Manual de instruções ProcessorCI Interface

Julio Nunes Avelar [1]

Departamento de sistemas de computação, Universidade estadual de campinas

29 de Agosto de 2024

[1]Endereço de e-mail: julio.avelar@students.ic.unicamp.br

Conteúdo

0.1 Introdução

No âmbito da validação de processadores em FPGAs, o desenvolvedor perde drasticamente a capacidade de analisar sinais internos e externos do processador durante a sua execução. Por este motivo, precisamos de um hardware externo ao processador com capacidade de ler e escrever informações na memória, enviar informações ao processador e controlar os principais sinais do mesmo. Além disso, este módulo precisa ser capaz de se comunicar com uma máquina Host para que os dados possam chegar ao programa de teste e/ou ao desenvolvedor que está testando. Por esse motivo, necessita-se do desenvolvimento de um protocolo onde a máquina Host consiga enviar comandos e informações ao hardware auxiliar e vice-versa.

0.2 Funcionalidades Necessárias

Para controlar o Core em teste na FPGA, é essencial ter domínio sobre seus sinais de entrada e saída, como os sinais de *clock* (CLK), *halt*, *reset*os dados do barramento de memória, entre outros. Para isso, o controlador precisa executar algumas funções específicas, que são:

1. Controlar o sinal de CLK.
2. Controlar o sinal de RESET.
3. Escrever na memória.
4. Ler da memória.
5. Controlar a prioridade de acesso à memória.
6. Realizar o controle de timeout de execução do Core.
7. Verificar o término da execução de um programa no Core.
8. Realizar leitura e escrita em lote na memória.
9. Enviar informações da memoria via Serial para uma maquina *Host*.

0.3 Instruções

Para a comunicação entre o hardware e o software, a interface é realizada por meio de um protocolo baseado em instruções (comandos).

0.3.1 Formato das Instruções

As instruções são compostas por 32 bits, onde os 8 bits menos significativos representam o *opcode* e os 24 bits mais significativos são utilizados para o campo imediato, conforme ilustrado na Tabela 1. Para instruções que não possuem um campo imediato, esses bits são preenchidos com zeros.

31:8	7:0
Imediato	Opcode

Tabela 1: Formato da Instrução

0.3.2 Opcodes

A Tabela 2 apresenta uma lista de todas as instruções disponíveis, incluindo seus respectivos opcodes em binário, ASCII e hexadecimal. A especificação detalhada de cada instrução pode ser encontrada na Seção 0.6.

Descrição	Opcode	ASCII opcode	Hex opcode	2º pacote
Enviar N pulsos de CLK	01000011	C	0x43	
Parar o CLK do Core	01010011	S	0x53	
Retomar o CLK do Core	01110010	r	0x72	
Resetar o Core	01010010	R	0x52	
Escrever na posição N de memoria	01010111	W	0x57	Y
Ler a posição N de memoria	01001100	L	0x4C	
Carregar bits mais significativos no Acumulador	01010101	U	0x55	
Carregar bits menos significativos no acumulador	01101100	l	0x6C	
Somar N ao acumulador	01000001	A	0x41	
Escrever Acumulador na posição N	01110111	w	0x77	
Escrever N na posição do acumulador	01110011	s	0x73	
Ler a posição do acumulador	01110010	r	0x72	
Setar timeout	01010100	T	0x54	
Setar tamanho da pagina de memoria	01010000	P	0x50	
Executar testes em memoria	01000101	E	0x45	
Obter o ID e verificar funcionamento do modulo	01110000	p	0x70	
Definir endereço N de termino de execução	01000100	D	0x44	
Definir o valor do Acumulador como endereço de termino	01100100	d	0x64	
Escrever N posições a partir do acumulador	01100101	e	0x65	
Ler N posições a partir do acumulador	01100010	b	0x62	
Obter o acumulador	01100001	a	0x61	
Alterar prioridade de acesso a memoria para o Core	01001111	O	0x4F	
Executar até ponto de parada	01110101	u	0x75	

Tabela 2: Listas de comandos suportados pelo protocolo

0.4 Implementação

0.4.1 Funcionamento

O protocolo de interface opera sobre um protocolo físico responsável pela transmissão de dados entre a FPGA e a máquina host. Em configurações de comunicação Master-Slave, como no protocolo SPI, uma linha de sinal denominada CAL (*Callback*) é utilizada. Esta linha informa à máquina host que o controlador possui informações prontas ou está preparado para executar um novo comando. Para protocolos bidirecionais, como o UART, os dados são enviados pela FPGA sem sinais de *callback*.

As tabelas abaixo mostram os regimes de comunicação entre a FPGA e a máquina host ao longo do tempo, considerando os três casos possíveis: envio de instrução, envio de instrução e dados, e envio de instruções com recebimento de dados, onde em cada caso as ações ocorrem sequencialmente, utilizando o caso 2 como exemplo, os dados são enviados apenas após o envio da instrução.

Caso 1: Apenas Envio

Master	Instrução
Slave	

Caso 2: Envio com Dados

Master	Instrução	Dados
Slave		

Caso 3: Envio e Recebimento

Master	Instrução	
Slave		Dados

0.4.2 Formas de Comunicação

E possível estar utilizando a interface sobre diversos protocolos de comunicação, sendo os com algum suporte ou em fase de implementação suportados pela infraestrutura citados na tabela 3.

Nome	Velocidade
UART	115200 bps
SPI	10 MHz
PCIe	2.5 GB/s
USB	

Tabela 3: Formas de Comunicação

0.5 Registradores Internos

A infraestrutura possui alguns registradores internos especializados e multifuncionais que podem ser utilizados para interação com a memória e execução dos testes. Esses registradores são:

1. **Acumulador (ACC)**: O acumulador é um registrador de 32 bits de propósito geral que pode ser utilizado como ponteiro de memória, ser escrito na memória, e ser definido como *breakpoint*[1].

2. **Timeout**: O registrador de timeout é um registrador de 32 bits utilizado para definir o tempo máximo de execução do processador em um determinado teste, medido em ciclos de clock.

3. **NumOffPages**: O registrador de número de páginas é um registrador de 24 bits utilizado para a execução de testes que envolvem paginação de memória. Ele é responsável por definir a quantidade de páginas de testes disponíveis para execução.

4. **EndPosition**: O registrador EndPosition é um registrador de 32 bits utilizado como *breakpoint* para a execução dos testes. A partir do acesso a esse endereço pelo processador, a infraestrutura identifica que a execução foi concluída e pode interromper o processador.

0.6 Especificação das instruções

1. **Enviar N pulsos de CLK (Opcode: 01000011, Hex: 0x43)**: Envia N pulsos de clock para o processador, permitindo o avanço de N ciclos de clock. Após o término dos N ciclos, o clock do processador é interrompido.

2. **Parar o CLK do Core (Opcode: 01010011, Hex: 0x53)**: Interrompe o clock do núcleo do processador, pausando a execução.

3. **Retomar o CLK do Core (Opcode: 01110010, Hex: 0x72)**: Retoma o clock do núcleo do processador, continuando a execução a partir do ponto de parada.

4. **Resetar o Core (Opcode: 01010010, Hex: 0x52)**: Realiza o reset do processador, utilizando um valor constante de ciclos de clock denominado `RESET_CLK_CYCLES`, que por padrão é 20. Ou seja, o processador é resetado por 20 ciclos de clock, ou pelo valor configurado em `RESET_CLK_CYCLES`.

5. **Escrever na posição N de memória (Opcode: 01010111, Hex: 0x57)**: Escreve um valor na posição N da memória. A operação necessita do envio de dois pacotes de dados de 32 bits, com o primeiro contendo o opcode e o endereço, e o segundo contendo os dados a serem escritos.

[1] O termo *breakpoint* refere-se a um ponto de parada onde o programa será interrompido quando esse ponto for atingido.

6. **Ler a posição N de memória (Opcode: 01001100, Hex: 0x4C)**: Lê o valor armazenado na posição N da memória e retorna o valor lido.

7. **Carregar bits mais significativos no Acumulador (Opcode: 01010101, Hex: 0x55)**: Carrega os 24 bits mais significativos (superiores) do acumulador.

8. **Carregar bits menos significativos no acumulador (Opcode: 01101100, Hex: 0x6C)**: Carrega os 8 bits menos significativos (inferiores) do acumulador.

9. **Somar N ao acumulador (Opcode: 01000001, Hex: 0x41)**: Adiciona o valor N ao conteúdo atual do acumulador. Esta operação é sinalizada utilizando complemento de 2.

10. **Escrever Acumulador na posição N (Opcode: 01110111, Hex: 0x77)**: Escreve o valor contido no acumulador na posição N da memória.

11. **Escrever N na posição do acumulador (Opcode: 01110011, Hex: 0x73)**: Escreve o valor N na posição de memória apontada pelo acumulador.

12. **Ler a posição do acumulador (Opcode: 01110010, Hex: 0x72)**: Lê o valor da memória na posição apontada pelo acumulador.

13. **Setar timeout (Opcode: 01010100, Hex: 0x54)**: Define um valor de tempo limite para a execução do processador, especificado em ciclos de clock.

14. **Setar tamanho da página de memória (Opcode: 01010000, Hex: 0x50)**: Define o tamanho da página de memória utilizada para os testes, configurando a quantidade de memória a ser usada para cada teste.

15. **Executar testes em memória (Opcode: 01000101, Hex: 0x45)**: Inicia a execução de um conjunto de testes na memória especificada. Esses testes são executados de forma paginada, com a possibilidade de executar um lote de testes de forma automatizada. A infraestrutura executa um teste até um determinado ponto de parada ou até o timeout. Após o término da execução do teste, o processador é "resetado"e a página de memória é alterada, repetindo todo o processo. Após o término da execução, é enviada uma mensagem de confirmação (0x676F6F64 - "good"). O número de páginas a serem utilizadas é passado como o valor imediato da instrução.

16. **Obter o ID e verificar funcionamento do módulo (Opcode: 01110000, Hex: 0x70)**: Recupera o ID do módulo e verifica se ele está funcionando corretamente. O ID é um número de 32 bits, que contém informações como a FPGA em utilização, a identificação da infraestrutura, entre outros.

17. **Definir endereço N de término de execução (Opcode: 01000100, Hex: 0x44)**: Define o endereço N como ponto de término para a execução (*breakpoint*).

18. **Definir o valor do Acumulador como endereço de término (Opcode: 01100100, Hex: 0x64)**: Utiliza o valor atual do acumulador para definir o endereço de término de execução (*breakpoint*).

19. **Escrever N posições a partir do acumulador (Opcode: 01100101, Hex: 0x65)**: Escreve valores em N posições consecutivas de memória a partir do endereço apontado pelo acumulador. Essa instrução recebe N + 1 palavras[2] de 32 bits, com a primeira sendo a instrução em si e as próximas N palavras sendo as informações a serem escritas na memória.

20. **Ler N posições a partir do acumulador (Opcode: 01100010, Hex: 0x62)**: Lê N posições consecutivas de memória a partir do endereço apontado pelo acumulador. Essa instrução retorna N palavras, com essas N palavras sendo os dados da memória.

21. **Obter o acumulador (Opcode: 01100001, Hex: 0x61)**: Recupera o valor atual armazenado no acumulador.

[2]Para fins de esclarecimento, o termo "palavra"se refere a um bloco de informações de 32 bits ou 4 bytes.

22. **Alterar prioridade de acesso à memória para o Core (Opcode: 01001111, Hex: 0x4F)**: Modifica a prioridade de acesso à memória, permitindo que o processador tenha acesso prioritário à memória.

23. **Executar até ponto de parada (Opcode: 01110101, Hex: 0x75)**: Permite a execução do processador até que um ponto predefinido (*breakpoint*) seja alcançado. Ao executar esta instrução, o processador é resetado, recebe prioridade de acesso à memória e opera até alcançar o ponto de parada ou atingir o timeout de execução. Após o término da execução, é enviada de volta uma mensagem de confirmação (0x6C75636B - "luck") e uma informação indicando se o término ocorreu por timeout ou fim da execução, além da quantidade de ciclos gastos pelo processador para a execução. Essa informação é enviada no formato: 24 bits mais significativos indicam os ciclos gastos, e os 8 bits menos significativos indicam se ocorreu timeout.

0.7 Técnicas de Utilização

Com base nas instruções existentes, diversas técnicas podem ser utilizadas para a execução dos testes, incluindo: execução manual, execução até um ponto de parada, e execução por páginas.

0.7.1 Execução Manual

A execução manual é realizada através da execução manual do fluxo após o carregamento dos testes na memória. Isso inclui tarefas como "resetar"o processador e gerar manualmente N pulsos de *clock*. O pseudo código abaixo ilustra essa abordagem:

```
Obter o ID e verificar o funcionamento do modulo;

Escrever N posicoes apartir do acumulador;

Resetar o Core;

Enviar N pulsos de CLK;

Ler a posicao N de memoria.

# Se necessario mais M pulos de clk

Enviar N pulsos de CLK
```

Listing 1: Pseudo codigo com um exemplo de execução manual

0.7.2 Execução Até Ponto de Parada (UNTIL)

A infraestrutura permite a execução automática de um teste utilizando a técnica de ponto de parada (*breakpoint*). Com essa técnica, após carregar os testes na memória, basta definir um *breakpoint* e um *timeout* em ciclos de *clock*. Com esses parâmetros definidos, é possível enviar a instrução para executar até o ponto de parada e aguardar a conclusão. O pseudo código para esse processo é apresentado abaixo:

```
Obter o ID e verificar o funcionamento do modulo;

Escrever N posicoes apartir do acumulador;

# Carregar endereco do breakpoint

Definir endereco N de termino da execucao;

Setar timeout;

Executar ate ponto de parada.
```

Listing 2: Pseudo codigo com um exemplo de execução até ponto de parada

0.7.3 Execução por Páginas

Para a execução de vários testes em sequência, é possível utilizar um sistema baseado em paginação. Nesse sistema, os testes são armazenados em blocos de tamanho fixo, por padrão 256 posições, e o controlador navega por esses blocos controlando os bits mais significativos do endereço. O funcionamento é semelhante ao da execução até um ponto de parada, mas o processo é repetido até a execução de todas as páginas. O pseudo código a seguir ilustra esse processo:

```
Obter o ID e verificar o funcionamento do modulo;

Escrever N posicoes apartir do acumulador;

# Carregar endereco do breakpoint - o endereco precisa ser menor ou igual ao
    endereco maximo da pagina por padrao 0xFF

Definir endereco N de termino da execucao;

Setar timeout;

Executar testes em memoria.
```

Listing 3: Pseudo código para execução por páginas

0.8 Fluxo de Dados

É possível enviar e ler dados para a memória de duas formas: atômica (palavra por palavra) ou em lote, enviando N palavras de uma vez.

0.8.1 Carregamento Atômico

O carregamento atômico é realizado lendo e escrevendo dados palavra por palavra. Esse método pode ser executado utilizando as seguintes instruções: "Escrever na posição N de memória", "Escrever N na posição do acumulador", "Ler a posição N de memória"e "Ler a posição do acumulador". Ao utilizar instruções baseadas em valores imediatos, é necessário enviar a instrução M vezes para ler ou escrever M palavras. Quando se utiliza o acumulador, é necessário definir o ponteiro do acumulador M vezes e executar as instruções de leitura e escrita M vezes. Dessa forma, o uso dessas instruções é mais adequado para pequenas modificações, como ler um resultado ou ler/escrever uma ou duas palavras na memória.

0.8.2 Carregamento em Lote

O carregamento em lote permite ler ou escrever M palavras utilizando apenas uma ou duas instruções. Com esse método, é necessário apenas carregar o endereço base no acumulador e utilizar as instruções de leitura em lote para ler ou escrever M palavras. As instruções de operação em lote são: "Ler N posições a partir do acumulador"e "Escrever N posições a partir do acumulador".