Simulação e síntese de controladores paralelos a partir de especificações baseadas em redes de Petri

João M. Fernandes **António M. Pina**
Alberto J. Proença
Dep. Informática, Universidade do Minho
Largo do Paço, 4709 Braga codex, Portugal
Email: miguel@di.uminho.pt

RESUMO

As Redes de Petri (RdP) mostram-se uma metodologia poderosa na modelação de sistemas de eventos discretos. Tal deve-se, em grande medida, ao conjunto disponível de técnicas formais para análise estrutural e dinâmica dos sistemas modelados. Apresentam-se as vantagens na utilização de RdP, relativamente a outros paradigmas de modelação, na especificação de controladores com comportamento paralelo. São também propostas algumas alterações ao comportamento habitual das RdP, de forma a conseguir modelar eficientemente os controladores. É apresentado um novo ambiente de desenvolvimento que permite especificar, analisar, animar, simular e sintetizar estruturas de controlo digitais, a partir de uma RdP. Finalmente, é considerado e analisado detalhadamente um exemplo.

Palavras-chave: Redes de Petri, controladores digitais, VHDL, sistemas ECAD.

ABSTRACT

Petri Nets prove to be an efficient methodology for modelling discrete-event systems with parallel activities. This is due to the availability of a set of formal techniques, for validation of the modelled system. The advantages on using Petri Nets in relation to other modelling paradigms are considered. Modifications to the standard PN behaviour are proposed which allow fast specification of synchronous parallel controllers. A new software framework is presented which allows complex parallel controllers, to be specified, analysed, animated, simulated and synthesized for a Petri Net-based controller. The analysis of a detailed example closes this communication.

Keywords: Petri Nets, Parallel Controllers, VHDL, ECAD tools.

1 Introdução

A análise de um sistema digital complexo inicia-se, naturalmente, dividindo-o em duas estruturas distintas —a de controlo e a de dados (*Control + Data Path*) [1]— que cooperam através de sinais gerados por cada uma delas.

As máquinas de estado finitas (FSM de "Finite State Machines"), usadas na modelação de sistemas sequenciais [2], permitem especificar controladores em que está unicamente activo um estado em cada momento. Embora as FSM se mostrem vocacionadas para a especificação de controladores sequenciais, é também possível usá-las para especificar controladores paralelos (vários estados activos simultaneamente).

Nas FSM de controladores paralelos, o controlo do sistema digital é subdividido por um conjunto de controladores sequenciais, sendo a sua interligação realizada através de sinais comuns ou bits de semáforo. Esta abordagem pode não ser de fácil aplicação, nem garante que a divisão resultante corresponda a uma implementação eficiente. Pode ainda acarretar outros problemas, tais como bloqueio (dois controladores esperam indefinidamente um pelo outro) ou acesso múltiplo (uma operação na unidade de dados é pedida por múltiplos controladores em simultâneo), provenientes da dificuldade em verificar a existência de erros de sincronização.

Embora as técnicas baseadas em FSM sejam adequadas para sistemas puramente sequenciais, a sua utilização torna-se mais complexa, na modelação de controladores de sistemas digitais potencialmente paralelizáveis. A utilização de RdP [3, 4] na modelação de controladores digitais apresenta-se como uma forma viável de ultrapassar os problemas surgidos com a especificação realizada com FSM.

2 Redes de Petri

Segundo Valette et al. [5], de entre as vários metodologias de modelação, a que se baseia em RdP é a único que permite facilmente especificar subsistemas cooperantes, além de tornar possível a utilização de procedimentos formais de validação. As RdP são também uma linguagem universal,unívoca e gráfica, fácil de compreender e de manipular. Ver exemplo, na figura 1, da representação gráfica de uma RdP.

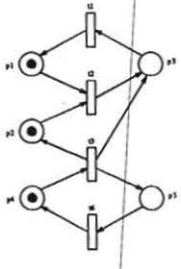

Figura 1: Exemplo de uma Rede de Petri.

Numa RdP, o grafo associado representa as propriedades estáticas ou estruturais do sistema [6]. O grafo é constituído por dois tipos de nodos, círculos e rectângulos, habitualmente designados por *lugares* e *transições*, respectivamente.

Os nodos do grafo são ligados por arcos dirigidos, de lugares para transições e vice versa, não podendo existir arcos a ligar nodos do mesmo tipo.

As características dinâmicas do sistema são modeladas através da atribuição de *marcas* (graficamente representadas por um pequeno círculo a cheio •) aos lugares. Uma *marcação* é uma atribuição de marcas aos lugares, cujo número e posição podem ser alterados durante a execução da RdP.

A partir de uma determinada marcação, uma RdP é executada, através do *disparo* de transições. Uma transição está *habilitada* (i.e. pode disparar), se todos os seus lugares de entrada contêm, pelo menos, uma marca. A transição, ao disparar, remove uma marca de cada um dos seus lugares de entrada e põe uma marca em cada um dos seus lugares de saída.

Numa RdP não são definidas quaisquer regras para determinar univocamente qual a transição a disparar, quando num dado instante, há várias habilitadas. A escolha da transição a disparar é feita não-determinísticamente [1] ou por forças que não são modeladas.

As RdP exibem três características principais: assincronismo, paralelismo e não determinismo. O *assincronismo* resulta da inexistência da noção de tempo no modelo. Existe apenas uma ordem parcial no disparo das transições (na ocorrência de eventos). O *paralelismo* assenta na possibilidade de duas transições habilitadas, que não interajam uma com a outra, poderem disparar independente e simultaneamente. O *não determinismo* surge da própria definição de disparo.

2.1 Redes de Petri em sistemas digitais

Considerando, como afirma Peterson [3], que uma das propriedades mais importantes a que uma RdP deve obedecer, no caso de sistemas digitais, é a segurança, resolveu-se escolher, como base para este trabalho, as RdP seguras (alternativamente denominadas por Sistemas Condição/Evento), de entre as várias variantes existentes. Uma RdP diz-se segura se todos os seus lugares forem seguros. Um lugar é seguro, se o número de marcas que pode conter não excede 1 (é 0 ou 1), o que permite implementá-lo directamente por um *flip-flop*. Se a RdP for limitada, o flip-flop dá lugar a um contador[2].

Para que as RdP seguras possam ser usadas na especificação de controladores digitais paralelos foi necessário introduzir algumas alterações ao modelo padrão de RdP. As

[1] Esta característica das RdP reflecte o facto de em situações da vida real, onde vários factos podem suceder simultaneamente, a ordem da ocorrência dessses acontecimentos não ser única. Se o não-determinismo é vantajoso do ponto de vista da modelação, introduz todavia alguma complexidade na análise do comportamento da RdP [6].

[2] Se um lugar é $k-$ *limitado*, então pode ser implementado com um contador de, pelo menos, $\log_2 k$ bits.

alterações incluem a possibilidade de disparar síncrona e simultaneamente várias transições (reguladas por um sinal de relógio), a atribuição de guardas (expressões lógicas formadas por sinais de entrada) às transições e a geração de sinais de saída nos nodos (lugares e transições). Estas alterações que caracterizam uma RdP Síncrona e Interpretada (RdPSI) exprimem-se com base nas seguintes regras:

Regra 1 (Habilitação de uma transição guardada) *Uma transição guardada de uma RdPSI diz-se habilitada a disparar, se se verificam os três factos seguintes: cada um dos seus lugares de entrada tem uma marca, cada um dos seus lugares de saída não contém qualquer marca e, a guarda associada a essa transição é verdadeira.*

Regra 2 (Disparo Simultâneo) *Todas as transições habilitadas, num dado instante de sincronismo, disparam em simultâneo.*

A utilização de RdPSI na modelação de controladores digitais pode ser vista como uma extensão natural às FSM, facilmente adaptável ao projecto de controladores paralelos. As RdPSI, comparativamente mais simples que outras variantes das RdPs (Estocásticas, de Alto Nível ou Temporizadas), permitem modelar as FSM, como casos particulares de controladores paralelos, em que o projectista beneficia dos conhecimentos previamente adquiridos na modelação por FSM.

2.2 Propriedades das RdPSI

As RdPSI que modelam os controladores paralelos devem ser vivas, seguras, determinísticas e livre de conflitos [2, 7].

Uma RdP é *viva* se é possível, a partir de qualquer marcação, habilitar todas as transições através de uma dada sequência de disparo. Este facto assegura a inexistência de situações de bloqueio e código morto (transições nunca habilitadas ou lugares nunca marcados), que garantem não haver desperdício de silício na implementação final.

Uma RdP diz-se *segura* se cada lugar não contém mais do que uma marca, em cada instante, podendo os lugares ser implementados por *flip-flops*. As transições numa RdP segura só podem disparar se os seus lugares de saída estiverem vazios, pelo que a segurança garante que as operações na unidade de dados não podem ser re-invocadas, antes de terem terminado completamente.

Uma RdP *determinística* garante que os recursos disponíveis na unidade de dados não podem ser simultaneamente partilhados ou invocados por mais do que um nodo da RdP. Deste modo, para cada marcação e conjunto de transições habilitadas, os sinais de controlo gerados devem ser mutuamente exclusivos.

Uma RdP é *livre de conflitos* se para todo o lugar marcado, não mais do que uma das suas transições de saída está habilitada, em qualquer momento, garantindo que o controlador se comporta de uma forma determinística.

3 Ambiente CAD

A consideração que o projecto de sistemas digitais tem de contemplar a realização física do sistema foi a motivação para a criação de um ambiente integrado de desenvolvimento, ECAD (Electronic CAD).

O ambiente proposto [8] (ver figura 2), desenvolvido como um conjunto de módulos independentes, parte do modelo RdPSI de um controlador para a síntese digital através de um processo com o seguinte faseamento:

1. modelação e validação das propriedades,
2. animação,
3. representação em VHDL.

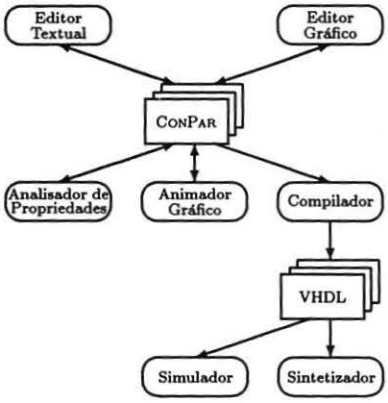

Figura 2: O ambiente completo de desenvolvimento.

3.1 Modelação

Os módulos *Editor Textual* e *Editor Gráfico* são utilizados para gerar a descrição do modelo inicial em linguagem ConPar [8]. Esta linguagem, que se baseia em regras de sequentes lógicos de Gentzen e que faz também uso da lógica de decisão e da lógica temporal, é especialmante adequada para a descrição de modelos RdPSI.

No ambiente, o módulo *Analisador de Propriedades* tem por função validar, de uma forma sistemática, as propriedades viva e livre de conflitos, de um controlador descrito em ConPar, evitando o recurso exclusivo à heurística e à intuição. O analisador gera um grafo de alcançabilidade [3], estrutura de dados que representa

todas as marcações alcançáveis a partir da marcação inicial, que em conjunto com o grafo que representa o próprio controlador permite a validação das propriedades.

A prova de que uma RdPSI não viola nenhuma das propriedades não é condição suficiente para validar a especificação do controlador face ao objectivo do projectista; contudo, se a RdPSI violar alguma das propriedades, a especificação está incorrecta.

3.2 Animação

O módulo *Animador Gráfico* é uma aplicação gráfica desenvolvida inteiramente em SCBA (Simulação Concorrente Baseada em Agentes) [9], que permite interactivamente, criar, visualizar e animar RdPSI. Neste ambiente, o agente, unidade básica de computação no modelo MCBA (Modelo de Computação Baseado em Agentes) [10], pode ser visto como um objecto concorrente, cujo comportamento é descrito por regras de produção e que interactua com outras entidades semelhantes, enviando e recebendo "mensagens" através de canais e portas.

No animador, o comportamento de cada entidade de uma RdPSI é modelado por um agente. Cada agente é entregue a um processador virtual, para ser executado independente e concorrentemente. Cada processador virtual gere a totalidade dos processos associados à actividade global de um agente, incluindo computação e comunicação. Um escalonador central, baseado numa esquema de prioridades, controla todos os processadores virtuais que executam concorrentemente.

3.3 Representação em VHDL

O módulo *Compilador* converte a descrição do modelo escrita em linguagem CONPAR para a correspondente em VHDL [11].

O código VHDL gerado corresponde ao nível de abstracção RTL (fluxo de dados), cujo grau de detalhe é suficiente para identificar e descrever os componentes a sintetizar.

As vantagens de usar a linguagem VHDL, standard IEEE 1076-1987, reflectem-se na possibilidade de beneficiar de todo o conjunto de ferramentas ECAD já existentes, desde que o sub-conjunto da linguagem utilizado seja aceite por essas ferramentas.

4 Exemplo de aplicação

Um exemplo paradigmático de um controlador paralelo, *reactor* [12], foi seleccionado, para mostrar a viabilidade da metodologia proposta. O reactor é usado para controlar o funcionamento de um sistema de mistura e transporte de reagentes — ver figura 3(a).

No exemplo, os silos $S1$ e $S2$, onde estão armazenados os reagentes, alimentam

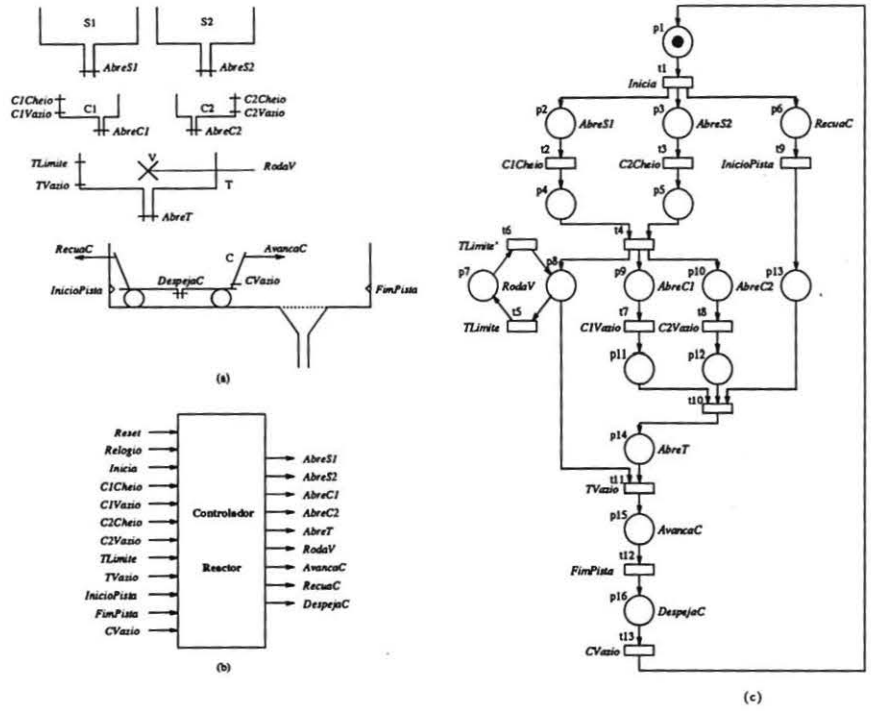

Figura 3: Reactor. (a) Sistema de Mistura e Transporte. (b) Esquemático do Controlador. (c) RdPSI para especificação do reactor.

os dois contentores $C1$ e $C2$, respectivamente. Os seus conteúdos são despejados para o tanque T e misturados, através do accionamento da ventoínha V (pás de uma misturadora). Quando se esgotam os reagentes nos contentores, a mistura é despejada para o carro C e descarregada, quando o carro atingir o fim da pista.

Os sinais `AbreS1` e `AbreS2` accionam as válvulas que controlam a queda dos reagentes nos contentores. O controlo da queda dos reagentes para o tanque é feito pelos sinais `AbreC1` e `AbreC2`. Os sinais `C1Cheio` e `C2Cheio` (`C1Vazio` e `C2Vazio`) indicam se os contentores $C1$ e $C2$, respectivamente, estão cheios (vazios). O sinal `TVazio` indica se o tanque T está vazio.

Quando o sinal `TLimite` está activo, o sinal `RodaV` é activado para accionar a ventoínha V. O sinal `TLimite` indica se há reagentes em quantidade suficiente para produzir a mistura que é despejada para o carro, activando o sinal `AbreT` que abre a válvula V.

Os sinais `AvancaC` e `RecuaC` controlam o movimento do carro. O seu conteúdo é vertido, accionando o sinal `DespejaC` que abre a válvula colocada no fundo do carro, até não haver mais mistura no carro. Nesta situação o sinal `CVazio` fica

activo. O momento em que o carro suspende o seu movimento é determinado pelos sinais `InicioPista` e `FimPista` que indicam quando o carro atingiu os extremos da pista.

O funcionamento do reactor corresponde a uma actividade cíclica que se inicia sempre que o sinal `Inicia` estiver activo, enquanto que os sinais `Reset` e `Relogio` são utilizados para iniciar e o sincronizar o controlador, respectivamente.

4.1 Modelação

O comportamento do controlador, representado pelo diagrama na figura 3(b), é modelado pela RdPSI na figura 3(c). Abaixo apresenta-se em linguagem CONPAR fragmentos do código do controlador do reactor.

```
.clock RELOGIO
.input INICIA C1CHEIO C1VAZIO ... FIMPISTA CVAZIO
.output ABRES1 ABRES2 ABREC1 ... RECUAC DESPEJAC

.part REACTOR
.place p1 p2 p3 ... p16
.transition t1 t2 t3 ... t13

.net
t1:  p1  * INICIA    |- p2 * p3 * p6;
t2:  p2  * C1CHEIO   |- p4;
t3:  p3  * C2CHEIO   |- p5;
...
t13: p16 * CVAZIO    |- p1;

.MooreOutput
p2   |-  ABRES1;
p3   |-  ABRES2;
...
p16  |-  DESPEJAC;

.marking p1
.e
```

Neste exemplo, o *Analisador de Propriedades* permite detectar um potencial conflito entre as transições *t5* e *t11* (partilham o lugar de entrada *p8*). No entanto, essas transições nunca poderão estar simultaneamente habilitadas, por estarem guardadas pelos predicados `TLimite` e `TVazio` que não podem fisicamente estar activos no mesmo instante.

O uso do grafo de alcançabilidade, de que se apresenta abaixo um fragmento, pode ser exemplificado pelo que sucede com a partilha do lugar de saída *p8* pelas transições *t4* e *t6*.

```
p1          : t1        ->  p6 p3 p2
p6 p3 p2    : t9 t3 t2  -> p13 p5 p4
            : t9 t3     -> p13 p5 p2
            : t9 t2     -> p13 p4 p3
            : t9        -> p13 p3 p2
            : t3 t2     ->  p6 p5 p4
            : t3        ->  p6 p5 p2
            : t2        ->  p6 p4 p3
```

Pode parecer possível, à primeira vista, que essas transições estejam simultaneamente habilitadas, uma vez que a transição *t4* não é guardada por qualquer sinal e a transição *t6* está guardada pelo predicado `TLimite'`. No entanto, uma análise do grafo de alcançabilidade completo leva a concluir que as duas transições nunca estarão simultaneamente habilitadas.

4.2 Simulação e síntese

O código VHDL, apresentado de seguida, foi gerado pelo módulo *Compilador* para a RdPSI do exemplo considerado. Foi usada uma opção de compilação de forma a criar um BLOCK; em alternativa poder-se-ia ter usado uma outra opção para gerar um PROCESS.

```
ENTITY controller IS
  PORT (reset, inicia, cicheio, ..., cvazio, relogio : IN BIT;
        abres1, abres2, abrec1, ..., recuac, despejac : OUT BIT);
END controller;

ARCHITECTURE dataflow OF controller IS
-- Place Signals
  SIGNAL p1, ..., p16  : REG_BIT REGISTER;
  SIGNAL Np1, ...,  Np16 : BIT;
-- Transition Signals
  SIGNAL t1, ..., t13 : BIT;

BEGIN
  PART : BLOCK (relogio='1' AND NOT relogio'STABLE)
  BEGIN
    p1  <= GUARDED Np1  WHEN reset='0' ELSE '1';
    ...
    p16 <= GUARDED Np16 WHEN reset='0' ELSE '0';
  END BLOCK;
-- Dataflow description for transitions
  t1  <= inicia AND p1 AND NOT p2 AND NOT p3 AND NOT p6;
  t2  <= c1cheio AND p2 AND NOT p4;
  t3  <= c2cheio AND p3 AND NOT p5;
  ...
  t13 <= cvazio AND p16 AND NOT p1;
-- Dataflow description for next place markings
  Np1  <= t13 OR (p1 AND NOT t1);
  Np2  <= t1 OR (p2 AND NOT t2);
  Np3  <= t1 OR (p3 AND NOT t3);
  ...
  Np16 <= t12 OR (p16 AND NOT t13);
-- Output Signals Equations
  abres1 <= p2;
  abres2 <= p3;
   ...
  despejac <= p16;
-- Transitions in conflict
  ASSERT NOT (t4 AND t6)
    REPORT "t4 and t6 are in conflict, due to output place p8"
    SEVERITY ERROR;
  ASSERT NOT (t5 AND t11)
    REPORT "t5 and t11 are in conflict, due to input place p8"
    SEVERITY ERROR;
-- No Enabled Transitions
  ASSERT NOT (t1='0' AND t2='0' AND ... AND t13='0')
    REPORT "Petri Net may be deadlocked"
    SEVERITY WARNING;
END dataflow;
```

O funcionamento do controlador foi testado com o simulador ASIMUT [13], usando o código VHDL e vectores de teste. A simulação foi executada com sucesso, produzindo apenas mensagens de aviso por haver situações em que, na RdPSI considerada, não há momentaneamente nenhuma transição habilitada a disparar.

O código VHDL obtido pelo *Compilador* foi usado, como entrada no sintetizador LOGIC [14], para produzir uma outra especificação VHDL ao nível estrutural que descreve claramente os circuitos lógicos do controlador. Esta especificação lista os componentes do sistema e respectivas interligações, não sendo a funcionalidade dos componentes evidente porque estes são, apenas, vistos como caixas negras com um interface bem definido.

Do código VHDL, ao nível estrutural gerado para o controlador do reactor, apresenta-se abaixo um fragmento.

```
ENTITY reactor IS PORT (
  reset : in BIT;        -- reset
  inicia : in BIT;       -- inicia
  ...
  despejac : out BIT;    -- despejac
  vdd : linkage BIT;     -- vdd
  vss : linkage BIT      -- vss
  );
END reactor;

-- Architecture Declaration
ARCHITECTURE structural_view OF reactor IS
  COMPONENT nao3_y port (
    i0 : in BIT;         -- i0
    i1 : in BIT;         -- i1
    i2 : in BIT;         -- i2
    f : out BIT;         -- f
    vdd : in BIT;        -- vdd
    vss : in BIT         -- vss
    );
  END COMPONENT;
  ...
  COMPONENT ms_y port (
    i : in BIT; -- i
    l : in BIT; -- l
    t : out BIT;         -- t
    vdd : in BIT;        -- vdd
    vss : in BIT         -- vss
    );
  END COMPONENT;

  SIGNAL t13 : BIT;      -- t13
  SIGNAL t11 : BIT;      -- t11
  ...
  SIGNAL auxreg4 : BIT; -- auxreg4

BEGIN
  ...
  auxsc117 : nao3_y PORT MAP (
       vss => vss, vdd => vdd, f => auxsc117,
       i2 => reset, i1 => auxsc121, i0 => auxsc119);
  ...
  p16 : ms_y PORT MAP (
       vss => vss, vdd => vdd, t => despejac,
       l => auxsc9, i => auxsc8);
  ...
  p1 : ms_y PORT MAP
       vss => vss, vdd => vdd, t => auxreg16,
       l => auxsc9, i => auxsc139);
end structural_view;
```

5 Conclusões

Neste trabalho apresentaram-se as RdPSI, resultantes da introdução de algumas alterações ao comportamento habitual das RdP, que permitem a especificação de controladores paralelos como uma extensão natural às FSM. Foi apresentado um ambiente de desenvolvimento para analisar, animar, simular e sintetizar estruturas de controlo digitais, tomando como entrada inicial uma especificação em RdPSI.

A possibilidade de utilizar mecanismos modulares e hierárquicos na especificação e a sua tradução para VHDL é um objectivo futuro, que se reflectirá positivamente na modularizaçao e reutilização de código já testado, diminuindo substancialmente os custos de desenvolvimento.

A necessidade de modelar controladores de elevada complexidade faz prever que, no futuro, o cálculo do grafo de alcançabilidade será transferido do *Analisador de Propriedades* para o *Animador*. Pretende-se, desta forma, aproveitar o suporte à concorrência disponibilizado pelo modelo MCBA para tornar mais rápido o cálculo do grafo de alcançabilidade e correspondente análise de propriedades. Actualmente, é alvo de um projecto I&D a realização do MCBA num ambiente de computação paralelo e distribuído de que também beneficiará a animação de RdPSI baseadas em agentes.

Referências

[1] Alberto J. Proença. Advanced Controller Design. In *Microcomputer '91 - design, practice, education*, number 39 in Konferencje, pp. 191–205. Prace Naukowe Instytutu Cybernetyki Technicznej Politechniki Wroclawskiej, 1991.

[2] James Pardey and Martin Bolton. Logic Synthesis of Synchronous Parallel Controllers. *Proceedings of the IEEE International Conference on Computer Design*, pp. 454–7, 1991.

[3] James L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, New Jersey, E.U.A., 1981.

[4] Tadao Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–80, Abril 1989.

[5] R. Valette, M. Courvoisier, J.M. Bigou, and J. Albukerque. A Petri Net Based Programmable Logic Controller. In *IFIP First International Conference on Computer Applications in Production and Enginnering*, Abril 1983.

[6] James L. Peterson. Petri Nets. *Computing Surveys*, 9(3):223–52, Setembro 1977.

[7] Pierre Azema, Robert Valette, and Michel Diaz. Petri Nets as a Common Tool for Design Verification and Hardware Simulation. In *Proceedings of the 13th ACM/IEEE Design Automation Conference*, pp. 109–16, Junho 1976.

[8] João M. Fernandes. Redes de Petri e VHDL na Especificação de Controladores Paralelos. Dissertação de Mestrado, Dep. Informática, Universidade do Minho, Braga, Portugal, Julho 1994.

[9] António M. Pina. SCBA — Simulação Concorrente Baseada em Agentes. In *XX SEMISH*, Florianópolis, Brasil, 1993.

[10] António M. Pina. MCBA — Modelo de Computação Baseado em Agentes. In *OOP'94*, Lisboa, Portugal, 1994.

[11] Douglas L. Perry. *VHDL*. McGraw-Hill, 1991.

[12] Marian Adamski. Direct Implementation of Petri Net Specification. In *7th International Conference on Control Systems and Computer Science CSCS7*, pp. 74–85, 1987.

[13] P. Bazargan Sabet, A. Greiner, and H. N. Vuong. ASIMUT: A Public Domain VHDL Simulation Tool. In *Fourth Eurochip Workshop on VLSI Design Training*, pages 62–5, Toledo, Espanha, 1993.

[14] Alain Greiner and François Pêcheux. Alliance: A complete set of CAD tools for teaching digital VLSI design. In *3rd Eurochip Workshop on VLSI Design Training*, pp. 230–237, Toledo, Espanha, Setembro 1992.