Implementação/Simulação unidades funcionais processador PowerNet

Gustavo Guedes de Azevedo Barbosa – 2016097471
Semar Augusto da Cunha Mello – 2016006905
Vítor Gabriel Reis Caitité - 2016111849

1) INTRODUÇÃO

No relatório a seguir são tratadas e reunidas todas as etapas da arquitetura e organização das unidades funcionais de um processador PowerNet. Para construir e planejar o funcionamento de um processador estudou-se cada parte do sistema que o constitui: somador, banco de registadores, extensor de sinais, memória de dados, memória de instruções, contador de programa e multiplexadores de duas e três entradas.

Visando compreender como o processador funciona, será fornecida uma explicação específica de cada componente, verificando seu funcionamento específico através de simulações.

2) SOMADOR

2.1) Descrição e decisões fundamentais de projeto

Um somador de N-bits consiste em um circuito que consegue somar duas entradas, cada uma delas com N-bits, e mais um sinal Carry de entrada e produz um resultado também com o mesmo tamanho, N-bits e um sinal Carry de Saída. Esse circuito somador é conhecido como Somador com Propagação do Carry, porque o sinal de Carry se propaga por todo o circuito do primeiro ao último bloco de hardware.

A forma mais simples, comum e intuitiva de criar um Somador é conectar vários somadores de 1-bit um após o outro, conectando o Carry de saída do anterior no Carry de entrada do atual e conectar o Carry de saída do atual no próximo bloco, totalizando uma cadeia de N blocos para um somador de N bits. Este então é o circuito do Ripple-Carry Adder. Ele é o somador mais modular que existe, já que podemos utilizar cada um dos somadores de 1-bit como módulos e escalarmos esse somador infinitamente. Com essa ideia pôde se implementar a lógica de se somar cada um dos 56 bits do primeiro número com os bits correspondentes do segundo número.

2.2) Simulação

Figura 1 - Simulação da Unidade Lógico-Aritmética

2.3) Comentário

A partir da simulação do código Verilog programado foi possível notar que o somador funcionou corretamente realizando as operações desejadas.

Foram testadas todas as operações possíveis de serem realizadas pela somador verificando o correto funcionamento.

3) BANCO DE REGISTRADORES

3.1) Descrição

Um banco de registradores 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. Este componente é um dos componentes mais importantes do fluxo de dados em um processador. As informações que estão sendo processados em um determinado momento devem estar armazenadas no banco de registradores.

Abaixo pode-se visualizar a representação de um banco de registradores com as entradas “Registrador de leitura número 1” e “Registrador de leitura número 2” que são responsáveis por receberem os endereços dos registradores usados na instrução em questão. Os conteúdos desses registradores são passados adiante através das saídas “Dados da leitura 1” e “Dados da leitura 2”, respectivamente. A entrada “Registrador para escrita” recebe o endereço do registrador que deve ter seu conteúdo alterado para o valor que entra em “Dados para escrita” caso necessário na instrução já que ela só será realizada se “escreve” estiver ativa.

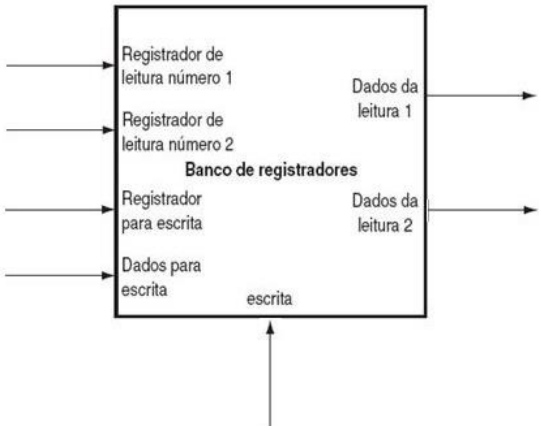

Figura 2 – Representação do Banco de Registradores

3.2) Simulação

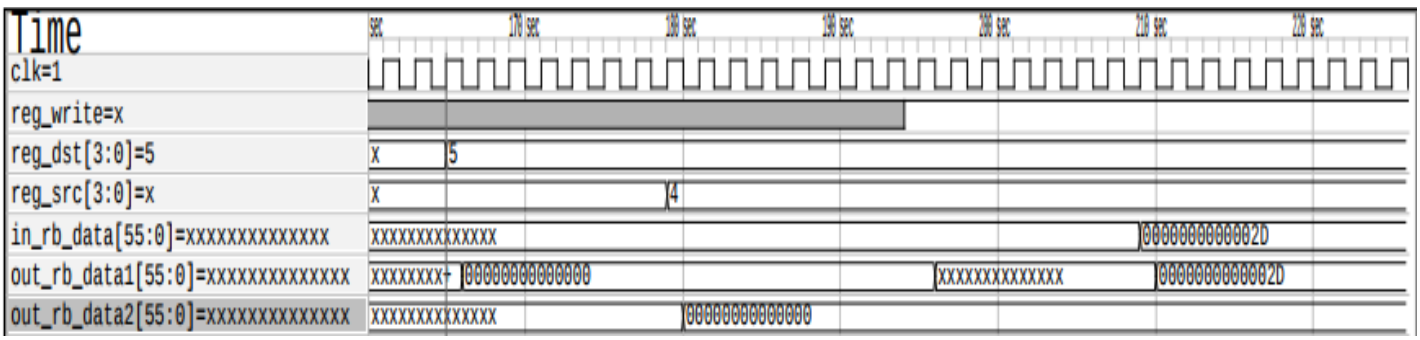

Figura 3 - Simulação do Banco de Registradores

3.3) Comentário

A figura 3 mostra uma simulação do funcionamento do Banco de Registradores. Nela vemos os valores armazenados no Banco devido à habilitação de sua respectiva entrada de controle. Esse armazenamento é ordenado, de modo que os registradores são armazenados em ordem crescente a partir do zero. Vemos também os registradores que devem ter seus dados colocados nas suas respectivas saídas. Isso significa que os valores armazenados no registrador de armazenamento, estarão na saída do Banco.

4) MEMÓRIA DE DADOS

4.1) Descrição

A figura abaixo é uma representação da Memória de Dados. Nela podemos observar quatro entradas e uma saída. A entrada “Endereço” recebe o endereço do local da memória que deve ter seu conteúdo alterado, se “EscreveMem” estiver ativo, ou lido , se “LeMem” estiver ativo. No primeiro caso (escrita) o novo valor associado a esse endereço é recebido por “Dados para escrita”. Já no segundo caso (leitura), o dado lido é passado através de “Dados da leitura”. “EscreveMem” e “LeMem” nunca estarão ativos ao mesmo tempo (no mesmo ciclo de clock).

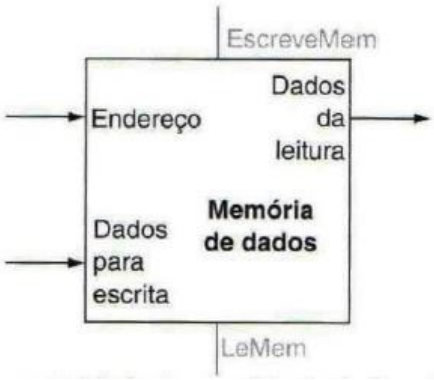

Figura 4 – Representação da Memória de Dados

Com isso a implementação do código torna-se algo simples. Recebe-se como sinais o clock, um sinal que indica se será realizada uma Escrita e um sinal que indica se será realizada uma leitura, recebe-se tambem o valor que será escrito(em caso de escrita), e um endereço de onde o dado será lido ou onde o dado será escrito. Se o sinal de leitura estiver ativo então a partir do endereço selecionamos o dado e o colocamos na saída. Se o sinal de escrita estiver ativo, então pega-se o valor da entrada e coloca-o na posição descrita pelo endereço.

4.2) Simulação

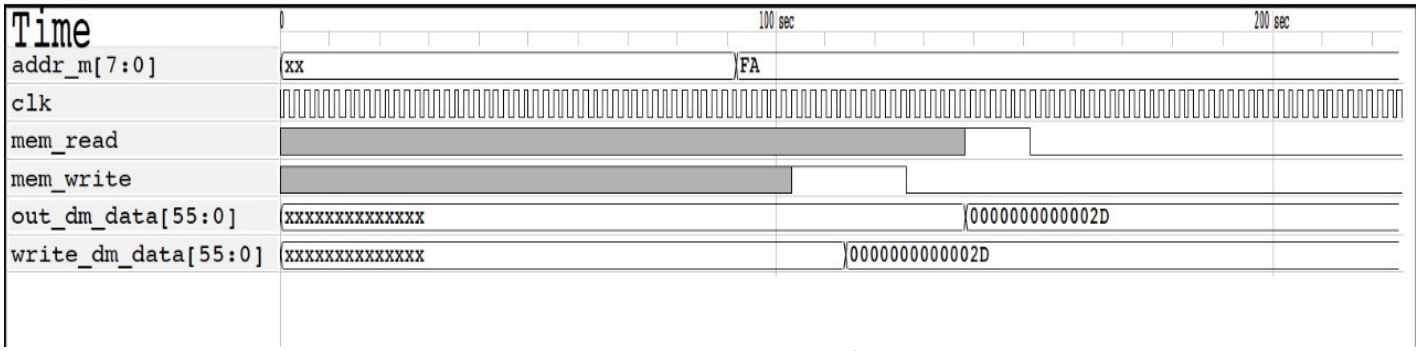

Figura 5 - Simulação da Memória de Dados

4.3) Comentário

A figura 5 mostra a simulação da Memória de Dados. Nela podemos visualizar quais dados devem ser guardados na memória de dados. A partir disso também visualizamos a saída dessa memória. Pode-se ver também que nesse caso a memória está vazia.

5) MEMÓRIA DE INSTRUÇÕES

5.1) Descrição

Essa unidade armazena as instruções a serem lidas na execução de um programa. Durante o Ciclo de Busca, é a Unidade de Controle que atua. Uma nova instrução é buscada da Memória de Instruções para que possa ser decodificada.

Para a criação do código deste bloco deve levar em conta que ele tem como entradas o sinal de clock e um endereço, então a partir desse endereço(de 8 bits), seleciona-se o dado requisitado e coloca-o na saída.

5.2) Simulação

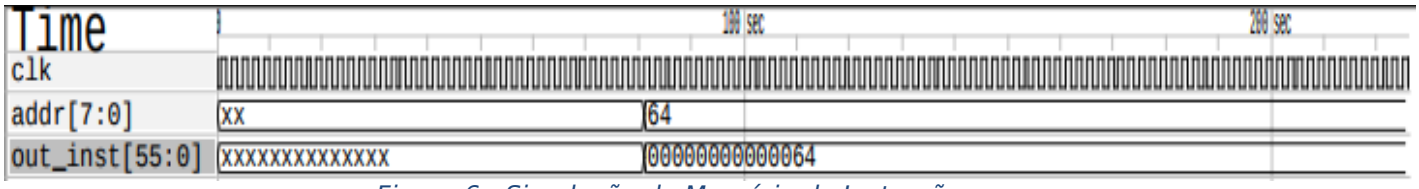

Figura 6 - Simulação da Memória de Instruções

5.3) Comentário

A figura 6 mostra a simulação da Memória de Instruções. Percebe-se que a partir de um endereço recebido, essa unidade retorna um dado de 56 bits.

6) ESTENSOR DE SINAIS

6.1) Descrição

A extensão do sinal é a operação, na aritmética do computador, de aumentar o número de bits de um número binário, preservando o sinal do número (positivo / negativo) e o valor. Isso é feito adicionando dígitos ao lado mais significativo do número, seguindo um procedimento dependente da representação de número particular usada.

Com isso, para o desenvolvimento desse bloco que estende o sinal de 16 bits para 56 bits, bastou copiar os primeiro 15 bits da entrada para a saída, e então replicar o último bit da entrada em todos os outros bits mais significativos da saída.

6.2) Simulação

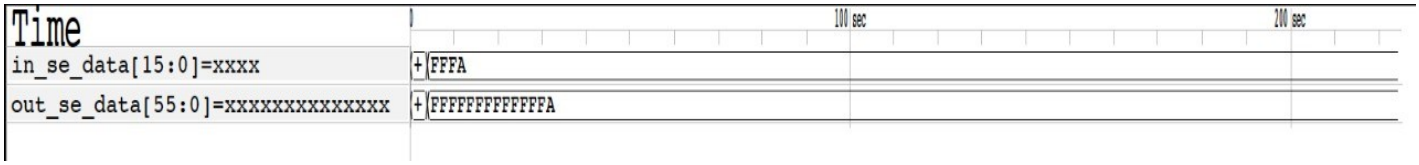

Figura 6 - Simulação do Estensor de Sinais

6.3) Comentário

Observa-se facilmente que o estensor de sinais está cumprindo seu papel de maneira correta replicando o bit mais significatico de modo a estender um sinal de 16 bits para um de 56 bits.

7) MUX 2x1 E MUX 3x1

7.1) Descrição

Um multiplexador, ou mux, é um dispositivo que seleciona as informações de duas ou mais fontes de dados num único canal. São utilizados em situações onde o custo de implementação de canais separados para cada fonte de dados é maior que o custo e a inconveniência de utilizar as funções de multiplexação.

A implementação do código é bem simples, recebe-se os dados na entrada (2 ou 3 dados dependendo do mux) e a partir de uma entrada de seleção selecionar uma das entradas e colocá-las na saída.

7.2) Simulação

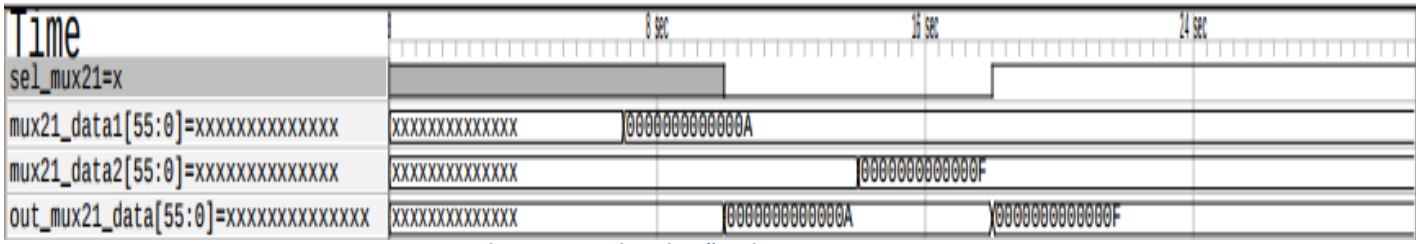

Figura 7 - Simulação do MUX 2x1

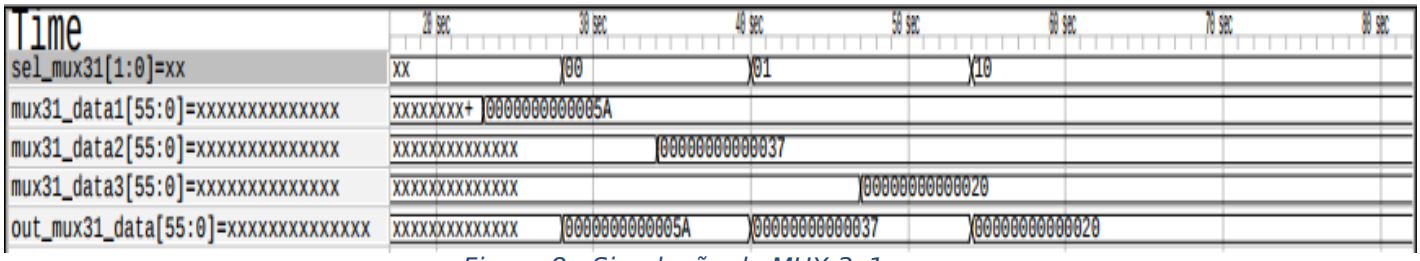

Figura 8 - Simulação do MUX 3x1

7.3) Comentário

Pôde-se percerber claramente que ambos os mux selecionam uma das entradas e a colocam na saída com base na entrada de seleção. Esse é exatamente o comportamento esperado para um mux.

8) CONTADOR DE PROGRAMA (PC)

8.1) Descrição

Contador de programa é um registrador de uma Unidade Central de Processamento que indica qual é a posição atual na sequência de execução de um processo. O contador de programa é automaticamente incrementado para cada ciclo de instrução de forma que as instruções são normalmente executadas sequencialmente a partir da memória, sendo que o contador de programa deve ser colocado a zero no inicio da execução do mesmo.

A lógica para a criação do código é bem simples, visto que se trata de apenas um registrador, ou seja é necessário apenas que esse bloco receba um valor e o guarde. Além disso ele ainda recebe um sinal reset que quando ativado zera o PC.

8.2) Simulação

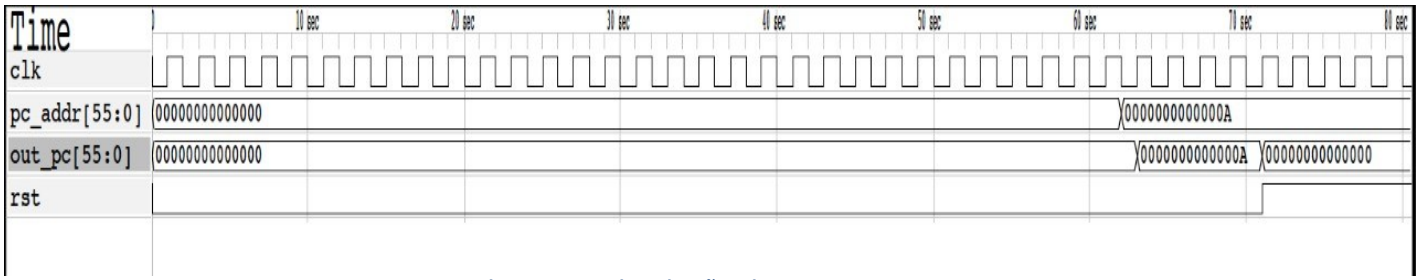

Figura 9 - Simulação do PC

8.3) Comentário

Percebe-se que o PC recebe uma entrada e a coloca na saída, percebe-se ainda que quando ele é carregado com outro valor na entrada ele o coloca na saída a partir da próxima borda de clock. Por fim pôde-se visualizar que quando rst é 1 (ou seja o reset está ativado), então o PC volta ao seu valor inicial.

9) CONCLUSÃO

O desenvolvimento das unidades de um processador foi fundamental para conseguirmos verificar, compreender e unir todo o conhecimento adquirido na disciplina teórica de Organização de Computadores aplicando-o na prática, além de entender melhor a arquitetura e o funcionamento real de um processador deste tipo.

A programação de cada uma das partes do processador foi realizada com sucesso e sem maiores problemas, já que os testes individuais são mais imediatos de serem realizados. Pôde-se perceber o funcionamento de cada um dos blocos de maneira conclusiva e efetiva para o aprendizado.