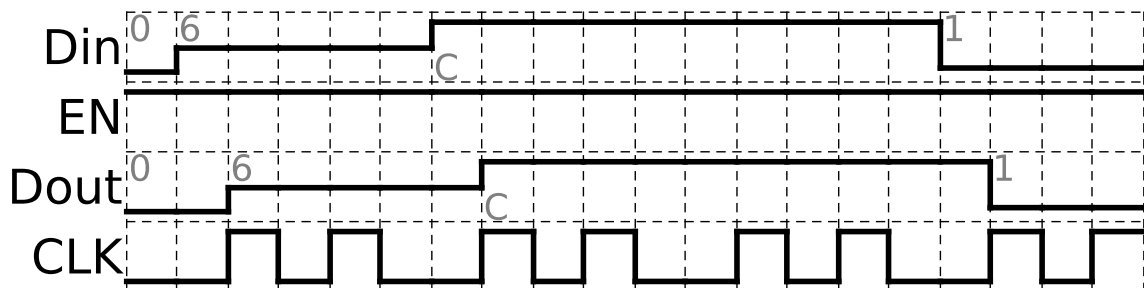
1. Criar um circuito novo, designado por reg4.
2. O circuito usa um *flip-flop* do tipo D. Este componente existe na biblioteca Flip-Flops.
3. Criar um registo de 4 bits com sinal de habilitação conforme indicado na figura. Para facilitar a utilização, entradas e saídas de dados estão agrupadas em barramentos de 4 bits.



Explicar como é implementada a habilitação (*enable*) do registo.

4. Para simular o circuito, variam-se as entradas (neste caso, DIN e EN). Para fazer o gerador de sinal de relógio avançar meio período, usar o a tecla **C** em modo de simulação.

O comportamento temporal do circuito pode ser exibido num diagrama temporal através do comando de menu **Simulation >> Show measurement graph**.



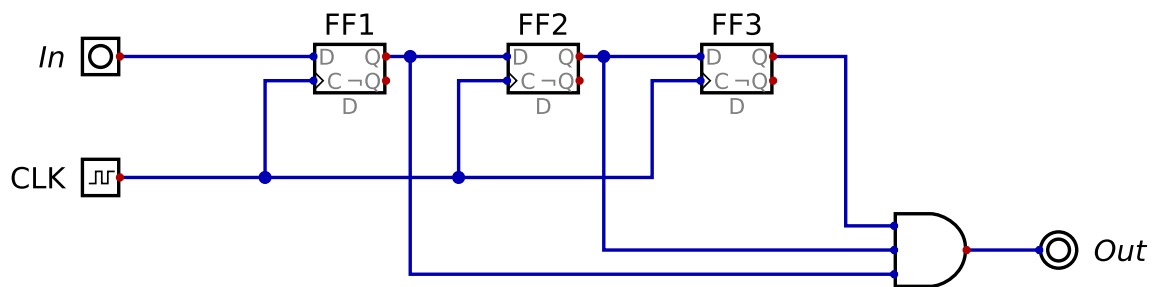
Usar o simulador para confirmar que os valores apresentados à entrada DIN são armazenados nos *flip-flops* e ficam disponíveis na saída DOUT apenas quando a entrada EN está a 1 e ocorre um flanco ascendente do sinal de relógio.

A biblioteca Memory do DIGITAL contém já um componente chamado Register, que realiza a funcionalidade deste exemplo. A respetiva propriedade Data Bits permite definir o número de bits a armazenar (i.e., a capacidade de armazenamento do circuito).

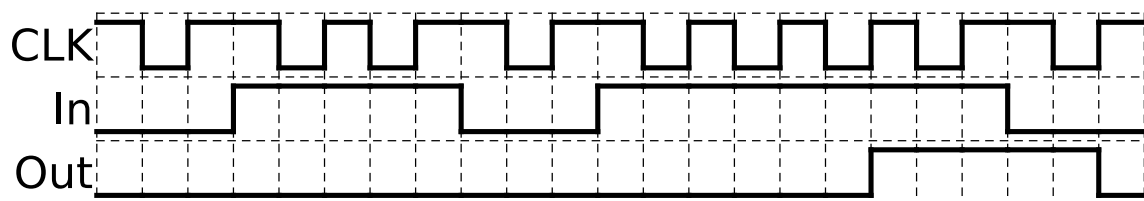
7.2 Exemplo 2: detecção de uma sequência

Como segundo exemplo, vamos ver um circuito que deteta a presença de uma sequência de três 1s seguidos.

1. Criar um circuito novo, designado por det111.
2. Completar o circuito conforme indicado na figura.



3. Simular o circuito como no exemplo anterior.



Confirmar e explicar o funcionamento do circuito.

Fim