****

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Falc-On MM**

**ALUNOS:**

**João Pedro Oliveira Silva - 1201524403**

**Talles Bezerra de Assunção - 1201524410**

**Janeiro de 2018**

**Boa Vista/Roraima**

****

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Falc-On MM**

**Janeiro de 2018**

**Boa Vista/Roraima**

**RESUMO**

**Conteúdo**

[1 Especificação 7](#_Toc444681789)

[1.1 Plataforma de desenvolvimento 7](#_Toc444681790)

[1.2 Conjunto de instruções 8](#_Toc444681791)

[1.3 Descrição do Hardware 9](#_Toc444681792)

[1.3.1 ALU ou ULA 9](#_Toc444681793)

[1.3.2 BDRegister 9](#_Toc444681794)

[1.3.3 Clock 9](#_Toc444681795)

[1.3.4 Controle 9](#_Toc444681796)

[1.3.5 Memória de dados 10](#_Toc444681797)

[1.3.6 Memória de Instruções 10](#_Toc444681798)

[1.3.7 Somador 10](#_Toc444681799)

[1.3.8 And 10](#_Toc444681800)

[1.3.9 Mux\_2x1 10](#_Toc444681801)

[1.3.10 PC 10](#_Toc444681802)

[1.3.11 ZERO 11](#_Toc444681803)

[1.4 Datapath 11](#_Toc444681804)

[2 Simulações e Testes 13](#_Toc444681805)

[3 Considerações finais 14](#_Toc444681806)

**Lista de Figuras**

[Figura 1 - Especificações no Quartus 6](#_Toc444681815)

[Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus 8](#_Toc444681816)

[Figura 19 - Resultado na waveform. 13](#_Toc444681817)

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador XXXX. 7](#_Toc444681822)

[Tabela 2 - Detalhes das flags de controle do processador. 9](#_Toc444681823)

[Tabela 3 - Código Fibonacci para o processador Quantum/EXEMPLO. 12](#_Toc444681824)

# Especificação

O Falc-On MM foi desenvolvido para a disciplina de Arquitetura e Organização de Computadores da Universidade Federal de Roraima no semestre 2017.2. O Processador é multiciclo, ou seja, executa cada tipo de instrução com diferentes ciclos de clock.

## Plataforma de desenvolvimento

Para a implementação do processador Falc-On MM foi utilizado a IDE Quartus Prime Lite Edition versão 17.0 fazendo uma simulação do dispositivo xxxx da família Cyclone V.

Toda a descrição do hardware foi feita utilizando a linguagem VHDL que é usada para facilitar o design de circuitos digitais. A figura 1 mostra as especificações do projeto.



Especificações no Quartus

## Conjunto de instruções

O processador Falc-On MM possui 4 registradores: S0, ... e SN. Assim como 13 formatos de instruções de 8 bits cada, Instruções do **tipo R, tipo I e tipo J**, seguem algumas considerações sobre as estruturas contidas nas instruções:

* **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
* **Reg1**: o registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
* **Reg2**: o registrador contendo o segundo operando fonte;

### Instruções do tipo R:

Abrange instruções de carregamento e gravação de dados na memória primária e instruções baseadas em operações aritméticas. A divisão de bits está descrita na tabela 1 abaixo.

Table 1 - Formato de intruções do tipo R

|  |  |  |
| --- | --- | --- |
| Opcode | Registrador 1 | Registrador 2 |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Instruções do tipo J:

Abrange instruções onde é necessário fazer um salto condicional ou não entre os endereços, como um *if* ou um *goto*, por exemplo. A divisão de bits está descrita na tabela 2 abaixo.

Tabela 2 - Formato de instruções do tipo J

|  |  |
| --- | --- |
| Opcode | Endereço |
| 7-4 | 3-0 |
| 4 bis | 4 bits |

### Instruções do tipo I:

Abrange instruções de gravação de dados na memória primária e operações imediatas. A divisão de bits está descrita na tabela 3 abaixo.

Tabela 3 - Formato de instruções do tipo I

|  |  |  |
| --- | --- | --- |
| Opcode | Registrador 1 | Registrador 2 |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Visão geral das instruções do Processador Falc-On MM:

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () de 16 **Opcodes (0-15)** que são distribuídos entre as instruções, assim como é apresentado na tabela 4.

Tabela 4 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Falc-On MM.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | Nome | Formato | Breve Descrição | Exemplo |
| 0000 | ADD | R | Soma | Add $r1, $r2 -> $r1 = $r1 + $r2; |
| 0001 | SUB | R | Subtração | Sub $r1, $r2 -> $r1 = $r1 - $r2; |
| 0010 | ADDI | I | Soma imediata | Addi $r1, XX -> $r1 = $r1 + 000000XX; |
| 0011 | MULT | R | Multiplicação | Mult $r1, $r2 -> $r1 = $r1 \* $r2; |
| 0100 | AND | R | AND lógico | And $r1, $r2 -> $r1 = $r1 and $r2; |
| 0101 | OR | R | OR lógico | Or $r1, $r2 -> $r1 = $r1 or $r2; |
| 0110 | NOT | R | NOT lógico | Not $r1, $r2 -> $r1 = not $r1, $r2 é ignorado; |
| 0111 | XOR | R | XOR lógico | Xor $r1, $r2 -> $r1 = $r1 xor $r2; |
| 1000 | LW | I | Load word | Lw $r1, $r2 -> $r1 = 0($r2); |
| 1001 | SW | I | Store word | Sw $r1, $r2 -> 0($r2) = $r1; |
| 1010 | LI | I | Load imediato | Li $r1, XX -> $r1 = 000000XX; |
| 1011 | JMP | J | Jump | Jmp XXXX -> salto para XXXX; |
| 1100 | JR | J | Jump registrador | Jr $r1, $r2 -> salto para o valor em $r1, $r2 é ignorado; |

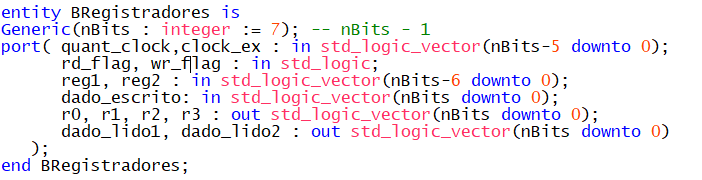
## Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Falc-On MM, incluindo uma descrição de suas funcionalidades, valores de entrada e saída. Segue a descrição de cada um.

### Banco de registradores

O Banco de Registradores do Falc-On MM conta com 4 registradores com 8 bits cada um, como foi citado na sessão **1.2 Conjunto de Instruções**. Foi declarado com 7 pinos de entrada e 6 pinos de saída.

Figure 1 - Declaração do banco de registradores



* quant\_clock –
* rd\_flag –
* wr\_flag –
* reg1 e reg2 –
* dado\_escrito –
* r0,...,r3 –
* dado\_lido1 e dado\_lido2 –

### Extensores de sinais

Existem dois extensors de sinal, um que recebe 2 bits e outro que recebe 4 bits, ambos transformam a entrada em uma saída de 16 bits. Os extensores de 2 e 4 bits para 8 bits vão preencher os bits extras com 0’s.

Figure 2 - Declaração de um extensor de sinal de 2 bits para 8 bits

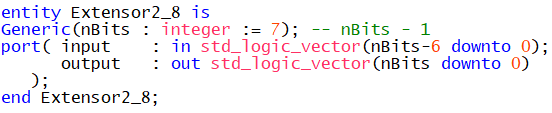
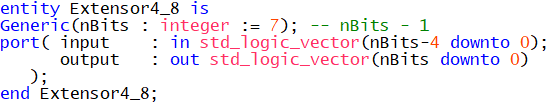


Figure 3 - Declaração de um extensor de sinal de 4 bits para 8 bits

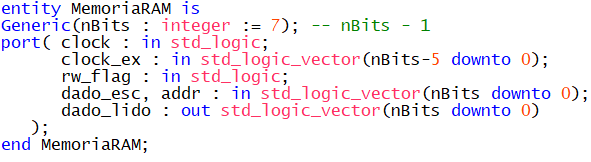


* **input** – Sinal de entrada em 2 / 4 bits
* **output** – Sinal de saída em 8 bits

### Memória de dados

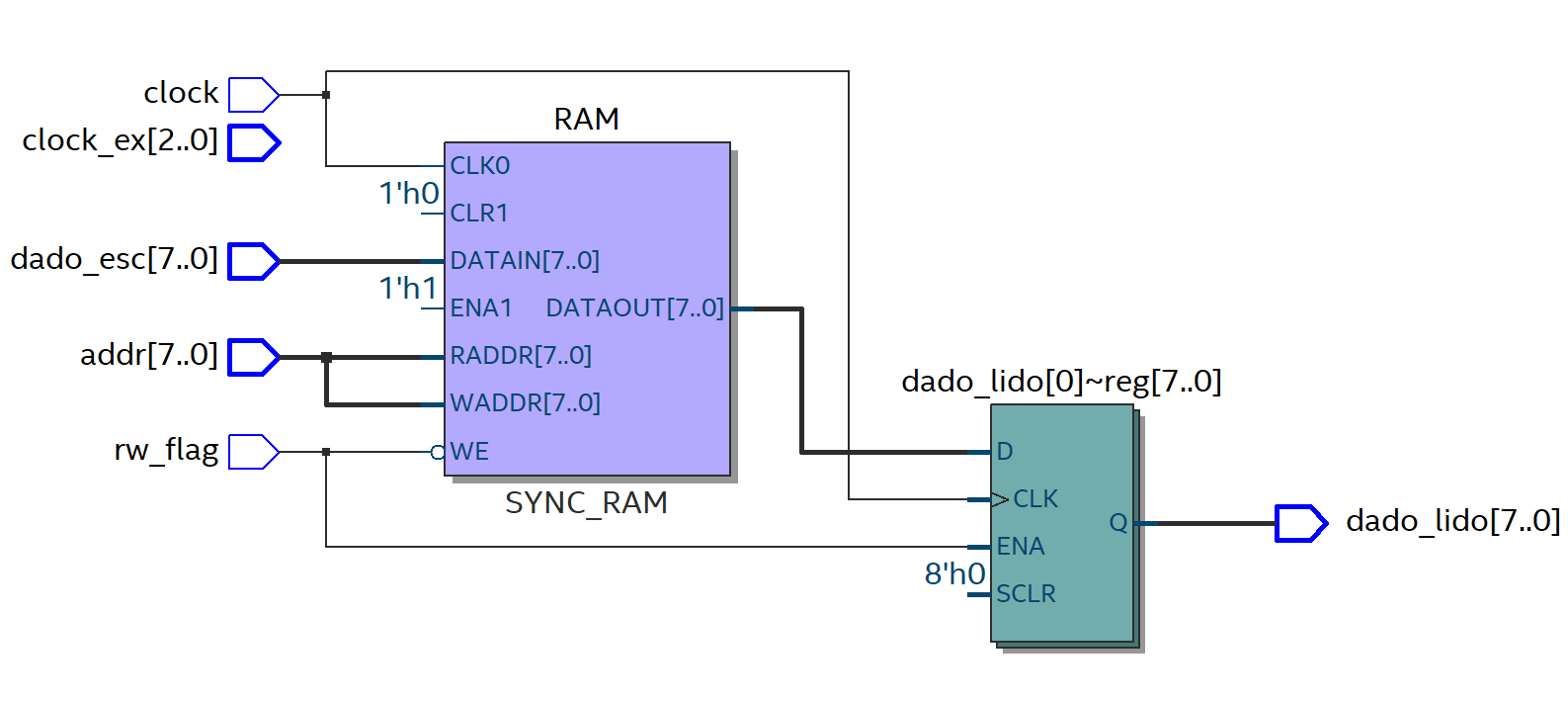
A memória de dados foi implementada como uma memória síncrona de acesso dinâmico aleatório, ou seja, é uma memória sincronizada com o barramento do sistema com a transição de subida do clock.

Figure 4 - Declaração da memória de dados



* **clock** – sinal do clock do processador
* **rw\_flag** – flag para liberar o acesso a memória de dados
* **dado\_esc** – dado a ser escrito
* **addr** – endereço de memória que está sendo escrito
* **dado\_lido** – dado escrito na memória no endereço **addr**

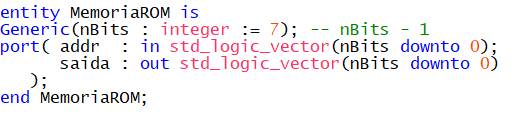
Figure 5 - Vizualização da memória de dados



### Memória de instruções

A memória de instruções foi implementada com uma memória de somente leitura, ou seja, suas informações são gravadas pelo fabricante uma única vez e após isso não podem ser alteradas ou apagadas, somente acessadas.

Figure 6 - Declaração da memória de instruções

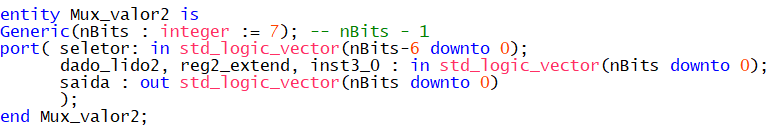


* **addr** – Endereço do qual se desejar ler
* **saida** – valor armazenado na memória no endereço **addr**

### Multiplexadores

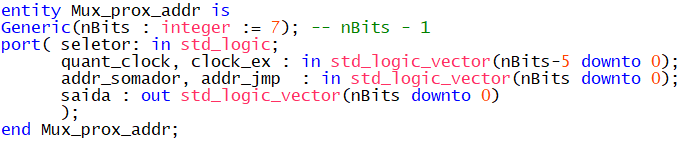
Existem 3 multiplexadores, um para saber qual o tipo de instrução será utilizado (**Tipo** **R**, **J** ou **I**), outro para incrementar para ir para a próxima instrução e outro para escrever o dado do registrador no banco de registradores.

Figure 7 - Declaração do multiplexador para saber o tipo de instrução à ser utilizada



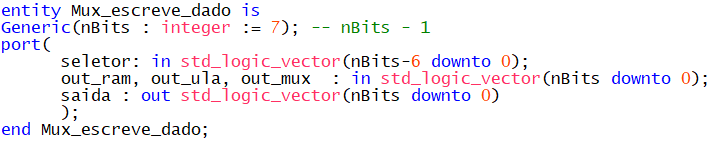
* selector –
* dado\_lido2 –
* reg\_extend –
* inst3\_0 –
* saida –

Figure 8 - Declaração do multiplexador para incrementar para a próxima instrução



* selector – seletor de saída que define qual das entradas será a saída
* quant\_clock –
* addr\_somador –
* addr\_jmp –
* saida – sinal de saída contendo

Figure 9 - Declaração do multiplexador que escreve o dado do registrador no banco de registradores

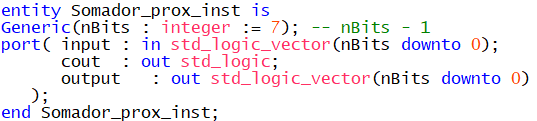


* seletor – seletor de saída que define qual das entradas será a saída
* out\_ram – entrada que recebe a saída da memória de dados
* out\_ula – entrada que recebe a saída da ULA
* out\_mux – entrada que recebe a saída do multiplexador descrito na figura 6
* saida – sinal de saída contendo o valor que está no registrador para graver no banco de registradores

### Somador

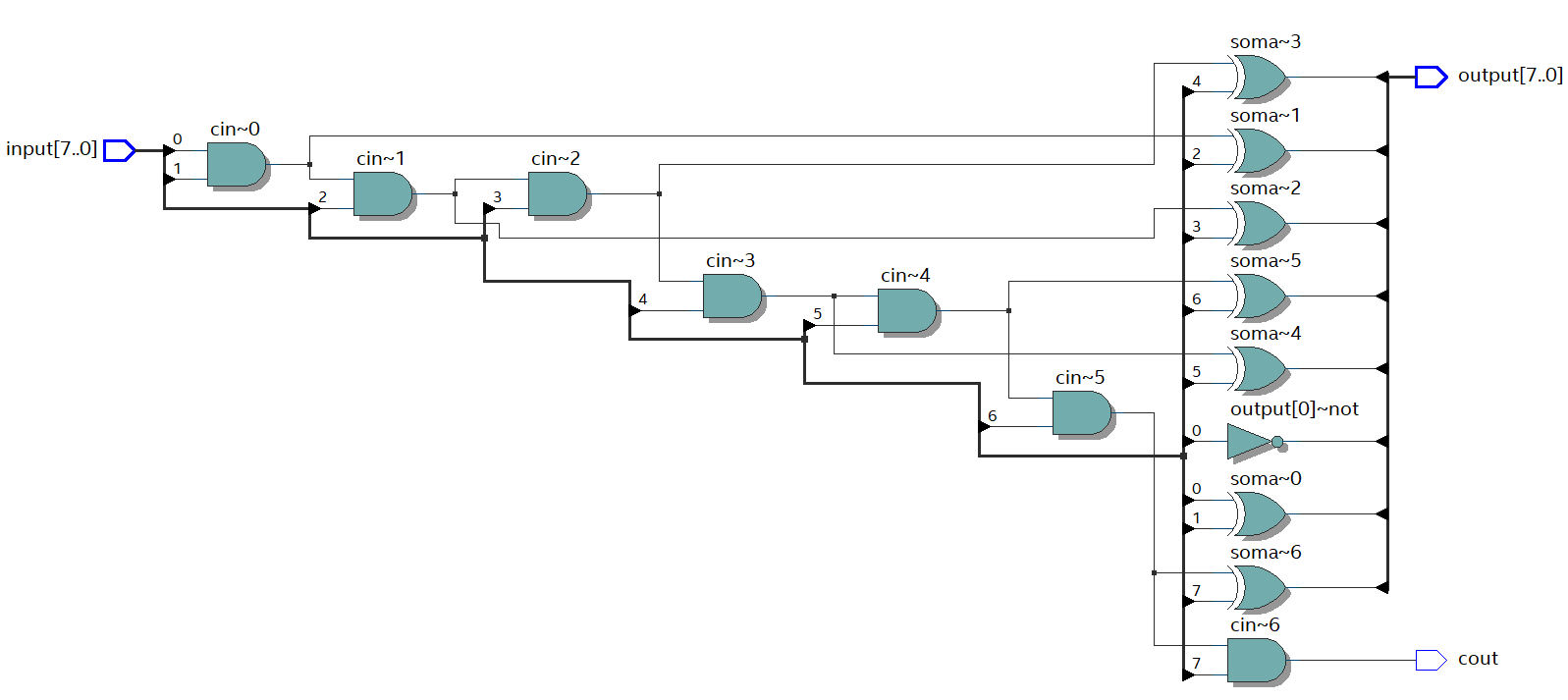
Um somador simples para ir a próxima instrução.

Figure 10 - Declaração do somador



* input – Valor de entrada
* cout – Indicador de *overflow*
* output – Resultado da soma

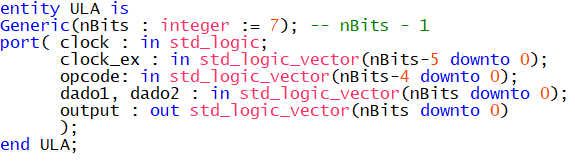
Figure 11 - Visualização do somador



### ULA

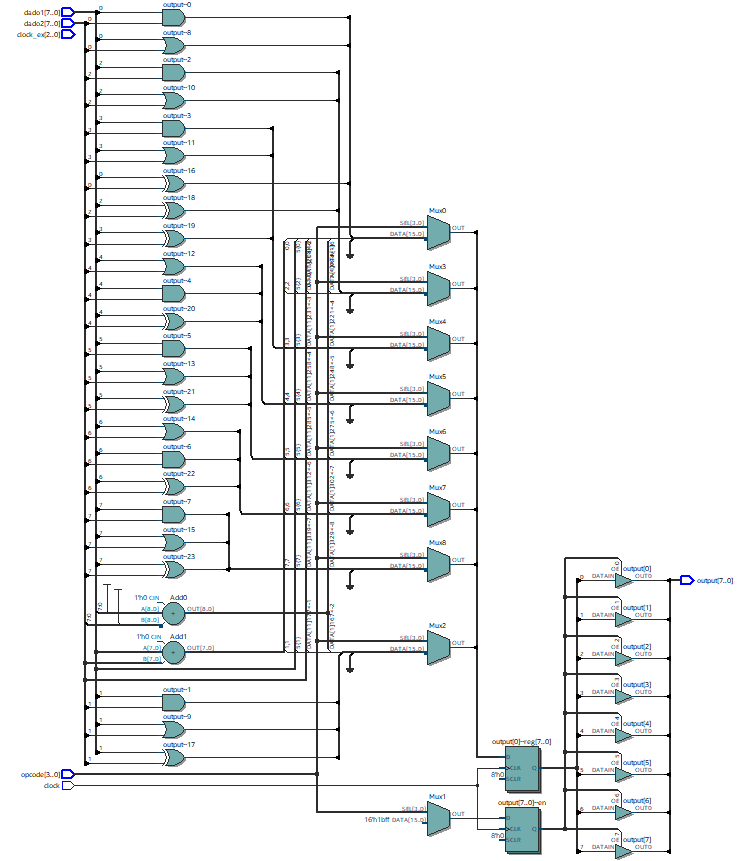
A Unidade Lógica e Aritmética (ULA) desenvolvida conta com 13 operações aritméticas, todas listadas na tabela 4.

Figure 12 - Declaração da Unidade Lógica e Aritmética



* clock – sinal do clock do processador
* opcode – seletor de operação
* dado1 e dado2 – operandos
* output – resultado da operação aritmética definida por **opcode**

Figure 13 - Vizualização da Unidade Lógica e Aritmética

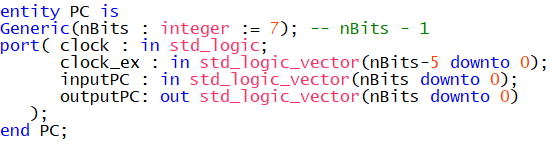


Na vizualização da ULA, Podemos ver que a mesma é composta por xxx componentes 7 componentes diferentes, entre eles temos: 8 portas lógicas simples AND, 8 portas lógicas simples OR, 8 portas lógicas simples XOR, 2 somadores, 9 multiplexadores, 2 registradores D Flip-Flop de 8 bits e 8 buffers.

### PC

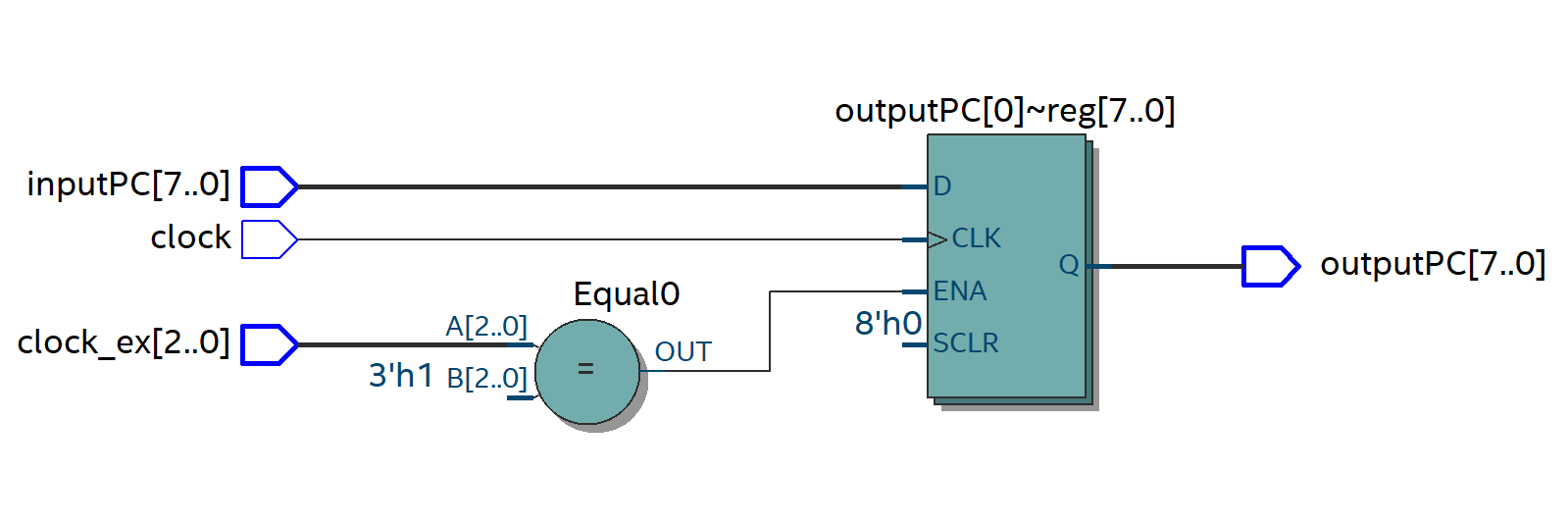
O Contador do programa (PC) é apenas um registrador D Flip-Flop, ele conta com 2 pinos de entrada, o inputPC de 8 bits e o clock, e 1 pino de saída de 8 bits.

Figure 14 - Declaração do contador do programa



* clock – Sinal de clock do processador
* inputPC – Valor de entrada
* outputPC – Valor armazenado no PC

Figure 15 - Visualização do contador do programa



### Unidade de controle

## Datapath

<AquiDatapath>

# Simulações e Testes

# Considerações finais