MÓDULO I.P. DE UM PROCESSADOR PARA APLICAÇÕES EMBARCADAS SEM FIO

J. D. Costa[+], M. D. B. Melo[+], W. H. Veneziano, R. P. Jacobi[+], A. F. Rocha, J. C. Costa[+]*

[+]Universidade de Brasília-Brasil, *CEFET-PR
janaina_costa@ig.com.br, maxwell@ene.unb.br, wilsonhe@unb.br, rjacobi@cic.unb.br, adson@unb.br, camargo@ene.unb.br

RESUMO

A complexidade do desenvolvimento de sistemas em silício (*System on Chip* - SoC) vem crescendo rapidamente, atingindo em 2002 cerca de um bilhão de transistores CMOS por chip. Esta complexidade pode ser simplificada pela utilização de módulos IP, que provêm componentes reutilizáveis. Este artigo apresenta um módulo IP em linguagem VHDL de um microprocessador RISC, com dezesseis instruções e dezesseis registradores, com suporte para execução de procedimentos e de interrupção. Este microprocessador é parte integrante de um sistema de controle de irrigação, capaz de melhorar a gestão no uso da água e aumentar a produtividade. Através de comunicação sem fio, dados sobre a umidade do solo são medidos e transmitidos por nós coletores a estações de campo e, posteriormente, a uma estação base. Cada nó é composto de coletor solar, bateria, antena, atuador eletromecânico, sensor de umidade e um SoC (microprocessador, transceptor de RF, interface e conversor A/D).

ABSTRACT

The complexities of today's systems on chips (SoC's) are rapidly increasing, resulting in long, expensive design and verification cycles. One way to reduce this complexity is through the use of I.P's (intellectual property). In this paper a RISC microprocessor I.P. was generated in VHDL language. It has 16 instructions (arithmetic, logic, data transfer, jump and branches) and 16 registers. It also provides support for procedures and interruptions. This microprocessor is a part of an irrigation control unit for use in farms. It allows optimization of water use and an enhancement of the productivity. Collector nodes measure and transmit (fully wireless) ground humidity data to intermediate stations, and then, to a central station (human-machine interface). Each node has a solar collector, a battery, an antenna, an electromechanical transducer, a humidity sensor, and a Soc (a microprocessor, a RF transceiver, a serial interface and an A/D converter).

MÓDULO I.P. DE UM PROCESSADOR PARA APLICAÇÕES EMBARCADAS SEM FIO

J. D. Costa[+]*, M D. B. Melo*[+]*, W. H. Veneziano*, R. P. Jacob*[+]*i, A. F. Rocha*[+], J. C. Costa[+]

[+]Universidade de Brasília-Brasil, *CEFET-PR

RESUMO

A complexidade do desenvolvimento de sistemas em silício (*System on Chip* - SoC) vem crescendo rapidamente, atingindo em 2002 cerca de um bilhão de transistores CMOS por chip. O tratamento de problemas desta complexidade pode ser simplificado pela utilização de módulos I.P.(*Intellectual property*), que provêm componentes reutilizáveis de variadas complexidades. Este artigo apresenta um módulo I.P. em linguagem VHDL de um microprocessador RISC, que apresenta dezesseis instruções (aritméticas, lógicas, de transferência de dados e desvios) e dezesseis registradores, bem como suporte para execução de procedimentos (sub-rotinas) e de interrupção. Este microprocessador é parte integrante de um sistema de controle de irrigação em propriedades rurais, capaz de melhorar a gestão no uso da água e aumentar a produtividade. Através de comunicação sem fio, dados sobre a umidade do solo são medidos e transmitidos por nós coletores a estações de campo e, posteriormente, a uma estação base central (interface com o usuário). Cada nó é composto de coletor solar, bateria, antena, atuador eletromecânico, sensor de umidade e um SoC (microprocessador, transceptor de RF, interface serial e conversor A/D).

1. INTRODUÇÃO

A crescente complexidade de projetos de produtos eletrônicos e a necessidade de reduzir o tempo necessário para a chegada desses produtos ao mercado (*time to market*) têm tornado cada vez mais comum o uso de módulos I.P. (*intelectual property*) [1] e [2]. A integração desses módulos para o desenvolvimento de um projeto maior, ASIC ou FPGA, tem sido proposta como uma forma de obter sistemas complexos de alta qualidade com um curto tempo de projeto [3]. Muitos desses módulos I.P.s são dedicados a aplicações sem fio e elementos de processamento, de importância crescente.

Um exemplo dessas aplicações é um sistema de controle de irrigação que está sendo desenvolvido na Universidade de Brasília, capaz de otimizar a utilização de água por propriedades rurais agrícolas, além de aumentar a produtividade da cultura.

Neste trabalho, foi desenvolvido um módulo I.P. em linguagem VHDL (VHSIC - *Very High Speed Integrated Circuit - Hardware Description Language*) de um microprocessador para o sistema de irrigação. Essa linguagem, padronizada pelo IEEE, é utilizada para descrever hardwares em vários níveis de abstração [4],[5].

Nas seções seguintes deste texto, serão descritos o sistema de irrigação, a arquitetura do processador, a implementação em linguagem VHDL e os resultados das simulações.

2. SISTEMA DE CONTROLE DE IRRIGAÇÃO

O sistema destina-se a agricultura de precisão e tem o objetivo de propiciar melhor gestão do uso da água. São utilizados informações georreferenciadas e dados climatológicos e hidrológicos para programação do sistema. Ele é composto por uma estação base, estações de campo e por nós. A estação base, única na propriedade rural, concentra as informações oriundas das estações de campo (sem fio) e faz a interface com o usuário (Figura 1). Os dados coletados são processados, armazenados em mídia de massa e apresentados ao usuário. O sistema também permite que o usuário programe a sua atuação.

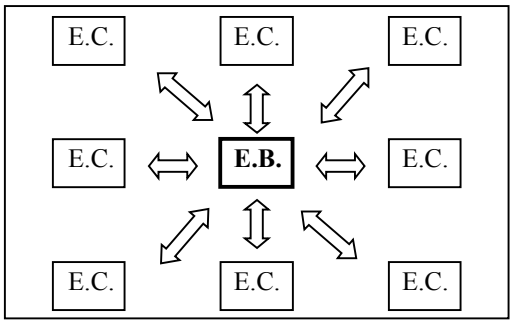

Figura 1: estação base e estações de campo

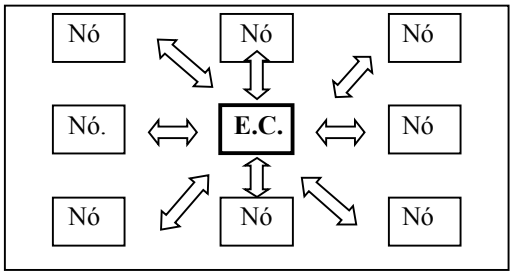

Figura 2: estação de campo e nós

As estações de campo, por meio de comunicação sem fio, coletam dados dos nós e os retransmitem para a estação base (Figura 2).

As funções de medição no solo e atuação nos aspersórios de água são executadas pelos nós. Eles têm uma área de cobertura de 100 hectares e compõem-se basicamente de um sistema em chip CMOS 0,35μm (microprocessador RISC, oscilador, interfaces serial RS232 e A/D, transceptor de RF em 915 a 927,75MHz e 1mW), sensor de umidade do solo (tensiômetro), antena, coletor solar, fonte de alimentação, bateria, atuador eletromecânico e programas computacionais. O arranjo físico está apresentado na Figura 3.

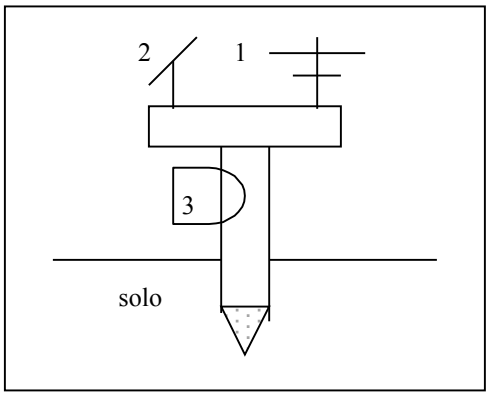

Figura 3: nó composto por Antena (1), painel solar (2), sensor e placa de circuito impresso (3)

3. ARQUITETURA DO PROCESSADOR

Baseado em arquitetura RISC de 16 bits, o processador possui um banco de registradores com 16 registradores de 16 bits, especificados na Tabela 1. Doze posições de memória são utilizados como registradores para comunicação com as interfaces das unidades de RF, de comunicação serial e de conversão A/D.

Estão contempladas 16 instruções lógicas, aritméticas, de transferência de dados e de desvios condicionais e incondicionais, apresentados na Tabela 2.

A arquitetura do processador também provê a execução de procedimentos (sub-rotinas), três tipos de interrupção e duas sinalizações de erro. A unidade lógica e aritmética opera em ponto fixo e seu somador é do tipo *carry lookahead.* Estão presentes também, unidades de memória do tipo RAM e ROM. Na Figura 4 estão apresentados o caminho de dados e as linhas de controle do processador.

Foram adicionadas estruturas de hardware para teste do processador, através da filosofia de projeto voltado para a testabilidade. Foi desenvolvido um montador para este processador.

Tabela 1: registradores [6],[7]

Símbolo	Função	obs
$zero	Constante 0	Constante 0 de 16bits
$t0,$t1,$t2	Temporários	Registr. Auxiliares
$a0, $a1	Argumento	Argumentos para operações aritméticas e procedimentos
$a2		
$s0, $s1	Salvos	Armazena valores durante chamadas de procedimento
$s2, $s3		
$int	Interrupção	Contém status das interrupções e ender. da instrução com erro
$gp	Apontador Global	Aponta para as variáveis globais no interior da pilha
$sp	Apontador Pilha	Aponta para o topo da pilha
$pc	Contador de Programa	Aponta para a próxima instrução
$ra	Endereço de Retorno	Aponta para o endereço de retorno de uma subrotina

Tabela 2:conjunto de instruções do processador [6],[7]

Instr	Exemplo	Significado
Add	Add $s1,$s2,$s3	Adiciona $s2 a $s3 e armazena o resultado em $s1
Sub	Sub $s1,$s2,$s3	Subtrai $s3 a $s2 e armazena o resultado em $s1
Addi	Addi $s1,100	Adiciona $s1 a constante e armazena o resultado em $s1
Shift	Sft $s1,8	Desloca $s1 da constante e armazena o resultado em $s1. Se o valor da constante for negativo, desloca à esquerda.
And	And $s1,$s2,$s3	AND booleano bit a bit entre $s2 e $s3 e armazena resultado em $s1
Or	Or $s1,$s2,$s3	Or booleano bit a bit entre $s2 e $s3 e armazena resultado em $s1
Not	Not $s1	NOT booleano bit a bit de $s1 e armazena resultado em $s1
Xor	Xor $s1,$s2,$s3	XOR booleano bit a bit de $s2 e $s3 e armazena resultado em $s1
Slt	Slt $s1,$s2,$s3	Torna $s1 = 1 se $s2 < $s3 senão $s1 = 0
Lw	Lw $s1,$s2,$s3	Carrega palavra armazenada no endereço $s2 deslocado de $s3 e salva o resultado em $s1
Sw	Sw $s1,$s2,$s3	Carrega palavra armazenada em $s1 e salva o resultado no endereço $s2 deslocado de $s3
Lui	Lui $s1,100	Carrega a constante nos oito bits mais significativos de $s1
Beq	Beq $s1,$s2,5	Se $s1 = $s2 desvia programa para $pc + CONST
Blt	Blt $s1,$s2,5	Se $s1 < $s2 desvia programa para $pc + CONST
J	J $s1,100	Desvia para o endereço $s1 deslocado da constante
Jal	Jal $s1,100	Desvia para end. $s1 deslocado da constante salvando origem em $ra

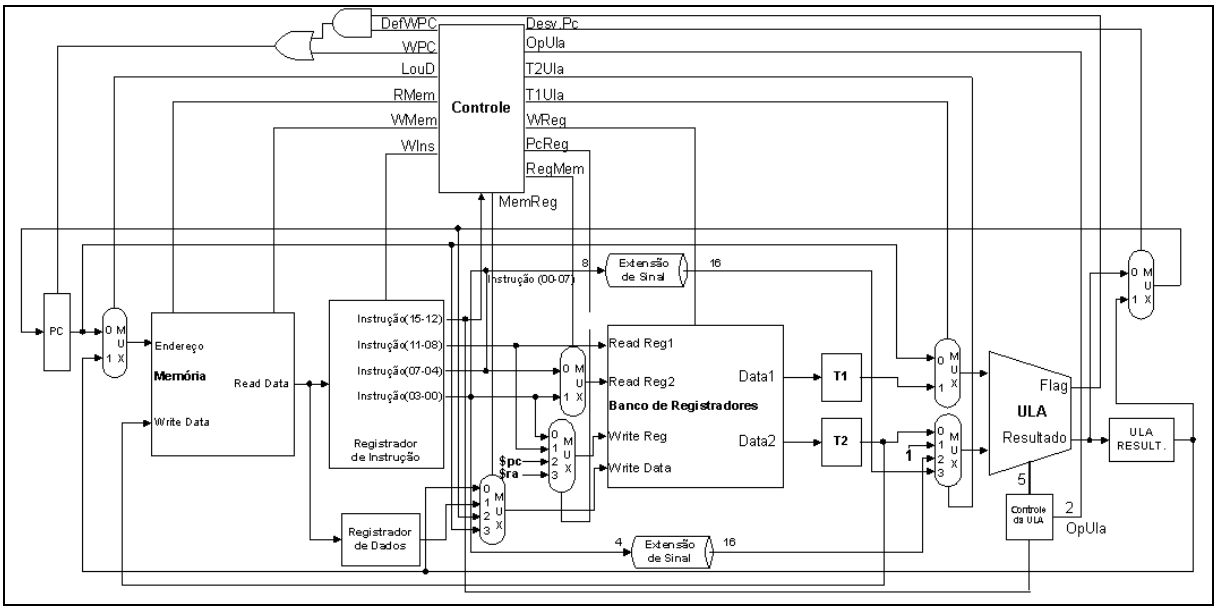

Figura 4: caminho de dados e linhas de controle [6]

4. IMPLEMENTAÇÃO

A abordagem utilizada na implementação do microprocessador foi realizar uma descrição comportamental baseada no modelo FSMD [9]. Neste modelo o comportamento do microprocessador é definido por uma máquina de estados onde em cada estado a microinstrução correspondente é apresentada de forma similar a um programa em linguagem de alto nível. Ou seja, em vez de se trabalhar diretamente com componentes e seus sinais de controle, especifica-se a parte operativa utilizando-se variáveis, arranjos e operadores lógico-aritméticos. A síntese do processador é realizada pela ferramenta Max+Plus II, versão 9.23 da Altera Corporation. O código criado para modelar o microprocessador é composto de um caminho de dados, de uma unidade de controle (FSM) e de uma unidade de memória. O caminho de dados reflete a parte computacional do processo, e compreende o banco de registradores, a unidade lógico-aritmética, os recursos de interconexão e lógica auxiliar. A máquina de estados finita gera os sinais de controle para direcionar o caminho de dados de forma temporizada. A figura 5 apresenta parte do código usado para realizar a instrução **addi** a título de ilustração. O modelo de FSMD que se mostrou mais adequado à ferramenta Max+Plus II foi aquele baseado em um único processo. Contrariamente as recomendações usuais encontradas na literatura, a descrição baseada em dois processos, um para o registrador de estado e outro para a lógica de saida e de próximo estado, não foi sintetizado corretamente pela ferramenta.

Para verificar se as funções realizadas pelo processador estavam sendo executadas conforme o esperado, foi elaborada uma rotina de teste contendo todas as 16 instruções previstas. Este programa foi carregado na memória RAM do processador projetado e simulado juntamente com o código VHDL.

A simulação pode ser dividida em dois estágios: análise e execução. Na análise, a descrição VHDL é examinada, e erros de sintaxe e semântica são localizados e corrigidos [8]. No segundo estágio ocorre a simulação propriamente dita.

```
Process (clock, reset)
   Begin
     if reset = '1'  then
        State <= RESET_PC;
     elsif  clock'event and  clock = '1'  then
        case  state is
                          .
                          .
                          .
           when execute_addi =>
                     resultado <= banco_reg(reg) + reg_ext2;
                     state <= execute_addi1;
           when execute_addi =>
                     banco_reg(reg) <= resultado;
                     memory_address_reg <= prog_counter;
                     state <= fetch;
                    .
                    .
                    .
end process;
```

Figura 5 – Código VHDL para realizar a instrução ADDI

A passagem de tempo é realizada em passos discretos. A simulação começa com um estágio de inicialização (RESET_PC), onde são dados valores

iniciais para o contador de programa, para o banco de registradores e para a unidade de memória. Após a inicialização, seguem os demais ciclos da simulação.

5. RESULTADOS

Para um FPGA modelo EPF10K130EQC240-1, da família FLEX10KE Altera, a implementação requisitou 5.779 células lógicas e possibilitou uma freqüência máxima de operação de 15,92 MHz.

Nas Figuras 6 a 8, estão apresentados os diagramas de tempo capturados do simulador para a instrução ADDI. No estado reset_pc, o registrador de endereço da memória (memory_address_reg) e o PC são carregados com zero. O banco de registradores também é iniciado com zero. Em seguida, no estado FETCH uma operação de leitura da memória é realizada, o registrador $PC é incrementado de uma unidade e a instrução fica disponível na entrada do registrador de instruções. Esses novos valores serão atualizados somente na próxima subida do clock, conforme pode ser visto nas Figuras 6 e 7.

Na figura 7 o registrador de instruções é atualizado com o valor 8305. O valor 8 corresponde ao OPCODE da instrução ADDI e o valor 3, ao registrador onde será armazenado o resultado da operação. Os campos seguintes, com valores 0 e 5, correspondem ao valor da constante de que será adicionada ao conteúdo do registrador 3.

Conforme é possível observar na figura 8, o resultado da operação somente está disponível no próximo ciclo de clock (fetch da instrução seguinte).

6. CONCLUSÕES

Foi gerado com sucesso um módulo IP composto de uma descrição de um microprocessador RISC em linguagem VHDL. As simulações comprovaram o perfeito funcionamento da estrutura para todas as dezesseis instruções. Este trabalho foi utilizado também para validar a arquitetura antes de serem confeccionados os leiautes das máscaras para fabricação em silício.

Trabalhos futuros poderão envolver a geração de um código totalmente estrutural e a inclusão de *pipeline*, com vistas a baixar a potência consumida pelo circuito integrado e aumentar seu desempenho.

7. AGRADECIMENTOS

Ao CNPq e à CAPES pelo apoio financeiro.

8. REFERÊNCIAS

[1] S.J.E. Wilton, R. Saleh, “Programmable logic IP cores in SoC design: opportunities and challenges”, *Custom Integrated Circuits, IEEE Conference on*, pp. 63 –66, 2001.

[2] P. Coussy, A. Baganne and E. Martin, “A design methodology for integrating IP into SOC systems”, *Proceedings, Custom Integrated Circuits Conference*, IEEE, pp. 307 –310, 2002.

[3] P. Varhol, “The IP revolution Reaches Programmable Logic”, *Computer Design*, pp. 70-73, 1997.

[4] D. Perry, *VHDL*, 3rd Ed., McGraw-Hill, New York, 1998.

[5] Hamblen, J. O.; M. D. Furman, *Rapid Prototyping of Digital Systems,* Kluwer Academic, Boston, 2001.

[6] G. M. Benício, *Projeto de Microprocessador RISC 16-Bit para Sistema de Comunicação sem Fio em Chip,* (dissertação de mestrado em engenharia elétrica), Universidade de Brasília, Brasília, 2002.

[7] R. R. Linder*, Linguagem de Máquina para um processador num sistema em chip (SoC),* (dissertação de mestrado em engenharia elétrica), Universidade de Brasília, Brasília, 2002.

[8] Ashenden, Peter J., *The Designer´s Guide to VHDL*, Morgan Kaufmann, California, 1996.

[9] Gajski, D. *High Level Synthesis: Introduction to Chip and System Design*. Kluwer Academics Publisher, 1992.

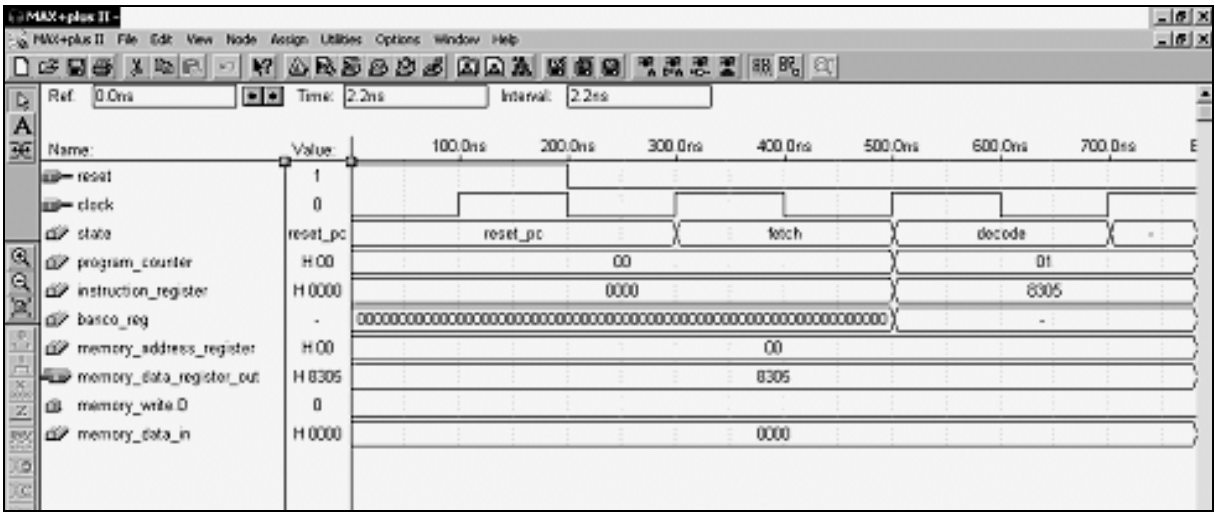

Figura 6: instrução ADDI (estados reset_pc, fetch e decode)

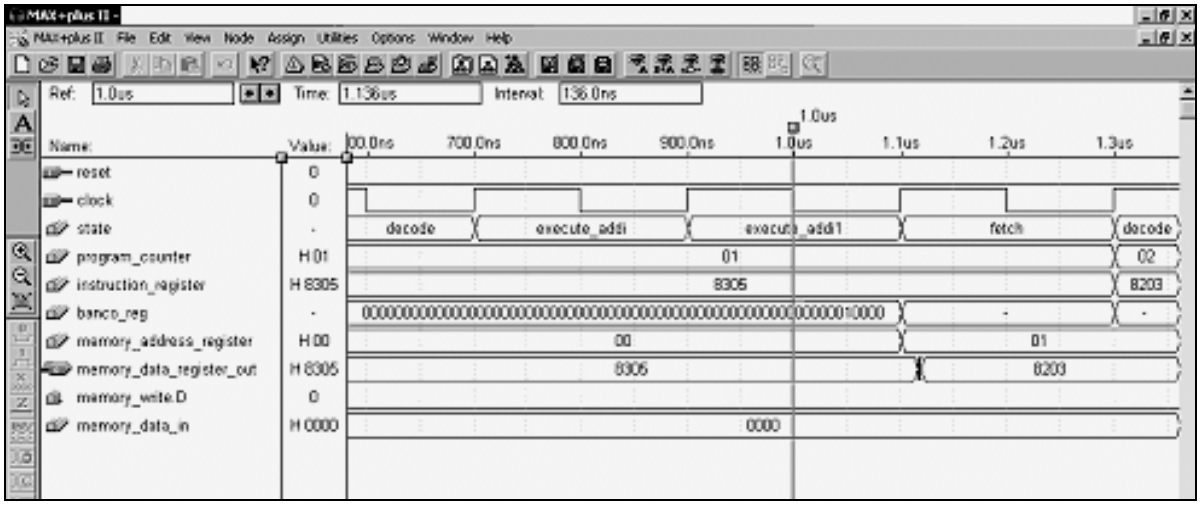

Figura 7: instrução ADDI (decode, execute_addi, execute_addi1 e fetch)

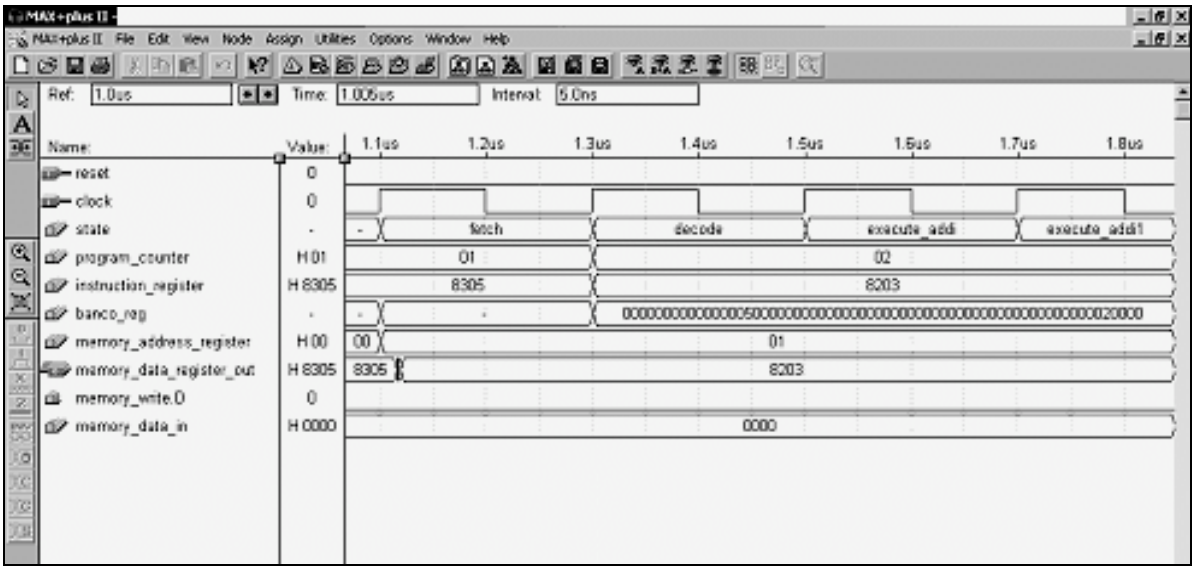

Figura 8: instrução ADDI (finalização)