UM BARRAMENTO ESPECIAL PARA SINCRONIZAÇÃO DE BARREIRA EM MÁQUINAS DE MEMÓRIA COMPARTILHADA

Martha X.T.Delgado, Edward D.M.Ordoñez, Celso A.S.Santos, Sergio T.Kofuji [1]

Laboratório de Sistemas Integráveis
Departamento de Engenharia Eletrônica
Escola Politécnica da Universidade de São Paulo

RESUMO

Para poder realizar uma sincronização de barrreira de alta velocidade é necessário suportar sua execução por meio de hardware, para produzir uma sobrecarga de tempo mínima. O objetivo deste trabalho é desenvolver uma solução de sincronização de barreira ao nível de hardware para uma máquina de memória compartilhada que seja econômica, eficiente e que melhore as características das soluções existentes. Este trabalho apresenta o projeto de um barramento especial para realizar a sincronização de barreira. Descreve-se seu princípio de funcionamento, sua implementação, e uma comparação com as soluções existentes. Também é desenvolvido um modelo analítico para obter quantitativamente o desempenho da solução proposta.

ABSTRACT

In order to carry out a high velocity barrier synchronization that produces a minimal overhead time it is need a hardware implementation. The goal of the present work is to develop a barrier synchronization solution on hardware for shared memory parallel machine with economics, efficient and flexible characteristics when compared to traditional solutions. The paper describes the design of a special bus for barrier synchronization including their basic principles and implementation, and a comparison with several traditional solutions. Also, it is developed an analytical model for compute the general performance.

[1] Pesquisadores do Laboratório de Sistemas Integráveis (LSI/EPUSP)
E-mail: (mxtd, edmoreno, saibel, kofuji) @lsi.usp.br

1. INTRODUÇÃO

A Sincronização de Barreira estabelece um ponto lógico (chamado barreira) onde um conjunto de tarefas dentro de um programa paralelo deve chegar, antes de permitir a execução de outras tarefas [1].

A sincronização de barreira é uma das formas mais críticas da sincronização porque obriga a comunicação de um processo com cada um dos outros processos, além de precisar que todos os processos esperem na barreira até que o último chegue. Sua grande utilização e sua potencial influência negativa no desempenho fazem da barreira um importante alvo de pesquisa [1]. Por exemplo, em [2] foi feita uma análise que mostra como a sobrecarga de tempo gerada pela sincronização de barreira influi negativamente no tempo de execução de um laço paralelo. Consequentemente desenvolver um mecanismo eficiente de sincronização de barreira contribui para aumentar a velocidade de execução dos algoritmos em máquinas paralelas. Muitas pesquisas têm sido realizadas com o objetivo de fazer implementações eficientes tanto ao nível de software [3] como de hardware [4][5].

Existem muitas soluções de hardware para máquinas de memória compartilhada mas poucas são flexivéis o suficeinte para permitir que qualquer número de processadores possam ser sincronizados, para realizar sincronizações de barreira simultaneamente, e realizar sincronizações de barreiras diferentes consecutivas. As soluções em software são muito mais flexíveis do que as soluções em hardware, mas as primeiras são menos eficientes principalmente porque na sua execução utilizam o mesmo elemento de interconexão que é usado para transmissão de dados. Portanto, uma solução em hardware que reúna a maioria das características anteriormente mencionadas é mais desejável.

O presente trabalho apresenta uma solução em hardware para sincronização de barreira em máquinas de memória compartilhada que pretende ser econômica, eficiente e que melhore as características das soluções existentes. Também apresenta o desempenho da solução obtido mediante um modelo analítico e uma comparação com as soluções existentes. Na seção 2 descreve-se o seu princípio de funcionamento, e suas características. Na seção 3 apresenta-se a implementação da solução de maneira detalhada, ne seção seguinte mostra-se o modelo analítico, a análise dos resultados e finalmente na seção 5 as conclusões e os trabalhos futuros.

2. DESCRIÇÃO BÁSICA DA SOLUÇÃO

Este projeto é feito com o objetivo de implementar um hardware de sincronização de barreira flexível, econômico, e eficiente. Escolhe-se implementar a solução sobre um barramento especial porque se pode aproveitar suas propriedades de "difusão" para obter uma solução flexível, além de ser uma opção ecônomica. Esta solução é projetada para realizar sincronização de barreira exclusivamente em máquinas paralelas de memória compartilhada. Por ser baseada em barramento, não é uma solução escalável.

Permite fazer a sincronização de qualquer número de processadores, sincronizar concorrentemente várias barreiras, além de suportar a execução de barreiras diferentes consecutivas. Pode ser usada tanto para sincronizar comandos como para sincronizar laços.

Esta solução, mostrada na figura 1, consiste em usar um barramento especial de sincronização e um bloco de hardware (b_i) adicional para cada processador (P_i). No momento em que um processador chega na barreira emite através do barramento o nome de sua barreira. Os demais processadores emitem uma resposta e o bloco de hardware realiza a sincronização de barreira.

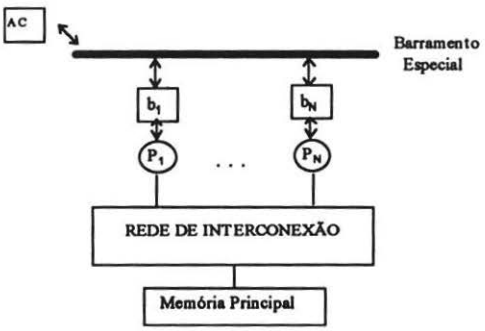


Figura 1. Diagrama do barramento especial de sincronização de barreira.

2.1. Princípio de Funcionamento

O barramento de sincronização é usado para realizar a comunicação entre os blocos de hardware, emite-se para ele o nome da barreira do processador e um sinal chamado de SB (sinronização de barreira), o qual é usado para emitir a resposta de sincronização de barreira (este é um fio "wired-or"). Há dois registradores de sincronização importantes: NB, onde está o nome da barreira na qual o processador sincroniza-se e CH1, que é um registrador de 1 bit no qual o processador avisa no momento que chega na barreira. O bloco de hardware monitora os sinais no barramento e realiza as ações correspondentes. O processador não participa dessas decisões.

No momento que o processador (P_i) chega à barreira, o bloco de hardware (b_i) acessa o barramento (existe um protocolo de arbitração para dar acesso ao barramento a um bloco de hardware por vez, neste caso usa-se um arbitro central (AC na figura 1)), emite o nome de sua barreira, e coloca um 0 na linha sincronização de barreira (SB). Ao mesmo tempo os demais blocos de hardware comparam o nome da barreira no barramento com o valor que está em seu registrador NB. Se os dois valores são iguais e o processador já tiver chegado, o bloco de hardware é encarregado de colocar um 0 na linha SB. Se são iguais e o processador ainda não chegou, o bloco de hardware coloca um 1 na linha SB. E, por fim, se são diferentes o bloco de hardware coloca um 0 na linha SB. Depois de um tempo, cada bloco de hardware lê o estado da linha SB, se a linha está em 0 então todos os processadores que se estão sincronizando nessa barreira sabem que a sincronização foi feita; se está em 1 significa que ao menos um processador não chegou ainda na barreira. Então o bloco de hardware tentará de novo pegar o barramento.

2.2. Características

Este esquema permite realizar sincronização de barreira de qualquer número de processadores. O processador deve escrever o nome da barreira na qual ele vai participar, antes de começar a execução de uma tarefa paralela que precise sincronização de barreira. Cada barreira diferencia-se da outra através de um número. Esta informação deve ser fornecida pelo compilador.

Barreiras Concorrentes:

Como foi dito anteriormente, o barramento apresenta um protocolo de arbitração que não tem preferência para dar acesso a um processador que tenha uma determinada barreira. Assim, cada processador que chega em sua barreira tem uma oportunidade para acessar o barramento e desta forma o sistema pode tolerar várias sincronizações de barreira ao mesmo tempo. A concorrência das barreiras

está limitada pelo fato de usar as mesmas linhas do barramento para todos os processadores. Esta solução foi adotada para que o projeto seja mais econômico e escalável no espaço.

Barreiras diferentes consecutivas:

Em alguns programas apresenta-se situações nas quais é necessário usar a sincronização de barreira consecutivamente [6]. Isto é mostrado na figura 2. Neste caso o número atribuído para cada barreira consecutiva deve seguir a ordem em que se devem ser executadas. Na figura 2 a sincronização de barreira 2 só é possível depois que for feita a sincronização de barreira 1. Supondo que vai se usar o barramento especial para realizar estas sincronizações de barreira, e que o processador P_3 chega na barreira 2 antes de que P_1 começe a executar T4, P_3 emite o nome de sua barreira no barramento e a resposta que vai obter é que todos os processadores chegaram na barreira (porque tanto P_1 como P_2 vão por um 0 na linha SB). Esta resposta falsa vai gerar erros na execução do programa.

Para que o barramento especial realize satisfatoriamente este tipo de sincronização, é necessário diferenciar uma barreira consecutiva de uma barreira normal. Quando se tem uma barreira consecutiva o controlador de sincronização de barreira deve esperar até que todas as barreiras anteriores tenham sido executadas. Isto é feito adicionando-se um comparador que observa as barreiras anteriores, e um contador o que é habilitado quando se sincronizam cada uma delas. Para diferenciar uma barreira consecutiva de uma normal deve-se pôr um bit adicional ao nome da barreira: bit 1 para barreira normal e bit 0 para barreira consecutiva.

Com esta implementação pode-se fazer a sincronização de barreira consecutiva mostrada na figura 2. Neste caso P_3 apresenta uma barreira consecutiva chamada barreira 2 e espera até que seja realizada a sincronização da barreira 1, para começar sua sincronização.

O barramento especial precisa de $\log_2(P-1)$ linhas para pôr o nome da barreira; com este número podem ser suportadas P - 1 barreiras simultâneas (P é o número de processadores). Também precisa de 3 linhas de controle (para o protocolo de arbitramento) e a linha SB. Em total o número de linhas do barramento especial de sincronização de barreira é $\log_2(P-1) + 4$.

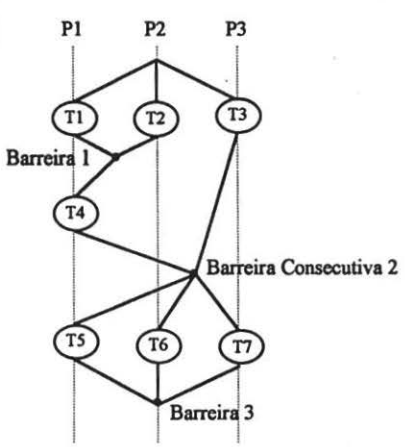


Figura 2. Diagrama de fluxo de um programa

3. IMPLEMENTAÇÃO

A visão geral do barramento especial de sincronização é mostrado na figura 1. O protocolo de arbitramento utilizado no barramento é *"round robin"*. Usa-se um arbitro central (AC na figura), o qual gera um sinal de relógio que é transmitido para todos os blocos de hardware mediante uma linha do barramento chamada "relógio". Em cada bloco há um contador que é acrescentado em 1 em cada borda de subida deste sinal. Um comparador analisa o valor deste contador e o valor ID (o número do processador associado). Os dois valores iguais indicam que é a vez do bloco de hardware acessar o barramento, mas este acesso é feito somente se sua barreira correspondente chegou. Neste caso o bloco deve parar o relógio para evitar que outro processador pegue o barramento então emitindo um 1 na linha do barramento chamada "parada". O arbitro central por sua vez, "limpa" (põe em 0) o sinal "relógio".

Detalhadamente, o bloco de hardware está dividido em 5 sub-blocos, como é mostrado na figura 3. O sub-bloco de inicialização, o sub-bloco de arbitramento, o sub-bloco de emissão, o sub-bloco de recepção e o sub-bloco de emissão/recepção da resposta da barreira. Todos os sub- blocos comunicam-se para realizar a sincronização de barreira. O sub-bloco de inicialização armazena o nome da barreira e a chegada dela (este dados são emitidos pelo processador). O sub-bloco de arbitramento define o controle sobre o barramento para seu bloco de hardware. O sub-bloco de emissão é encarregado de emitir o nome da barreira (no caso que o bloco tenha o controle do barramento). O sub-bloco de recepção é encarregado de receber o nome da barreira que se encontra no barramento. O sub-bloco de emissão/recepção da resposta da barreira é encarregado de emitir seu estado com respeito à sincronização de barreira e de receber a resposta total da sincronização de barreira.

Implementação do sub-bloco de inicialização:

O bloco de hardware deve receber dois dados do processador associado. O nome da barreira na qual o processador vai participar e o momento de chegada nela. O sub-bloco de inicialização é encarregado de armazenar esses dados. Como se mostra na figura 4, o registrador NB (nome da barreira) armazena o nome da barreira no momento que o processador emite uma escrita para NB (inicialmente o registrador está em 0), o registrador CH1(inicialmente está em 0) armazena a chegada da barreira.

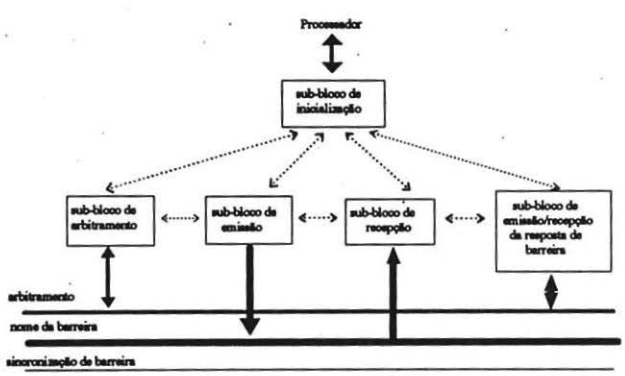


Figura 3. Diagrama de blocos do hardware de sincronização de barreira

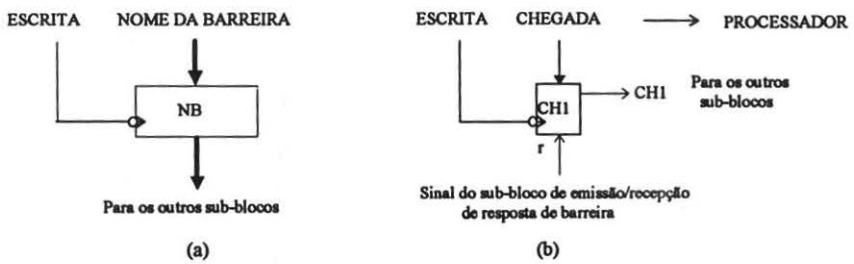


Figura 4. Implementação do sub-bloco de inicialização

Implementação do sub-bloco de arbitramento:

Em cada bloco de hardware há um sub-bloco de arbitração (figura 5(a)). Como citado anteriormente o contador é acrescentado em 1 em cada borda de subida do sinal "relógio". O comparador analisa o valor deste contador e o valor ID (o número do processador associado). Se os dois valores forém iguais, significa que é a vez do bloco de hardware acessar o barramento, porém, este acesso somente é feito se sua barreira correspondente chegou. Sabe-se que chegou, se o sinal CH1 (sinal que vem do sub-bloco de inicialização) está em 1. Neste caso o sinal P será 1, indicando que o bloco tem o controle sobre o barramento. No caso que CH1 seja 0 e/ou C1 seja 0, o sinal P será 0 e o bloco não acessará o barramento.

A lógica de acesso ao barramento é apresentada na figura 5(b). CH1 e C1 são lidos na borda de descida do sinal "relógio" (para efeitos de sincronização); o sinal CH1 pode mudar de valor asíncronicamente portanto deve-se ter cuidado com o problema de metaestabilidade. Os registradores C1 e CH2 são incializados em 0. P é 1 somente se CH2 e C1 são 1.

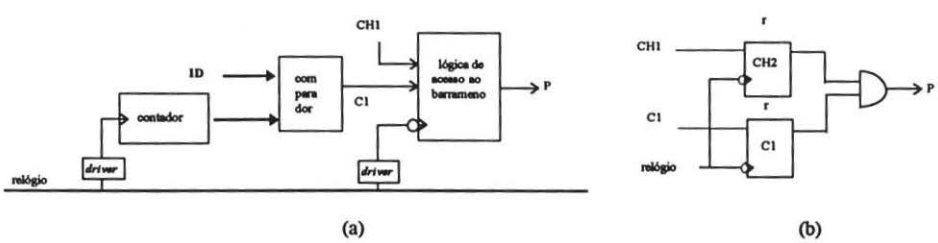


Figura 5. Implementação do sub-bloco de arbitramento(a). Implementação da lógica de acesso ao barramento(b).

Após o bloco de hardware ter obtido o controle do barramento (P = 1), deve emitir o nome de sua barreira no barramento, colocar um 1 na linha "parada" e começar o protocolo de comunicação com os demais blocos. Nesta implementação o protocolo de comunicação usado é síncrono. O bloco de hardware gera um sinal de sincronização "relógio$_1$" o qual é transmitido para os demais blocos. Este sinal é mantido até que o bloco termine o acesso ao barramento. Quando a linha "relógio$_1$" passa para 0, a linha "parada" passa para 0 permitindo que o sinal relógio seja gerado novamente, passando o controle para o processador seguinte. Este protocolo garante que todos os processadores realizem suas ações antes de passar o controle do barramento para outro processador.

Implementação do sub-bloco de emissão:

O sub-bloco de emissão é encarregado de emitir o nome de sua barreira no barramento, levar para um 1 a linha "parada" e de gerar o sinal "relógio$_i$". Na figura 6 mostra-se a implementação deste sub-bloco.

A figura 6(a) mostra como é emitido o sinal "parada" ao barramento. Usam-se duas unidades de atraso ("*delay*"): A_1 e A_2. A unidade de atraso A1 é programada para um tempo T_1 e a unidade de atraso A2 para um tempo T_1+T_2 (estes tempos serão explicados em detalhe no decprrer deste trabalho). A saída P_i que gera o sinal "parada" obedece ao diagrama de tempo mostrado na figura 7(a), e a tabela verdade que corresponde à lógica combinacional da figura 6(a) é mostrada na figura 7(b). A figura 6(b) mostra a implementação do sinal "relógio$_i$" que vai para o barramento. A saída r_i que gera o sinal "relógio$_i$" obedece ao diagrama de tempo mostrado na figura 7, e a tabela verdade que corresponde à lógica combinacional da figura 6(a) é mostrada na figura 7(b). Finalmente, a figura 6(c) mostra que o sinal P_i habilita a saída do nome da barreira (NB) no barramento. Esta saída fica ativa até que P_i seja 0.

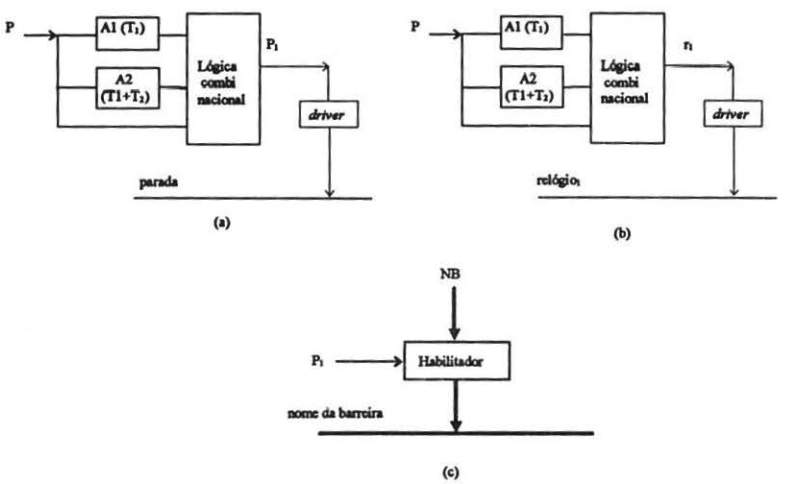


Figura 6. Implementação do sub-bloco de emissão

O período do sinal "relógio" deve ser tal que permita que o sinal "parada" pare o relógio do arbitro central antes de iniciar outro ciclo de relógio, evitando que mais de um bloco acesse o barramento ao mesmo tempo.

O período mínimo do sinal relógio T é:

$$T = \mathbf{T} + T_{\text{sinal de relógio}} + \Delta_{\text{lógica de acesso ao barramento}} + \Delta_{\text{lógica de emissão fig. 6(a)}} + \mathbf{T}_{\text{parada}} + \Delta_{\text{atraso no AC}} \quad (1)$$

$\mathbf{T}$: tempo de transmissão para todo o barramento.

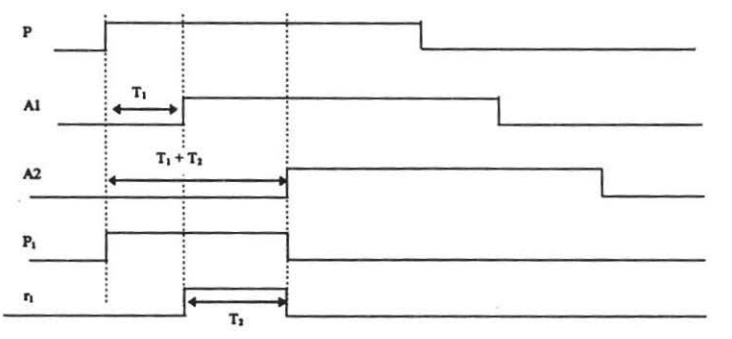
$T_{\text{sinal de relógio}}$: tempo que o sinal de relógio fica em 1

$\Delta_{\text{sub-bloco}i}$: tempo de atraso do sub-bloco$_i$.

O primeiro $\mathbf{T}$, na expressão (1), corresponde à emissão do sinal relógio a todos os blocos (incluindo o último). O segundo $\mathbf{T}$ corresponde ao tempo que demora o arbitro central em receber o sinal de parada (supondo que o sinal vem do último bloco e o arbitro está ao lado do primeiro bloco, que é o caso crítico). O tempo T deve ser maior que o atraso do contador , do comparador e do tempo "*setup*" do registrador C1(da figura 5(b)) para garantir sincronização na resposta.

Implementação do sub-bloco de recepção:

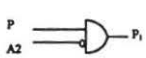
Quando um bloco de hardware emite o nome de sua barreira, todos os blocos devem receber este valor. O sub-bloco de recepção é encarregado de receber o nome da barreira e verificar se ele coincide com o nome de barreira interno, como mostrado na figura 8. A borda de subida do sinal "relógio$_1$" permite armazenar o valor emitido pelo comparador no registrador C2 de 1 bit. Para garantir que na borda de subida do sinal "relógio$_1$" o valor no comparador seja correto e estável, é necessário que o nome de barreira esteja estável. Para assegurar isto, o bloco de hardware que tem o controle do barramento envia o nome da barreira, T_1 unidades de tempo antes de enviar o sinal "relógio$_1$". T_1 é a soma dos tempos de atraso do "driver" e do comparador (correspondentes à figura 8).



(a)

P	A1	A2	P1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(b)



P	A1	A2	r1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(c)

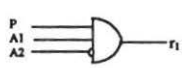


Figura 7. Diagrama de tempo (a). Tabelas de verdade dos sinais P_1 (b) e r_1 (c).

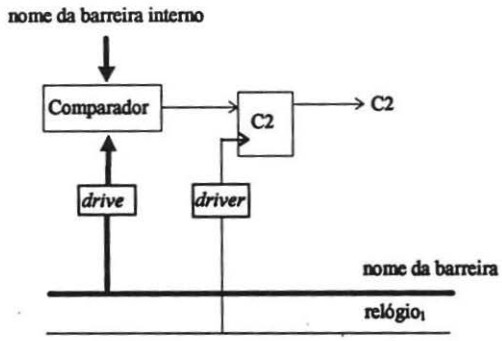


Figura 8. Implementação do sub-bloco de recepção

Implementação do sub-bloco de emissão/recepção de sincronização de barreira

Uma vez que o nome da barreira seja recebido no sub-bloco de recepção, o sub-bloco de emissão/recepção emite sua resposta na linha de sincronização de barreira (LSB). A figura 9 mostra a implementação deste sub-bloco. A lógica 1 transmite uma resposta pela LSB e a lógica 2 recebe a resposta total na mesma linha.

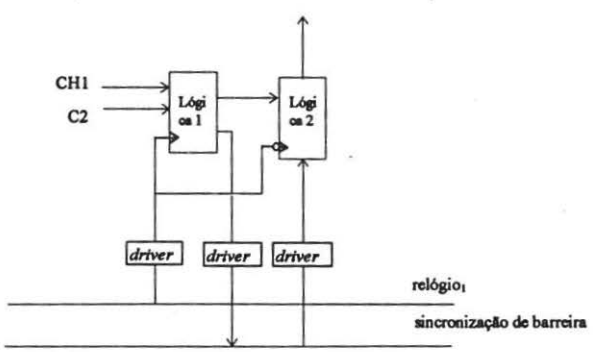


Figura 9. Implementação do sub-bloco de emissão/recepção de sincronização de barreira

Implementação da lógica 1:

A lógica 1 é encarregada de transmitir uma resposta pela LSB. Na borda de subida do sinal "relógio$_1$" é testado o valor de chegada CH1. O sinal emitido pela LSB é sempre 0 e somente muda para 1 quando o nome da barreira do barramento é igual ao nome da barreira interna (C2 = 1) e o processador associado ainda não chegou à barreira (CH1 = 0).

Implementação da lógica 2:

A lógica 2 é encarregada de receber a resposta da LSB. A leitura é feita na borda de descida do sinal "relógio$_1$". Como se observa na figura 7(a), a borda de descida do sinal "relógio$_1$" chega T_2 unidades de tempo depois da borda de subida. O tempo T_2 deve garantir que a leitura seja feita depois que todos os blocos tenham transmitido suas respostas. Para determinar T_2 , é necessário considerar o caso mais crítico, que ocorre quando o primeiro bloco tem o controle do barramento. Neste caso, deve-se fazer o percurso total do barramento para que todos emitam a resposta: na borda de subida do sinal "relógio$_1$" e depois dos atrasos correspondentes no sub-bloco de recepção e na lógica 1 (estes atrasos são chamados de Δ), é emitido uma resposta na LSB. O último bloco recebe esta borda de subida um tempo $\mathbf{T}$ depois, portanto sua resposta estará no barramento um tempo ($\mathbf{T} + \Delta$) depois. O primeiro bloco receberá a resposta total no barramento ($\mathbf{T} + \Delta + \mathbf{T}$) unidades de tempo após da borda de subida do sinal "relógio$_1$", então este deve ser o valor mínimo de T_2.

A resposta do barramento é útil somente no caso que se a barreira interna (NB) no bloco seja igual àquela do barramento (C2 = 1) e além disso se tenha chegado a ela (CH1 = 1). Portanto, o sinal do barramento somente é lido neste caso, no qual o sinal S (que vem da lógica 1) fica em 1 e habilita a entrada do sinal "relógio$_1$". LSB é uma linha "*wired-or*" na qual quando todas suas entradas estão em 0, a linha fica em 0 e quando alguma de suas entradas está em 1, a linha fica em 1.

Como citado anteriormente, no caso crítico, o primeiro bloco lê a LSB T_2 unidades de tempo após da colocação de sua resposta na linha. O último bloco vai ler a linha "sincronização de barreira" $\mathbf{T}_{\text{relógio1}}$ unidades de tempo depois, que corresponde ao atraso de propagação da borda de descida do sinal "relógio$_1$". É necessário que o tempo entre a borda de descida do sinal "relógio$_1$" e sua seguinte borda de subida (chamado t) seja maior do que $\mathbf{T}$ para garantir que o último bloco leia uma resposta certa e estável. O caso no qual t é mínimo ocorre quando um bloco obtém o controle do barramento imediatamente após o bloco anterior. Neste caso:

$$t = \Delta_{\text{arbitro central}} + T_{\text{sinal de relógio}} + \Delta_{\text{lógica de acesso ao barramento}} + T_1 \qquad (2)$$

$\Delta_{\text{arbitro central}}$ = tempo de resposta do arbitro central quando chega o sinal "parada".

$T_{\text{sinal de relógio}}$ = explicado anteriormente.

$\Delta_{\text{lógica de acesso ao barramento}}$ = tempo de atraso da lógica de acesso ao barramento.

T_1 = explicado anteriormente.

4. DESCRIÇÃO DO MODELO ANALÍTICO

O modelo analítico é desenvolvido com o objetivo de analisar o comportamento e o desempenho da solução em diferentes situações. Particularmente o modelo analisa o desempenho da solução quando há uma sincronização de barreira no tempo. Consideram-se duas situações diferentes: quando todos os N processadores do sistema participam na sincronização de barreira e quando P processadores participam. Em cada situação analisa-se o melhor e o pior caso, ou seja, o modelo vai proporcionar dois resultados o límite superior e o límite inferior do desempenho da solução descrita anteriormente. Portanto o comportamento generico da solução sempre vai estar dentro desses límites. Os tempos de atraso considerados no modelo são aproximados e o resultado deve ser considerado como um tempo menor do que o real.

O modelo analítico calcula o tempo de sincronização de barreira que é definido como o tempo decorrido desde o momento que o último processador chega na barreira até que todos os processadores saibam que todos chegaram, este tempo mede o desempenho da solução. Desta mesma maneira é medido o desempenho de outras soluções [5].

1. Todos os processadores participam: neste caso existem duas situações para analisar, quando todos os processadores chegam simultaneamente à barreira e quando não chegam simultaneamente. Para cada uma das duas situações existem dois casos extremos:

1.1. Melhor caso: Se todos os processadores chegam simultaneamente o melhor caso acontece quando todos os processadores chegam imediatamente antes que seja a vez de algum bloco de hardware obter o acesso ao barramento. Neste caso os processadores chegam exatamente antes da borda de descida do sinal "relógio" (sinal da figura 5(a) depois do "*driver*") desta maneira o bloco de hardware obtem o acesso ao barramento e começa a sincronização de barreira.

Analisando detalhadamente este caso temos o seguinte. Para que o bloco de hardware emita o sinal "parada" (e assim obter o barramento) deve ler em seu registrador CH2 da figura 5(b) o valor 1 que vem do registrador CH1 (figura 4 (b)). Na figura 10, T = 0 mostra o momento em que o processador emite ao bloco a chegada da barreira. No melhor caso, T = 0 deve ocorrer antes do atraso (T_a na figura 10) do registrador CH1 da figura 4(b) e do tempo "*setup*" (T_b na figura 10) do registrador CH2 da figura 5(b). Assim na borda de descida do sinal "relógio", CH2 lê um sinal estável e igual a 1. O sinal "relógio" na figura 10 é o sinal depois do "*driver*" da figura 5(b).

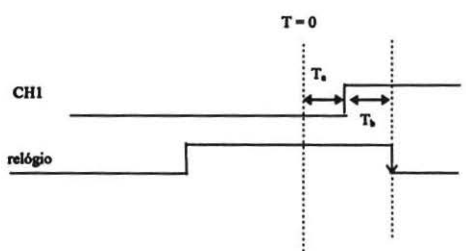


Figura 10. Chegada da barreira no melhor caso

O tempo de sincronização de barreira neste caso será:

$$T_{\text{melhor caso}} = T_a + T_b + T_c + T_1 + T_2 + \mathcal{T}_{\text{relógio1}} + T_d \qquad (3)$$

O tempo de sincronização de barreira como foi definido começa desde T = 0 da figura 10. T_a e T_b são respectivamente o atraso do registrador de figura4(b) e o "*setup*" do registrador da figura 5(b). T_c é o atraso que o bloco de hardware demora para emitir o sinal "relógio$_1$" que corresponde à somo dos atrasos na lógica do barramento da figura 5(a) e na lógica da figura 6(b). T_1 é o tempo necessário para que se escreva satisfatoriamente na LSB e T_2 é o tempo necessário para que o primeiro bloco leia satisfatoriamente a resposta total na LSB. Estes tempos foram explicados anteriormente e são mostrados na figura 7(a). $\mathcal{T}_{\text{relógio1}}$ é o tempo de propagação da linha "relógio$_1$" desde o primeiro bloco até o último bloco, depois deste tempo é que o último bloco lê a resposta total na LSB. T_d é o atraso da lógica 2 da figura 9. Neste caso acaba-se a sincronização de barreira pois todos chegaram ao mesmo tempo. Então T é o tempo de sincronização de barreira.

Se os processadores não chegam simultaneamente, o melhor caso ocorre quando o último processador chega imediatamente antes que seja á vez de seu bloco associado acessar o barramento. A situação é exatamente a mesma que no caso anterior portanto a expressão (3) também é válida.

1.2. Pior caso: Se todos os processadores chegam simultaneamente o pior caso acontece quando todos os processadores chegam imediatamente depois que seja a vez de algum bloco de hardware obter o acesso ao barramento. Neste caso todos os processadores devem esperar até a seguinte borda de descida do sinal "relógio" para começar a sincronização de barreira. Na figura 11 mostra-se esta situação na qual o processador chega à barreira (T = 0) Tm unidades de tempo depois da borda de descida do sinal "relógio". Tm é o tempo de manuntenção (T_{hold}) do registrador CH2 da figura 5(b). Portanto o bloco de hardware obtem o barramento e todos os blocos devem esperar aproximadamente um período T do sinal "relógio" para que o bloco seguinte obtenha o barramento na seguinte borda de descida e recomeçe de novo a sincronização de barreira como no melhor caso.

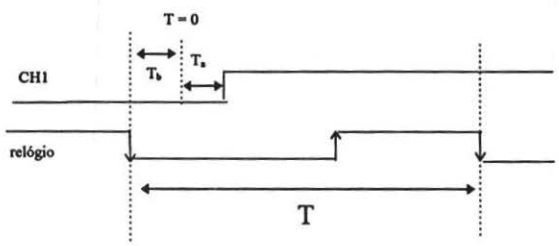


Figura 11. Chegada da barreira no pior caso

O tempo de sincronização de barreira para o pior caso é:

$$T_{\text{pior caso}}(ns) = (T_{\text{fig. 11}} - T_m) + (T_{\text{melhor caso}} - (T_a + T_b)) \quad (4)$$

Substituindo $T_{\text{fig. 11}}$ da expressão (1) e $T_{\text{melhor caso}}$ da expressão (3), tem-se:

$$T_{\text{pior caso}}(ns) = \mathbf{T}_{\text{relógio}} + T_{\text{sinal de relógio}} + \Delta_{\text{lógica de acesso ao barramento}} + \Delta_{\text{lógica de emissão fig. 4.10(a)}} + \mathbf{T}_{\text{parada}} + \Delta_{\text{atraso da parada no arbitro central}} - T_m + T_c + T_1 + T_2 + \mathbf{T}_{\text{relógio1}} + T_d \quad (5)$$

Se os processadores não chegam simultaneamente, o pior caso ocorre quando o último processador chega imediatamente depois que seja á vez de seu bloco associado obter o barramento. A situação é exatamente a mesma que no caso anterior, portanto a expressão (4) também é válida.

2. P processadores participam: neste caso além de considerar quando os P processadores chegam simultaneamente à barreira e quando não chegam simultaneamente deve-se considerar trambém como foram alocados os processadores. Existem dois casos extremos.

2.1. Melhor caso: Se os P processadores chegam simultaneamente, o melhor caso acontece quando chegam imediatamente antes que seja a vez de algum bloco de hardware (pertencente aos P

processadores) obter o barramento. Neste caso os processadores chegam exatamente antes da borda de descida do sinal "relógio" (sinal da figura 4.6 depois do "*driver*") desta maneira o bloco de hardware obtem o acesso ao barramento e começa a sincronização de barreira. O tempo é exatamente o mesmo do caso 1.1, considerando que P na expresssão (3) é N. Se os P processadores não chegam simultaneamente o melhor caso ocorre quando o último processador chega imediatamente antes que seja a vez de seu bloco associado obter o barramento. A situação é exatamente a mesma que no caso anterior, portanto a expressão (3) também é válida.

2.2. Pior caso: Se os P processadores chegam simultaneamente, o pior caso ocorre quando eles são alocados consecutivamente com respeito ao barramento especial de sincronização de barreira e chegam imediatamente depois que seja a vez do P-esimo bloco de hardware (dos P porcessadores) obter o barramento. Neste caso os P processadores devem esperar até que seja a vez do primeiro bloco de hardware (pertencente aos P processadores) obter o barramento. A figura 12 mostra esta situação, onde o primeiro bloco obtem o controle do barramento depois de N - P + 1 ciclos do sinal relógio e recomeça a sincronização de barreira como no melhor caso. T_a é o atraso do registrador da figura 4(b). T_m é o tempo de mantenimento (T_{hold}) do registrador CH2 da figura 5(b). T_c é o atraso que o bloco de hardware demora para emitir o sinal "relógio$_1$".

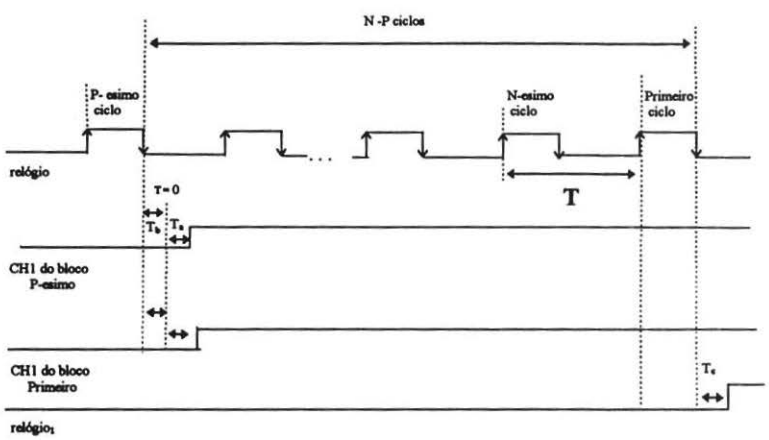


Figura 12. Tempo de sincronização de barreira de P processadores no pior caso

O tempo de sincronização de barreira no pior caso está dado então por:

$$T\,(ns) = (N - P + 1)T - T_m + (T_{melhor\ caso} - (T_a + T_b)) \qquad (6)$$

Se os P processadores não chegam simultaneamente, o pior caso ocorre quando eles são alocados consecutivamente com respeito ao barramento especial de sincronização de barreira e o último processador é o P-ésimo e chega imediatamente depois que seja a vez do P-ésimo bloco de hardware (dos P porcessadores) obter o barramento. A situação é exatamente a mesma que no caso anterior, portanto a expressão (6) também é válida.

Para determinar quantitativamente o tempo de sincronização de barreira é necessário especificar os atrasos dos blocos lógicos. Estes atrasos dependem principalmente da tecnologia escolhida na implementação. No modelo, escolhe-se a tecnologia ECL 10KH [7] por ser de alta velocidade para obter um melhor desempenho. Os atrasos estimados foram calculados baseando-se na informação fornecida pelo Handbook de 1985[7].

O tempo de propagação no barramento depende do material, do comprimento da linha, e da carga. Para sistemas lógicos de alta velocidade é necessário escolher linhas de transmissão especiais que garantam o bom funcionamento do circuito. Para realizar conexões entre placas é preferivel usar linhas do tipo coaxial ou par trançado[7]. Portanto no modelo é necessário considerar um destes tipos de linha. Em geral T é:

$$T = T_{pd0}\sqrt{1+Cd/(Co \times L)} \quad (7)$$

T_{pd0} : tempo de propagação do tipo de linha quando não tem carga em ns/ft. Co : capacitância do tipo de linha por unidade de comprimento quando não tem carga em pfarads/ft. Cd : capacitância de carga na linha em pfarads. L : comprimento da linha em ft. T_{pd0} e Co são características intrinsecas da linha. Cd e L dependem do tipo de tecnologia usada, do projeto e da quantidade de blocos que se usam.

A linha "relógio$_1$" e a linha "sincronização de barreira" são linhas "*wired-or*" e apresenta 2P "*drivers*" (onde P é o número de processadores no sistema) dos quais a metade são de entrada e a outra metade de saída, cada *driver* apresenta uma carga de 5 pfarads na tecnologia ECL 10KH. Portanto Cd = (2P)x5 pfarads. A linha "relógio" e a linha "parada" apresentam P + 1 "*drivers*" (onde P é o número de processadores no sistema). Na linha "relógio" existem P "drivers" de entrada e 1 de saída (para o arbitro central) e na linha "parada" existem P "drivers" de saída e 1 de entrada (para o arbitro central). Cada "*driver*" apresenta uma carga de 5 pfarads na tecnologia ECL 10KH. Portanto Cd = (P+1)x5 pfarads.

Supondo uma distância de separação de 2 cm entre dois blocos e considerando que o arbitro está ao lado do primeiro bloco, L = 0.065x(P-1) ft. A linha par trançadoAWG-26 escolhida para o modelo apresenta T_{pd0} = 1.33 ns/ft e Co = 11.56 pf/ft. O tempo de propagação de $T_{relógio1}$ é igual a $T_{sincronização\ de\ barreira}$ e o tempo de propagação de $T_{relógio}$ é igual a T_{parada}.

A expressão (3) para o tempo de sincronização de barreira no melhor caso será:

$$T(ns) = 25 + 3x0.086\,(P-1)\sqrt{1+13.3\,[P/(P-1)]} \quad (8)$$

A expressão (4) para o tempo de sincronização de barreira no pior caso será:

$$T\,(ns) = 37.5 + 3x0.086\,(P-1)\sqrt{1+13.3\,[P/(P-1)]} + 2x0.086\,(P-1)\sqrt{1+6.65\,[P/(P-1)]} \quad (9)$$

A expressão (6) para o tempo de sincronização de barreira com P processadores no pior caso será:

$$T\,(ns) = (N - P + 1)(13 + 2x0.086\,(N-1)\sqrt{1+6.65\,[N/(N-1)]}) + 3x0.086\,(N-1)\sqrt{1+13.3\,[N/(N-1)]} + 24.5 \quad (10)$$

A figura 13 mostra o comportamento do barramento especial de sincronização de barreira para o caso em que todos os processadores do sistema participam da sincronização de barreira; considera-se sistemas desde P = 2 processadores até P = 100. Baseando-se nas expressões (8) e (9) para o melhor e o pior caso respectivamente.

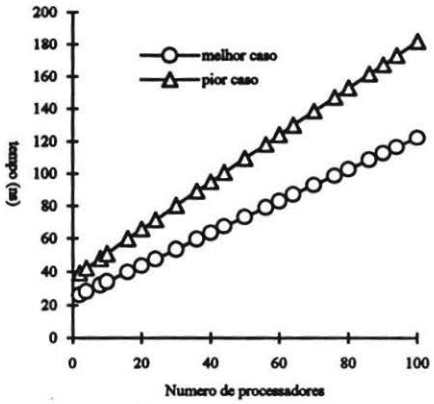


Figura 13. Tempo de sincronização de barreira para sistemas desde P = 2 até P = 100 processadores.

O tempo de sincronização de barreira aumenta quando se usam máquinas com maior número de processadores, isto é devido principalmente aos atrasos produzidos nas linhas de transmissão já que um sistema com maior número de processadores implica, no modelo, um acrescimo no comprimento da linha e da carga. A figura 13 mostra os límites de desempenho da sincronizaçãso de barreira para um sistema com N processadores. Por exemplo, para um sistema com P = 60 processadores, o tempo de sincronização de barreira encontra-se entre 83 ns e 123.8 ns.

A figura 14 mostra o comportamento do barramento especial de sincronização de barreira, para uma máquina de memória compartilhada de N =100 processadores com P processadores realizando sincronização de barreira. Baseando-se nas expressões (8) e (10) para o melhor e o pior caso respectivamente.

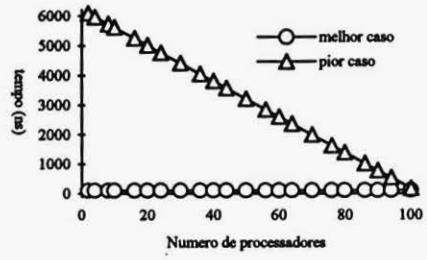


Figura 14. Tempo de sincronização de barreira em um sistema de N = 100 processadores.

O tempo de sincronização para o melhor caso é uma constante e representa o límite inferior para este sistema, este límite é o mesmo da figura 13 para P = 100. O maior tempo ocorre quando se sincronizam 2 processadores do sistema no pior caso, porque os 2 processadores devem esperar um maior número de ciclos de "relógio" para realizar sua sincronização, como é mostrado na expressão (10). Isto significa que entre maior número de processadores participem na sincronização de barreira em um sitema de N processadores, menor será o tempo de sincronização. Por exemplo para sincronizar P = 60 processadores o tempo de sincronização de barreira encontra-se entre 122 ns e 2593.8 ns, observa-se que estes tempos aumentaram com respeito ao caso anterior, porque no caso anterior considera-se um sistema de P = 60 processadores.

5. CONCLUSÕES E TRABALHOS FUTUROS

O barramento especial de sincronização de barreira permite que qualquer número de processadores realizem a sincronização de barreira, permite realizar várias sincronizações de barreira ao mesmo tempo e adicionando alguns outros elementos no bloco de hardware permite realizar sincronizações diferentes consecutivas. O modelo analítico avalia esta solução no caso que somente uma sincronização de barreira realiza-se. Os resultados obtidos com ele dão uma tempo de sincronização aproximado que permite concluir que o barramento especial de sincronização é uma solução razoavelmente eficiente comparando-a com os tempos obtidos com outras soluções de hardware [5]. No futuro pretende-se avaliar o desempenho do barramento especial de sincronização de barreira considerando várias barreiras simultâneas e barreiras diferentes consecutivas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] AXELROD, T.S. Effects of synchronization barriers on multiprocessor performance. Parallel Computing, v. 3, n. 2, p. 129-140, may 1986.

[2] TORRES, M.X., et al. Estudo do efeito da sincronização de barreira implementada em software no desempenho de máquinas paralelas. In: SIMPOSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES PROCESSAMENTO DE ALTO DESEMPENHO,5. Florianópolis, 1993. **Anais**. Florianópolis, 1993. v.2, p. 243-258.

[3] MELLOR-CRUMMEY, J. M.; SCOTT, M. L. Algorithms for scalable synchronization on shared-memory multiprocessors. ACM Transaction on Computer Systems, v. 9, n. 1, p. 21-65, Fev. 1991.

[4] LUNDSTROM, S.F. Applications considerations in the system design of highly concurrent multiprocessors. IEEE Transactions on Computers, v. c-36, n. 11, p. 1292-1309, nov. 1987.

[5] DAVIS, M.H.; RAMACHANDRAN U. Synchronization primitives on an optical broadcast ring. Georgia Institute of Technology/College of Computing, 1992. (PhD) Thesis - Georgia Institute of Technology.

[6] GUPTA, R.; EPSTEIN, M. Achieving low cost synchronization in a multiprocessor system. In: 3rd CONFERENCE ON PARALLEL ARCHITECTURES AND LANGUAGES EUROPE, 3., Eindhoven, 1989. Proceedings. Springer-Verlag, 1989. v.1, p. 70-84.

[7] BLOOD, W.R. MECL System Design Handbook. MOTOROLA Semiconductors Products Inc. 1983.