DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Linguagem de descrição de Hardware Verilog

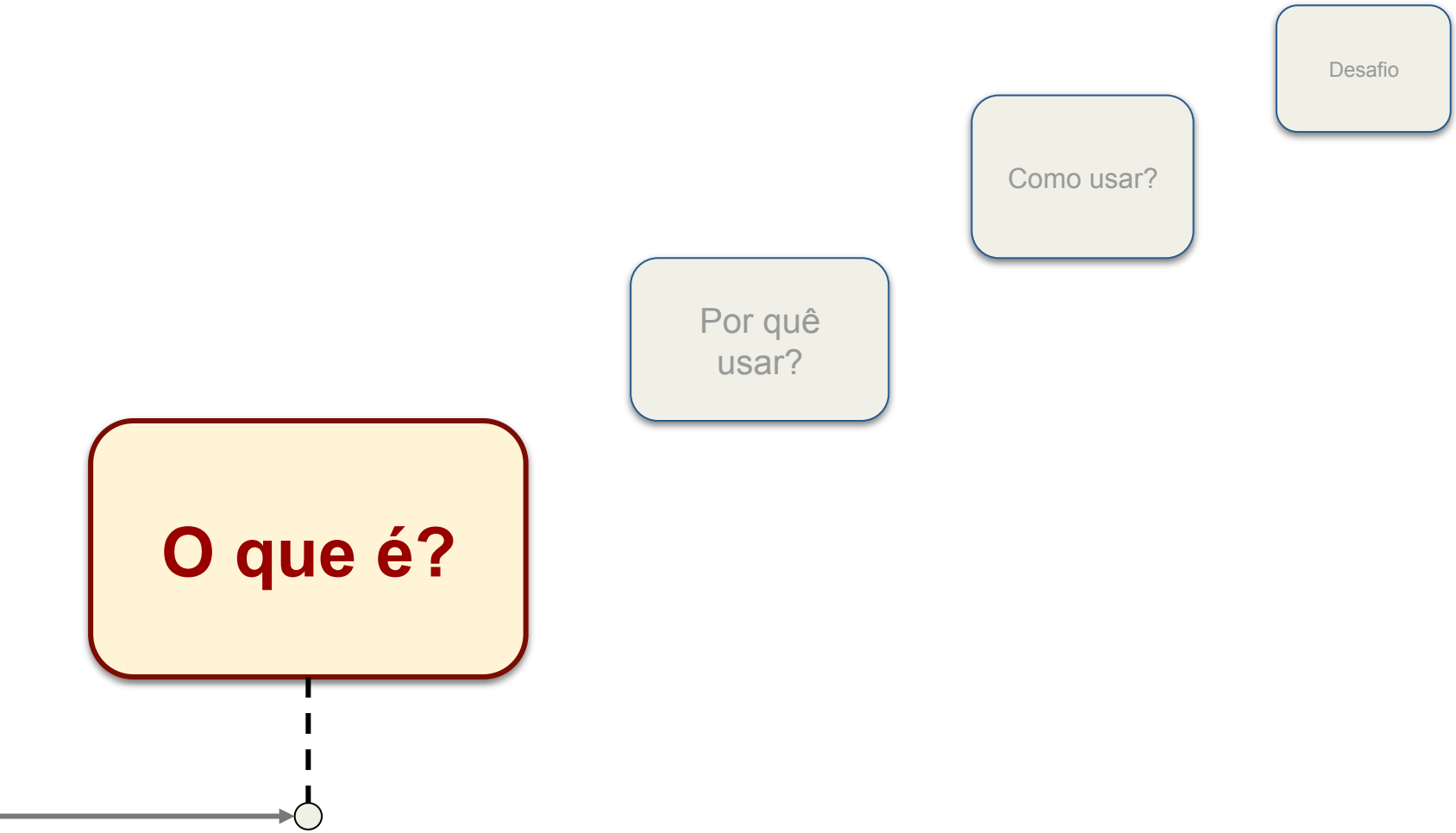
Arquitetura de Computadores

Professor: Omar Paranaíba Vilela Neto

Monitor: Pedro Arthur R. L. Silva

Objetivos

- ➔ Apresentar a linguagem Verilog.
- ➔ Demonstrar como utilizá-la em um projeto.
- ➔ Ao final da aula você será capaz de projetar e simular *hardware* utilizando *Verilog.*

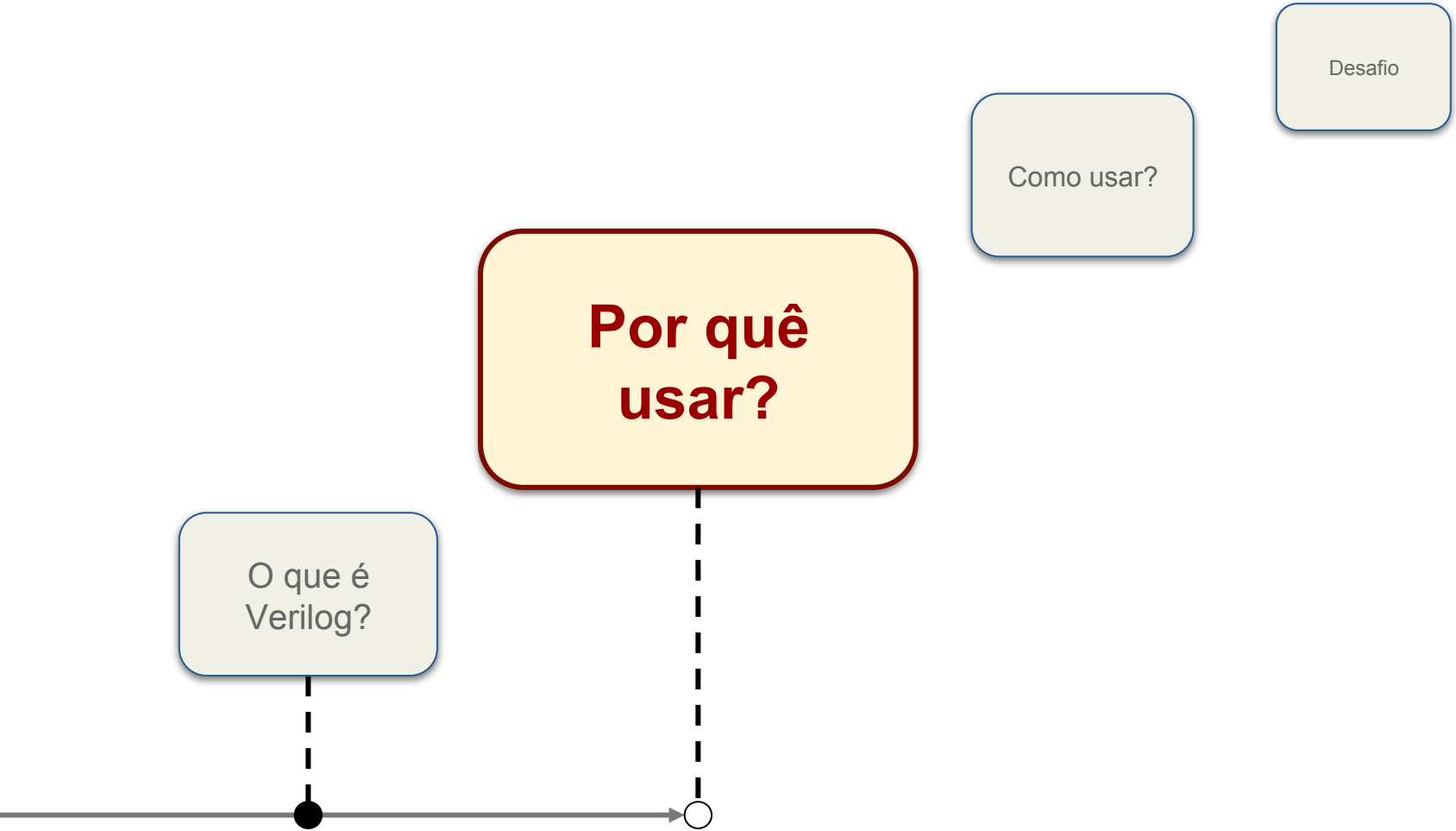

Desafio
Como usar?
Por quê usar?
O que é?

O que é Verilog?

- Linguagem de descrição de *hardware* (HDL).
  - **Não descreve instruções** que o computador deve executar.
  - Descreve o comportamento de um sistema digital.

- Surgiu na década de 80. Padronizada pela IEEE em 1995 e ratificada pela mesma em 2005.

- Baseada na linguagem C.


Desafio
Como usar?
Por quê usar?
O que é Verilog?

Por quê usar Verilog?

- Você já parou para pensar em como os circuitos digitais são construídos?
  - Manualmente?
  - E se eu precisar do mesmo circuito mas em um dispositivo de arquitetura diferente?
  - E se o circuito tiver milhões de porta?
  - Entre outros muitos problemas.

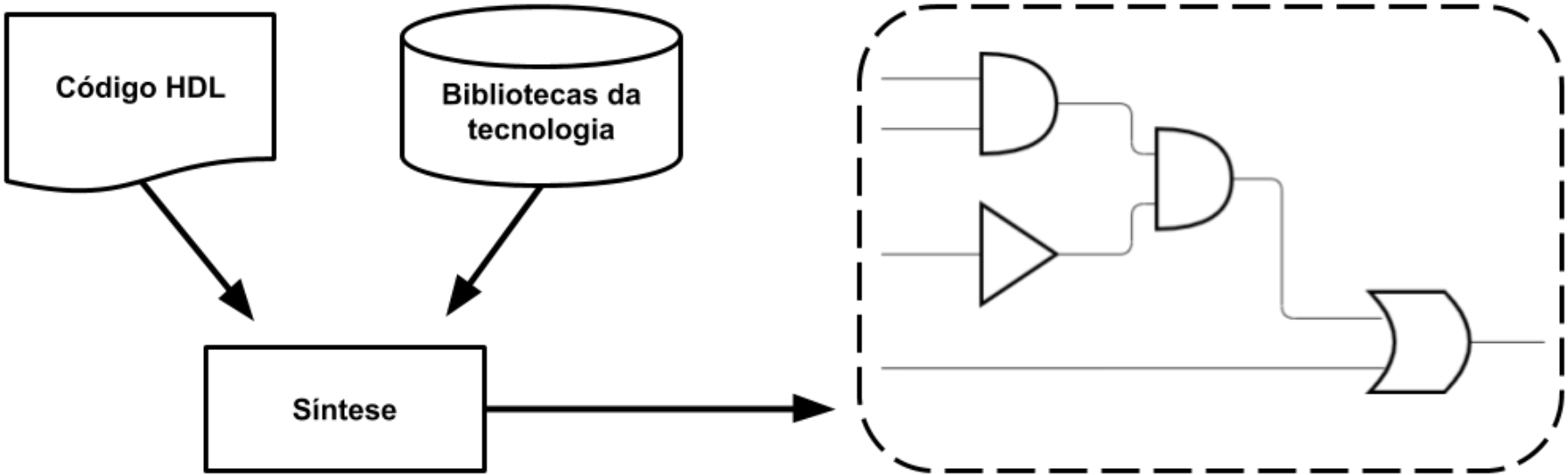
Por quê Verilog?

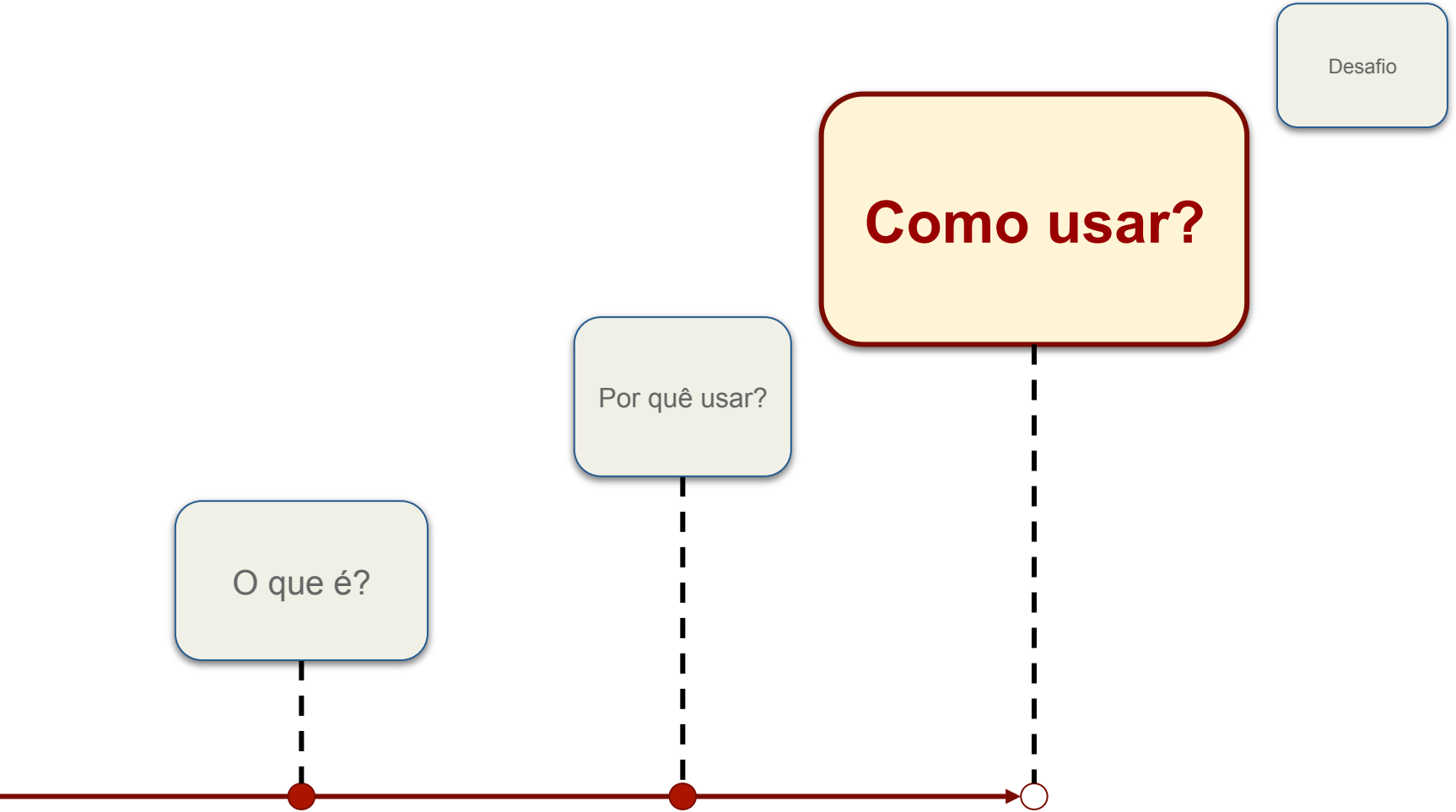
- HDL geralmente tem o papel de especificar o *design* do circuito, que será usado posteriormente para algoritmos de síntese.

- **Algoritmos de síntese** tem o papel de mapear a descrição em **HDL** para circuitos em tecnologias específicas.

- Muito mais confiável, rápido e eficiente.

Por quê Verilog?



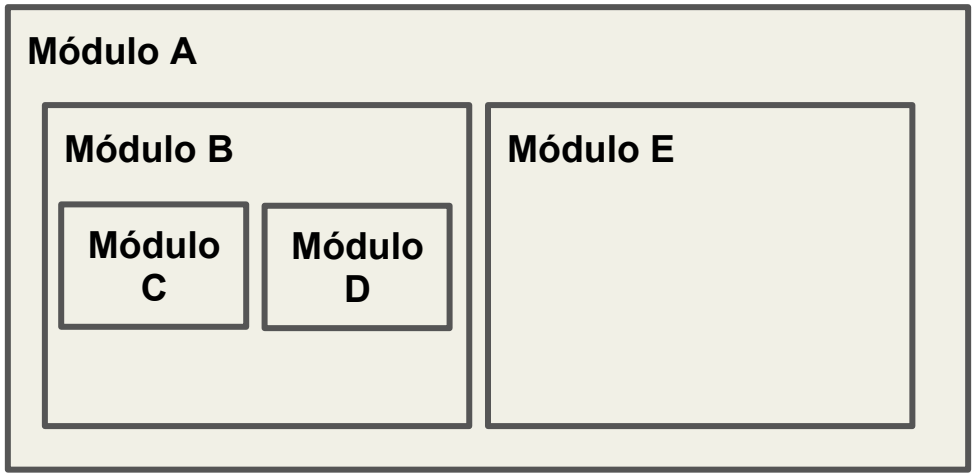

Desafio
Como usar?
Por quê usar?
O que é?

Como usar Verilog?

- Simuladores que vamos utilizar
  - Google Colabs
  - JDoodle
  - DigitalJS
- Simulador offline
  - Icarus Verilog
  - Windows, Linux e Mac.

Como usar Verilog?

- O projeto é composto por blocos, denominados **módulos**.
  - Utiliza-se a palavra reservada **module** para declarar um módulo.
  - Um módulo pode instanciar vários outros módulos.



Meu primeiro programa

Exemplo de Hello World

Representação numérica

Representação	Valor	Comentário
4’b1010	1010	binário de 4 bits
8’h5A	1011010	hexadecimal de 8 bits
5’o21	10001	octal de 5 bits
3’d2	010	decimal de 3 bits
4’b101x	101x	x representa *don’t care*
-5’b00001	11111	Complemento de 2

Operadores

TIPO	SÍMBOLO	FUNÇÃO	NÚMERO DE OPERANDOS
ARITMÉTICO	+	Adição	2
	-	Subtração	2
	*	Multiplicação	2
	/	Divisão	2
	%	Módulo	2
	**	Exponenciação	2

Operadores

TIPO	SÍMBOLO	FUNÇÃO	NÚMERO DE OPERANDOS
RELACIONAL	>	Maio que	2
	<	Menor	2
	>=	Maior ou igual que	2
	<=	Menor ou igual que	2
	==	Igual	2
	!=	Diferente	2

Operadores

TIPO	SÍMBOLO	FUNÇÃO	NÚMERO DE OPERANDOS
BIT A BIT	~	NOT	1
	&	AND	2
	|	OR	2
	^	XOR	2

Operadores

TIPO	SÍMBOLO	FUNÇÃO	NÚMERO DE OPERANDOS
LÓGICOS	!	NOT	1
	&&	AND	2
	||	OR	2

Operadores

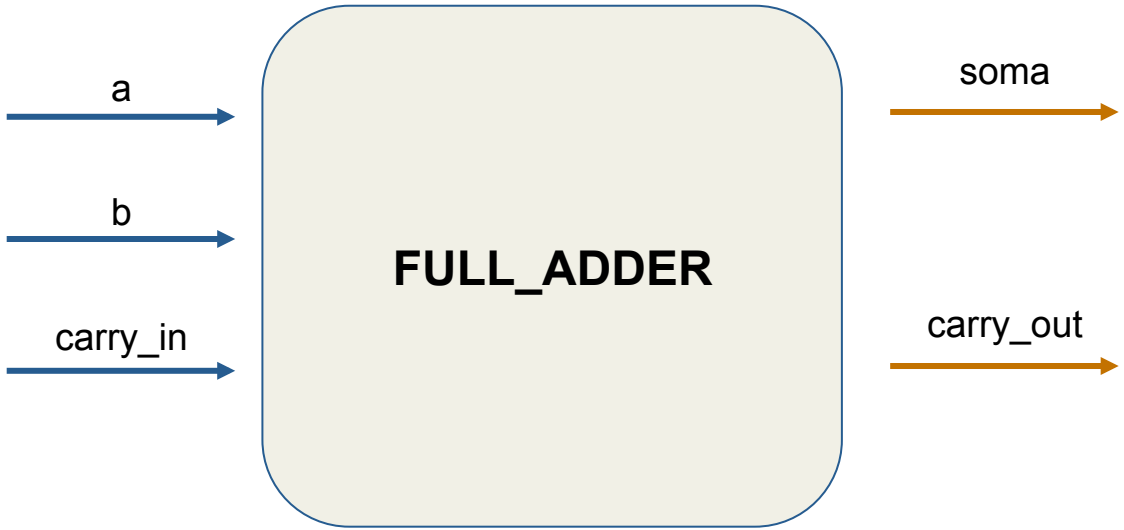
Exemplo de utilização de operadores

Como usar Verilog?

- Criar um projeto e atribuir um nome
- Criar o código *Verilog* do módulo principal (**main**) que descreve o circuito.
- Definir as entradas e saídas - **pinos** (interfaces externas)
- Definir sinais internos
- Definir comportamento do circuito

Projetando um somador completo

- 3 entradas
  - a
  - b
  - carry_in
- 2 saídas
  - soma
  - carry_out



Estrutura do módulo

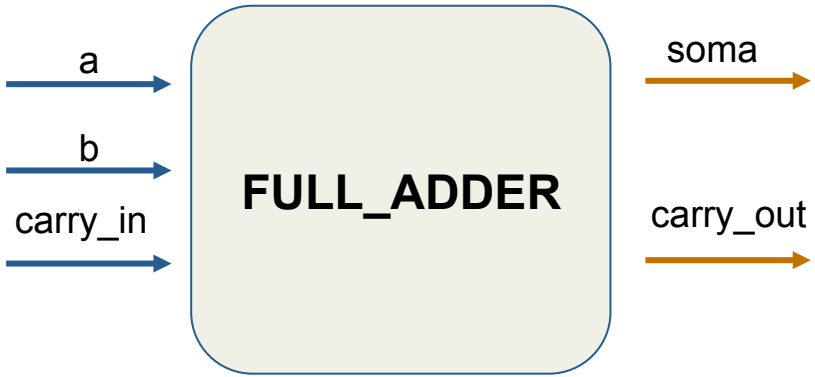
- **CASE** sensitive
  - in_a ≠ in_A
- Todas as **palavras-chave** são escritas em letras minúsculas (*lowercase*)
- **Comentários**
  - Uma linha: *//*
  - Múltiplas linhas: */* */*

```
module nome_do_modulo (
        //Declaração dos
pinos
);

    //Comportamento

endmodule
```

Pinos do módulo

- Caracterizados quanto a **tipo** e **direção**
- Por tipo, entende-se sinal **simples** (1 bit) ou **barramento** (2 ou mais bits).
- Em termos de direção, tem-se
  - **INPUT** - entrada do módulo
  - **OUTPUT** - saída do módulo
  - **INOUT** - bidirecional

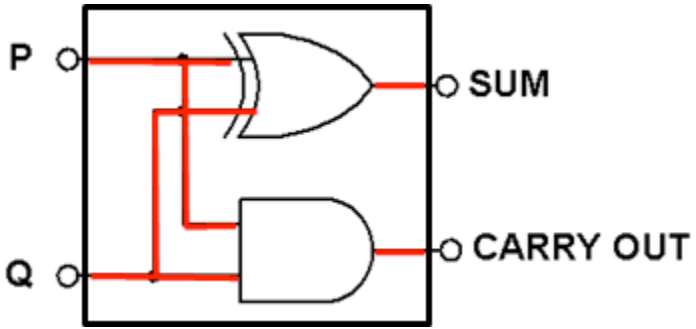
Projetando um somador completo




```
module full_adder(
            input a,
      input b,
            input carry_in,
            output soma,
      output  carry_out
);

                  .
                  .
                     .

endmodule
```

Tipos de Dados

- ***NETS***
  - Representam conexões físicas entre os elementos.
- São utilizados nas saídas de atribuições contínuas (***assign***) e sinais de conexão entre os módulos.
- Não armazenam valores.
- Tipo ***wire*** é o mais utilizado.



Tipos de Dados

- Declarações implícitas são sempre do tipo ***wire*** de um *bit.*
  - **input** a, **output** b, **inout** c
- **wire** [most_significant_bit : least_significan_bit]
- Exemplos:
  - **input** a - tipo implícito, 1 bit
  - **output** [3:0] b - tipo implícito, barramento de 4 bits
  - **inout wire** [0:7] c - tipo explícito, barramento de 8 bits, último dígito é msb
  - **wire** d - tipo explícito, 1 bit

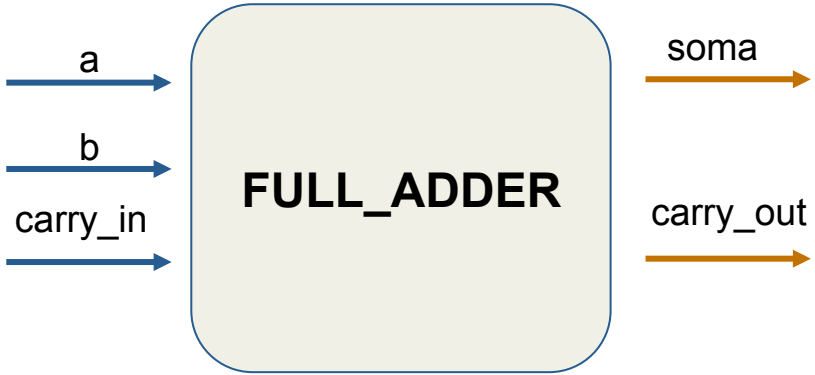
Tipos de Dados

- ***Register***
  - Representam elementos capazes de armazenar dados temporariamente
- São usados em saídas de atribuições procedimentais (**bloco initial** ou **always**);
- Existem 5 tipos: reg, integer, real, time e realtime;
- Podem ser usados para declaração de memória

Tipos de Dados

- ***Register***
  - Exemplos:
  - **reg** a - registrador de 1 bit
  - **reg** [0 : 4] meuBarramento - Um barramento de 5 posições com registradores de 1 bit
  - **reg** [0 : 3] minhaMemoria [0 : 63] - Memória de 64 posições com registadores de 4 bits

Tipos de dados no somador completo




```
module full_adder(
            input a,
      input b,
            input carry_in,
            output soma,
      output  carry_out
);
wire  a;
wire  b;
wire  carry_in;
wire  soma;
wire  carry_out;

                        .
                        .
                        .

endmodule
```

Atribuições

- Existem dois modos de atribuições em *Verilog*.
- Atribuições contínuas
  - Utilizadas para gerar circuitos combinacionais
  - Palavra-chave ***assign.***
  - Saída muda sempre que há alteração em qualquer uma das entradas
  - Lado esquerdo da atribuição tem que ser do tipo ***NET*** (*wire*).
  - Lado direito pode ser *NET* ou tipo registrador

Atribuições contínuas no somador completo

Exemplo de somador completo

Atribuições

- Atribuições procedurais
  - Efetuadas dentro de blocos de designação de procedimento
    - **bloco *initial*** ou **bloco *always***
  - Blocos *always* e *initial* não podem ser aninhados
  - Representam processos diferentes
  - Na simulação, iniciam no mesmo momento, no tempo 0
  - Todos os blocos *always* são executados em paralelo

Atribuições

- Bloco **initial**
  - Blocos *always* e *initial* não podem ser aninhados
  - Representam processos diferentes
  - Na simulação, iniciam no mesmo momento, no tempo 0
  - São executados em paralelo
  - Normalmente utilizado para inicializar declarações, para fins de simulação
  - É executado somente uma vez, no início da simulação, no tempo 0
  - Declarações dentro de um bloco *initial* são executadas sequencialmente

Atribuições

- Bloco **always**
  - Declarações dentro de um bloco *always* são executadas sequencialmente
  - **Iniciam no tempo 0** e executam continuamente na forma de ***loop***
  - Modelam processos que são continuamente repetidos
  - Só é possível modelar circuitos sequenciais utilizando blocos *always*

Atribuições

Exemplo de contador

Atribuição Bloqueante vs Não Bloqueante

Exemplo de atribuições bloqueantes

Comando CASE

Exemplo de multiplexador 4x1

Comando CASE

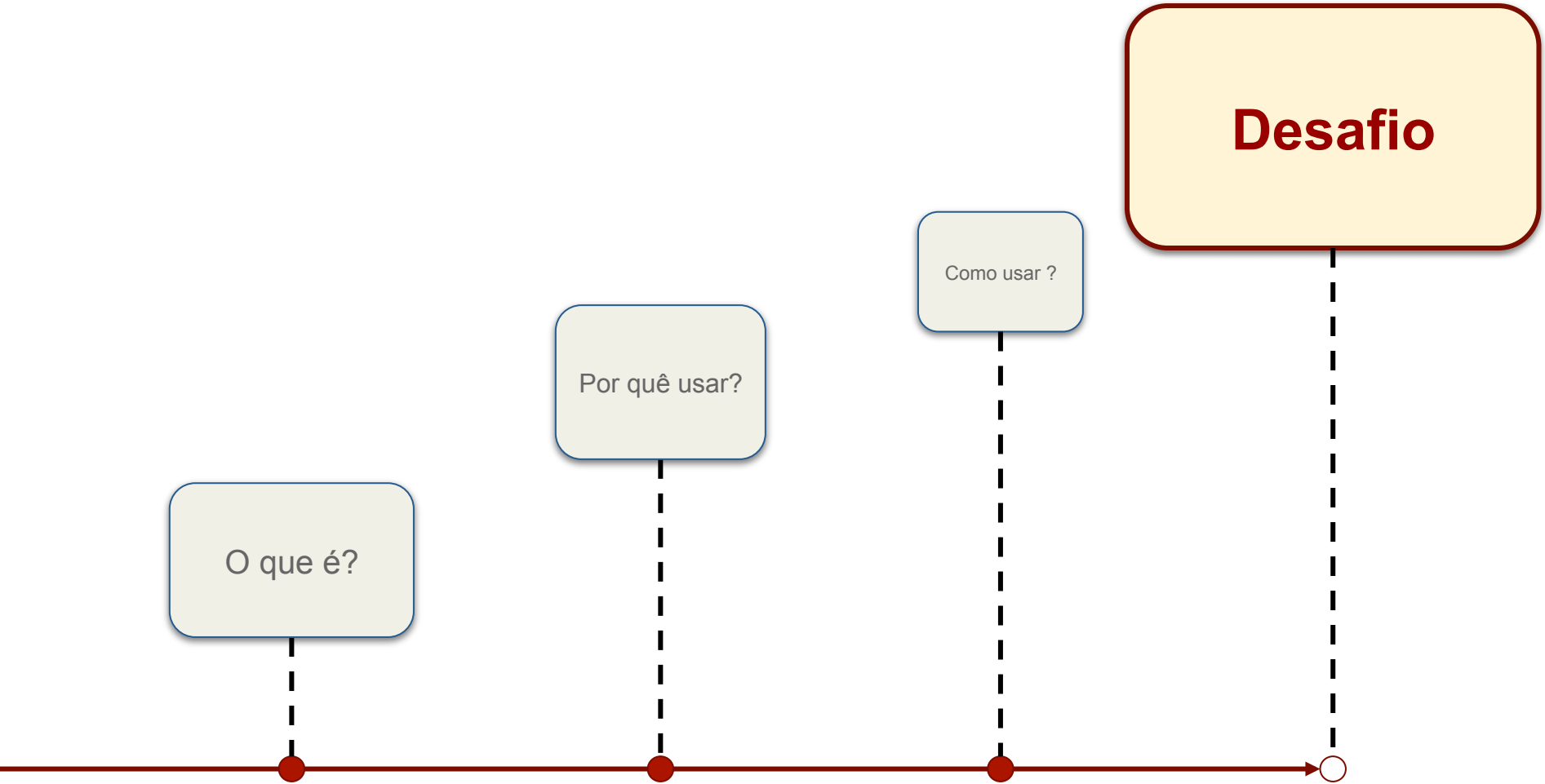
Exemplo de ALU

Como testar?

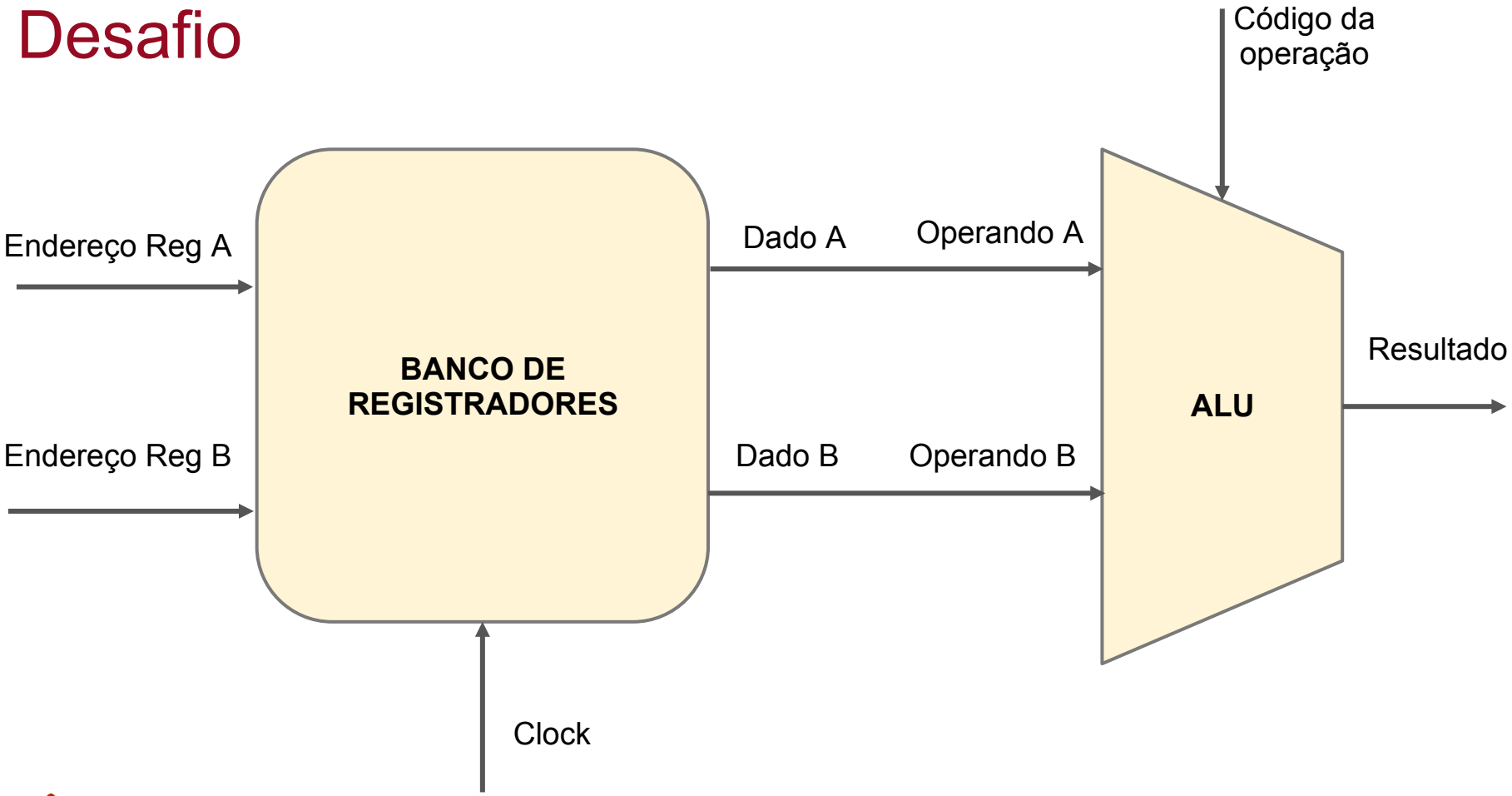
- Devemos **testar** nosso projeto para averiguar se o **funcionamento** está de acordo com as especificações.
- Precisamos de um módulo para **inicializar** e conduzir nossa variáveis de entrada e **monitorar** nossas saídas.
- Necessário instanciar o módulo que vai ser testado.
- Finalizar a simulação.

Como testar?

Exemplo de Testbench


Desafio
Como usar ?
Por quê usar?
O que é?

Desafio


Código da operação
Endereço Reg A
Endereço Reg B
BANCO DE REGISTRADORES
Dado A
Operando A
Dado B
Operando B
ALU
Resultado
Clock

Referências

- TALA, Deepak Kumar, ASIC WORLD . Acessado em Agosto, 2020.
  URL: http://www.asic-world.com/
- Notas de aula da Prof.ª Mara Cristina. DECOM. CEFET-MG
- Notas de aula do Prof.º Ricardo Ferreira. DPI. UFV
- BASTOS, Alex Vidigal. VERILOG. Acessado em Agosto, 2020.
  URL: http://www.decom.ufop.br/alex/arquivos/sist_emb/Verilog.pdf

Links dos simuladores

- JDoodle. Console, editor e testbench. Acessado em Agosto de 2020.
  URL: https://www.jdoodle.com/execute-verilog-online/
- Icarus. Console, editor e testbench. Acessado em Agosto de 2020.
  URL: https://www.tutorialspoint.com/compile_verilog_online.php
- Verilog Online + Formas de onda, . Acessado em Agosto de 2020.
  URL: http://www.techep.csi.cuny.edu/~zhangs/v.html
- DigitalJS, Digital Logic Simulator,. Acessado em Agosto de 2020.
  URL: https://digitaljs.tilk.eu/

Obrigado!

Pedro Arthur R. L. Silva -
pedro.rodrigues@dcc.ufmg.br

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS