
PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

# RELATÓRIO DO PROJETO: PROCESSADOR KL

ALUNOS:
Kevin Costa Aires Oliveira - 2201324412
Larissa Santos Silva - 1201324456

Março de 2016
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 KL

Março de 2016
Boa Vista/Roraima

**Resumo**

Esse projeto aborda a implementação de um processador baseado na arquitetura Mips. O processador KL apresenta as diversos componentes necessários para que o mesmo funcione e apresenta a caracteristica de ter memórias separadas, ou seja, memória para dados e memória para instruções; sendo este componentes a ULA, a unidade de controle, memória de dado, memória de instrução, extensor de sinal, multiplexador de duas entradas e o Program Counter ou PC.

Cada componente tem uma caracteristica especifica, onde a unidade de controle vai enviar os sinais pelas flags para que de acordo com a instrução que deve operar, o PC apresenta a união a um somador, para que o mesmo possa fazer o calculo da procima instrução, As memórias servem para armazenar as informações, sendo uma para os dados que serão utilizados e outro para as instruções que devem ser executadas. A ULA apresenta as operações basicas, com exceção da divisão, além de um componente para comparar se é maior ou maior e outro para comparar se o mesmo é igual a zero, semelhante ao $zero usado no mips.

O processador KL infelizmente apresenta um problema relacionado a salto incondicional, onde o mesmo não possui um componente que possa ser utilizado para que o mesmo efetue este tipo de operação. Em outras palavras, os componentes de modo separado funcionam de maneira correta, porém quando estão interligados apresentam problemas.

**Conteúdo**

**Lista de Figuras**



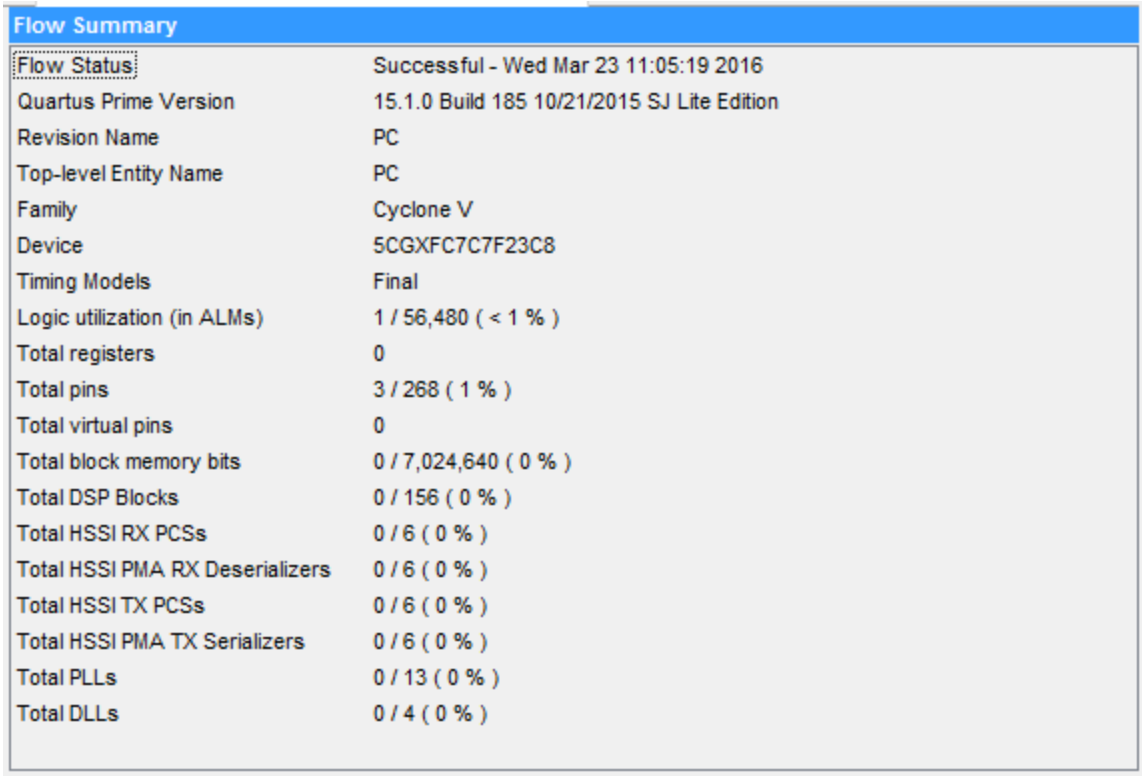
**Lista de Tabelas**

## 1. Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador KL, bem como a descrição detalhada de cada etapa da construção do processador.

### 1.1 Plataforma de desenvolvimento

Para a implementação do processador KL foi utilizado a IDE: Quartus Prime Vesion 15.1.0.



| Flow Summary | |
|---|---|
| Flow Status | Successful - Wed Mar 23 11:05:19 2016 |
| Quartus Prime Version | 15.1.0 Build 185 10/21/2015 SJ Lite Edition |
| Revision Name | PC |
| Top-level Entity Name | PC |
| Family | Cyclone V |
| Device | 5CGXFC7C7F23C8 |
| Timing Models | Final |
| Logic utilization (in ALMs) | 1 / 56,480 ( < 1 % ) |
| Total registers | 0 |
| Total pins | 3 / 268 ( 1 % ) |
| Total virtual pins | 0 |
| Total block memory bits | 0 / 7,024,640 ( 0 % ) |
| Total DSP Blocks | 0 / 156 ( 0 % ) |
| Total HSSI RX PCSs | 0 / 6 ( 0 % ) |
| Total HSSI PMA RX Deserializers | 0 / 6 ( 0 % ) |
| Total HSSI TX PCSs | 0 / 6 ( 0 % ) |
| Total HSSI PMA TX Serializers | 0 / 6 ( 0 % ) |
| Total PLLs | 0 / 13 ( 0 % ) |
| Total DLLs | 0 / 4 ( 0 % ) |

Figura 1 - Especificações no Quartus

### 1.2 Conjunto de Instruções

O processador KL possui 04 registradores: S0, ... e SN. Assim como 02 formatos de instruções de 8 bits cada, Instruções do tipo R (opção aritmeticas ) , I (loard e store), seguem algumas considerações sobre as estruturas contidas nas instruções:

**Opcode:** a operação básica a ser executada pelo processador, tradicionalmente

**Reg1:** o registrador contendo o primeiro operando fonte e adicionalmente para

**Reg2:** o registrador contendo o segundo operando fonte;

Tipo de Instruções:

- Formato do tipo R: Este formatado aborda instruções de Load (exceto load Immediately), Store e instruções baseadas em operações aritméticas.

Formato para escrita de código na linguagem Quantum:

| Tipo de Instrução | Reg1 | Reg2 |
|---|---|---|

Formato para escrita em código binário:

| Tipo da Instrução | 1° Registrador | 2° Registrador |
|---|---|---|
| 4 Bits | 2 Bits | 2 Bits |
| 7-4 | 3-2 | 1-0 |

**TABELA 1 – TABELA QUE MOSTRA A LISTA DE OPCODES UTILIZADAS PELO PROCESSADOR KL.**

| Opcode | Breve descrição | Nome | Formato | Exemplo |
|---|---|---|---|---|
| 0000 | SOMA | ADD | R | ADD $s0,$s1 \| $s0 + $s1 |
| 0001 | SUBTRAÇÃO | SUB | R | SUB $s0,$s1 \| $s0 - $s1 |
| 0010 | AND | AND | R | AND $s0,$s1 |
| 0011 | OR | OR | R | OR $s0,$s1 |
| 0100 | MULTIPLICAÇÃO | MULT | R | MULT $s0,$s1 \| $s0 * $s1 |
| 0101 | BEQ | BEQ | I | BEQ $s0,$s1 \| $s0 < $s1 |
| 0110 | STL | SLT | I | STL $s0,$s1 \| $s0 == $s1 |
| 0111 | STL/ZERO | SZ | I | SZ $s0,$s1 \| $s0 == 0 |
| 1000 | MOVE | MOVE | I | MOVE $s0,$s1 \| $s0 = $s1 |

| | | | | |
|---|---|---|---|---|
| 1001 | LOAD IMMED. | LI | I | LI $s0,2 \| $s0 = 2 |
| 1010 | LOAD | LW | I | LW $s0,12($s2) |
| 1011 | STORE | SW | I | SW $s0,12($s2) |
| 1100 | JUMP | JUMP | J | JUMP 0011 |
| 1111 | *EXIT | | | É um elemento especial para encerrar o programa. É representado juntamente com o JUMP. J<br>UMP EXIT. |

### 1.3 Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Quantum, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

#### 1.3.1 ALU ou ULA

O componente ULA (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: OR, AND, soma, subtração. Adicionalmente o ULA efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente ULA recebe como entrada três valores: dadoA – dado de 8 bits para operação; dadoB - dado de 8 bits para operação e seletorResult– identificador da operação que será realizada de 2 bits. O ULA também possui 1 saída: – de 8 btis na qual mostra o resultado da operação.

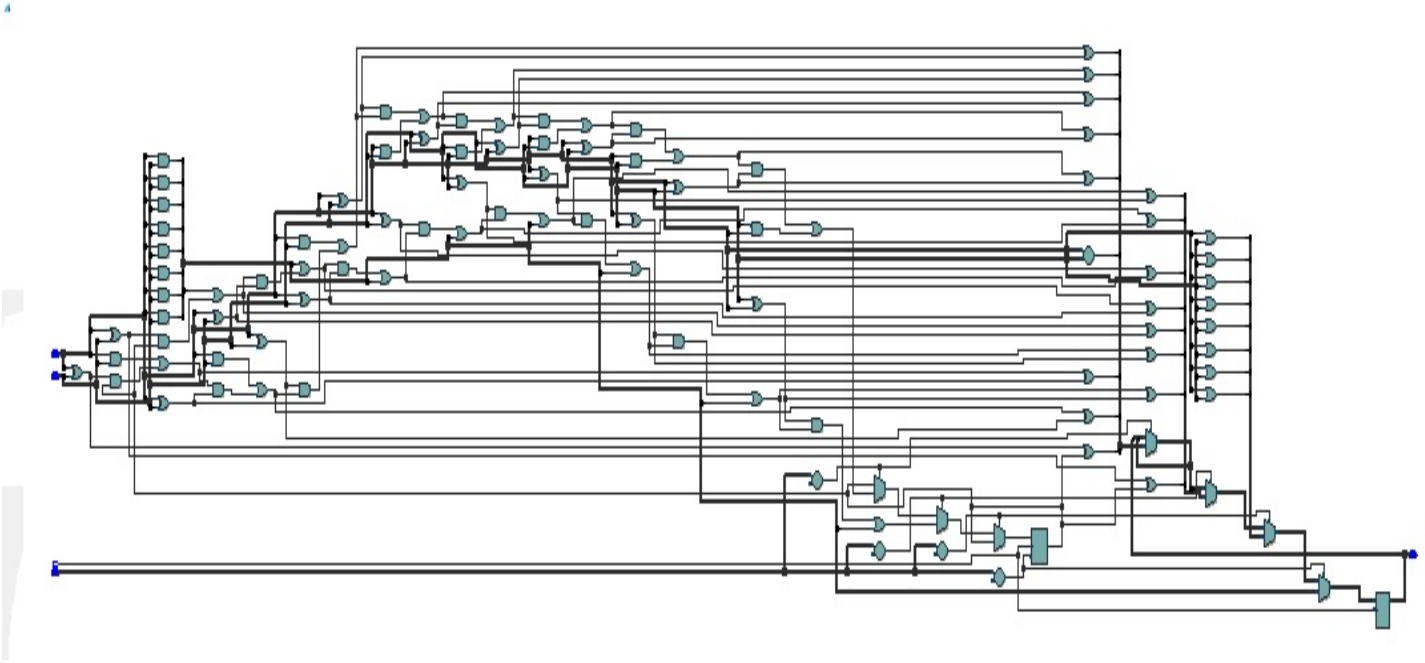


Figura 2 - Bloco simbólico do componente ALU gerado pelo Quartus

### 1.3.2 Banco de Registrador

O registrador é um componente digital composto por um conjunto de registradores que podem ser acessados de forma organizada. De uma maneira geral, podem ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas. As informações que estão sendo processado em um determinado momento devem estar armazenadas no banco de registradores. Possuindo assim: dado - com 8 bits responsavel pelo o armazenamento de dados; saida - possui 8 bits e mostra o resultado.

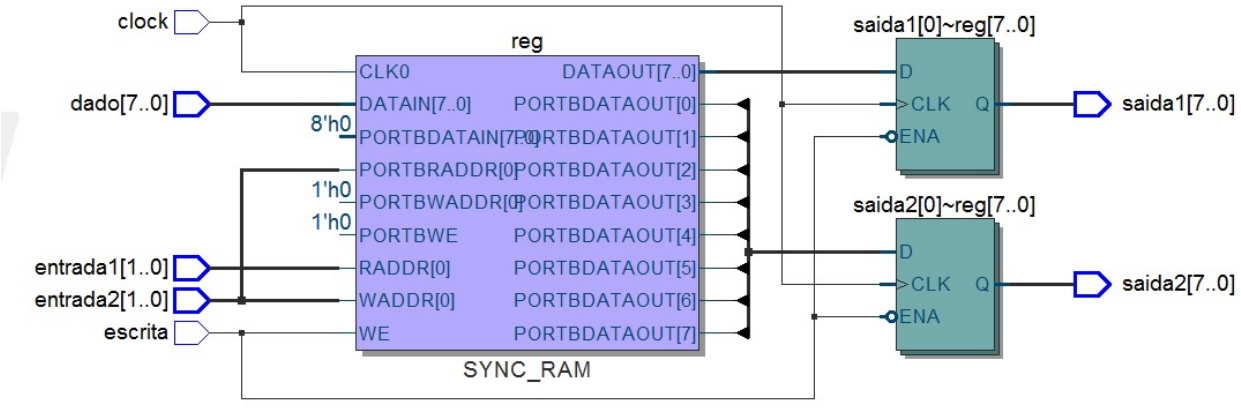



Figura 3 - Bloco simbólico do componente registrador gerado pelo Quartus

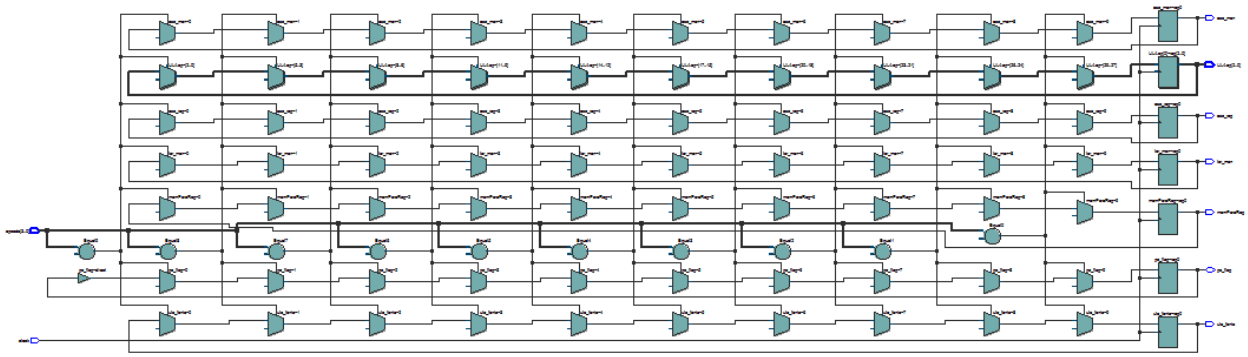
### 1.3.3 Clock

O clock é responsavel por coordenar ações de dois ou mais componentes. Um sinal de clock oscila entre os estados alto e baixo (no estado de 1 e 0), normalmente usando um para definir algum tipo de operação.

### 1.3.4 Controle

A unidade de controle é responsável por enviar os comando para as demais unidade. Este comando são necessários para a execução de uma determinada instruçã. Para isso este componente tem como entrada o Opcode responsável pela instrução, ou seja, para a instrução de soma o opcode que será entrada da unidade de controle será o 0000. Com a entrada do opcode é ativado as flags de saida e com isso os demais componente serão “configurados” para que a operação de soma seja executada.

| Flag de controle | Necessidade |
|---|---|
| pc_flag | Flag utilizada para que ativar e desativar o PC |
| ler_mem | Quando 1 habilita que alguma informação possa ser acessada dentro da memoria de dados. |
| esc_mem | Quando 1 habilita que alguma informação possa ser escrita dentro da memoria de dados. |
| memParaReg | Quando 0 faz com que uma informação saida da ula seja enviada direto para o registrador. Quando 1 permite que um dado saia pa memoria para um registrador |
| ULAop | Define qual operação deve ser realizada pela ULA |
| esc_reg | Quando for 1 permite que uma informação seja escrita na ula |
| ula_fonte | Quando 0 permite que o segundo dado vindo do Banco de registradores seja acessado pela ULA. Quando 0 permite que seja que o dado vindo do extensor de sinal seja enviado a ULA. |



### 1.3.5 Memória de dados

Mem_dado e usada para pelo processador para armazenar os arquivos e programas que estão sendo processada. A principal característica da memoria de dados é que ela é volátil, ou seja, os dados se perdem ao reiniciar o computador. Ao ligar é necessário refazer todo o processo de carregamento, em que o sistema operacional e aplicativos usados são transferidos do HD para a memória, onde podem ser executados pelo processador.

A mem_dado possui os seguintes componentes: clock - possui 1 bits e atua como um sinal para sincronismo; reset - 1 bits e usado para reiniciar; leitura - 1 bits responsavel pela leitura do dado; escrita - 1 bits recebe os dados escritos; endereco - 8 bits ele pode armazenar dados ou buscar um determinado valor, depende da instrução; dado - 8 bits responsavel pelo o armazenamento de dados ; saida - 8 bits, onde mostra o valor sinal da operação.

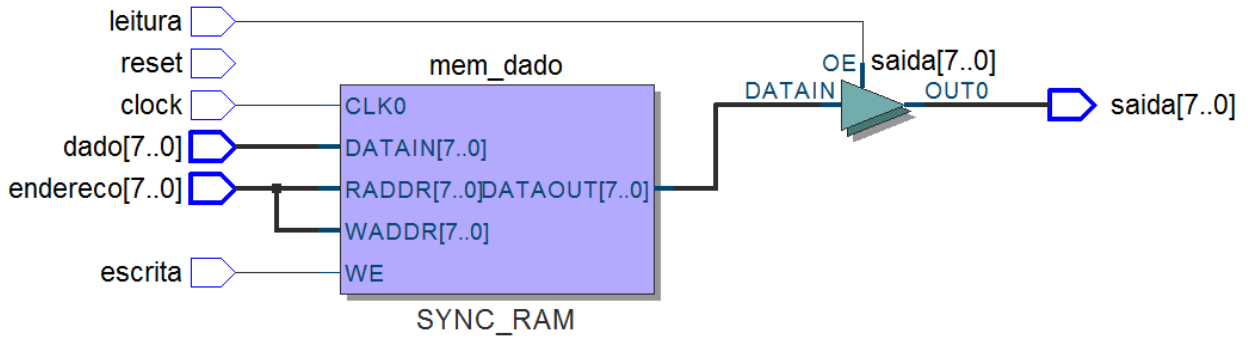



Figura 4 - Bloco simbólico do componente da mem_dado gerado pelo Quartus

### 1.3.6 Memória de Instruções

A mem_instrucao é um termo genérico usado para designar as partes do computador ou dos dispositivos periféricos onde os dados e programas são armazenados. Sem uma memória de onde os processadores podem ler e escrever informações, não haveria nenhum computador digital de programa armazenado. Sendo formada por: 2 reg - com 2 bits responsavel pelo conjunto de dados armazenados; opcode - possui 4 bits para conseguir realizar determinadas tarefas; endereco - com 8 bits para armazenar dados.

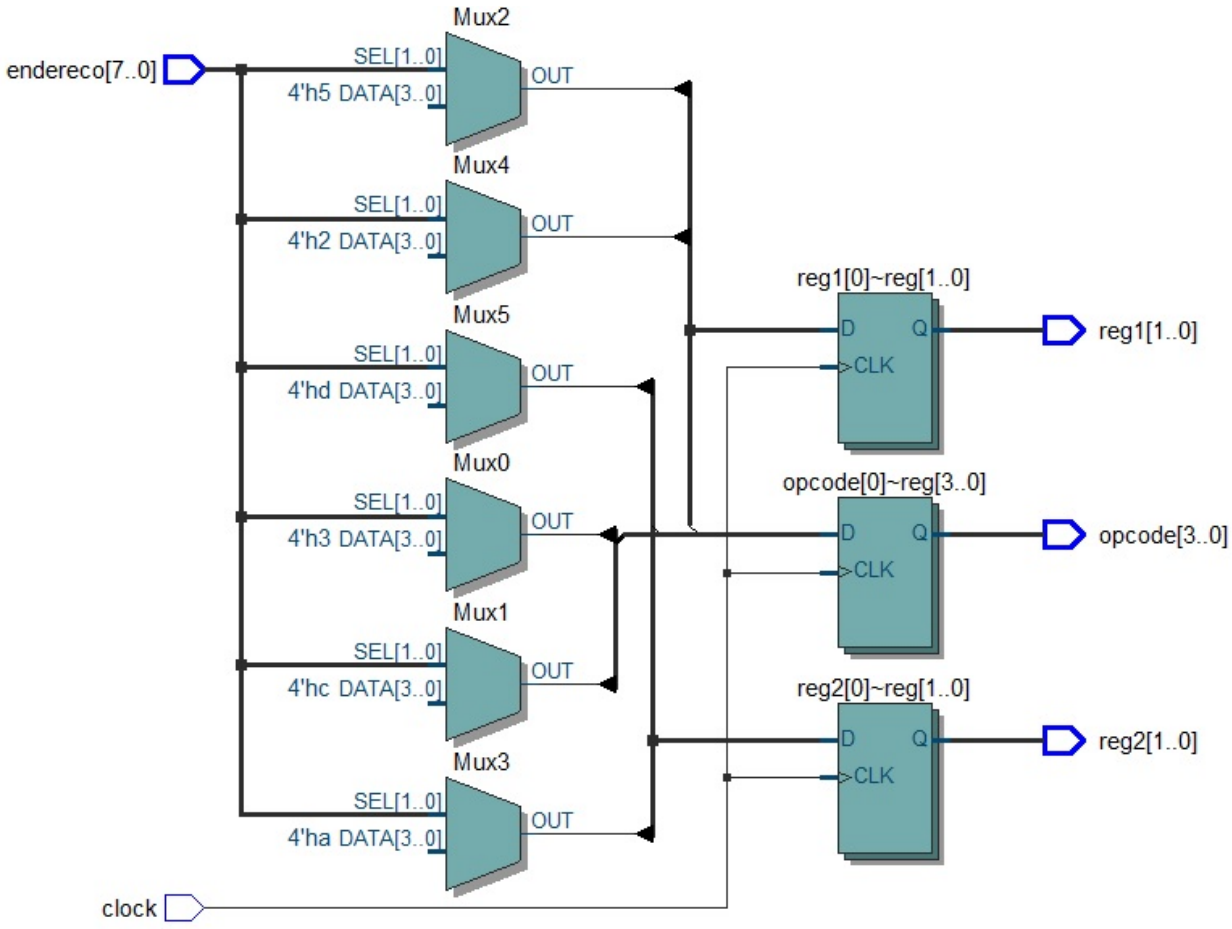


Figura 5 - Bloco simbólico do componente da mem_instrucao gerado pelo Quartus

### 1.3.9 Mux_2x1

O multiplex2x1 e responsavel por seleciona as informações de duas ou mais fontes de dados em uma unica saida. Contendo assim: dadoA – dado de 8 bits para operação; dadoB - dado de 8 bits para operação e seletorResult– identificador da operação que será realizada de 2 bits; saida - contendo 8 bits a saida e responsavel pelo valor final da operação.

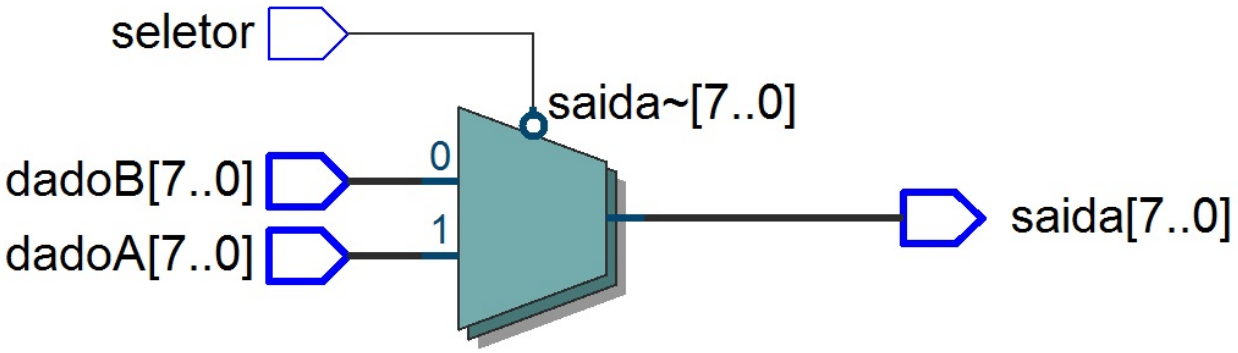



Figura 6 - Bloco simbólico do componente da multiplex2x1 gerado pelo Quartus

### 1.3.10 PC

O pc é a unidade responsavel por armazenar instruções a ser executada, alem disso pode ser feita algumas operações para descobrir a proxima instrução, pois tem um somador integrado. Sendo assim: clock - possui 1 bits e atua como um sinal para sincronismo; reset - 1 bits e usado para reiniciar; flag - 1 bits e usado como interruptor (isto é, valores 1/0, ligado/desligado, ativo/inativo) permite otimizar as estruturas de dados; entrada - 8 bits e uma informação que é recebida e processada; saida - 8 bits mostra a saida do dado processado ; saida2 - 8 bits mostra a saida do dado processado.

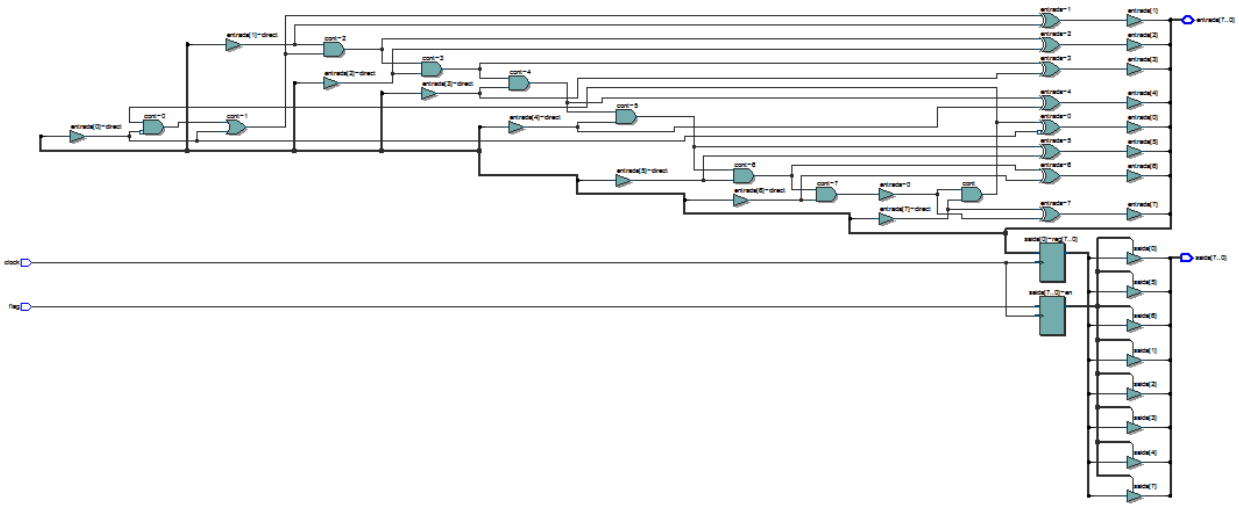


Figura 7  - Bloco simbólico do componente da pc gerado pelo Quartus

### 1.4 Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.

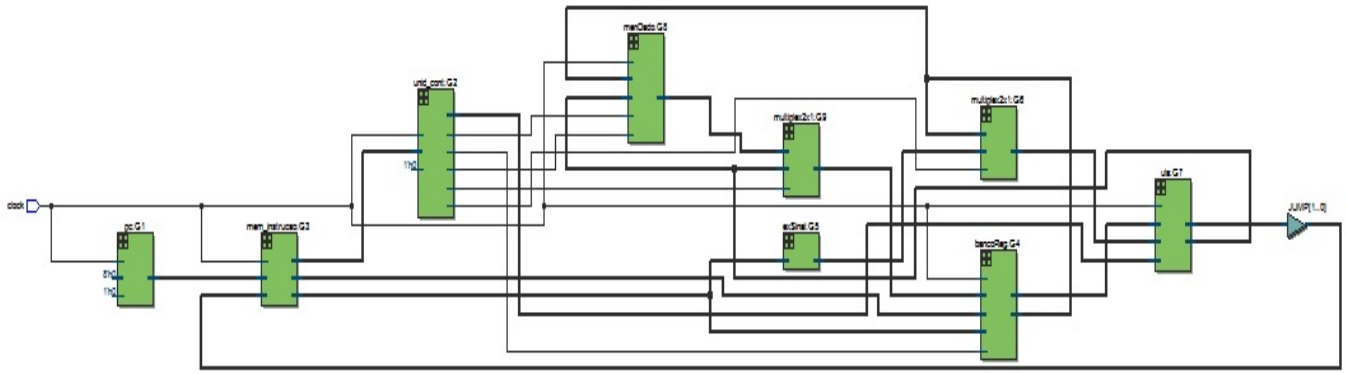


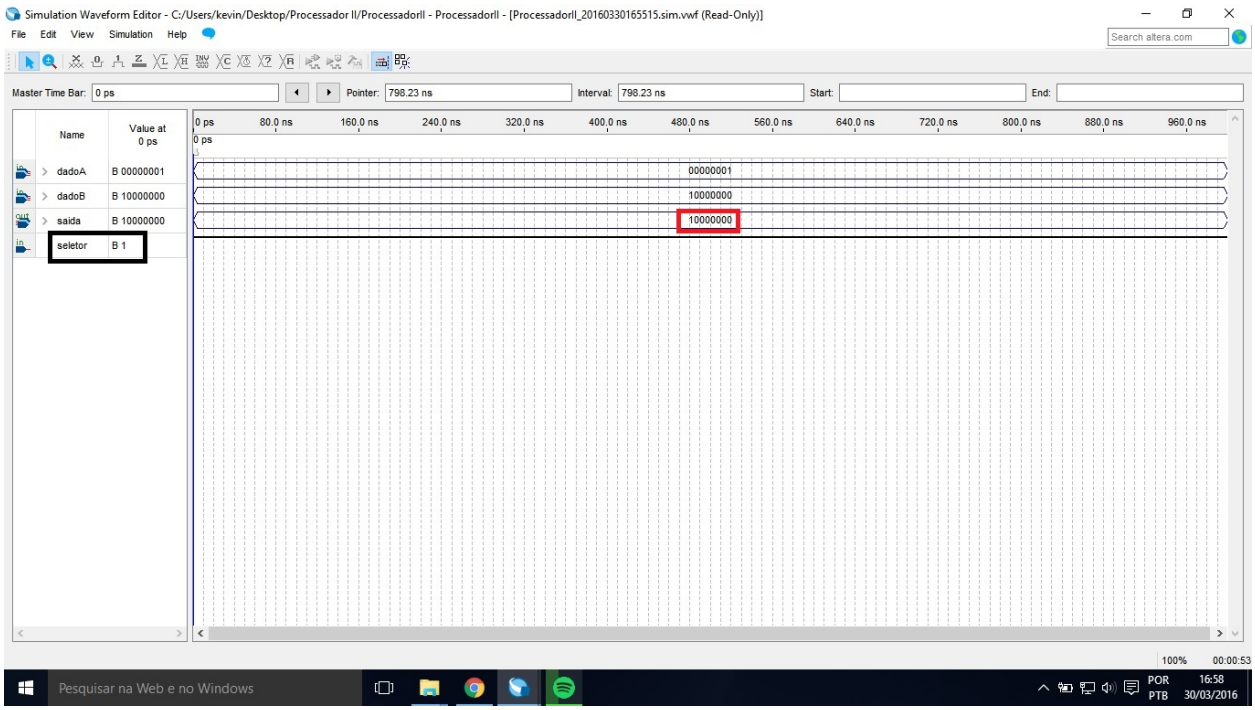
Figura 8 - Bloco simbólico do componente da Datapath gerado pelo Quartus

## 2. Simulações e Testes

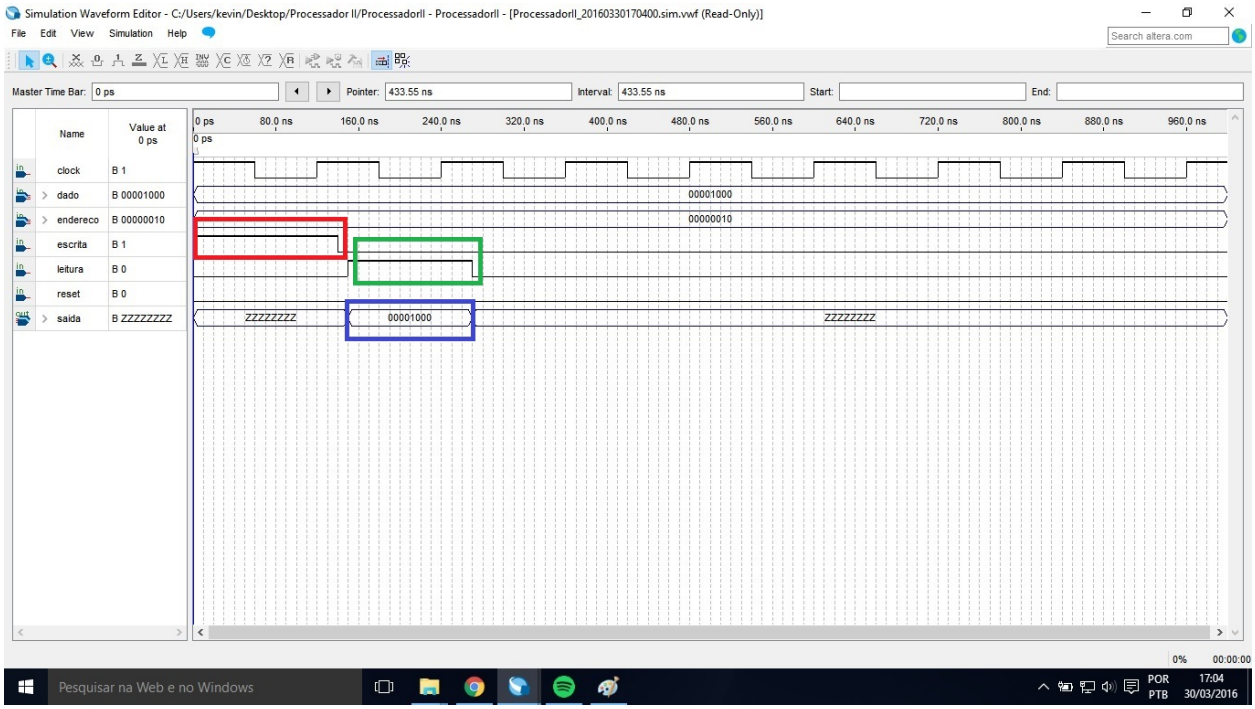
Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador KL, mais os testes foram realizados sem sucesso.

| **Endereço** | **Linguagem de alto nivel** | **Binário** | | |
|---|---|---|---|---|
| | | **Opcode** | **Reg1** | **Reg2** |
| | | | **Endereço** | |
| 1 | LI $s0,2 | 1001 | 00 | 00 |
| 2 | LI $s1,4 | 1001 | 01 | 00 |
| 3 | add $s2,$s0 | 0000 | 10 | 00 |
| 4 | add $s2,$s1 | 0000 | 10 | 01 |
| 5 | slt $s2,$s0 | 0110 | 10 | 00 |
| 6 | jump 0010 | 1100 | 0010 | |
| 7 | jump 0011 | 1100 | 0011 | |
| 8 | move $s2,$s0 | 0111 | 10 | 00 |
| 9 | jump exit | 1100 | 1111 | |

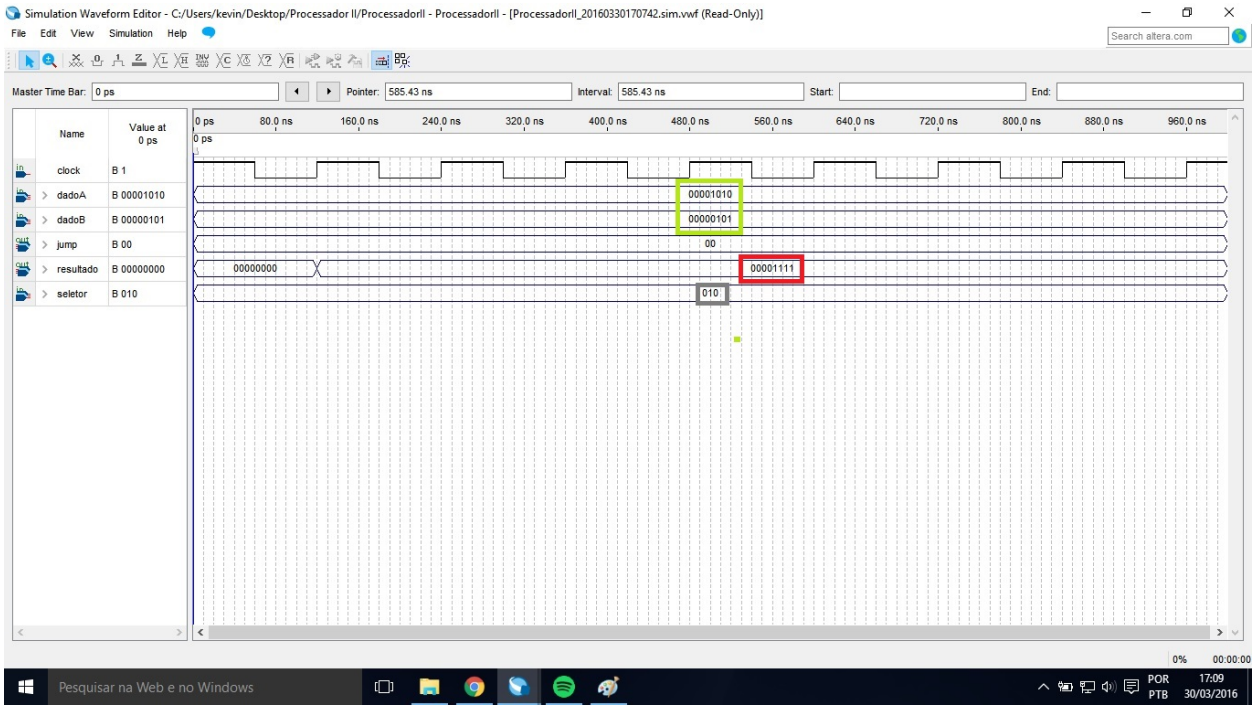
| | | | | |
|---|---|---|---|---|
| 10 | move $s2,$s1 | 0111 | 10 | 01 |
| 11 | jump exit | 1100 | 1111 | |




Teste multiplexador
Preto mostra o seletor e vermelho, o resultado

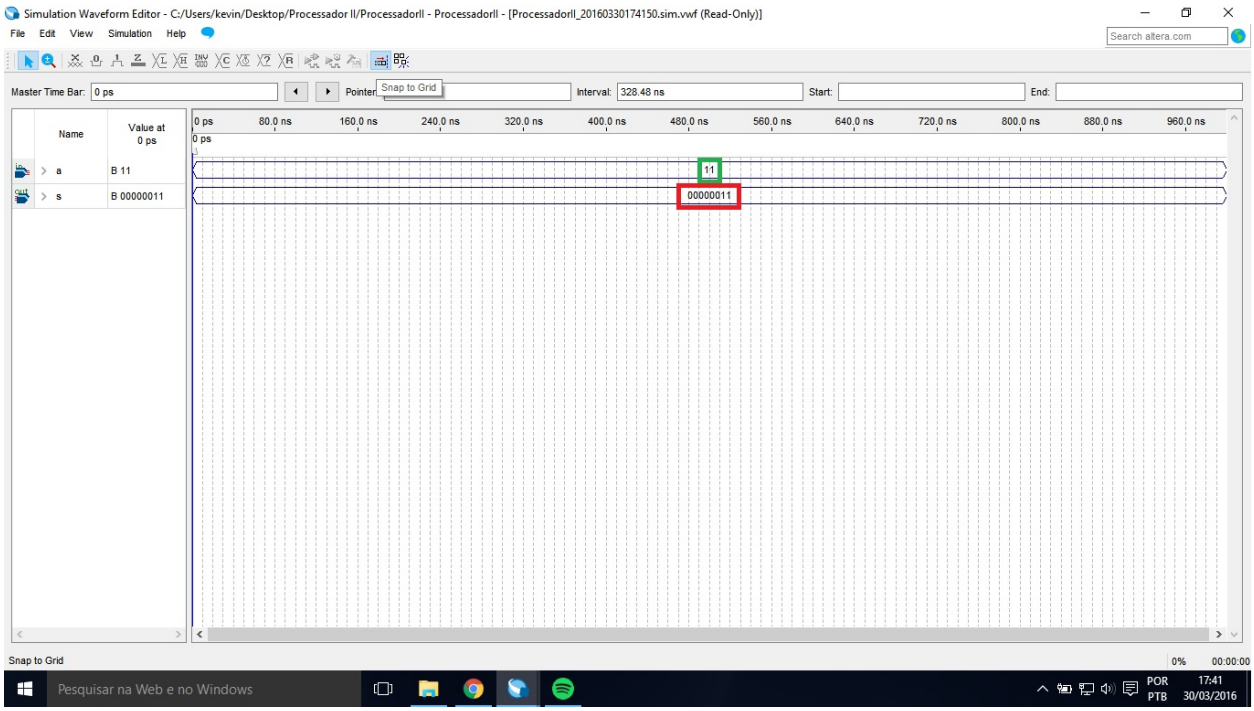



Teste memoria de dados
Vermelho: escrita ativa
Verde: leitura ativa
Azul: resultado




Teste ula

Verde: dados de entrada
Vermelho: resultado
Cinza: Seleção da operação, neste caso a soma




Teste extensor de sinal
Verde: valor a receber uma extenção
Vermelho: valor extendido.