Relatório Final

NUMA – Non Uniform Memory Access

Aluno: Renan Galeane Alboy

RA: 11415088

1.0– Introdução

A arquitetura NUMA (Acesso não uniforme a memória), é utilizada para multiprocessadores. Esse tipo de projeto é caracterizado por cada processador possuir sua memória e poder acessa-la, porém as demais memórias dos outros processadores também podem ser utilizadas, ou seja, trata-se de uma memória compartilhada. O acesso a sua memória local ocorre de maneira mais rápida do que o acesso na memória não local, que seria a memória dos demais processadores, pois é necessário um controlador de acesso para tratar situações de possível acesso simultâneo a uma mesma memória.

Entre as vantagens do uso NUMA esta no aumento da banda de memória proporcionado pelas múltiplas memórias. Isso pode auxiliar a melhorar o desempenho diminuindo o tempo de execução. Porém, a complexidade na comunicação entre os processadores ser alta, pois requerer um pouco mais de esforço do software para tirar vantagem da memória distribuída de forma que não prejudique o desempenho.

1.1 – Objetivo

Objetivo do trabalho é a implementação de uma arquitetura NUMA em processador mips. Realizando também melhoria da parte de vídeo e a implementação de novas instruções.

2.0–Implementação

Para a realização do trabalho foi utilizada a placa Cyclone II, representada na Figura 2.1. Para a sinterização e análise foi utilizado o software Quartus II.

Figura 2.1 – Placa FPGA Cyclone II .

2.1– Melhorias de vídeo

Para a melhoria do vídeo foram feitas alterações que permitam que mais cores sejam mostradas na tela. Originalmente eram mostradas as cores azul e amarela que

representavam o pixel desligado e ligado respectivamente. Foi escolhida essa melhoria para possibilitar mais testes e também para evitar ocupar demais a memória.

A implementação foi feita triplicando a memória de vídeo de forma que seja possível tratar cada uma como um canal com uma variação diferente, tal que seja possível representar oito cores no padrão rgb. Cada memória possui espaço de 24 words, totalizando 72 words, e o acesso a memória ocorre a partir da primeira a partir de um único ponto, somando ao ponto inicias em que houve o acesso 24, para alcançar a segunda parte da memória, e em seguida 48 ao ponto inicial para alcançar a terceira parte da memória.

2.2– Implementação de instruções

Na parte de instruções foram adicionadas as instruções ROR, ROL, NOR e MUL, sendo todas do tipo R. As instruções ROR e ROL foram implementadas com o campo função das instruções SLLV e SRLV respectivamente. Já a instrução MUL foi adicionada com o campo de função 27 e NOR com o campo 26. Foram escolhidas essas instruções por permitirem a manipulação da memória de maneira mais visual, tornando o exemplo mais claro.

2.3- Implementação do NUMA

Para a implementação do NUMA foi utilizado dois processadores mips e utilizado uma memória de vídeo. A arquitetura planejada é vista na Figura 2.2.

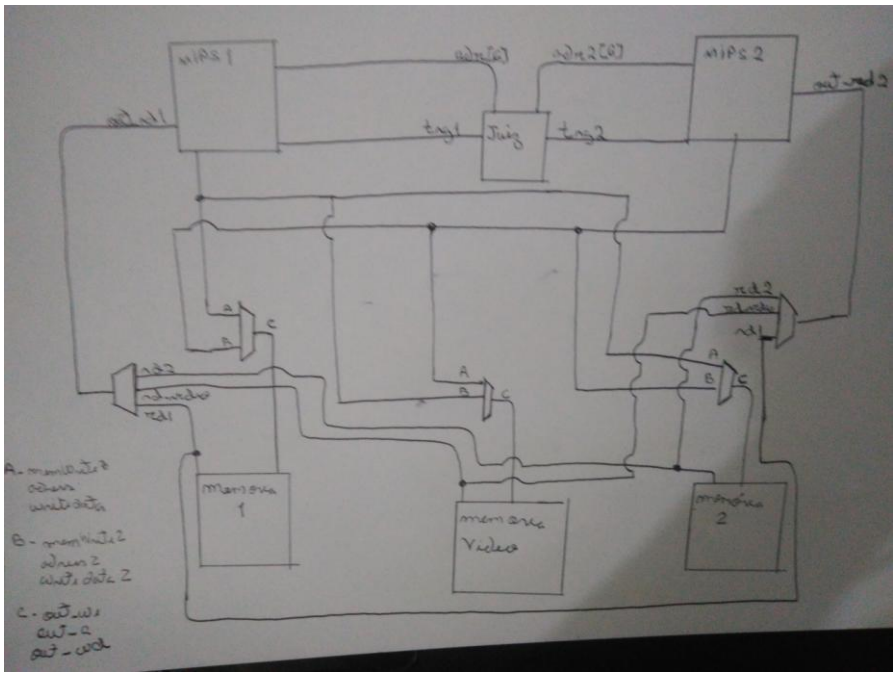

Figura 2.2 – Arquitetura planejada para o trabalho.

Como pode ser observado, cada mips possui a sua memória. Inicialmente a memória de cada mips também continha parte de memória de vídeo, porém a memória de vídeo foi separada das memórias dos mips e criada de forma independente para ser utilizada por ambos.

A comunicação de ambos os mips com suas memórias a memória de vídeo ocorre com a utilização de um barramento, direcionando os dados e a partir do que foi recebido para o caminho correto. Em cada memória há um mux que serve como seletor para a entrada direcionar o acesso a memória, como é mostrada na Figura 2.3. Esse mux recebe como input o memwrite, adress e writedata de ambos os mips, também recebe como sinal de controle o 7 bit do endereço de cada mips para permitir ou não a escrita na memória. Dependendo da combinação destes 2 bits é permitida ou não o uso da memória.

É importante resaltar que o cada mips tem prioridade sobre o acesso a sua memória, assim, necessitando de permissão para executar em uma memória não local e somente se essa não estiver sendo usada.

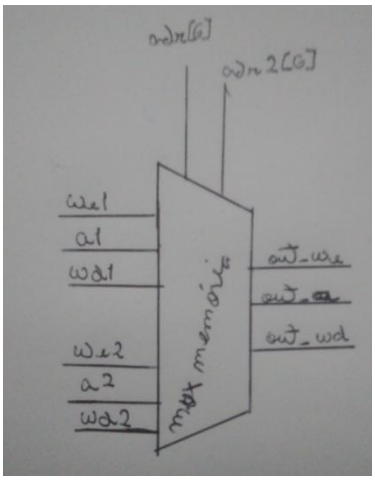

Figura 2.3 – Mux de seleção entre mips e memória.

O outro mux feito foi o responsável por levar o sinal de readdata da memória até o mips. O sinal de readdata que chega ao mips vindo de um mux recebe como input as saídas da memórias, como mostra a Figura 2.4.

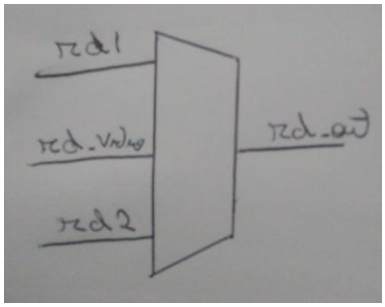

Figura 2.4 – Mux de envio do sinal readdata ao mips.

Para que tal controle sobre o acesso a memória foi implementado um módulo denominado de juiz. Essa parte é responsável por receber dos mips os pedidos de escrita na memória, a estrutura do juiz pode ser vista na Figura 2.5.

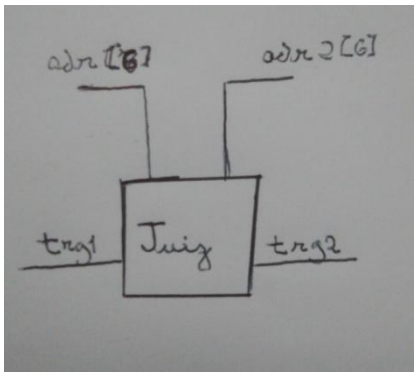

Figura 2.5 – Componente Juiz .

Como resposta ao sinal recebido é enviado a cada mips a permissão ou não para escrever na memória. Caso a memória que um mips tente acessar já esteja sendo utilizada, é provocada uma espera, passando-o para estado de stall, até que ele consiga a devida permissão para executar.

3.0–Testes

3.1 – Testes da implementação e problemas

Para a realização dos teste foram feitos dois arquivos .dat diferentes, um para cada MIPS instanciado, em que cada um utiliza parte das instruções implementadas. No arquivo que é executado no primeiro mips são utilizadas as instruções ROR e MUL, já o arquivo que é executado no segundo mips são executadas as instruções ROL e NOR.

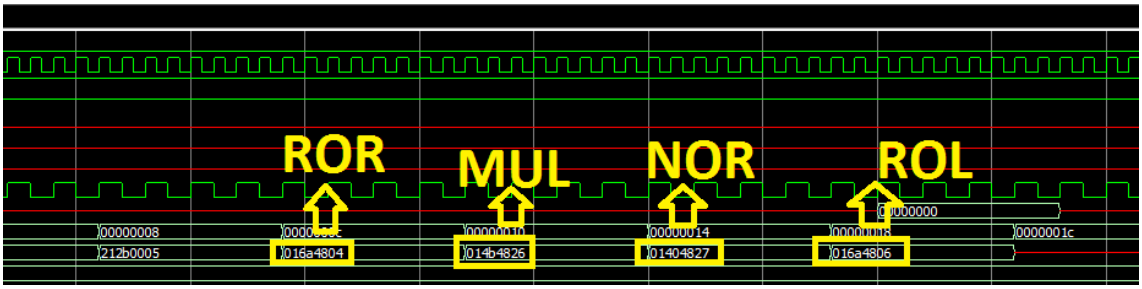

Figura 3.1 – Wave form das instruções implementadas.

A diferença de cada execução pode ser percebida pela execução forma em que se apresentam no monitor. A execução do primeiro mips apresenta duas linhas, uma mais lenta do que a outra, indo da direita para a esquerda, enquanto que a execução do segundo apresenta uma linha que vem da esquerda para a direita e causa alternância de pixels ligados e desligados na parte azul da tela. Os teste dos programas mostram

os resultados da implementação das instruções da melhoria de vídeo, porém nestes testes a memória de vídeo ainda estava junto com a memória do mips.

Durante os testes com as memórias separadas houve dificuldades com a comunicação do mips com a memória de vídeo. Essa dificuldade pode ter ocorrido pela maneira com que a memória de vídeo foi estruturada. Porém, analisando os programas utilizados nos testes em que a memória de vídeo ainda estava junto com a memória que continha o programa é possível ver que tanto a alteração de cores como as instruções estavam funcionando. Isso corrobora com a ideia de que a dificuldade está na comunicação do mips em fazer acesso a memória de vídeo. Na imagem 3.2 é mostrada a imagem do monitor com a execução. Nele é visto a melhoria que foi implementada, estando na tela as oito cores possíveis.

Figura 3.2 – Imagem do monitor com melhoria de vídeo.

3.2 – Testes comparativos

Em relação a testes comparativos foram utilizadas a NUMA com outra do mips em uma versão somente multiciclo. Na execução somente com o mips multiciclo houve uso 29% de uso de memória. Enquanto que a vesão em NUMA o uso de memória da FPGA alcançou 41%, porém devido ao problema com a comunicação, que pode fazer com que vários elementos lógicos não sejam contabilizados, esse valor pode ter sido afetado. Considerando que são instanciados 2 mips e 3 memórias o valor de uso da memória tenderia crescer.

No quesito desempenho a versão com NUMA apresenta um melhor desempenho do multiciclo por permitir que as os programas rodem em paralelo. Porém, caso o um mips tente acessar uma memória não local pode cair no estado de stall e ter que aguardar o mips que está utilizando a memória terminar de executar. O que pode causar uma queda no desempenho caso tal situação seja muito frequente.

4.0- Conclusão

O uso da arquitetura NUMA mostra-se muito interessante para a melhoria do desempenho quando comparado com o mips multiciclo. Porém, deve-se atentar a quantidade de memória que sua implementação irá ocupar em comparação a quantidade de memória da placa. Outro ponto importante é o programa que ira ser executado em cada mips, eles podem causar variações de desempenho caso tentem acessar com muita frequência a memória não local, isso ocorre porque existem algoritmos que não são otimizados para serem executados em uma arquitetura NUMA.

5.0- Bibliografia

Patterson, David, and John L. Hennessy. Arquitetura de Computadores: uma abordagem quantitativa. Vol. 5. Elsevier Brasil, 2014.

Blagodurov, Sergey, et al. "A case for NUMA-aware contention management on multicore systems." Proceedings of the 19th international conference on Parallel architectures and compilation techniques. ACM, 2010.

Manchanda, Nakul, and Karan Anand. "Non-uniform memory access (NUMA)." New York University (2010).

Majo, Zoltan, and Thomas R. Gross. "Memory system performance in a NUMA multicore multiprocessor." Proceedings of the 4th Annual International Conference on Systems and Storage. ACM, 2011.