AVALIAÇÃO DE DESEMPENHO DE UM COMPUTADOR A FLUXO DE DADOS MASSIVAMENTE PARALELO

JORGE LUIZ E SILVA
Universidade Federal de São Carlos
CCT/ Departamento de Computação
São Carlos, S.P., Brasil
(jorge@power.ufscar.br)

SHUSABURO MOTOYAMA
Universidade de Campinas
Departamento de Telemática
Campinas, S.P., Brasil
(motoyama@tisl.ukans.edu)

CLÁUDIO KIRNER
Universidade Federal de São Carlos
CCT/ Departamento de Computação
São Carlos, S.P., Brasil
(dcki@power.ufscar.br)

ABSTRACT - This paper presents a massively parallel dataflow computer based on hierarchical parallel buses, and its performance avaluation. In this computer, a dataflow program is executed by splitting and mapping program parts into computer nodes. The computer is composed of a set of parallel buses placed in a hierarchical manner that permit easy dataflow program execution and expansion flexibility. The computer has also mechanims for providing hardware and software fault tolerance so that it can work as a reliable machine. Several aspects of the computer such as: architecture, software structure, fault tolerance strategies and performance evaluation are discussed in this paper.

RESUMO - Este artigo apresenta a avaliação de desempenho de um Computador a Fluxo de Dados Massivamente Paralelo baseado em barramentos paralelos hierárquicos. Neste Computador, um programa a fluxo de dados é executado através da alocação das operações a fluxo de dados em vários elementos de processamento do computador. O computador é composto por um conjunto de barramentos paralelos dispostos de maneira hierárquica, que permite a execução de programas a fluxo de dados, é de fácil expansão e flexibilidade. O computador tem um mecanismo de tolerancia a falhas por software e hardware, tal que ele pode trabalhar como uma máquina confessável. Vários aspectos do computador tais como: arquitetura, estrutura do software, estratégias de tolerancia a falhas e avaliação da performance são discutidos neste artigo.

Palavras chaves: Paralelismo, Arquitetura, Simulação.

1- INTRODUÇÃO

As arquiteturas de computadores propostas para operar no modelo a Fluxo de Dados, teve seu inicio com Dennis [7,8,9], com o objetivo de explorar ao máximo o paralelismo existente na execução de um programa.

A partir de Dennis, outras maquinas foram propostas para operar no modelo a fluxo de dados. Gurd e Watson [10,11] propuseram a Data-Driven System for High Parallel for High Speed Parallel Computing em 1980. Por volta de 1986 ITO [13] propôs

a Inference Machine PIM-D, com uma proposta para tratar o paralelismo próximo a ideia fluxo de dados. Nesta mesma época, SHIMADA [17] propôs a Prototype Dataflow Processor of the Sigma-1 for Scientific Computations. Mais recentemente foi proposto a EM-4 Hybrid Dataflow Machine, por Sakai [15].

As maquinas a fluxo de dados portanto vem sendo desenvolvidas nos últimos 20 anos, com o objetivo de explorar o paralelismo natural de um processamento em aplicações que possuem essa característica, permitindo assim alto desempenho nas maquinas implementadas segundo o modelo a fluxo de dados.

Este trabalho descreve o mecanismo de avaliação de desempenho da arquitetura do CPER (Computador Paralelo Estruturado Recursivamente), que possui uma arquitetura baseada em barramentos paralelos hierárquicos usando processamento a fluxo de dados para a execução de programas, caracterizando o CPER como uma Maquina a Fluxo de Dados Dinâmica Massivamente Paralela (MPDC).

Essa avaliação se dá a partir de um programa simulador do hardware e da modelagem e analise de filas para alguns programas executados no simulador.

Na sessão 2 é mostrada a arquitetura do CPER. Na sessão 3 é mostrado como o CPER pode trabalhar como uma maquina a fluxo de dados dinâmica (MPDC). Na sessão 4 é mostrado o mecanismo de avaliação do desempenho da MPDC, a partir do simulador e do mecanismo de analise. Finalmente na sessão 5 é apresentado uma conclusão.

2 - ESTRUTURA DO CPER - COMPUTADOR PARALELO ESTRUTURA RECURSIVAMENTE.

A estrutura do CPER é mostrada na Figura 2.1 [3,4,5,6]. Um conjunto de N processadores, cada um chamado de Elemento de Processamento (EP), são interconectados ao barramento paralelo constituindo uma estrutura básica ou cluster. O conjunto dos barramentos destas estruturas básicas correspondem aos barramentos paralelos do nivel 1. Várias estruturas básicas do nivel 1 são interconectados para um barramento constituindo assim um cluster do nivel 2. Esta mesma idéia pode ser utilizada recursivamente para obtermos o nivel M. Os N processadores em cada cluster são tambem interconectados por uma rede em anel com características de tolerância a falhas. Uma outra rede em anel é utilizado para interconectar todos os clusters. Cada EP foi proposto para ser implementado utilizando o processador Transputer da INMOS.

O Gateway possui uma estrutura similar ao do EP com duas interfaces. Cada interface tem um mesmo conjunto de linhas e o mesmo mecanismo de acesso usado em cada EP.

Cada barramento paralelo é constituído de 97 linhas: 64 para transferir dados, 8 para overhead, 8 para endereço fonte, 8 para endereço destino e 9 para controle. Ele possui tambem 8 linhas sobressalentes para tolerância a falhas. É possível interconectar até 64 EPs em um barramento paralelo.

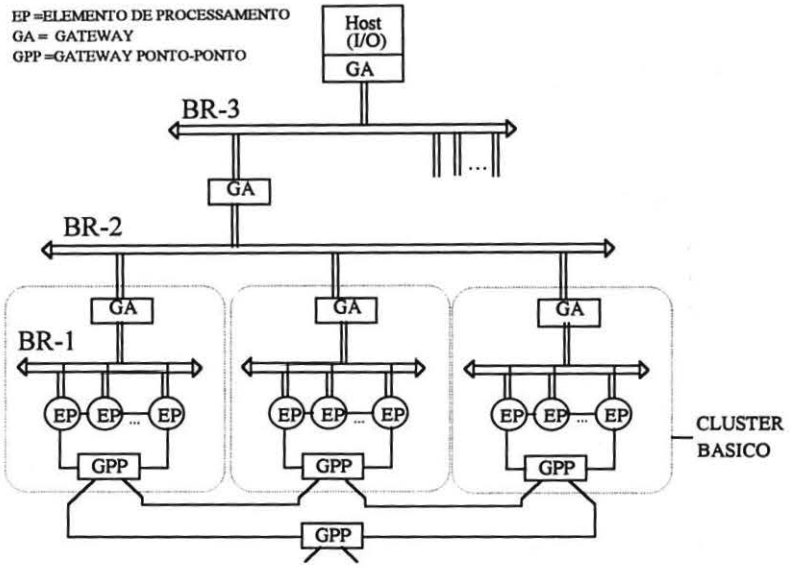


FIGURA 2.1 - ORGANIZAÇÃO BÁSICA DO CPER

O Acesso ao barramento é controlado por um sinal de prioridade que circula entre todos os EPs, em um esquema round-robin. O mecanismo de acesso é tal que: em um registrador circulante de prioridade, controlado através de um EP específico para essa atividade (EP gerenciador do barramento), é registrado o EP selecionado, em função de um conjunto de linhas que ligam os EPs ao EP gerenciador do barramento. Estas linhas são utilizadas para a requisição do barramento, e consequentimente utilizado pelo EP gerenciador do barramento na definição do registrador circulante de prioridade. Da mesma forma existe um conjunto de linhas que ligam o EP gerenciador do barramento, a cada um dos EPs para informar qual dos elementos receberá o barramento. Por exemplo, se dois ou mais EPs requerem o barramento, somente o EP de maior prioridade é habilitado pelo EP de decisão de prioridade. Um EP fica habilitado até que o barramento é reservado para ele. Tão logo ocorra a reserva, o EP é desabilitado e o sinal de prioridade é enviado para o EP com menor prioridade. Assim, neste mecanismo de acesso, a transmissão de dados e controle de acesso (circulação do sinal de prioridade) ocorrem simultaneamente, diminuindo o overhead de comunicação.

Com esse mecanismo de acesso e transferindo em paralelo: dados, cabeçalho do pacote, endereços fonte e destino, e controle, podemos diminuir o overhead na comunicação inter-processos e inter-cluster no CPER.

3 - ESTRUTURA DO SOFTWARE

O Software do CPER está dividido atualmente em duas camadas: o sub-sistema de comunicação e o gerenciador de programas fluxo de dados. No Sub-Sistema de

comunicação estão as rotinas básicas de comunicação entre os EPs em um mesmo cluster, entre os EPs em cluster diferente envolvendo então os gateways, e uma rotina de comunicação serial entre o Host e um dos gateways do sistema. No gerenciador de programas a fluxo de dados estão todas as rotinas básicas que permitem a execução de uma programa a fluxo de dados, distribuído entre os EPs.

3.1 - EXECUTANDO PROGRAMAS A FLUXO DE DADOS NO CPER

Na MPDC, como vimos, configurada através de um software que roda sobre o CPER, cada dado apresenta um tagged-token associado, e as operações dos programas a fluxo de dados são disparadas por dados que tenham o mesmo tag associado [1,2,3,4,5,6].

Esses tags são gerados em cada entrada de um programa, função ou procedimento, representando uma ativação de dados. No caso de existirem laços, implementados com operações iterativas WHILE, REPEAT, FOR, a entrada em cada uma dessas operações também geram um novo tag que será associado aos parâmetros de entrada com os tags antigos. Para cada ciclo do loop, o novo tag deverá ser ajustado para indicar a iteração. Como as operações iterativas podem ser aninhadas é útil que o tag tambem contenha o nivel de aninhamento. Ao término de cada operação iterativa, função, procedimento ou programa, a parcela de tag correspondente é destruído.

Antes da execução de um programa a fluxo de dados na MPDC, é necessário que suas operações sejam alocadas nos EPs por um carregador instalado no hospedeiro. O carregador deve alocar as operações nos EPs da MPDC de tal forma a se obter o melhor desempenho da maquina.

Durante a execução, cada EP recebendo dados de entrada necessários com o mesmo tag associado, executará sua operação e remeterá o resultado para outro EP. Esta troca de informação ocorrerá até que o resultado final de cada ativação do programa seja produzido.

3.2 - ESTRUTURA DE FUNCIONAMENTO DOS OPERADORES

A maioria dos operadores mais comuns usados em grafos a fluxo de dados foram mapeados em um operador genérico com 3 entradas e 3 saídas (Figura 3.2a).

A implementação deste operador genérico pode ser feita utilizando alocação dinâmica de memória como mostrado na Figura 3.2b. A parte principal da memória possui campos indicando: tipo da operação, numero de entradas, numero de saídas, numero de ativações, endereço do resultado (3 lugares), e endereço da primeira ativação. As outras partes da memória não são fixadas e podem ser expandidas de acordo com o numero de ativações e iterações. O gabarito de memória da ativação possui campos indicando endereços do elemento anterior, numero de iterações, ativação X, endereço dos tags antigos, endereço da próxima ativação e endereço da próxima iteração. O gabarito de memória da iteração possui os seguintes campos: endereço do elemento anterior, iteração X, estado do dado, dados (3 lugares) e endereço da próxima iteração. O Campo estado do dado indica se todos os dados estão presentes para execução.

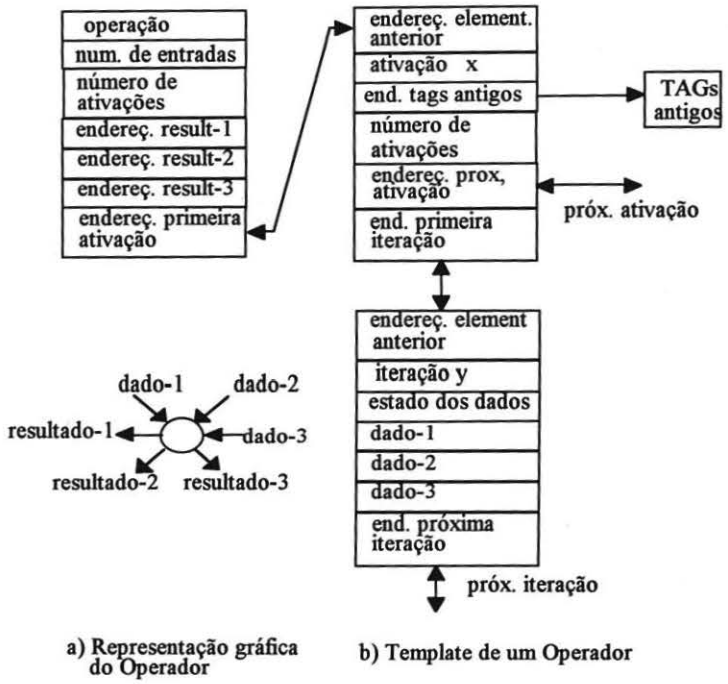


FIGURA 3.2 ESTRUTURA GENÉRICA DE UM OPERADOR

Uma operação simples, quando recebe um dado de uma ativação, gera uma locação na memória correspondente quela ativação e também gera uma locação na memória correspondente a iteração inicial, ficando com o gabarito de memória igual ao da Figura 3.2b. No caso de operações em loop, vão sendo geradas outras locações de memória correspondentes a novas iterações. Quando chegam todos os dados de uma operação, ela é executada, e a locação de memória para aquela iteração é eliminada; se não restar nenhuma locação de memória correspondente a alguma iteração, ligada a memória correspondente aquela ativação, então esta locação de memória também é eliminada.

Desde que a MPDC utiliza tagged-tokens, podem existir várias ativações e várias iterações para cada operação. Neste caso, as memórias são ligadas em uma lista encadeada com os endereços principais indicando as sequências de ativações e iterações conforme mostrado na Figura 3.2b.

A localização de um dos vários conjuntos de dados desta lista encadeada é realizada pela pesquisa nas ativações e em seguida pela pesquisa nas iterações. O desempenho na execução de operações na MPDC depende da existência de um

mecanismo de busca eficiente. Um sistema matching store foi proposto para otimizar esse processo de busca através de dispositivos de alta velocidade [6], obtendo um sucesso em aproximadamente 70ns.

4 - AVALIAÇÃO DE DESEMPENHO DA MPDC

A avaliação de desempenho da MPDC foi desenvolvido utilizando análise de filas, onde os modelos foram elaborados para analisar a execução a fluxo de dados, mas principalmente modelar a operação do barramento de dados da MPDC, que é o elemento de gargalo do sistema.

A entrada de dados para as redes de filas de todos os programas analisados, foi considerada a partir de uma taxa que representa 30% da capacidade máxima do barramento, e em função do número de filas que recebem dados externos, foi feita uma divisão proporcional (Pi) da taxa para cada fila.

O modelo básico analisado foi o modelo de Rede de Filas que elabora um mecanismo de acesso em forma de anel com slots no barramento [16]. Nesse modelo a mensagem é dividida em vários pacotes e a transmissão é feita por pacotes utilizando o mecanismo de aceso em anel por slots. A expressão utilizada para a obtenção do tempo total de transmissão de todos os slots é a descrita a seguir:

$$wp = \left\{ \frac{\lambda\ G\ Tu\ (N+1)\ +\ 2\ G\ (N+1) - 1}{2\ [\ 1\ -\ \lambda\ G\ Tu\ (N+1)\ ]} \right\} Tu$$

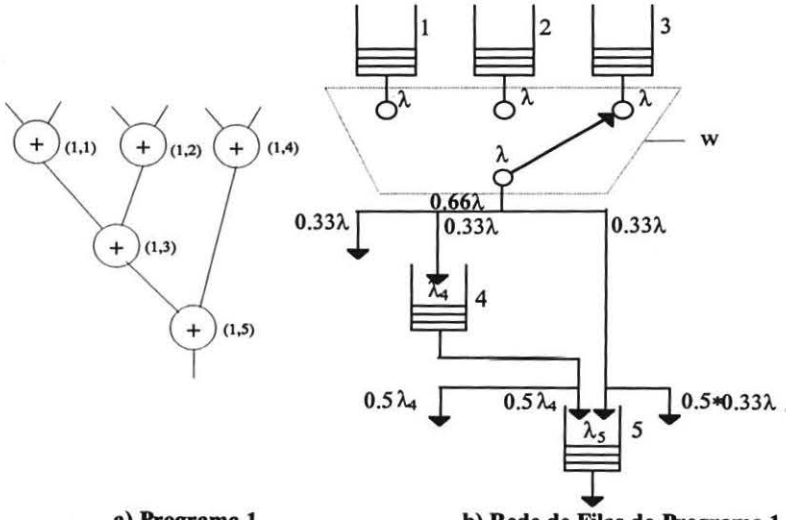
Onde:

N o número de processadores;
G o número médio de pacotes por mensagem (10 na MPDC)
Tu o tempo de transmissão do último pacote (61 ns)

Na Figura 4.1 e 4.2 é descrito o modelo de Rede de Filas com mecanismos de acesso em anel com slots, para dois programas a fluxo de dados diferentes.

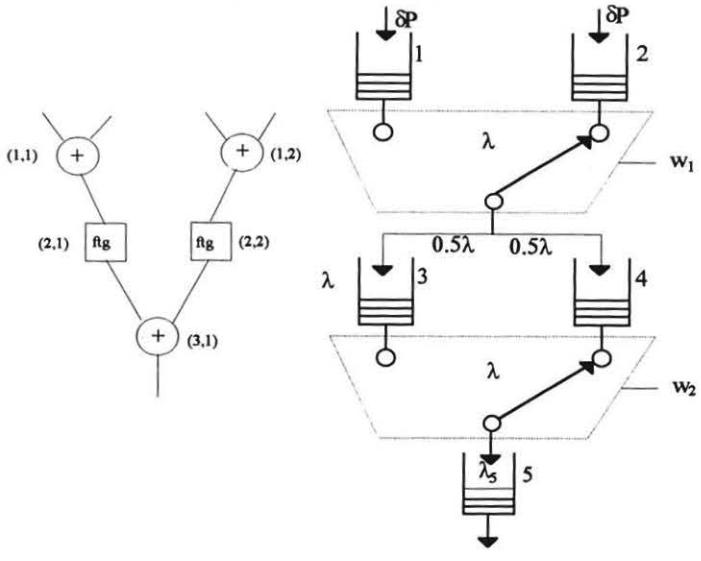
Na Figura 4.1 é analisado um programa fluxo de dados simples, onde as operações estão distribuídas em um único cluster. Já na Figura 4.2 é analisado um programa também simples, exceto que algumas operações estão distribuídas em cluster diferentes, sendo necessário o uso das operações gateway (ftg).

Foi desenvolvido um simulador da MPDC que permite a edição, analise, e execução de programas a fluxo de dados. Os objetivos do simulador foram: validar as estruturas da maquinas proposta, estudar o desempenho da máquina e criar um ambiente de desenvolvimento do software básico da MPDC [4].



a) Programa 1 **b) Rede de Filas do Programa 1**

FIGURA 4.1 - REDE DE FILAS COM MECANISMO DE ACESSO EM ANEL POR SLOTS DO PROGRAMA 1



a) Programa 2 **b) Rede de Filas do Programa 2**

FIGURA 4.2 - REDE DE FILAS COM MECANISMO DE ACESSO EM ANEL POR SLOTS DO PROGRAMA 2

O Simulador, chamado FDsim - Fluxo de Dados Simulador, foi implementado em PASCAL e possui os seguintes módulos: o Módulo Editor para edição dos programas a fluxo de dados; o Módulo Analisa para a execução de todas as operações a fluxo de dados; o Módulo Relógio para a sincronização e gerenciamento dos eventos durante a execução dos programas; o Módulo Recepção para o matching store de memória das operações; o Módulo Execução para a execução das operações a fluxo de dados; o Módulo Barramento que simula o mecanismo de acesso; e o Módulo Tolerância a Falhas para a implementação do protocolo de Tolerância a Falhas.

O FDsim tem uma estrutura básica baseada em arrays. Um array representando o hardware lógico da MPDC cujas linhas e colunas indicam os clusters e EPs, respectivamente. Cada elemento deste array descreve a estrutura de um EP (buffer de entrada, buffer de saída, ponteiros de buffers e ponteiros de listas encadeadas). Um outro array representando o barramento lógico cujas linhas e colunas tambem indicam clusters e EPs, respectivamente. Cada elemento deste array descreve o sinal de prioridade e a indicação de qual EP espera para transmitir. Existem tambem um array de eventos onde cada elemento descreve o estado de cada EP. Este estado pode indicar três diferentes situações: recepção, execução e transmissão.

A TABELA 1 mostra os resultados da simulação para alguns programas a fluxo de dados selecionados usando o FDsim.

A tabela leva em consideração os seguintes parâmetros e símbolos:

- Taxa de Transmissão para pacotes de 64 bits 61ns
- Tempo médio de execução para uma operação a fluxo de dados 500ns
- Tempo de transferência pelo gateway 250ns
- Tempo de busca na matching store 66ns
- EXEC. = Tempo de execução: SEQ (sequencial), PARAL (paralelo)
- P = Numero de EPs
- G = Numero de gateways- C = Numero de Clusters
- PIPEi = Tempo de execução usando i dados em pipeline (ns)
- Sp = Speed-up
- Ep% = Eficiência (%)

TABELA 1 - Performance de dados relacionados com a execução de programas simples

EXEC	P	G	C	PIPE1	SEQ	Sp	Ep%	PIPE3	SEQ	Sp	EP %	PIPE5	SEQ	Sp	Ep%
PARAL	11	0	1	4240	5500	1.30	11.8	6360	16500	2.59	23.6	8480	27500	3.24	29.5
PARAL	15	0	1	5300	7500	1.42	9.4	7420	22500	3.03	20.2	9540	37500	3.93	26.2
PARAL	23	2	3	9975	11500	1.15	5.0	13965	34500	2.47	10.7	15960	57500	3.60	15.7
PARAL	27	2	3	11970	13500	1.13	4.2	15960	40500	3.54	9.4	19950	67500	3.38	12.5

Na tabela podemos observar a simulação de alguns programas considerando dados em pipeline, granularidade fina para a distribuição das operações a fluxo de dados em EP's num mesmo cluster e EP's em cluster diferentes, e comparação dos resultados com a execução de forma sequencial.

Na Figura 4.3 podemos verificar a plotagem do speed-up e eficiência relacionado com a execução de diferentes programas na MPDC.

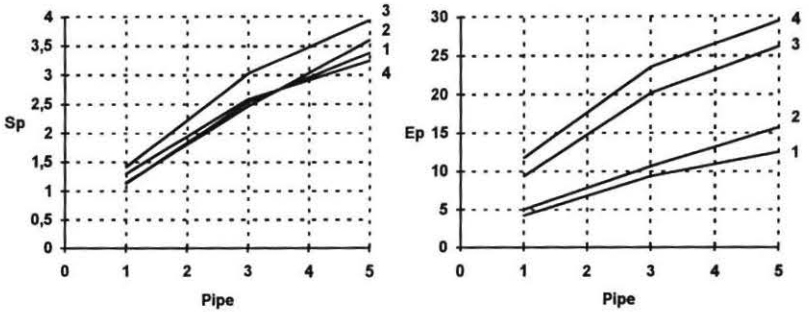


FIGURA 4.3 - SPEED-UP E EFICIÊNCIA PARA ALGUNS PROGRAMAS

Conforme a Figura 4.3 podemos observar que para programas alocados em um único cluster e com uso máximo da capacidade em operações para aquele cluster a eficiência é da ordem de 30% (curva 3 e curva 4). Para programas com grande número de operações, necessitando do uso de gateways e portanto distribuídos em clustes diferentes, existe um overhead imposto pela comunicação mas medida que temos mais operações e mais níveis de pipeline, a eficiência vai aparecendo com maior intensidade (curvas 1 e 2).

5 - CONCLUSÃO

Este trabalho descreveu o Computador a Fluxo de Dados Massivamente Paralelo (MPDC) e sua avaliação de desempenho. A arquitetura básica, com uma estrutura hierárquica composto por clusters interligados por barramentos de alta velocidade, e as características de tolerância a falhas tanto a nível de hardware como a nível de software permitem alta flexibilidade e confiabilidade do sistema. Foi apresentado também o mecanismo de execução de operações a fluxo de dados na máquina. A partir de um simulador implementado especificamente para a avaliação desse sistema e da análise de filas, foi apresentado alguns dados de desempenho do computador. Finalmente a performance analisada demonstrou que o sistema tem grande potencial medida que se aumenta o número de operações e o número de níveis pipeline de dados.

REFERÊNCIAS

[1] A.H. Veen, "Dataflow Machine ARchitecture", *ACM Computing Surveys*, Vol. 18, No. 4, Dec. 1986, pp. 365-396.

[2] Arvind and R.S. Kikhil, "Executing a Program on the MIT Tagged-Token Dataflow Architecture", *IEEE Transaction on Computer*, Vol. 39, No. 3, 1990, pp. 300-318.

[3] C. Kirner, "Design of a Recursively Structured Parallel Computer", *Prod. of the 17th Annual Computer Science Conference, ACM*, Louisville, USA, Feb. 1989, 1 pp.

[4] C. Kirner, J.L. Silva, and S. Motoyama, "A Fault-Tolerant Massively Parallel Dataflow Computer", *VI IASTED International Conference on Parallel and Distributed Computign Systems*, Washington DC, USA, Oct. 3-5, 1994, pp.323-326

[5] C. Kirner, E. Marques, "Design of Distributed System Suport Based in a Centralized Parallel Bus". *Computer Architecture News*, V.14, No.4, Sep. 1986, pp. 15-26.

[6] E. Marques and C. Kirner, "Design of the Matching Unit of a Massively Parallel Dataflow Computing System", *Conference on Massively Parallel Computing System, IEEE/EUROMICRO*, Ischia, Italy, May 2-6, 1994, 19pp.

[7] J.B. Dennis, "First Version of a Data Flow Produce Language" . *Lecture Notes in Computer Science*. Vol. 19, 1974, pp. 362-372.

[8] J.B. Dennis, & D.P. Misumas, " A Preliminary Architecture for a Basic Flow Processor". In: *Annual Symp. Computer Architecture, 2. Proceedings*, 1975, pp, 126-132.

[9] J.B. Dennis, "Data Flow Supercomputer". *IEEE Computer*, Vol. 13, No. 11, Nov. 1980, pp. 48-56.

[10] J. Gurd. & I. Watson, " Structuring software for Parallel Execution I. In: Data-Driven Systems for High Speed Parallel Computing". *Computer Design*, Vol. 19, No. 6, Jun. 1980, pp. 91- 100.

[11] J. Gurd, & I Watson, "Hardware Design. II. In: Data-Driven System for High Parallel Computing". *Computer Design*, Vol. 7, Jul. 1980, pp. 97-106.

[12] M.G. Sami and N. Scarabotolo, "Fault Tolerance in Parallel Architecture", *Lecture Notes in Computer Science*, No. 272, Future Parallel Computer, Springer-Verlag, 1987, pp.82-152.

[13] N. Ito, et. al. "The Architecture and Preliminary Evaluation Results of the Experimental Parallel Inference Machine PIM-D". In *Proc. of the 13th Annual International Symposium on Computer Architecture*,Tokyo, Japan, Jul. 1986, pp. 149-156.

[14] P.C. Trelevean, D.R. Brownbridge and R.P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture", *ACM Computing surveys*, Vol. 14, No. 1, Marc. 1982, pp. 93-143.

[15] S. Sakai, Y. Kodama, and Y. Yamaguchi, "Prototype Implementation of a Highly Parallel Dataflow Machine EM-4". *Proc. Int. Parallel Processing Symposium, 1991.*

[16] H. Takagi, "Analysis of Polling Systems", *London: MIT Press*, 1985, 197p.

[17] T. Shimada, K. Hiraki, K. Nishida and S. Sekigushi, "Evaluation of a Prototype Dataflow Procesor of the Sigma-1 for Scientific Computations", *Computer Architecture News*, Vol. 15, No. 2, 1986, pp.226-234.

[18] V.P. Srini, "A Fault-Tolerant Dataflow System". *IEEE Computer*, Vol. 18, No. 11, Mar. 1985, pp.54-68.

[19] V.P. Srini, Än Architectureal Comparation Of Dataflow Systems", *Computer*, Vol. 19, No. 3, 1986, pp. 68-88.