R2NP: Processador de Rede RISC Reconfigurável

Henrique Cota de Freitas
Pontifícia Universidade Católica de Minas Gerais
cota@pucminas.br

Carlos Augusto Paiva da Silva Martins
Pontifícia Universidade Católica de Minas Gerais
capsm@pucminas.br

Resumo

Este artigo apresenta o projeto do R2NP (Reconfigurable RISC Network Processor). Sua principal aplicação é como elemento de processamento em sistemas de comunicação de dados, efetivando e estabelecendo conexões e a comunicação entre os equipamentos e dispositivos de rede. As principais características deste processador são a reconfigurabilidade de alguns blocos lógicos, o suporte a multi-protocolo e a topologias dinâmicas, objetivando principalmente a não dependência e a então, flexibilidade de funcionamento. Para isto, o R2NP possui um conjunto de instruções específicas e microarquitetura dedicada para suportar as características mencionadas.

1. Introdução

Este trabalho tem por objetivo principal apresentar a evolução e estágio atual do R2NP - *Reconfigurable RISC (Reduced Instructions Set Computing) Network Processor*.

Durante o ano de 2001 foi consolidado o projeto da microarquitetura e conjunto de instruções do processador de rede RCNP [11,12] - *Reconfigurable CISC (Complex Instructions Set Computing) Network Processor*, projetado por nós e publicado nos anais do WSCAD'01. Suas características principais foram utilizadas como base para o desenvolvimento do R2NP e os resultados encontrados serviram de parâmetros de comparação para validar analiticamente a nova proposta de microarquitetura e novas instruções RISC.

Com o avanço das arquiteturas dos equipamentos de rede, a exigência por processadores mais eficazes se tornou essencial. A corrida pelo desenvolvimento destes tipos de processadores [28] provocou fusões entre empresas. Os projetos são motivo de segredo e a liberação dos *datasheets* ainda é lenta. As fusões aconteceram entre a Motorola [21] e a C-Port [7], a Agere [2] e a Lucent e entre a Sitera [24] e a Vitesse. Outras empresas também fabricam processadores de rede, são elas: EZChip [10], Intel [16], IBM [14], MMC Networks [19], Lexra [17] e a CISCO [4,6].

É possível encontrar processadores de rede em roteadores [5], *switches* [15] e equipamentos de telefonia [1,16], por exemplo. Não se descarta, obviamente, a possibilidade de ampla utilização destes tipos de processadores em atividades relacionadas ao processamento paralelo e distribuído usando clusters [3] de computadores. Cada elemento do cluster, que é responsável pelo roteamento entre os computadores, poderia ser um processador de rede utilizando protocolo leve para comunicação. Um tipo de rede adequada a este tipo de aplicação seria a *System Area Network* (SAN) [20], específica para aplicações em nível de sistema e que necessita de protocolos leves de comunicação para não perder em desempenho durante o tráfego de dados.

A característica principal do processador de rede é o suporte a comunicação de dados, que aumenta em função da necessidade crescente de conexões entre os diversos dispositivos de rede. Ou seja, estabelecer efetivamente a comunicação entre os diversos dispositivos e equipamentos da rede. É importante que se tenha um bloco de processamento, que suporte e flexibilize as conexões, através de programas previamente estabelecidos e baseados em protocolos de comunicação, que poderão em tempo de execução, garantir o tráfego de dados, manter e alterar topologias de rede. A grande exigência atualmente é com a qualidade de serviço em comunicação de dados. Roteadores ativos [26] são projetados para acelerar e distribuir o processamento de pacotes que trafegam pelos equipamentos de rede. Ou seja, neste tipo de rede [8,9] uma mensagem pode conter instruções para serem executadas em roteadores [26] ou *switches*. Um equipamento de rede [23] passa a ser elemento de processamento e atuar de forma ativa na execução do pacote enviado. A necessidade por qualidade de serviço (QoS) exige cada vez mais por melhorias de desempenho nos equipamentos de rede e a evolução destes equipamentos [5,26] está ligada diretamente aos Processadores de Rede.

O que será descrito neste artigo é como o R2NP pode contribuir, através de seu conjunto de instruções otimizadas e de sua microarquitetura dedicada, com as novas possibilidades de utilização (Clusters [3] e redes SAN [20]), apresentando a evolução de uma idéia e o modelo analítico no qual está baseado a comparação e os resultados que serão apresentados entre a proposta do processador de rede RCNP [11,12] e o R2NP.

2. Projeto R2NP

Este tópico irá apresentar as duas propostas e uma visão ampla das principais diferenças. O item 2.1 será mais resumido, já que foi apresentado no WSCAD'01 [11].

Os blocos principais das duas propostas são (figura 4):

i) *Buffers* reconfiguráveis: São *buffers* da porta de entrada que alocam os pacotes enquanto se espera a liberação de uma saída. Estes *buffers* adequam seus tamanhos de acordo com o tamanho dos pacotes que estão chegando, utilizando técnicas de reconfiguração. Os pacotes guardados nos *buffers* reconfiguráveis podem ser redirecionados para a saída através da chave *crossbar* ou para a memória, desde que o programa esteja escrito para tal.

ii)Chave *crossbar* configurável: Esta chave se equivale a uma matriz de conexões internas (entradas/saídas) responsável por redirecionar os pacotes de entrada para a porta de saída.

A idéia e o funcionamento básico permaneceram no processo de evolução da microarquitetura, porém o novo conjunto de instruções RISC tem a função de melhorar e otimizar o desempenho do processador, como descrito no item 4.

2.1. Microarquitetura e Conjunto ISA do RCNP

O processador RCNP [11,12] foi desenvolvido durante a graduação em Ciência da Computação. Neste período, o objetivo era utilizar um método de projeto capaz de demonstrar as características de funcionamento de um processador CISC dedicado. Para tal, o simulador (NPSIM – *Network Processor Simulator*) [13] desenvolvido através da linguagem C++, foi utilizado para validar funcionalmente o processador, possibilitando a visualização das operações executadas. As características básicas da microarquitetura do processador RCNP são as seguintes:

1. Oito portas de entrada
   - ✓ *Buffers* específicos para cada entrada
   - ✓ *Buffers* reconfiguráveis.
2. Oito portas de saída
3. Chave *crossbar* configurável (Conexão entre as portas e registradores)
4. DMA (Acesso Direto à Memória)
5. Oito registradores de propósito geral
6. Barramento de dados de 8 bits e 24 bits de endereçamento
7. Memória de tamanho máximo de 16Mbytes (palavra de memória de 1byte)

As principais características do conjunto de instruções:

1. Conjunto de instruções de propósito geral
   - ✓ Instruções aritméticas (Ex. ADD e SUB)
   - ✓ Instruções lógicas (Ex. AND e OR)
   - ✓ Instruções de acesso à memória (Ex. LOD e STO)
   - ✓ Instruções de desvio (Ex. JMP e JNZ)
2. Conjunto de instruções de rede
   - ✓ Leitura da porta de entrada (Ex. ENT)
   - ✓ Escrita na porta de saída (Ex. SAI)
   - ✓ Controle da *crossbar* (Ex. SEC)
   - ✓ Controle do reg. de status (Ex. LRS e SRS)

Por se tratar de uma microarquitetura CISC, algumas instruções além de *load* e *store*, também possuem no formato de instrução, um campo específico para acesso à memória. Instruções como ADI A,Imed e CALL End não mais existirão no projeto R2NP, uma vez que o acesso à memória será restrito a poucas instruções. Outra característica deste projeto é o microcódigo (palavra de controle) gerado pela unidade de controle. Cada estágio da instrução (Ex. busca, decodificação e execução) é representado por um microcódigo diferente. O caminho de dados do processador RCNP não possui estrutura em *pipeline*, o que obriga a execução de uma instrução após a outra, seqüencialmente.

O ponto forte desta microarquitetura é a unidade de comunicação composta por portas I/O, *crossbar* e *buffers* reconfiguráveis, que permaneceram no projeto R2NP, com poucas diferenças.

Detalhes do processador RCNP poderão ser obtidos através das referências [11,12,13].

2.2. Microarquitetura e Conjunto ISA do R2NP

Basicamente, o R2NP vem otimizar o caminho de dados, a microarquitetura de comunicação de dados, o formato e o conjunto de instrução, com o intuito de aumentar o desempenho de execução dos programas. Neste projeto cada instrução possui um formato fixo, o que facilita o estágio de decodificação da instrução. A figura 1 ilustra o formato de instruções do R2NP.

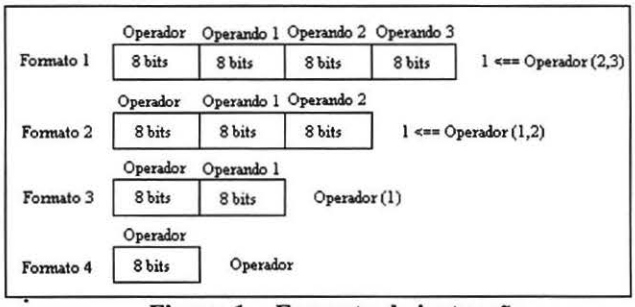

Figura 1 – Formato de instrução

Exemplos:

Formato 1: *ADD* (operador) *A* (operando 1), *B* (operando 2), *C* (operando 3)
Formato 2: *MOV* (operador) *A* (operando 1), *B* (operando 2)
Formato 3: *LRS* (operador) *A* (operando 1)
Formato 4: *HLT* (operador)

A tabela 1 relaciona todas as instruções do R2NP e a tabela 2 descreve a função das instruções de rede. Houve uma redução considerável na quantidade de instruções, conseqüência direta do projeto RISC e da otimização da microarquitetura, que possibilitou a exclusão de instruções não mais necessárias.

Tabela 1 – Instruções do R2NP

Propósito geral			
ADD A,B,C	*MOV A,B*	*SPUSH A*	*CONV*
SUB A,B,C	*INC A*	*LPOP A*	***Rede***
MUL A,B,C	*DEC A*	*JMP A*	*FCX A,B,C*
DIV A,B,C	*LOD A,End32*	*JZ A*	*LOB A*
AND A,B,C	*LDA A,End16*	*JMZ A*	*BRC A*
OU A,B,C	*LOX A,B*	*JMI A*	*SAI A,B*
XOR A,B,C	*LDI A,Imed16*	*JNZ A*	*LRS A*
NEG A	*STO End32,A*	*JNI A*	*SRS A*
ROD A	*STR End16,A*	*CALL A*	*SEC A,B*
ROE A	*STX A,B*	*RET*	*ENT A,B,C*

No caso de instruções de *load* e *store*, existem dois tipos de instruções, uma que acessa apenas a memória interna de 64 kbytes (LDA, LDI, STR) e outra (LOD, STO) que acessa memória interna e externa de tamanho máximo 16Gbytes. Quando se utiliza o segundo tipo, a próxima posição de memória guarda o endereço de 32 bits. Portanto as instruções LOD e STO utilizam duas células de memória e por isso é necessário um ciclo de busca a mais para o endereço. No entanto, as instruções LDA, LDI e STR, que precisam de apenas 16 bits para endereçar a memória interna, gastam apenas uma célula de memória e apenas um ciclo de busca.

Tabela 2 – Descrição das instruções de rede

Rede	***Descrição do funcionamento***
ENT A,B,C	Lê buffer (reg. B) posição (reg. C) e carrega no reg. A
BRC A	Broadcast do buffer (reg. A) para todas as saídas
SAI A,B	Valor do reg. A para saída (reg. B)
LRS A	Carrega no reg. Status valor do reg. A
SRS A	Carrega no reg. A valor do reg. de Status
SEC A,B	Redirecciona buffer (reg. A) para saída (reg. B)
FCX A,B,C	Fecha conexão entre buffer (reg.A) e saída (reg. B) Reg. C carrega byte de desconexão no buffer (reg. A)
LOB A	Carrega no reg. A, nº da porta de entrada que recebeu o pacote

A unidade de controle e conseqüentemente o microcódigo foram substituídos pelo caminho de dados com estrutura em *pipeline*, utilizando os bytes da própria instrução como forma de ativar os blocos lógicos necessários para cada estágio de instrução. Os estágios de instrução estão divididos em 5 etapas (figura 2):

i) Busca da instrução (B).
ii) Decodificação da instrução (D). Leitura em banco de registradores.
iii) Execução da instrução (E).
iv) Acesso de escrita ou leitura na memória (M) ou *Buffer* reconfigurável (BF).
v) Resultados (R). Escrita em banco de registradores.

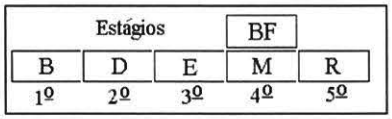

Figura 2 – Estágios do *pipeline*

A diferença básica para o modelo conceitual [22] dos estágios de *pipeline* é a inclusão do estágio BF (*Buffer* Reconfigurável). Este estágio está sobreposto ao estágio de acesso à memória, porque os dois não são executados para uma mesma instrução. Uma instrução *load* utiliza o estágio M e uma instrução de entrada (ENT) utiliza o estágio BF. É importante ressaltar que hierarquicamente os *buffers* estão mais próximos do Processador de Tarefas do que a memória (registradores ← *buffers* ← memória).

As características básicas da microarquitetura são:

1. Oito *microengines* (ASIC's – *Application-Specific Integrated Circuits*) substituindo cada buffer específico de entrada do projeto RCNP.
2. Multiplexador, *crossbar* e *buffers* reconfiguráveis
3. DMA (Acesso Direto à Memória) – Hardware dedicado (ASIC)
4. Memória interna e *cache* do Processador de Tarefas.
5. Quantidade total (máximo) de registradores: 256 (palavra de 64 bits)
6. Barramento de dados e de endereço de 32 bits
7. Memória de tamanho máximo de 16Gbytes (palavra de 32 bits / 4G células)

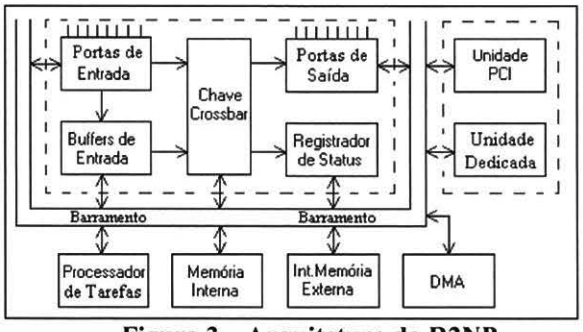

Figura 3 – Arquitetura do R2NP

O SoC (*System on Chip*) [29] é um chip composto por vários blocos lógicos, que normalmente são externos e trabalham em conjunto com processadores. Neste caso o R2NP é um SoC que possui um processador, oito *microengines*, portas de I/O, *buffers*, *crossbar*, multiplexador e memória todos internos (figuras 3 e 4) a uma única pastilha. A grande vantagem do SoC é a diminuição do tamanho, a proximidade dos blocos lógicos e o desempenho de processamento. O R2NP por ter características de um SoC, será visto com um processador internamente. Este o motivo de se dizer, que o *cache* está interno ao Processador de Tarefas, como mencionado no item 3 das características básicas da microarquitetura. O *cache* não é um bloco que está no mesmo nível da memória interna, o *cache* está em um nível mais baixo, mais próximo ou internamente a unidade de processamento.

Outra característica importante é a utilização de *microengines* e multiplexador (figura 4) no lugar de *buffers* temporários (estáticos) de entrada. A *microengine* é um hardware dedicado que tem a função de ler um espaço de memória, onde estão guardadas informações referentes ao protocolo, para tomar decisões de roteamento preliminares, redirecionando os pacotes para os *buffers* reconfiguráveis ou para saída, através da chave *crossbar*. A função do multiplexador é possibilitar a re-utilização de uma porta de entrada, que estava ligada a uma *microengine*. Pode-se perder uma *microengine*, mas continua-se com 8 portas.

No R2NP não existem mais as duas categorias de *buffers* (temporários e permanentes) [11]. Os *buffers* temporários foram substituídos pelas microengines e os *buffers* permanentes continuam reconfiguráveis como na versão RCNP. O seletor de conexão [11] passou a se chamar apenas Chave *Crossbar* Configurável. Seu funcionamento é similar a uma matriz de conexões internas capaz de redirecionar os pacotes das portas de entrada para as portas de saída, tal como na versão RCNP. A figura 4 ilustra a microarquitetura das portas de entrada, *microengines*, chave *crossbar*, multiplexador e *buffers* reconfiguráveis.

Ao contrário da ULA versão RCNP, o R2NP possui uma ULA capaz de fazer operação em ponto flutuante utilizando técnicas de *pipeline* internamente.

Existem duas interfaces de comunicação: i) Unidade dedicada e ii) Unidade PCI. A primeira é responsável pela interface entre processadores R2NP utilizando um protocolo leve de comunicação. A segunda interface é necessária para estabelecer uma comunicação padrão com outros tipos de processadores, que possam em determinadas aplicações, trabalhar em conjunto com o R2NP.

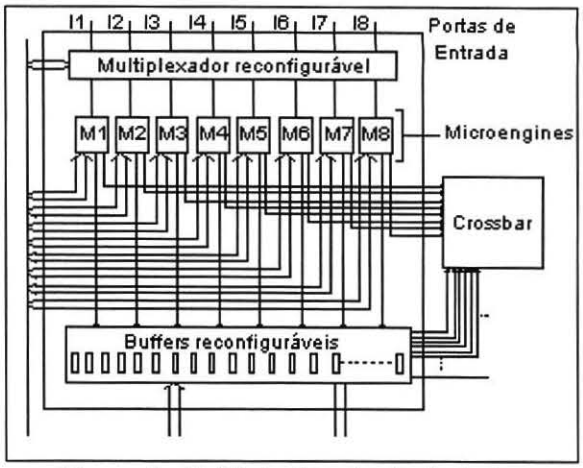

Figura 4 – Unidade de entrada de dados

3. Métricas de Desempenho

Algumas métricas [22] foram definidas com forma de analisar o R2NP. Estas métricas ajudarão a caracterizar melhor a qualidade de desempenho em relação ao modelo RCNP projetado. Basicamente, cada análise é feita com o objetivo de verificar a diminuição do tempo de processamento, variável principal e mais próxima do real para uma análise de desempenho.

3.1. Métricas para Análise

Como não existem fisicamente os modelos RCNP e R2NP, a freqüência de *clock* de 500Mhz será estipulada para fins de cálculos comparativos. Além da freqüência será utilizado o tempo de processador, o número de ciclos de *clock*, número médio de ciclos por instrução, número de instruções do programa é fator de performance. O objetivo será relacionar cada variável e definir ao final um fator ou uma relação, que indique o melhor desempenho entre os processadores projetados.

3.2. Relação entre as Métricas

Para as métricas estão definidas as seguintes variáveis:

Fc → Freqüência do *clock* (Hz)
Tp → Tempo de processador (s)
Ncc → Número de ciclos de *clock* do programa
Nmci → Número médio de ciclos por instrução
Nip → Número de instruções do programa
Fp → Fator de performance

Então, podemos relacioná-las da seguinte forma:

Nmci = Ncc / Nip
Tp = Nip * Nmci / Fc

É importante ressaltar que o modelo RCNP não utiliza *pipeline*, suas instruções são executadas seqüencialmente,

uma após o termino da outra. Para o modelo R2NP é utilizado o *pipeline* e em casos ideais, a diferença entre a execução de uma instrução e outra é de apenas 1 ciclo (*pipeline* cheio). Esta condição depende do programa que está sendo executado, já que instruções condicionais e de salto podem provocar bolhas (espaços vagos) durante a execução do *pipeline*.

O item 4 irá utilizar estas métricas para fins de comparação entre o projeto RCNP e R2NP.

4. Resultados

Os resultados apresentados são referentes a três tipos de análise:

1. Relação entre as métricas (RCNP x R2NP)
2. Quantidade de instruções de acesso à memória
3. Quantidade de instruções de acesso às portas de entrada e saída

As figuras 5 (anel unidirecional), 6 (hipercubo) e 7 (árvore balanceada) são as topologias utilizadas para análise de desempenho dos programas representados nas tabelas 3, 4 e 5.

Anel unidirecional: O objetivo desta topologia é criar uma condição em que cada micro esteja conectado com sua porta de saída na porta de entrada do micro à direita. Esta topologia foi criada por apenas um processador R2NP. A chave *crossbar* é responsável por estabelecer todas as conexões.

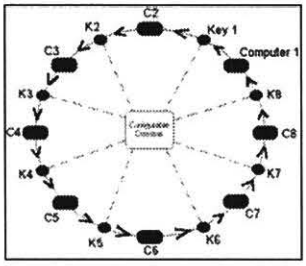

Figura 5 – Topologia anel unidirecional

Tabela 3 – *Assembly* anel unidirecional

	RCNP	Ciclos	R2NP	Ciclos
1	LDI A,07	5	LDI A,0008	5
2	LDI B,01	5	LDI B,0001	1
3	PUT B	4	LDI C,000A	1
4	LDI B,08	5	LDI D,0007	1
5	SEC	4	FCX A,B,C	1
6	FCX	4	DEC A	1
7	PUT B	4	LDI B,0006	1
8	DCR B	4 * 8	FCX A,B,C	1 + 1 * 7
9	SEC	4 * 7	DEC A	1 * 7
10	FCX	4 * 7	DEC B	1 * 7
11	DCR A	4 * 7	JMZ D	2 + 1 * 7
12	JMZ 000009	5 * 8	HLT	1
13	HLT	4	----------	----------

Hipercubo: Nesta topologia, o endereço de cada nó difere de apenas 1 bit para o nó adjacente. Esta diferença de bit é utilizada no programa de roteamento da tabela 4. Cada nó desta topologia representa um processador R2NP.

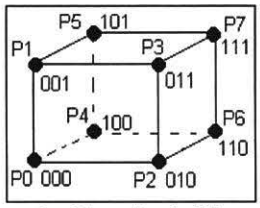

Figura 6 – Topologia hipercubo

Tabela 4 – *Assembly* hipercubo

	RCNP	Ciclos	R2NP	Ciclos
1	LDX F,000006	5	LDI N,0013	5
2	LDI D,03	5	LDI M,0015	1
3	PUT	4	LDI F,0007	1
4	SUI B,00	5	LDI A,0003	1
5	JZ 000000	5	LDI B,0002	1
6	LDI B,00	5	LDI C,0001	1
7	LDI C,05	5	LDI E,0005	1
8	ENT BC	4	ENT G,E,Trab	1
9	XOR D	4	MOV H,Trab	1
10	MOV E,A	4	DEC H	1
11	JZ F	4	JZ I	1
12	ANI A,01	5	XOR J,G,A	1
13	JMZ 000029	5	JZ F	1
14	ANI E,02	5 NE	AND L,J,C	1
15	JMZ 000025	5 NE	JMZ M	1
16	LDI B,03	5 NE	AND L,J,B	1 NE
17	SEC	4 NE	JMZ N	1 NE
18	JMP F	4 NE	SEC A,Trab	1 NE
19	LDI B,02	5 NE	JMP F	1 NE
20	SEC	4 NE	SEC B,Trab	1 NE
21	JMP F	4 NE	JMP F	1 NE
22	LDI B,01	5	SEC C,Trab	1
23	SEC	4	JMP F	1
24	JMP F	4	----------	----------

NE: Instrução não executada na simulação

Árvore balanceada: Os endereços de cada vértice crescem da esquerda para direita, servindo como referência para o programa de roteamento da tabela 5. Cada nó desta topologia, assim como na topologia hipercubo, representa um processador R2NP.

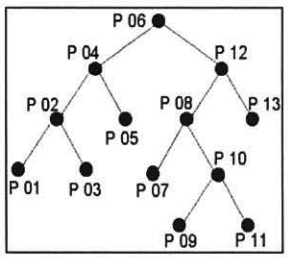

Figura 7 – Topologia árvore balanceada

Tabela 5 – *Assembly* árvore balanceada

	RCNP	Ciclos	R2NP	Ciclos
1	LDI D,06	5	LDI E,000B	5
2	PUT	4	LDI F,0017	1
3	SUI B,00	5	LDI H,0014	1
4	JZ 000025	5	LDI B,0000	1
5	LDI B,00	5	LDI D,0006	1
6	LDI C,05	5	LDI C,0005	1
7	ENT BC	4	MOV G,C	1
8	SUB D	4	ENT A,C,Trab	1
9	JMI 00001A	5	SUB I,Trab,B	1
10	LDI B,04	5	JZ H	1
11	SEC	4	SUB I,A,D	1
12	JMP 000002	5	JMI F	1
13	JZ 000025	5 NE	DEC G	1
14	LDI B,05	5 NE	SEC G,Trab	1
15	SEC	4 NE	JMP E	1
16	JMP 000002	5 NE	JZ H	1 NE
17	HLT	4 NE	SEC G,Trab	1 NE
18	----------	----------	JMP E	1 NE
19	----------	----------	HLT	1 NE

NE: Instrução não executada na simulação

4.1. Análise dos Resultados usando as Métricas

Para o programa da topologia Anel Unidirecional, os resultados são os seguintes:

Proposta RCNP

Nip = 45, Ncc = 191, Ncmi = 191 / 45 = 4,244

Tp = 191 / 500 10E6 = 0,382 μs

Proposta R2NP

Nip = 38, Ncc = 43, Ncmi = 43 / 38 = 1,131

Tp = 43 / 500 10E6 = 0,086 μs

Fp = 0,382 / 0,086 = 4,44

Podemos concluir que, para o programa da topologia anel unidirecional, o processador R2NP foi 4,44 vezes mais rápido.

Para o programa da topologia Hipercubo, os resultados são os seguintes:

Proposta RCNP

Nip = 17, Ncc = 73, Ncmi = 73 / 17 = 4,294

Tp = 73 / 500 10E6 = 0,146 μs

Proposta R2NP

Nip = 17, Ncc = 21, Ncmi = 21 / 17 = 1,235

Tp = 21 / 500 10E6 = 0,042 μs

Fp = 0,146 / 0,042 = 3,47

Neste caso o fator de performance mostrou que o processador R2NP foi 3,47 vezes mais rápido.

Para o programa da topologia Árvore Balanceada, os resultados são os seguintes:

Proposta RCNP

Nip = 12, Ncc = 56, Ncmi = 56 / 12 = 4,666

Tp = 4,666 / 500 10E6 = 0,112 μs

Proposta R2NP

Nip = 15, Ncc = 19, Ncmi = 19 / 15 = 1,266

Tp = 19 / 500 10E6 = 0,038 μs

Fp = 0,112 / 0,038 = 2,94

Para esta topologia o fator o fator de performance mostrou que o processador R2NP foi 2,94 vezes mais rápido.

Tabela 6 – Métricas de análise

	Nº médio de ciclos		Fator performance
	RCNP	R2NP	RCNP (μs) / R2NP (μs)
Anel	4,244	1,131	4,44
Hipercubo	4,294	1,235	3,47
Árvore	4,666	1,266	2,94

A diferença no fator de performance, para os três programas, é próprio da característica de projeto do processador R2NP. Como apenas instruções específicas (*load* e *store*) podem fazer acesso à memória em um processador RISC, um *loop* com acesso à memória foi facilmente evitado, o que não ocorreu com o RCNP. A linha 12 da tabela 3 mostra o acesso à memória feito pela instrução JMZ (8 vezes), que provocou um aumento da relação da performance. O segundo e terceiro programas são executados sem *loop* e nestes dois casos os fatores diminuíram, mas não deixaram de ser pertinentes. Apesar do número de instruções entre os processadores RCNP e R2NP serem muito próximos, a quantidade de ciclos do R2NP, reduzida pelo *pipeline*, favorece para um tempo menor de processamento (Tp) e também para um número médio de ciclos bem próximo a 1. No entanto estes números podem enganar. Se o programa da Árvore Balanceada não estivesse otimizada e possuísse uma instrução a mais, executada em 1 ciclo, o número médio de 1,266 ciclos iria cair para 1,250 e o fator de performance iria baixar de 2,94 para 2,80. Não adianta aproximar o número médio de ciclos para 1 se a quantidade de ciclos aumentar. O *loop* é uma das principais características de programa, quando se quer analisar performance. Em processadores CISC sem *pipeline*, como o RCNP, o número de ciclos em relação ao de instruções, aumenta consideravelmente em um *loop*, favorecendo o fator de performance em relação a um processador RISC (R2NP).

Neste artigo não é feita uma análise detalhada de conflitos, porém, algumas considerações podem ser feitas: Os conflitos estruturais são resolvidos através do caminho de dados, não ocorrendo, portanto, no R2NP. O conflito de controle é otimizado, adotando em instruções do tipo *jump* condicional, a técnica de predição de desvio. Ou seja, a princípio sempre haverá um desvio, caso contrário, existirá um ciclo de bolha. A linha 11 da tabela 3 exemplifica o ciclo a mais referente à bolha. O conflito de dados é resolvido, por exemplo, através de técnicas de

adiantamento de dados, não ocorrendo, portanto, no R2NP.

Com relação aos valores encontrados na tabela 6 podemos concluir que, para o processador RCNP ser mais rápido, sem alterar o projeto, seria necessário aumentar a freqüência de *clock*, o que poderia implicar em problemas externos ao projeto como temperatura, por exemplo. Os principais fatores de melhoria do projeto R2NP e que favoreceram a relação do fator de performance, são o formato de instrução e o caminho de dados baseado em *pipeline*.

4.2. Instruções de Acesso à Memória e às Portas de Entrada e Saída

A tabela 7 demonstra os resultados obtidos, relacionando o número total de instruções executadas com acesso à memória e às portas de entrada e saída, utilizadas para escrever os três programas.

Tabela 7 – Quantidade de instruções

	Memória		Portas de I/O	
	RCNP	R2NP	RCNP	R2NP
Anel	11	5	18	9
Hipercubo	9	7	3	2
Árvore	8	6	3	2

Os valores encontrados nesta tabela refletem diretamente no fator de performance encontrado no item 4.1. O número mais alto de acessos à memória pode prejudicar o desempenho do processador. Este detalhe fica bastante evidenciado nos resultados da topologia Anel. O dobro de instruções é utilizado, devido principalmente ao *loop*. O R2NP utiliza instruções de acesso à memória fora do *loop*, aproveitando as facilidades de otimização de código do modelo RISC. Apesar da diferença de quantidade instruções diminuir nos outros dois programas, a quantidade de ciclos foi responsável pela relevância no fator de performance.

Quanto menos acessos às portas de entrada e saída, mais liberdade o processador terá para executar outras tarefas. A análise do pacote fica mais eficiente e rápida com a utilização de menos instruções. A diferença é mínima do RCNP para o R2NP. O conjunto de instruções de rede possui boa otimização nos dois modelos.

No entanto, ao comparar os dois processadores com modelos GPP's (*General-purpose processor*), como apresentado no WSCAD'01 [11], a diferença aumenta consideravelmente. O RCNP apresentou um fator de performance, aonde o desempenho chegou a ser 7 vezes melhor em instruções de acesso à memória, comparando com um GPP CISC. Os processadores GPP's não possuem características dedicadas, que facilitariam e aumentariam o desempenho em aplicações de rede.

É importante ressaltar que os valores usados na análise de desempenho do RCNP foram obtidos usando o NPSIM [13] (simulador desenvolvido para testes funcionais). Portanto, os valores modelados analiticamente para o R2NP foram comparados com resultados de simulação em software (RCNP / NPSIM).

5. Conclusões

O objetivo deste artigo foi apresentar a evolução, características e performance do projeto de processador de rede (R2NP), que está sendo desenvolvido no Mestrado em Engenharia Elétrica da PUC Minas.

Os avanços da proposta se devem basicamente ao desenvolvimento de um modelo RISC que otimizou o conjunto de instruções, e a microarquitetura, usando novas técnicas para o caminho de dados e o acréscimo de *pipeline*.

As mudanças nos blocos lógicos com o acréscimo de unidades ASIC's (*microengines* e multiplexador) proporcionam o aumento de velocidade de processamento do R2NP. O multiplexador possibilita a utilização da mesma porta de entrada, mesmo que uma *microengine* se danifique. A *microengine*, por sua vez, acelera consideravelmente o processamento do R2NP liberando o Processador de Tarefas para os pacotes mais complicados ou que estejam em uma fila de processamento (*buffers* / memória).

Os blocos *buffers* reconfiguráveis e chave *crossbar* permaneceram basicamente inalteráveis, com exceção das instruções específicas destes blocos que estão mais otimizadas e aceleram a utilização dos mesmos.

Os resultados demonstraram coerência na evolução do projeto, evidenciando a melhoria do processador no seu novo modelo RISC. Foi constatado que o *pipeline* acelerou a utilização e execução das instruções. O acesso à memória, pelas instruções CISC (RCNP), contribuiu com o atraso para finalização dos programas, principalmente quando executadas dentro de um *loop*, aumentando o fator de performance em relação ao R2NP.

O projeto R2NP e seus resultados em relação à proposta RCNP é a principal contribuição deste artigo. Sua evolução e suas características de reconfigurabilidade (*buffers*, multiplexador e *crossbar*) são contribuições ainda não encontradas em processadores de rede comerciais.

Consultas ao sistema Lattes e aos *proceedings* do SBAC e WSCAD, revelaram que ainda não existem grupos de pesquisa envolvidos com processadores de rede no Brasil, o que ressalta a importância dos resultados apresentados e dos artigos já publicados, por nós, nos anais do WSCAD'00 [12] e WSCAD'01 [11].

Como trabalhos futuros podemos citar os seguintes:

1. Implementação de sub conjunto de instruções aritméticas de ponto flutuante e análise de impacto no desempenho do R2NP
2. Simulação do R2NP através de simulador reconfigurável [25]
3. Simulação e prototipação do R2NP em VHDL (*VHSIC Hardware Description Language*) e FPGA (*Field Programmable Gate Array*) [18]
4. Testes em um sistema de rede real [27]
5. Linguagem e compilador para o R2NP.

Estes trabalhos serão executados durante o mestrado e servirão de base para a dissertação. O simulador reconfigurável [25] é um trabalho de iniciação científica, desenvolvido no Instituto de Informática. A linguagem e o compilador são sugestões para uma dissertação ou trabalho de iniciação científica.

6. Agradecimentos

Agradecemos ao Datapuc Processamento de Dados, ao Programa de Pós-graduação em Engenharia Elétrica, aos colegas do GSDC (Grupo de Sistemas Digitais e Computacionais), pela colaboração e incentivo, que possibilitou o projeto do processador R2NP e nossa participação no WSCAD'02.

7. Referências

[1] ADC, Cuda Packet Telephony Module (PTM) Hardware Architecture, http://www.adc.com

[2] Agere System, Fast Pattern Processor (FPP) Product Brief, April 2001, http://www.agere.com

[3] Buya, R., High Performance Cluster Computing, Volume 1, Prentice Hall, 1999

[4] Cisco 7200 Series Network Processor Family, NPE-400, NPE-300 and NPE-225, Datasheet, 2000, http://www.cisco.com

[5] Cisco Systems White Paper, "The Evolution of high-end Router Architectures-Basic Scalability and Performance Considerations for Evaluating Large-Scale Router Designs", 2001, http://www.cisco.com

[6] Cisco, Route Switch Processor 8 for Cisco 7500 Series Routers, Datasheet, 2000, http://www.cisco.com

[7] C-Port, C5e Network Processor Product Brief, January 2002, http://www.motorola.com

[8] C. Ulmer, C. Wood, S. Yalamanchili, "Active SANs: Hardware Support for Integrating Computation and Communication", Workshop on Novel Uses of System Area Networks at HPCA (SAN 2002), February 2002

[9] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J. Minden, "A Survey of Active Network Research", IEEE Communications Magazine, Volume 35, Nº 1, pp.80-86, 1997

[10] EZChip Network Processors, http://www.ezchip.com

[11] H. C. Freitas, C. A. P. S. Martins, "Processador de Rede com Suporte a Multi-protocolo e Topologias Dinâmicas", II Workshop de Sistemas Computacionais de Alto Desempenho, WSCAD'2001, Pirenópolis - GO, pp.31-38

[12] H. C. Freitas, C. A. P. S. Martins, "Projeto de Processador com Microarquitetura Dedicada para Roteamento em Sistemas de Comunicação de Dados", I Workshop de Sistemas Computacionais de Alto Desempenho, WSCAD'2000, São Pedro - SP, pp.63, (Iniciação Científica)

[13] H. C. Freitas, C. A. P. S. Martins, "Simulation Tool of Network Processor for Learning Activities". Frontiers in Education Conference (FIE 2002), Boston, USA, 2002

[14] IBM PowerNP NP4GS3 Databook, http://www.ibm.com

[15] Intel WAN/LAN Access Switch Example Design for the Intel IXP 1200 Network Processor, May, 2001, http://www.intel.com

[16] Intel, "IXP 1200 - Network Processor", Datasheet, May 2000, http://www.intel.com

[17] Lexra, NetVortex Network Communications System Multiprocessor NPU, http://www.lexra.com

[18] M. Glesner, A. Kirschbaum, "State-of-the-Art in Rapid Prototyping", XI Brazilian Symposium on Integrated Circuit Design, SBCCI'98, Búzios, Rio de Janeiro, 1998, pp.60-65

[19] MMC Networks, "EPIF-105, EPIF-200, GPIF-207, XPIF-300, Packet Processors", http://www.mmcnet.com

[20] Myrinet Overview, http://www.myri.com/myrinet/overview/index.html

[21] Motorola's MPC8260 PowerQUICC II Integrated Communications Processor Family, Fact Sheet, 2001, http://www.motorola.com

[22] Patterson, D. A., J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann Publisher, 1997

[23] Sitara Networks, QoS Solutions for Enterprise and Service Provider Markets, http://www.sitaranetworks.com

[24] Sitera IQ2000, Network Processor Product Brief, http://www.sitera.com

[25] T. H. Medeiros, C. A. P. S. Martins, "Reconf_KMT, Uma Ferramenta Reconfigurável para a Simulação de Microprocessadores", III Workshop de Sistemas Computacionais de Alto Desempenho, WSCAD'2002, Vitória – ES, 2002

[26] T. Wolf and J. Turner, "Design Issues for High Performance Active Routers", International Zurich Seminar on Broadband Communications, Zurich, Switzerland, February 2000, pp. 199-205

[27] T. Wolf and M. A. Franklin, "CommBench – A Telecommunications Benchmark for Network Processors", International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, Texas, April 2000, pp. 154-162

[28] T. Wolf and M. A. Franklin, "Design of an Instruction Set for Modular Network Processors", IBM Research Report, RC21865, October 2000

[29] W. D. Mensch. Jr. and D. A. Silage, "System-on-chip Design Methodology in Engineering Education", International Conference on Engineering Education, ICEE2000 (IEEE/CS), Taipei, Taiwan, August 2000, pp. 224-228