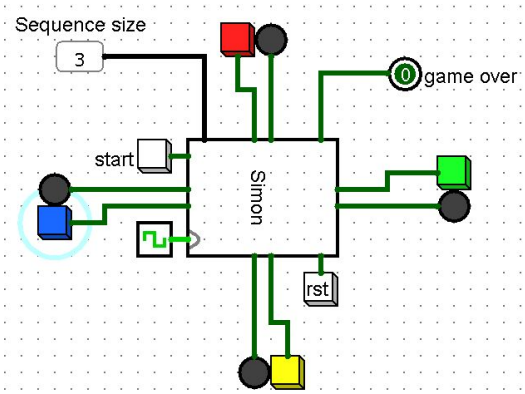
Trabalho 1

Trabalho 1

- Projetar o circuito do jogo Simon seguindo o modelo Bloco Operativo/Bloco de Controle
  - O funcionamento do jogo deve estar de acordo com www.freesimon.org
    - Sem som
  - O jogo deve armazena a sequência de cores em memória
  - Cada cor deve ser gerada de forma randômica através de um circuito LFSR (*Linear Feedback Shift Register*)
  - O jogo deve operar na frequência 4.1KHz do Logisim
  - A saída *Sequence size* mostra o tamanho da sequência atual
  - Antes de iniciar o jogo, o jogador deve pressionar o botão *start*
  - A cada botão pressionado, a cor correspondente deve ficar ativa tempo suficiente para o jogador identificar que o botão foi lido
  - Ao errar a sequência, a saída *game over* deve ficar ativa até o início de um novo jogo (*start*)

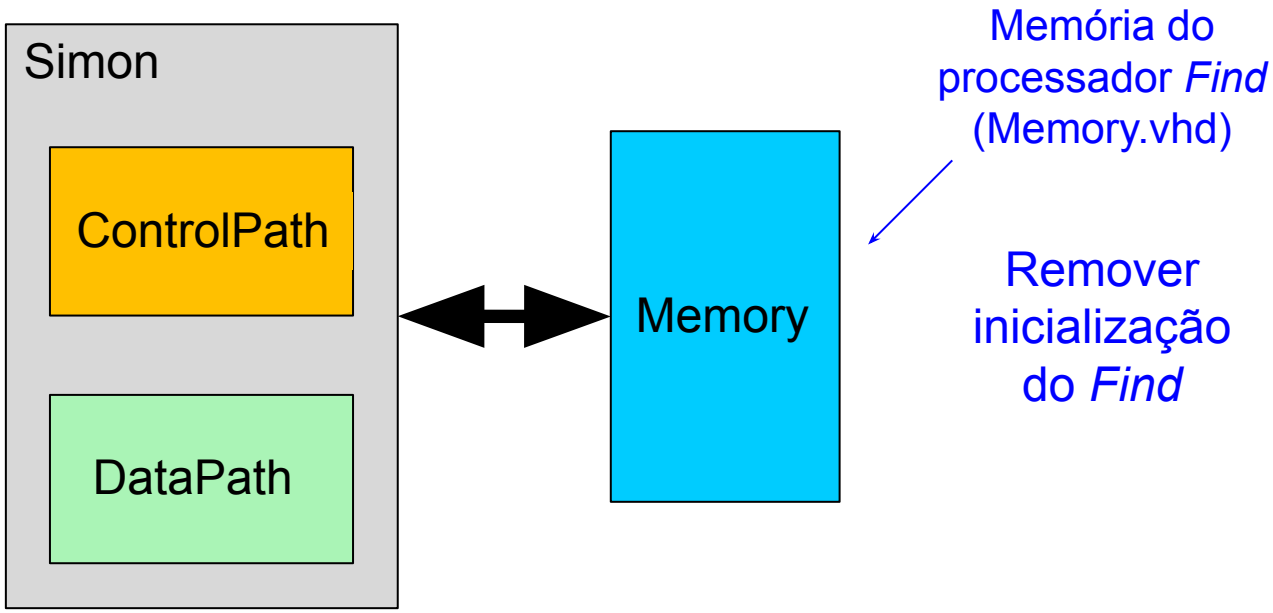


Trabalho 1

- Implementar em VHDL o jogo Simon
  - A implememementação deve ter duas *architectures*
    1. Comportamental + estrutural (*ControlPath* + *DataPath*)
       - A descrição VHDL deve ser baseada no projeto implementado na parte 1 do trabalho. O projeto da parte 1 pode ser alterado a fim de corrigir problemas.
    2. Totalmente comportamental
       - Deve-se explorar a descrição de alto nível
  - Tomar como referência do processador *Divider*

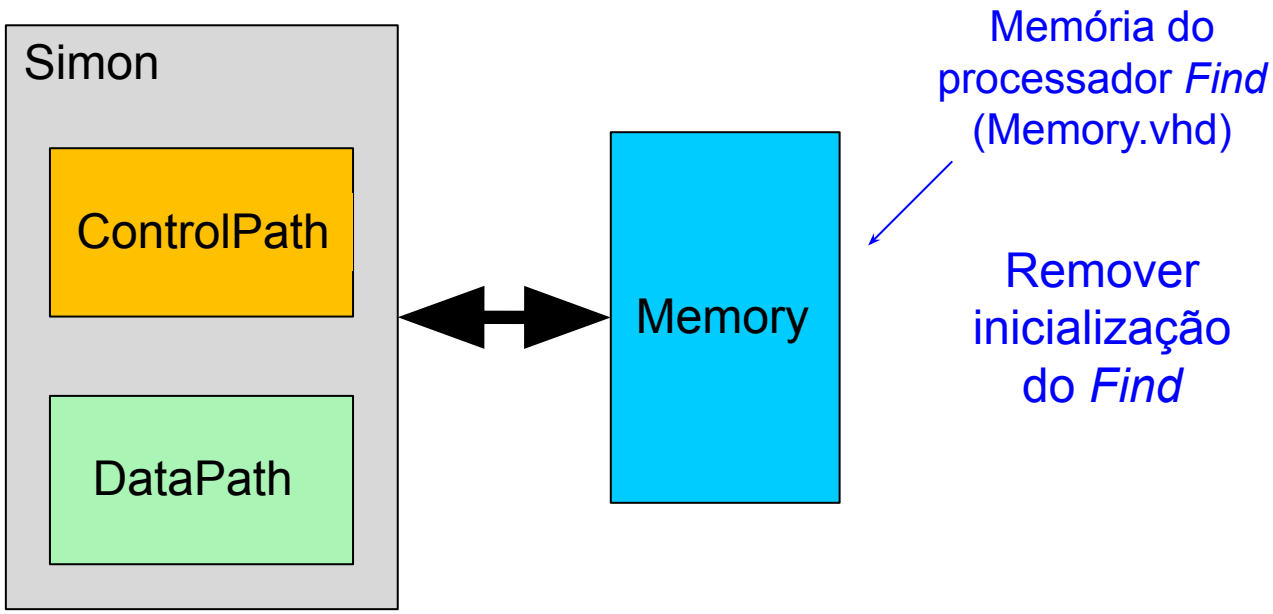
Trabalho 1

- *Architecture* Comportamental + estrutural
  - Estrutura da descrição
    - 4 entidades: *Simon*, *ControlPath*, *DataPath* e *Memory*
    - Cada entidade corresponde a um arquivo .vhd de mesmo nome



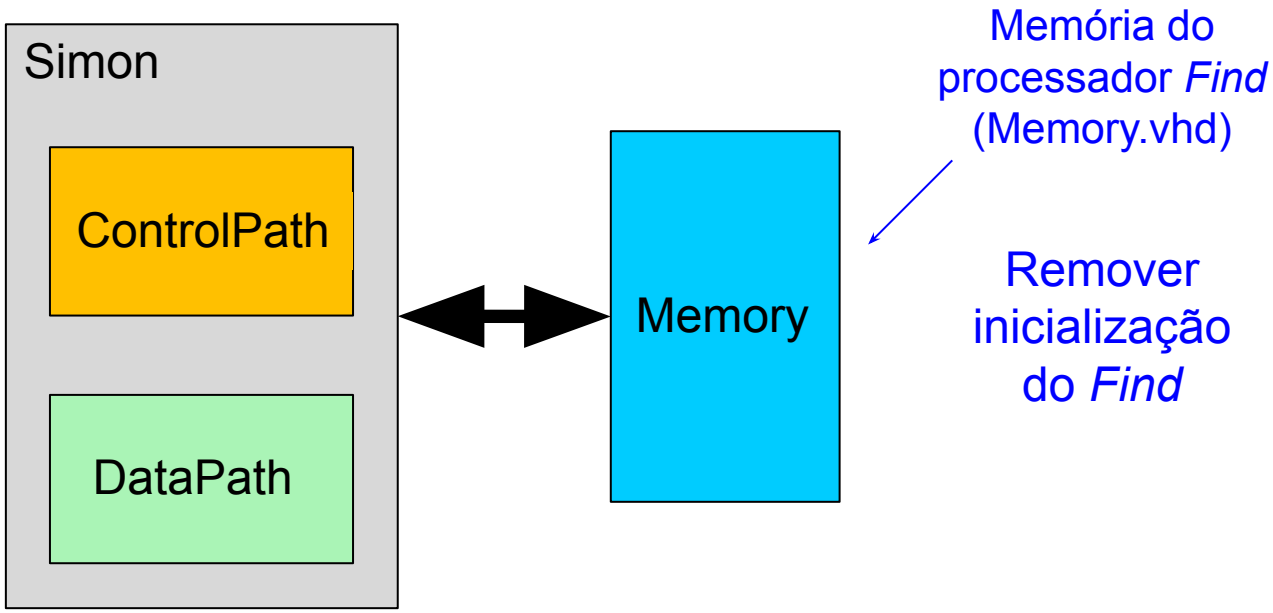
Trabalho 1

- *Architecture* Comportamental + estrutural
  - Estrutura da descrição
    - Criar *package* Simon_pkg contendo a definição de uma *record* com todos os comandos do *ControlPath* para o *DataPath* e outra contendo todos os sinais de *status* do *DataPath* para o *ControlPath*



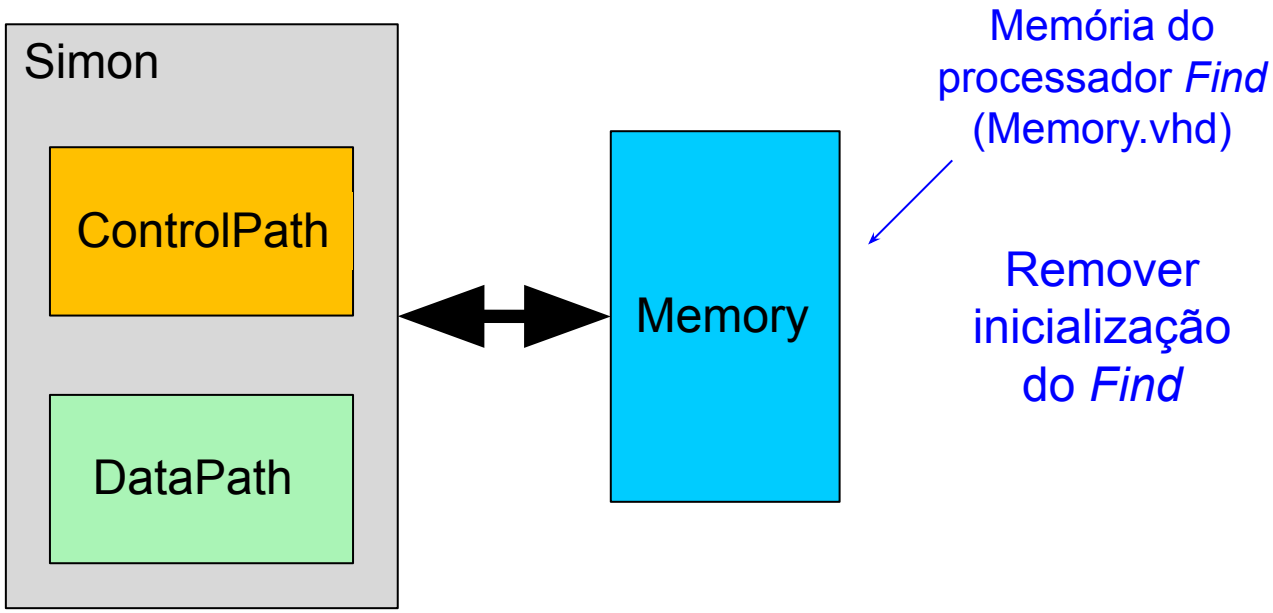
Trabalho 1

- *Architecture* Comportamental + estrutural
  - *DataPath* (DataPath.vhd)
    - Estrutural + comportamental
    - Registradores devem ser instâncias de RegisterNbits.vhd. Somadores devem ser instâncias de Adder_nbits.vhd. Comparadores devem ser instâncias do Comparador.vhd



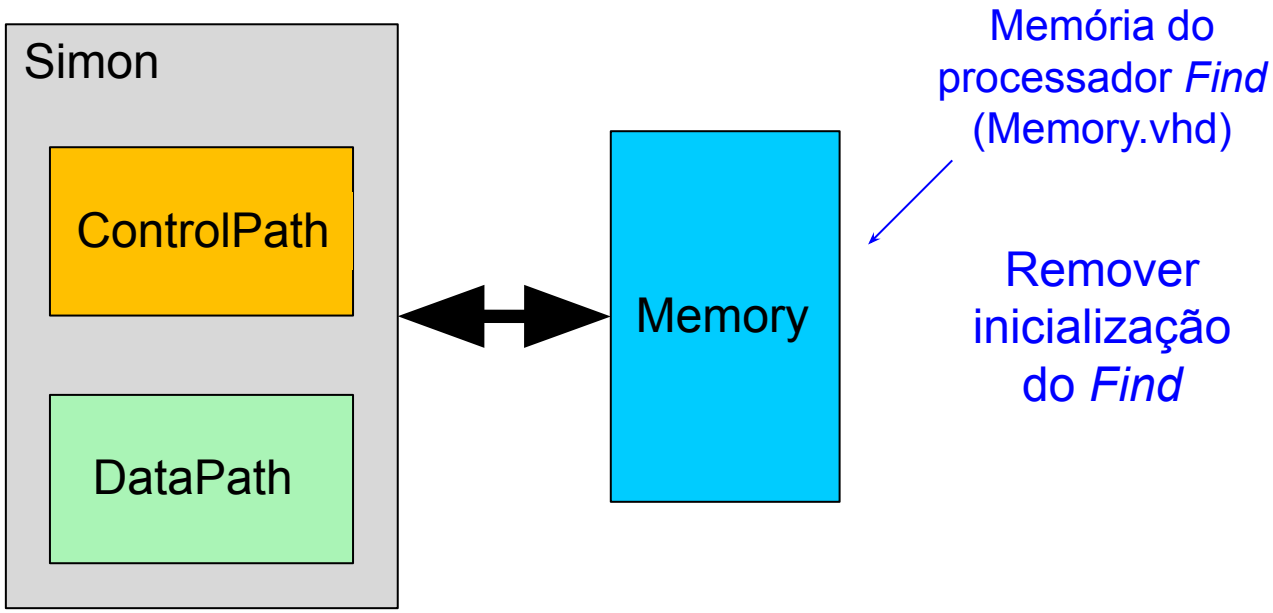
Trabalho 1

- *Architecture* Comportamental + estrutural
  - *ControlPath* (ControlPath.vhd)
    - Totalmente comportamental
    - Baseado nas descrições de FSMs vistas em aula



Trabalho 1

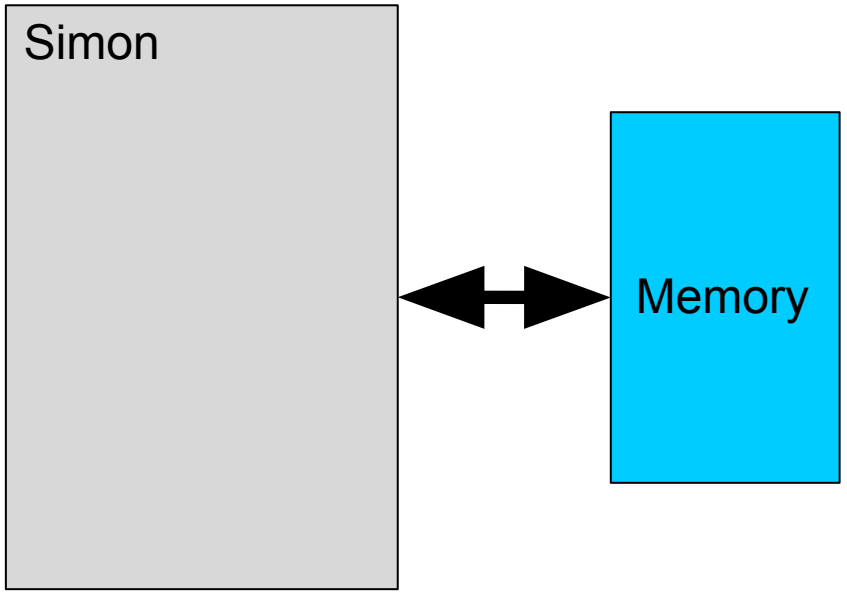
- *Architecture* Comportamental + estrutural
  - *Simon* (Simon.vhd)
    - Estrutural (ligação entre *DataPath* e *ControlPath*)
    - Utilizar as *records* definidas no Simon_pkg na conexão dos blocos



Trabalho 1

- *Architecture* Totalmente comportamental
  - *Simon* (Simon.vhd)
    - Totalmente comportamental
    - Um único *process*
    - O tempo de execução em ciclos de *clock* e o grafo de estados devem ser exatamente iguais aos da *architecture* comportamental + estrutural

Não há necessidade de implementar o sincronizador de botão visto que o *test bench* deve gerar pulsos sincronizados nas entradas dos botões

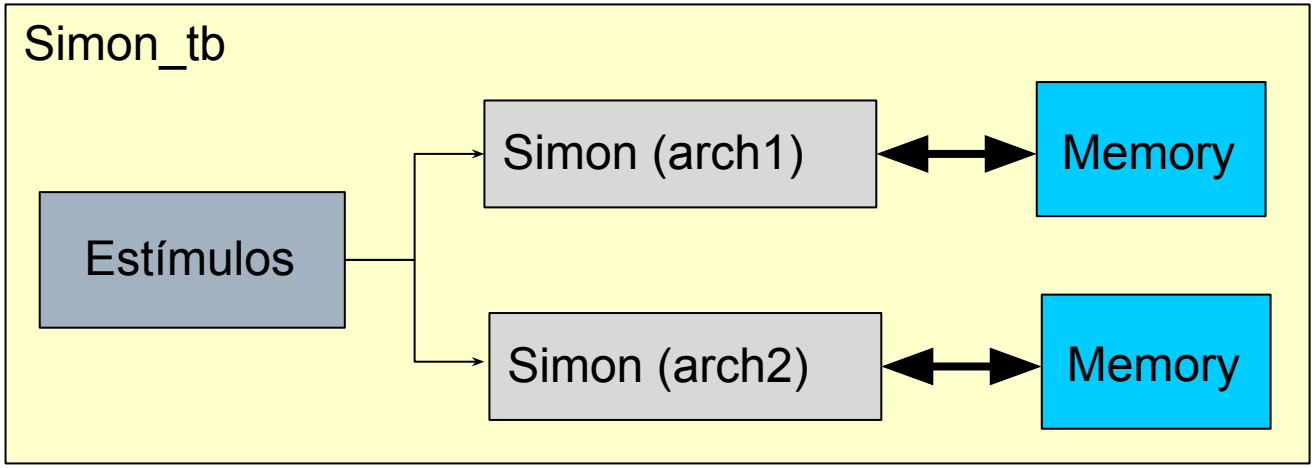


Memória do processador *Find* (Memory.vhd)

Remover inicialização do *Find*

Trabalho 1

- *Simon_tb* (Simon_tb.vhd)
  - Deve existir um único *test bench* no qual serão instanciadas as duas descrições do jogo e suas memórias
  - Os mesmos sinais de estímulos devem ser fornecidos às duas instâncias (*clock, reset, start, cores)*
  - Geração dos estímulos para o processador deve ser sincronizada com o *clk* (utilizar comando *wait until*)
  - Na apresentação, as duas descrições devem ser simuladas simultaneamente a fim de comparar a sincronia
  - Os tempos de execução devem ser idêntidos



Trabalho 1

- *Simon_tb* (Simon_tb.vhd)
  - Para a apresentação, o tempo que as cores ficam acessas/apagadas dever ser de poucos ciclos (1 ou 2) a fim de facilicar a vizualição nas formas de onda
  - Sinais a serem visualizados na apresentação para cada *architecture*: *clock*, *reset*, *start*, entradas dos botões de cores, saídas de cores, *game over* e o estado atual
  - Deve-se preprar um cenário onde o jogador acerta as três primeiras sequências e erra na quarta