Universidade Federal do Rio Grande do Norte- UFRN
Centro de Tecnologia
Departamento de Engenharia de Computação e Automação

Primeira avaliação

(Projeto de um processador programável)

Disciplina: : DCA0119 – Sistemas Digitais.
Professor: Heitor Medeiros Florencio.
Grupo: Alexandre Luz Xavier da Costa
Ana Jamile D. Barbosa
Jaime Cristalino Jales Dantas
Ramon Fava Souza

Natal, 21 de março de 2016

- **Objetivo**

Desenvolver em VHDL o projeto de um processador (múltiplos ciclos) programável de 16 bits, sendo os objetivos específicos

explicitados abaixo.

1. Projetar o caminho de dados (datapath) e o controlador de um processador programável de 16 bits.
2. Escolher e documentar um conjunto de instruções para o processador. Obs.: Instruções obrigatórias: ADD, SUB, AND, OR, NOT, LDA e STA.
3. Escolher o formato das instruções do processador.
4. Identificar os ciclos do processador projetado e descrever a temporização e os sinais gerador pelo controlador em cada ciclo de clock. Além de apresentar a máquina de estado do controlador.
5. Escrever em VHDL o datapath e o controlador do processador.
6. Testar o conjunto de instruções do processador (simulado).

- **Introdução**

Processadores programáveis, também conhecidos como processadores de propósito geral, são uma classe de circuitos digitais amplamente conhecida, cuja tarefa de processamento fica armazenada em uma memória ao invés de ser construída no próprio circuito, como é o caso dos processadores de proposito único.

Esse diferencial dos processadores programáveis em relação aos de proposito único, faz com que eles sejam capazes de serem construídos em massa e então programado para fazer inúmeras tarefas distintas. Sendo assim, o mesmo processador programável pode ser utilizado em diferentes aplicações, programando o processador para realizar a tarefa almejada de processamento e, consequentemente, o custo de projeto da fabricação de tais processadores em massa é amortizado.

- **Arquitetura Básica**

Um processador programável é constituído por duas partes principais, sendo estas um bloco operacional, também chamado de datapath) e uma unidade de controle.

O bloco operacional executa três ações básicas: carga de dados, transformação desses dados e armazenamento de novos dados. A carga consiste na leitura dos dados que serão utilizados, sendo estes vindos de alguns locais de entrada. A transformação é a realização de computações, operações, com esses dados, gerando

novos dados. Por fim, o armazenamento consiste em escrever os novos dados resultantes em alguns locais de saída.

A fim de guardar todos os dados, tanto de entrada como de saída, que o processador precisar acessar, faz-se necessária uma memoria de dados. Um processador programável deve carregar os dados da memoria em um dos diversos registradores do banco de registradores. Deve também ser capaz de alimentar unidades funcionais que por meio de um subconjunto de registradores executam operações de transformação dos dados (ALU). Os resultados das transformações devem ser carregados de volta no banco de registradores e, por fim, é preciso que o processador seja capaz de armazenar os dados dos registradores colocando-os na de volta na memoria.

Assim como foi descrito acima, a figura abaixo (Figura 01) retrata o circuito básico de um processador programável, também sendo este circuito conhecido como bloco operacional (datapath) do processador.

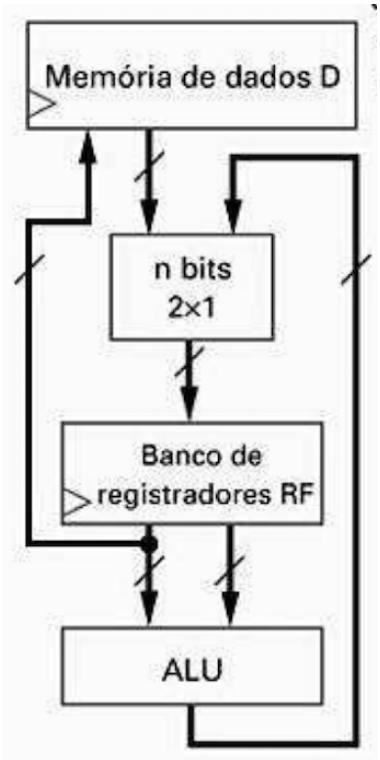


Figura 01: Bloco operacional básico de um processador programável

Em um determinado ciclo de clock, o bloco operacional pode executar uma das seguintes operações: carga, operação de ALU e armazenamento. Na primeira, um dado da memoria é carregado em

um registrador do banco de registrador (no caso deste trabalho, LDA). A segunda, operação de ALU, transforma dados de dois registradores, passando seus conteúdos pela unidade logico aritmética, e colocando o resultado de volta em um dos registradores. Neste trabalho, as operações de ALU são ADD, SUB, AND, OR, NOT. Por fim, operações de armazenamento são as que escrevem os dados armazenados nos registradores do banco de registradores (Register File- RF) na memoria de dados. (STA).

Partindo do raciocínio explicitado acima, não é possível que dados diretos da memoria sejam processados diretamente na ALU em um único ciclo de clock, uma vez que esses dados devem ser primeiramente carregados nos registradores. Devido a essa necessidade de todos os dados passarem pelo RF antes de serem transformados pela ALU, o bloco operacional deste trabalho que foi descrito acima possui arquitetura de carga e armazenamento.

A segunda parte principal de um processador programável, a unidade de controle (Figura 02), é responsável por ler cada instrução da memoria de instrução e coordenar a execução destas pelo bloco operacional. A unidade de controle executa as instruções, uma a uma de forma sequencial, por meio de estágios.

O primeiro estágio é o de busca. Neste estágio, a unidade de controle lê a instrução da memoria de instrução (controlando a sequencia por meio de um Contador de Programa- PC) e armazena essa instrução num registrador local (Registrador de Instrução- IR), levando todo o processo de busca um ciclo de clock.

O segundo estágio é a decodificação, onde a unidade de controle determina qual operação está sendo solicitada pela instrução, levando também um ciclo de clock.

Por fim, o terceiro estágio é a execução. Esse estagio é responsável pelo envio dos sinais que ativam o bloco operacional de modo a fazê-lo executar a operação solicitada pela instrução. Este estagio leva, também, um ciclo de clock.

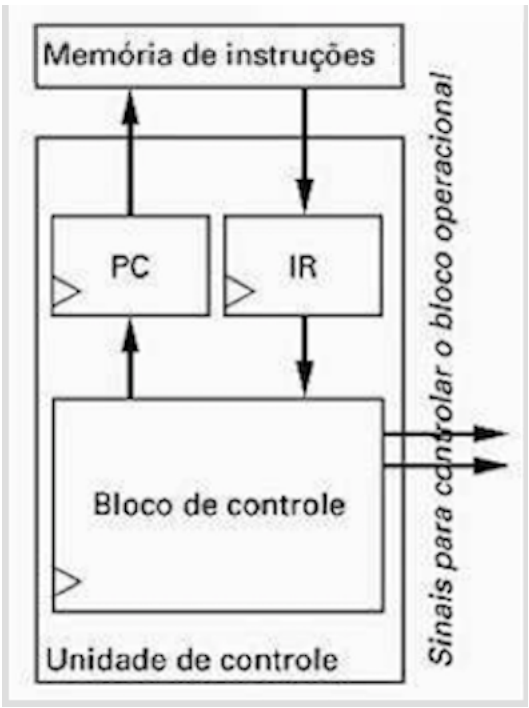


Figura 02: Unidade de controle de um processador programável

Cada uma das instruções da memória de instrução vai necessitar de três ciclos de clock para ser processada pela unidade de controle, sendo cada ciclo de clock referente a um respectivo estagio.

Para a execução dos três estágios, a unidade de controle conta com um bloco de controle, estando os estágios básicos do bloco de controle descritos na figura abaixo (Figura 03). Apos cada busca de instrução, o contador (inicialmente zerado PC=0 e referente a instrução I[0]) é incrementado para que no próximo estágio de busca seja obtida a próxima instrução da sequencia armazenada na memória de instrução.

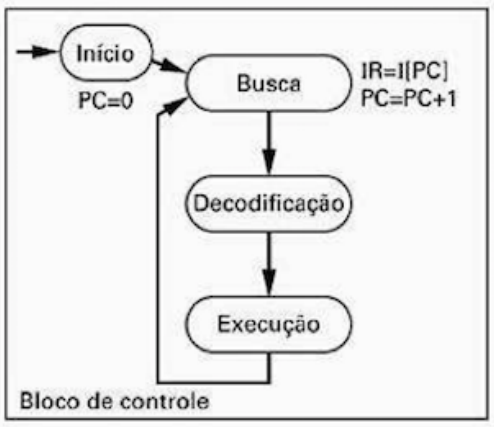


Figura 03: Estados básicos do bloco de controle da unidade de controle.

- **Instruções**

A lista de possíveis instruções e a maneira de representar essas instruções na memória de instruções são conhecidas como conjunto de instruções. Em nosso processador programável, o conjunto de instruções (Figura 04) se caracteriza por instruções de 16 bits, tendo a memória de instruções 16 bits de largura.

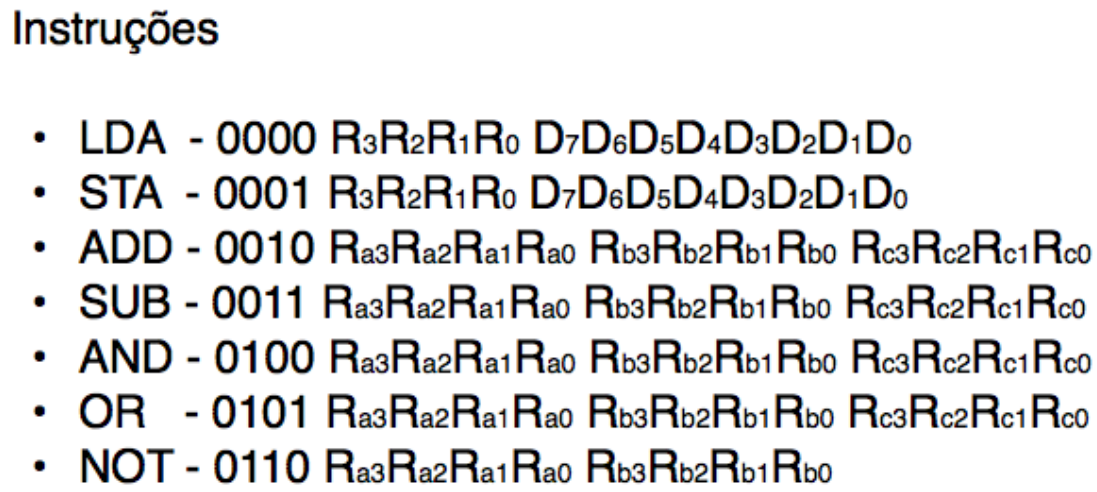


Figura 04: Conjunto de Instruções

Os 4 bits mais significativos (a esquerda) sao reservados para indicar o OPCODE da instrução, ou seja, qual instrução será realizada. Na operação LDA (load), o processador armazena no registrador $R_3R_2R_1R_0$ (registrador de destino) o dado lido da memoria na posição $D_7D_6D_5D_4D_3D_2D_1D_0$ (origem); na instrução STA (store) o dado do registrador $R_3R_2R_1R_0$ (origem) é lido e armazenado na posição de memória $D_7D_6D_5D_4D_3D_2D_1D_0$ (destino). Nas demais operações (ADD - adição, SUB -subtração, AND, OR e NOT), o registrador de índice “a” ($R_{a3}R_{a2}R_{a1}R_{a0}$) é o registrador de destino que receberá o resultado da operação (a ser efetuada na ALU), enquanto os registradores de índice “b” ou “c” são os registradores que contem os dados a serem operados.

É importante explicitar que nenhuma dessas operações modifica os conteúdos dos registradores de origem. Sendo assim, em uma operação de load, o dado lido na memoria permanece inalterado apos a leitura; na operação de store, o dado que se encontra no registrador nao é alterado mesmo apos ser salvo na memoria; a instrução ADD, nao altera o conteúdo dos registradores b e c; e assim sucessivamente para as demais operações.

- **Unidade de controle e bloco operacional para processador de 7 instruções**

Após a definição de nosso conjunto de 7 instruções, abaixo (Figura 05) está descrita a maquina de estado de alto nível, a qual explicita a sequencia de transições dos estados de nosso processador em funcionamento, sendo estas transições controladas por sinais de controle que vão posteriormente ser melhor explicados.

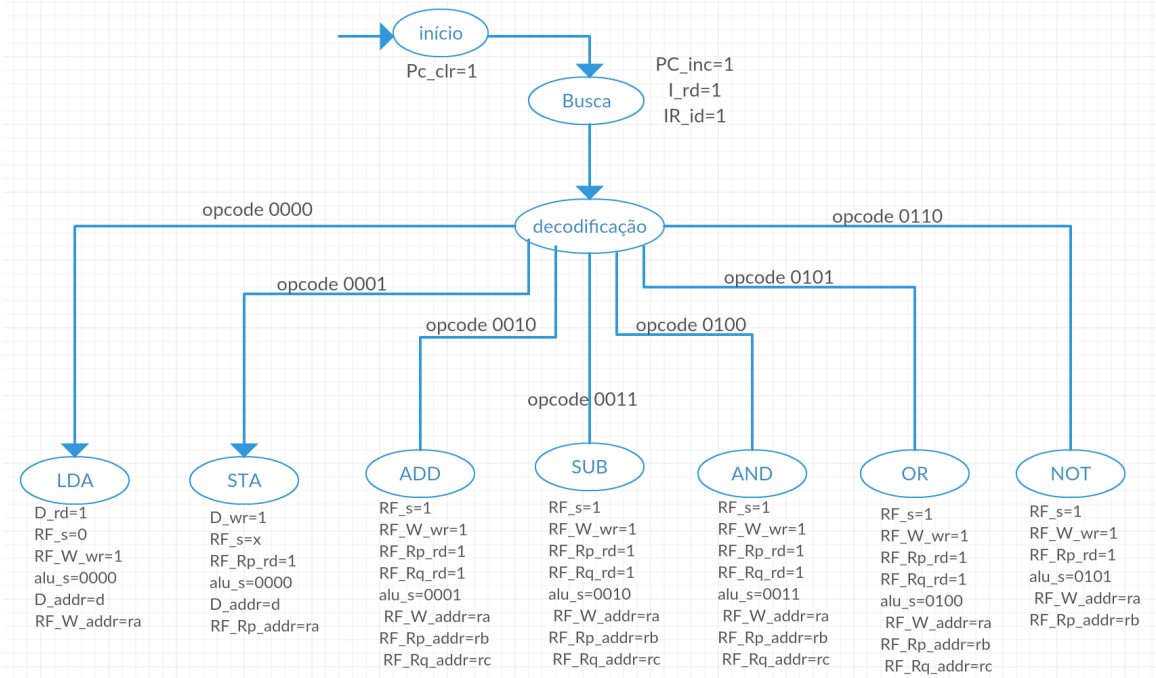


Figura 05: FSM para o bloco de controle do processador de sete instruções.

Como já foi explicitado, a interação entre a unidade de controle e o bloco operacional do nosso processador é dada por meio de sinais de controle, vindos da unidade de controle, que habilitam ou não as funções da memoria de dados e do bloco operacional. Dessa forma, o bloco operacional possui sinais de controle para cada porta de leitura e escrita do banco de registradores.

A figura abaixo (Figura 06) mostra a relação entre a unidade de controle com o bloco operacional refinado para o nosso processador de 7 instruções. Na figura, o sinal alu_s é composto por um sinal de 4 bits para que seja possível a diferenciação, por parte da ALU, da operação que vai ser realizada. Caso a operação nao necessite da ALU (operação de store por exemplo) os dados vindos do banco de registradores apenas passam pela alu sem ser

operados.

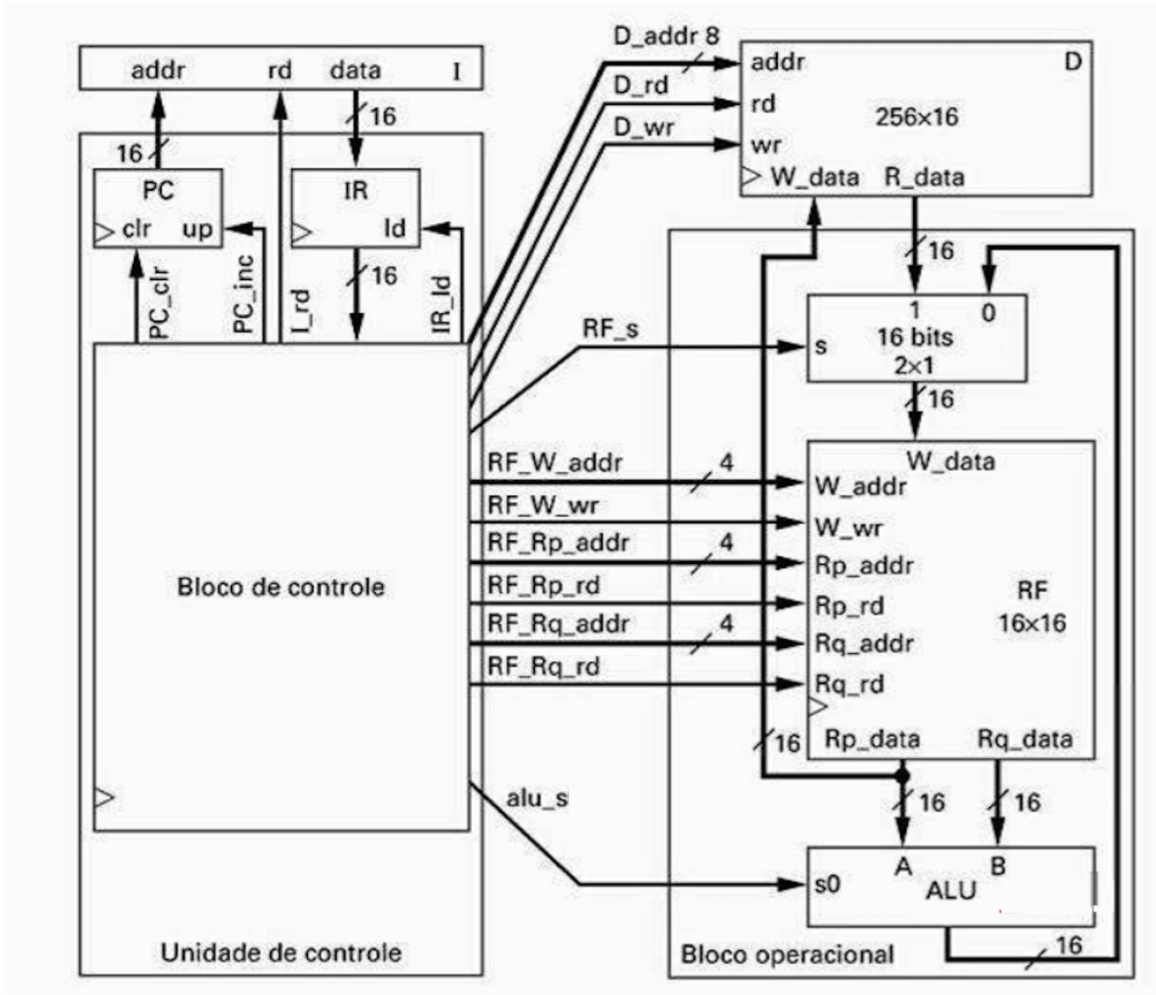


Figura 06: Bloco operacional refinado e unidade de controle para processador de 7 instruções

O banco de registradores (RF) possui 16 registradores (a instrução dispõe de 4 bits para endereça-los) com 16 bits cada. O bloco operacional também possui um multiplexador 2X1 que recebe o sinal RF_s em s e, se s=1, transfere para w_data (porta de escrita de dados do banco de registradores) o dado vindo da memoria e, se s=0, transfere para w_data o resultado da operação da ALU. Sobre a memoria, assumimos que esta possui apenas uma única porta de endereço, portanto podendo realizar ou uma leitura ou uma escrita de dados por vez, através da habilitação de sinais de controle para leitura ou escrita. A memoria possui 256 posições (a instrução dispõe de 8 bits para endereçar a memória de dados) e largura de 16 bits.

Abaixo (Figura 07) está a relação de cada estado com as

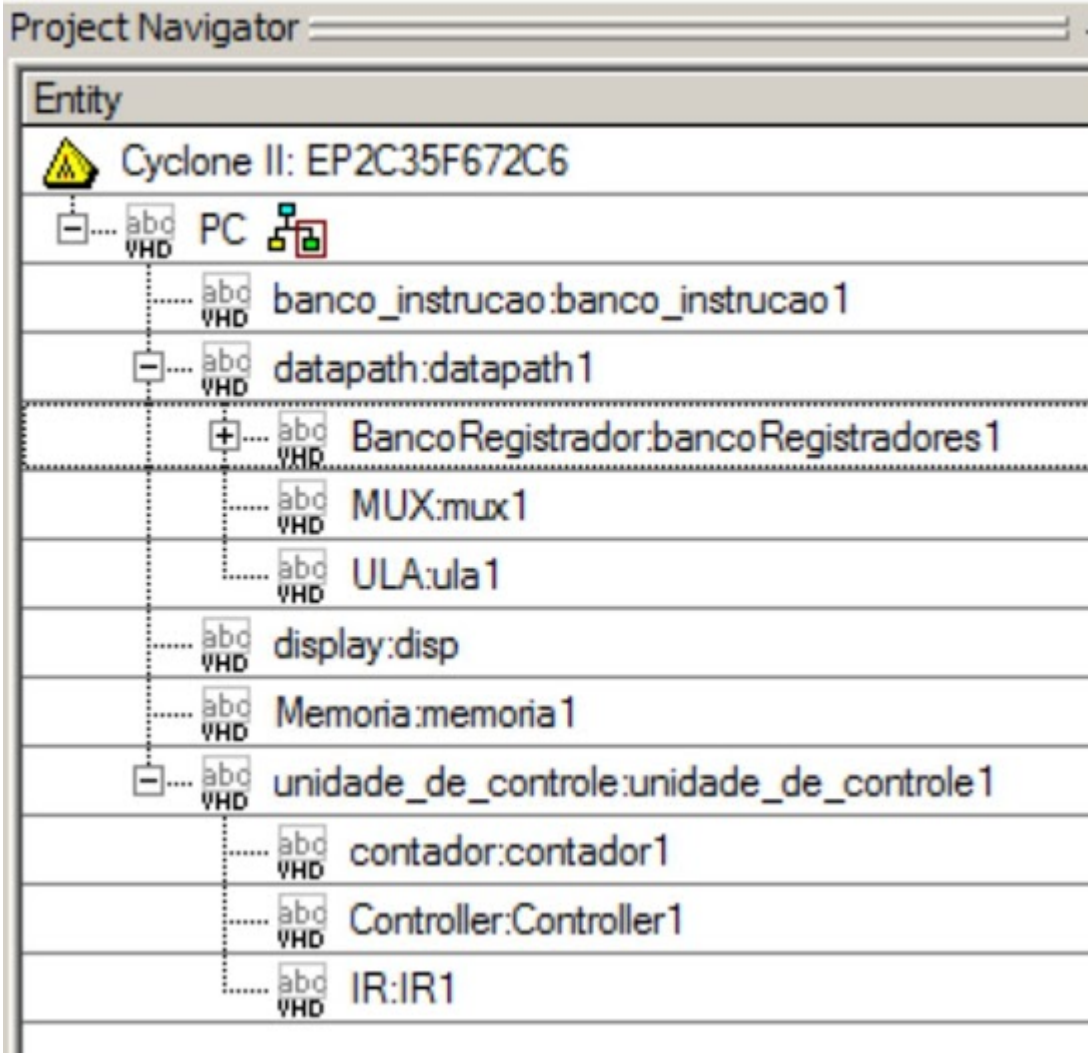
variáveis de controles mostradas na imagem acima (figura 06), sendo explicitadas quais das variáveis devem estar habilitadas ou não para o funcionamento de cada estado e realização das 7 instruções possíveis.

	Inicio	Busca	LDA	STA	ADD	SUB	AND	OR	NOT
PC_clr	1	0	0	0	0	0	0	0	0
PC_inc	0	1	0	0	0	0	0	0	0
I_rd	0	1	0	0	0	0	0	0	0
IR_ld	0	1	0	0	0	0	0	0	0
D_rd	0	0	1	0	0	0	0	0	0
D_wr	0	0	0	1	0	0	0	0	0
RF_s	0	0	0	X	1	1	1	1	1
RF_W_wr	0	0	1	0	1	1	1	1	1
RF_Rp_rd	0	0	0	1	1	1	1	1	1
RF_Rq_rd	0	0	0	0	1	1	1	1	0
alu_s	xxxx	xxxx	0000	0000	0001	0010	0011	0100	0101
D_addr	xxxx	xxxx	d(8bits)	d(8bits)	xxxx	xxxx	xxxx	xxxx	xxxx
RF_W_addr	xxxx	xxxx	ra	xxxx	ra	ra	ra	ra	ra
RF_Rp_addr	xxxx	xxxx	xxxx	ra	rb	rb	rb	rb	rb
RF_Rq_addr	xxxx	xxxx	xxxx	xxxx	rc	rc	rc	rc	xxxx
OPCODE	xxxx	xxxx	0000	0001	0010	0011	0100	0101	0110

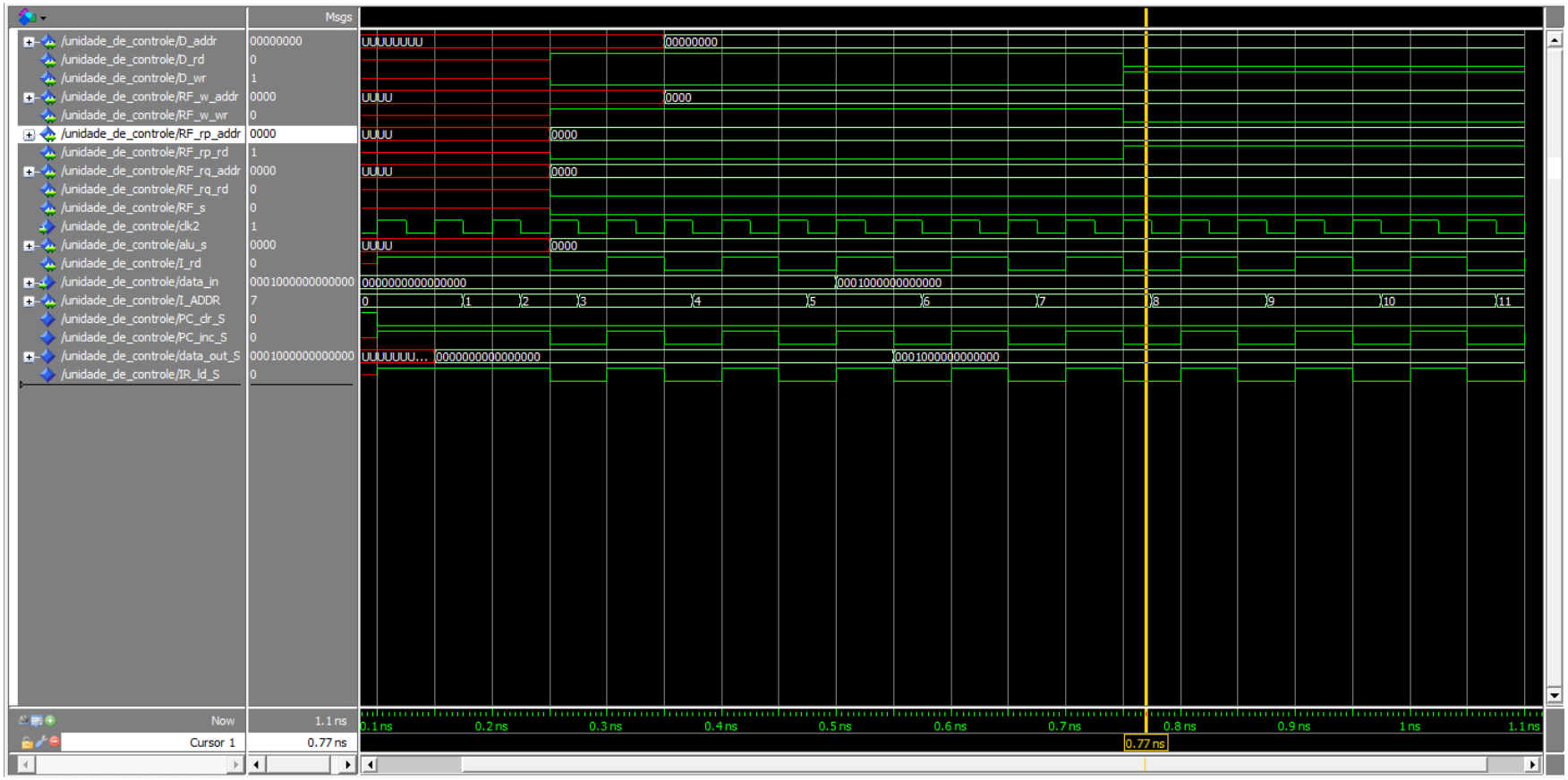
Figura 07: Sinais de controle

Abaixo se encontram as simulações:

Arquitetura do Projeto



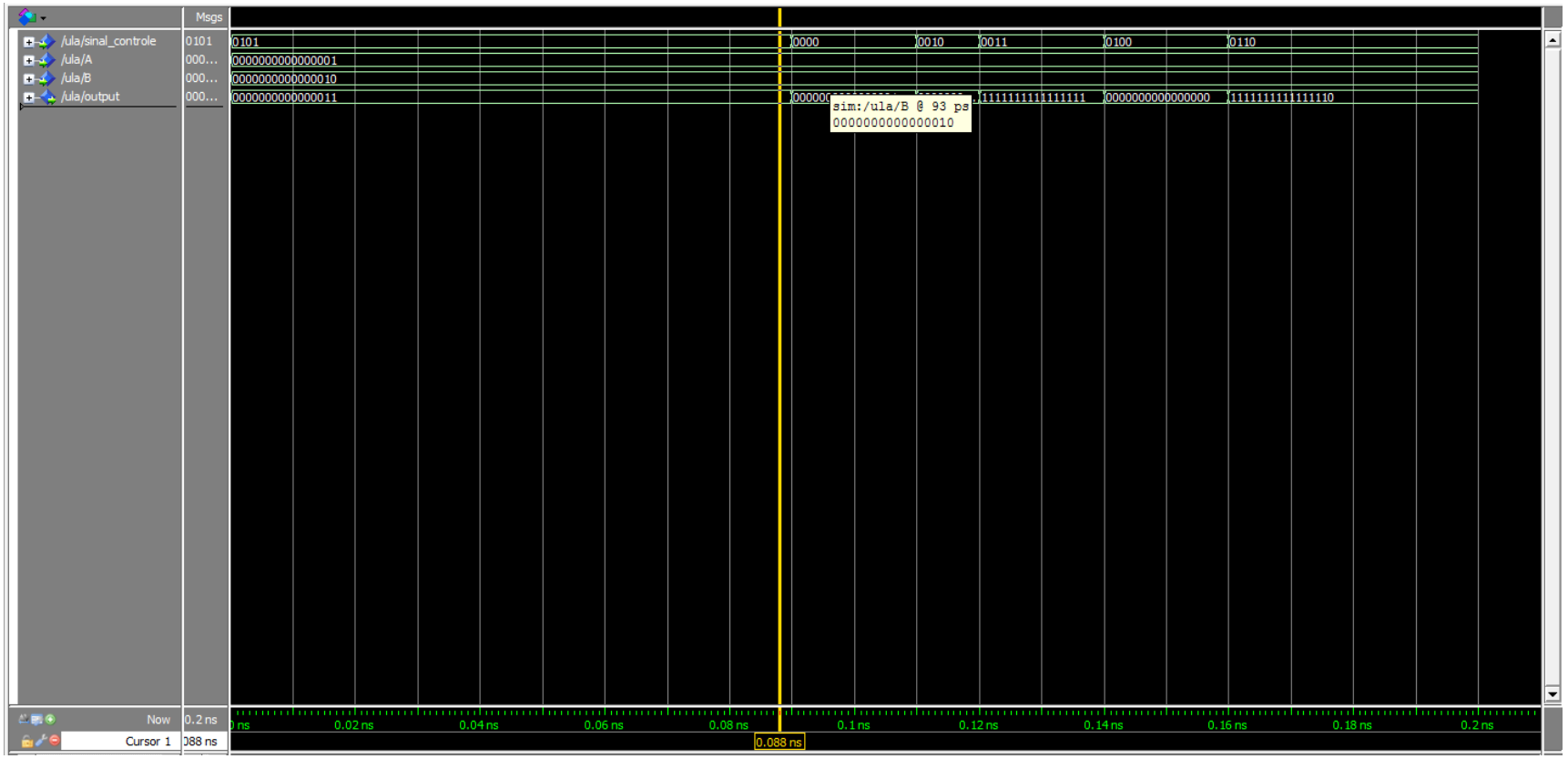
Unidade de Controle



Entrada: data

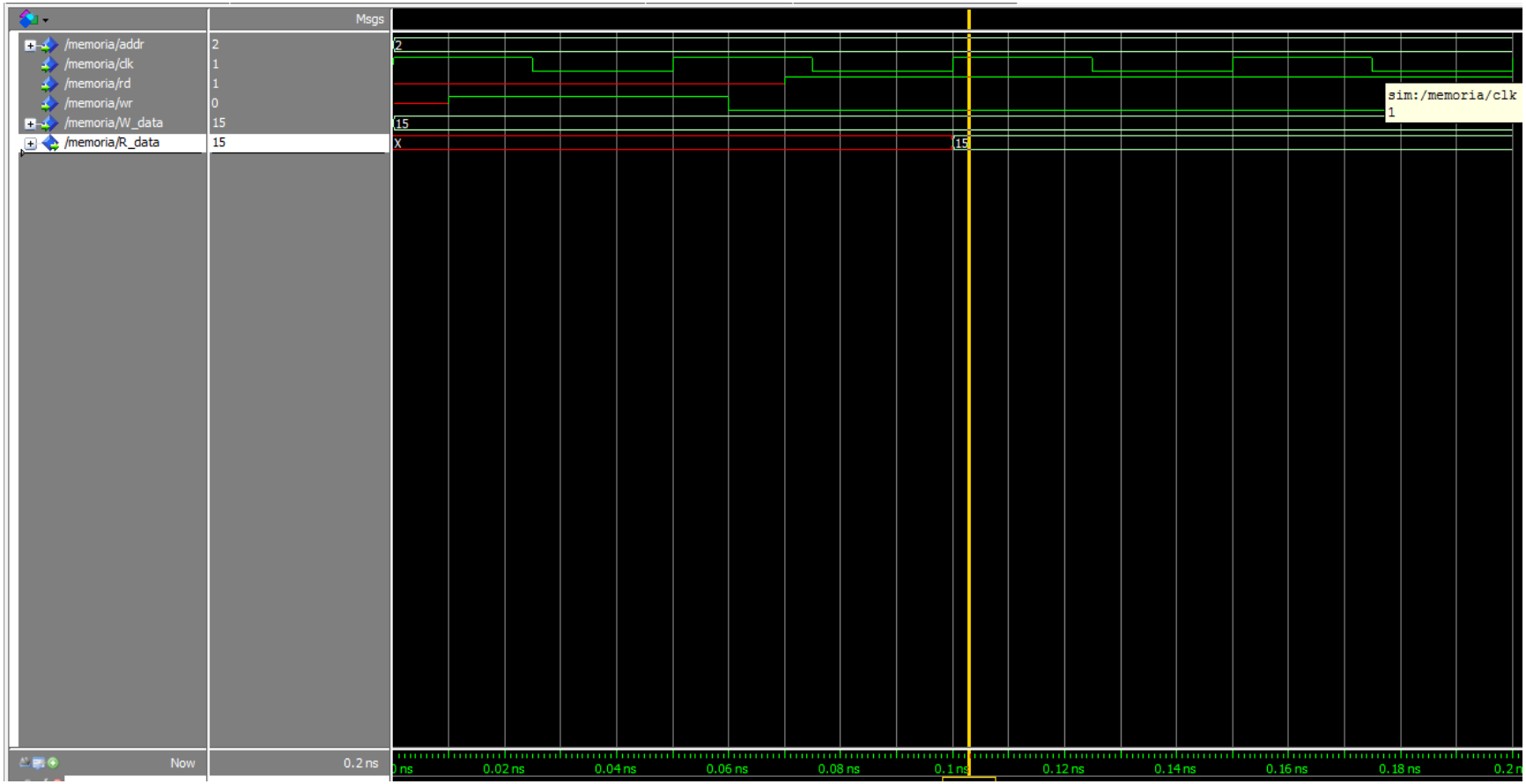
Saída: RF_w_wr, RF_rp_rd, RF_rq_rd, RF_s, d_dr, d_wr 1 bit; RF_w_addr, RF_rp_addr, RF_rq_addr, alu_s 4 bits; d_addr 8 bits; R_data 16 bits

ULA



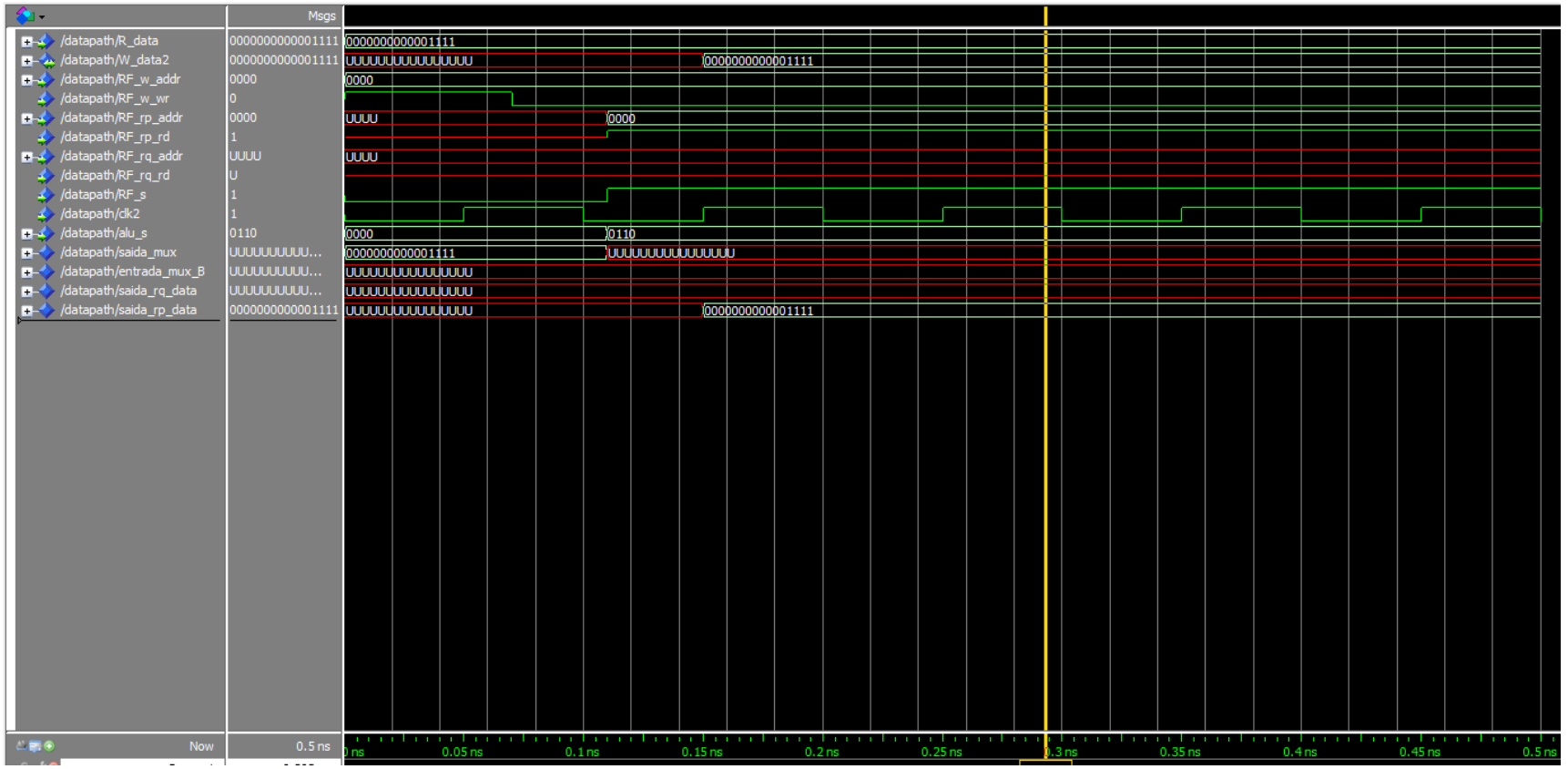
Entradas: A, B 16 bits; sinal_controle 4 bits
Saídas: Output 16 bits

Memória



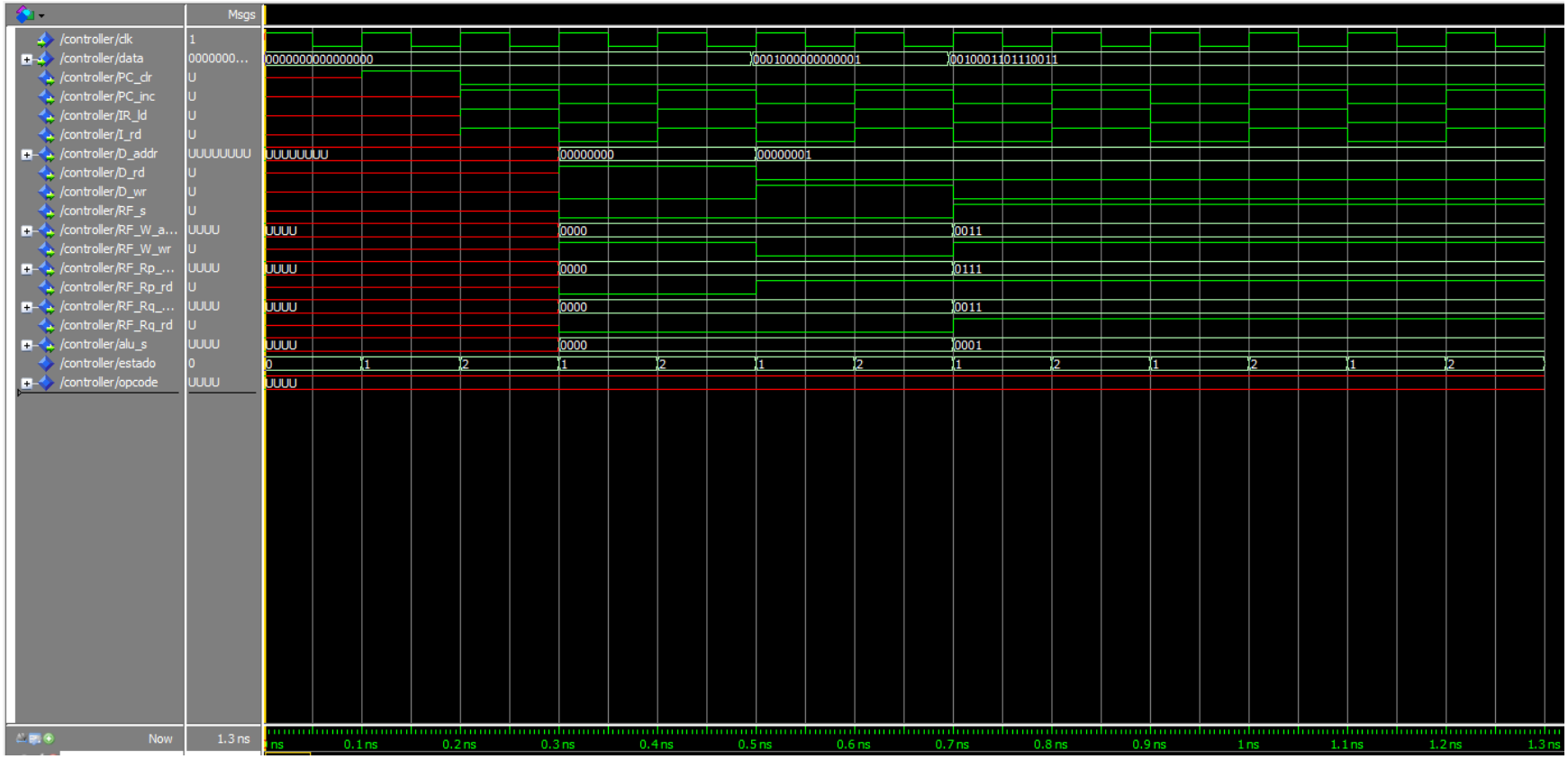
Entradas: addr 8 bits; rd, wr 1 bit; W_data 16 bits
Saídas: R_data 16 bits

Datapath



Entradas: RF_w_wr, RF_rp_rd, RF_rq_rd, RF_s 1 bit; RF_w_addr, RF_rp_addr, RF_rq_addr, alu_s 4 bits; R_data 16 bits
Saídas: W_data

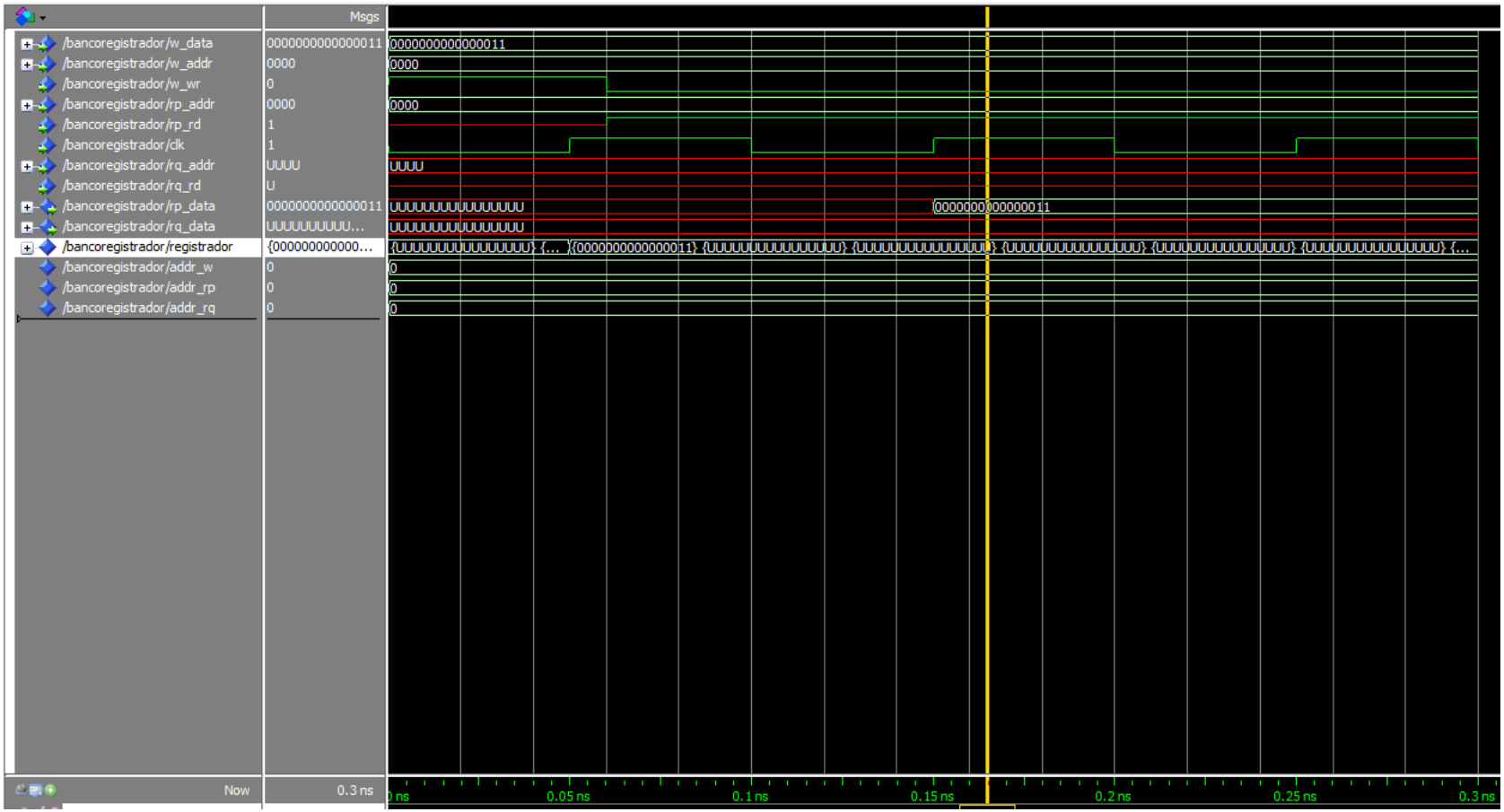
Controle



Entradas: data 16 bits

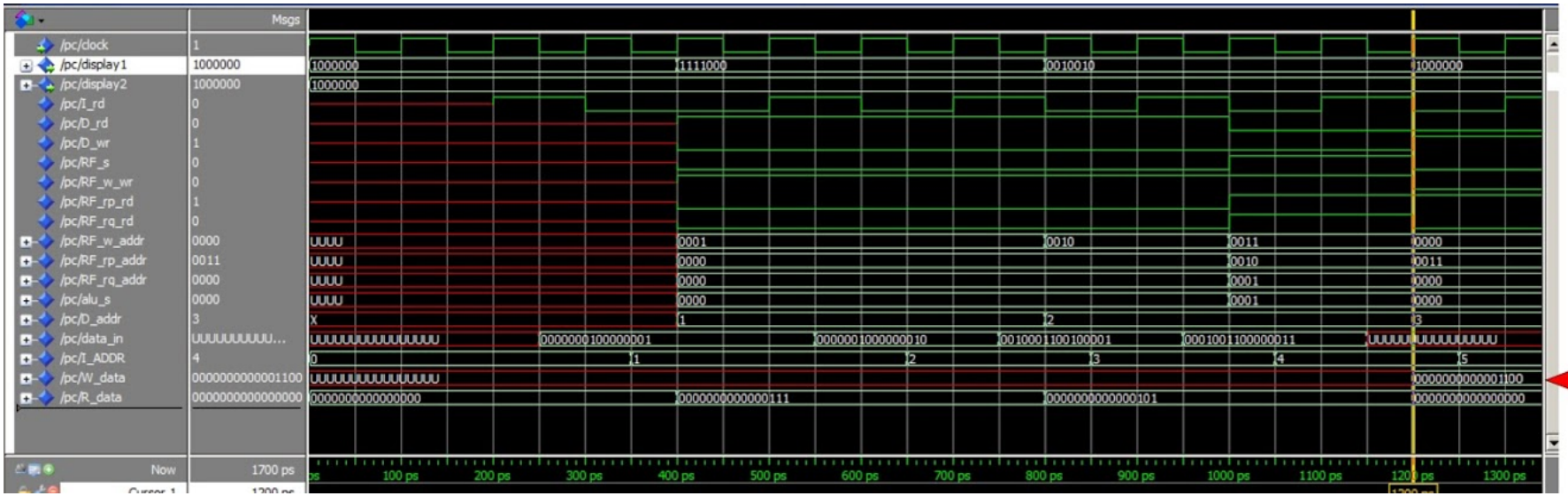
Saídas: pc_clr, pc_inc, ir_id, i_rd, d_rd, d_wr, rf_s, rf_w_wr, rf_rp_rd, rf_rq_rd 1 bit; rf_w_addr, rf_rp_addr, rf_rq_addr 4 bits; d_addr 8 bits

Banco de Registradores



Entradas: w_addr, rp_addr, rq_addr 4 bits; w_wr, rp_rd, rq_rd 1 bit; W_data 16 bits

Saídas: rp_data, rq_data 16 bits
PC



Entradas: data 16 bits
Saídas: i_rd, d_rd, d_wr, rf_s, rf_w_wr, rf_rp_rd, rf_rq_rd 1 bit; rf_w_addr, rf_rp_addr, rf_rq_addr 4 bits; d_addr 8 bits

O contador de instrução (I_ADDR) está começa do zero e vai aumentando a medida que as instruções são lidas. Nessa simulação tem-se load, load, soma e store. Para realizar a soma de 5 + 7. No final, tem-se no W_data o valor gravado da soma (12). Pode-se ver no R_data os valores que estão sendo gravados nos regritadores, primeiro o 7 depois o 5.

Operações:

```
memoria(0) <= "0000000100000001";--carrega memoria(1)
memoria(1) <= "0000001000000010";--carrega memoria(2)
memoria(2) <= "0010001100100001";--soma
memoria(3) <= "0001001100000011";--armazena o resultado na memoria(3)
```

Valores Iniciais da memória de dados:

```
memoria_RAM(1) => "0000000000000111";-- 7 em decimal
memoria_RAM(2) =>"0000000000000101";-- 5 em decimal

--O resultado da soma tem q ser 7 + 5 = 12 em decimal
```

Download de Códigos em VHDL:
http://www.4shared.com/zip/a8YvTX9-ba/codigos_SI.html

Referencias:

- VAHID, Frank. Sistemas Digitais. Porto Alegre: Bookman Editora, 2009.