1º Encontro Nacional do Colégio de Engenharia Electrotécnica
Ordem dos Engenheiros - Lisboa - Maio 1994

REDES DE PETRI NA ESPECIFICAÇÃO E VALIDAÇÃO DE CONTROLADORES PARALELOS

João Miguel Fernandes **Alberto José Proença**
Dep. Informática, Universidade do Minho
Largo do Paço, 4719 BRAGA CODEX, PORTUGAL
email: miguel@di.uminho.pt

Sumário

As Redes de Petri (abreviadamente RdP) têm mostrado ser uma metodologia eficiente na modelação de sistemas de eventos discretos. Este facto deve-se, em grande parte, à existência de um conjunto de técnicas para análise estrutural e dinâmica das RdP. Neste artigo mostram-se as vantagens na utilização de RdP, relativamente a outros paradigmas de modelação, na especificação de controladores que exibam comportamento marcadamente paralelo. São também propostas algumas alterações ao comportamento habitual das RdP, de forma a conseguir modelar eficientemente os controladores. Este artigo apresenta também um compilador que gera código VHDL, a partir de uma especificação de um controlador, baseada numa RdP. Finalmente, é considerado e analisado um exemplo.

1 INTRODUÇÃO

A especificação de uma estrutura de controlo digital relativamente complexa encontra-se normalmente dividida em duas partes distintas, mas cooperantes: uma estrutura de controlo e uma unidade de manipulação de dados (*Control + Data Path*) [1]. Essa perspectiva é ilustrada na figura 1. Estas duas partes estão relacionadas pelos sinais de actuação provenientes da estrutura de controlo para a unidade de dados e pelos qualificadores dos resultados das operações efectuadas sobre os dados, dirigidos em sentido inverso.

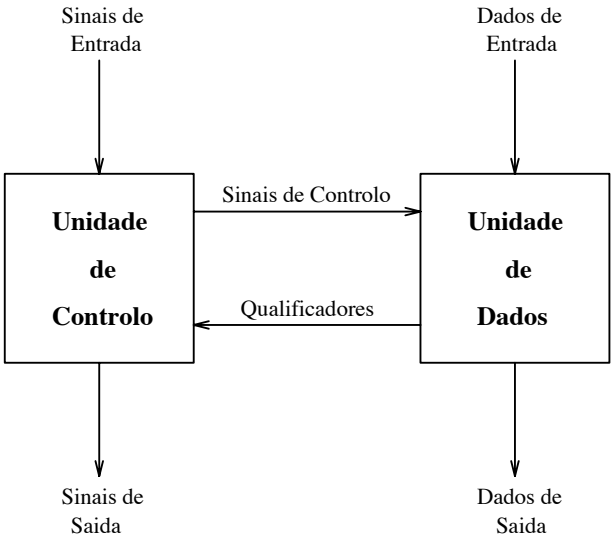

Figura 1: Estrutura Genérica de um Sistema Digital.

O primeiro passo no projecto de qualquer sistema digital é o da sua especificação. Normalmente, parte-se de uma descrição comportamental algorítmica e informal, e faz-se a sua tradução para uma representação mais formal. No caso particular dos controladores programáveis (PLCs), Gomes et al. [2] referem que existe uma série de métodos que permite especificar sistemas digitais como "*ladder diagrams*", linguagens lógicas e GRAFCET (*Graph de Commande Etape-Transition*). Algumas das linguagens de programação de controladores baseiam-se em formalismos de carácter gráfico, adaptados para a modelação de sistemas sequenciais ou paralelos [3]. Os diagramas de estado e os diagramas ASM (*Algorithmic State Machines*) [4] são formalismos bastante utilizados para sistemas sequen-

ciais, enquanto para sistemas paralelos, o sistema GRAFCET [5] (um *standard de facto* internacional) é baseado nas RdP.

As RdP têm mostrado ser um método eficiente na modelação de sistemas de eventos discretos. Este facto deve-se fundamentalmente à existência de um conjunto de técnicas matemáticas para análise estrutural e dinâmica das RdP. Assim, pretende-se mostrar as vantagens que resultam na utilização das RdP na especificação de controladores paralelos.

Este artigo está dividido em quatro secções. Na secção 2 são apresentadas as RdP. A secção 3 analisa um tipo específico de RdP — mais concretamente as RdP Síncronas e Interpretadas (RdPSI) — e apresentam-se as suas características principais. O compilador ConPar, em desenvolvimento na Universidade do Minho, é descrito na secção 4. A finalizar desenvolve-se um exemplo de uma RdPSI, na secção 5.

2 REDES DE PETRI

Os diagramas ASM são um método poderoso, compacto e eficiente para a especificação de unidades de controlo em sistemas sequenciais. Mais concretamente, podem-se especificar controladores em que apenas um estado está activo, em cada momento — controladores sequenciais. Alguns fabricantes permitem que se implemente de um modo quase directo os diagramas ASM. Por exemplo, a linguagem PALASM contém alguns comandos vocacionados para a escrita directa de diagramas ASM. Outros exemplos válidos são as linguagens ABEL, CUPL e OrCAD-PLD [6].

É precisamente a existência de ferramentas de apoio à especificação, aliada à generalização e uso fácil dos dispositivos de lógica programável, que tornou os diagramas ASM tão populares na especificação de sistemas sequenciais.

É possível, também, especificar um controlador paralelo (controlador em que podem estar activos vários estados simultaneamente), usando as técnicas inerentes aos diagramas ASM [7]. Para tal, há que dividir o controlador inicial num conjunto de controladores locais. Estes controladores locais serão sequenciais e a sua interligação faz-se através de técnicas que envolvem o uso de sinais comuns ou bits de semáforo. Qualquer uma dessas técnicas é contudo de difícil aplicação, além de que a divisão resultante pode corresponder a uma implementação ineficiente. Esta abordagem traz ainda outros problemas: a dificuldade de detecção de erros de sincronização, tais como bloqueio (*deadlock*) — quando dois controladores esperam indefinidamente um pelo outro — ou acesso múltiplo — quando uma operação na unidade de dados é pedida por dois controladores em simultâneo. Assim, embora as técnicas baseadas em diagramas ASM sejam apropriadas para sistemas puramente sequenciais, a sua aplicação é inadequada para sistemas que exibam comportamento paralelo.

Os testes para validação do comportamento de um controlador (e genericamente de um qualquer sistema digital), quando se usa uma ferramenta CAD, baseiam-se, normalmente, na utilização de vectores de teste, i.e., numa lista de valores de entrada e respectivos valores de saída esperados. Este tipo de testes apenas prova que o sistema funcionou (ou não) para os casos considerados. Se para sistemas de reduzidas dimensões a sua utilização pode aceitar-se, o mesmo não é válido para sistemas de maior porte.

De entre os vários paradigmas de modelação, aquele que se baseia em RdP é o único que permite facilmente especificar subsistemas cooperantes, além de tornar possível a utilização de procedimentos formais de validação [8]. Estes procedimentos são aplicados antes de se implementar o sistema e, baseiam-se numa teoria matemática [9]. Adicionalmente, as RdP constituem uma linguagem gráfica fácil de compreender e de manipular, unívoca e universal. A utilização do GRAFCET é análoga ao uso de RdP. Todavia, há ligeiras diferenças e ambiguidades no comportamento dos modelos GRAFCET, relativamente às RdP, que tornam difícil (ou mesmo impossível) a aplicação, a esses modelos, de técnicas de análise [3].

A progressiva complexidade dos sistemas de controlo torna a sua concepção de difícil entendimento para a equipa de projecto e para os seus utilizadores. Este facto, segundo Silva [3], sugere que:

- seja feita uma distinção bastante clara entre a

especificação do sistema (algo parecido a um contrato) e a sua implementação;

- se proceda a uma verificação formal da especificação (na prática apenas são verificadas algumas características da especificação) antes de se proceder à sua implementação.

Esta divisão torna indispensável a existência de um formalismo que permita uma análise matemática dos sistemas. Desta forma, além de claro e preciso, um "bom" formalismo deve respeitar as seguintes características [3]:

- poder de expressão, permitindo que o modelo seja construído modularmente.
- modelação clara e explícita de actividades e eventos.
- representação gráfica que permita um diálogo entre utilizadores e projectistas.
- existência de uma teoria forte de análise e verificação.
- existência de aplicações para computador para auxílio do projectista, durante o período de verificação.
- implementação fácil, usando um leque de soluções o mais vasto possível.

Neste sentido, o formalismo baseado em RdP parece ser o mais adequado.

3 REDES DE PETRI SÍNCRONAS E INTERPRETADAS

A especificação de controladores paralelos, usando RdPSI é feita a um nível bastante alto e, numa fase inicial, apenas se considera o comportamento do sistema, e não a sua implementação física. A especificação pode ser realizada de um modo hierárquico ou, se se preferir, modular, resultando ainda num maior nível de abstração.

A primeira alteração efectuada, em relação ao modelo genérico das RdP, reside na colocação de condições (expressões lógicas formadas por variáveis de entrada, a que se chama *guardas*) nas transições [10, 11]. A estas transições pode-se dar o nome de *transições guardadas*. A esta versão de RdP, chamar-se-á *Redes de Petri Interpretadas* (abreviadamente RdPI).

As guardas associadas às transições fazem alterar a regra que define se uma transição de uma RdP está ou não habilitada:

Regra 1 (Disparo de transição guardada) *Uma transição guardada duma RdPI 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.*

Adicionalmente, para representar as acções do controlador, tem de se associar os sinais de saída aos lugares ou transições, onde eles devem ser activados [10]. Um sinal de saída associado a um lugar representa um sinal do tipo Moore, enquanto que se a associação for entre uma transição e um sinal, este é do tipo Mealy [7].

Um dos problemas que as RdP exibem no seu comportamento assenta no assincronismo. Para obviar este problema, houve que proceder, novamente, a algumas alterações ao comportamento habitual das RdP. No presente trabalho, considera-se que os controladores, ao nível mais alto, trabalham sincronamente.

A solução adoptada assenta na seguinte filosofia: as transições habilitadas num dado instante, não disparam instantaneamente, mas mantêm-se habilitadas até que surja um pulso (uma transição activa) no sinal de relógio. Quando a transição no sinal de relógio surge, todas as transições habilitadas disparam e uma nova marcação é calculada. É, contudo, necessário garantir que em qualquer marcação alcançável, todas as transições habilitadas não estão em conflito. A uma RdPI que segue este princípio chama-se *Rede de Petri Síncrona Interpretada* (abreviadamente RdPSI).

Assim, foi adoptada a seguinte regra, definida em [12], e que pressupõe ou permite pressupor a existência de um sinal de relógio.

Regra 2 (Disparo simultâneo) *Todas as*

transições habilitadas, num dado instante, disparam simultaneamente.

4 APLICAÇÃO CONPAR

É fundamental que a especificação, baseada nas RdP, possa ser implementada. Se isso não se verificar, o processo de especificação será de utilização discutível; eventualmente, poderá servir para simulação. Embora esse uso seja de extrema importância, durante determinadas fases do projecto, é imprescindível a possibilidade de implementar fisicamente o sistema.

Neste contexto, a linguagem VHDL [13] mostra-se, verdadeiramente, uma solução a considerar. Primeiro, por se tratar de um *standard* que paulatinamente se tem imposto. Segundo, pelos diversos níveis de abstracção que admite. Finalmente, por permitir que um sistema seja simulado e sintetizado, desde que obedeça a determinadas restrições. A síntese corresponde, pois, a um processo que gera implementações físicas ou algo que tal permite obter (por exemplo, uma *netlist*).

Assim sendo, encontra-se presentemente em fase de desenvolvimento, como resultado de uma dissertação de mestrado [14], uma aplicação computacional (de nome ConPar), escrita em linguagem C, que permite traduzir directamente um controlador baseado em RdPSI para o equivalente em código VHDL.

Os sistemas são descritos a um nível alto, usando RdPSI. A descrição, neste primeiro protótipo, é conseguida usando uma linguagem textual (também de nome ConPar) construída para o efeito [14, 15]. Através de uma série de funções, algumas propriedades da RdPSI são verificadas: viva, segura, livre de conflitos e determinística. Posteriormente, um processo automático permite obter o código VHDL respeitante ao sistema modelado pela RdPSI. Esse código poderá ser então usado como entrada em ferramentas de simulação ou de síntese. O código VHDL gerado encontra-se ao nível fluxo de dados e é facilmente legível por um humano.

Para breve, prevê-se ainda a construção de uma interface gráfica que liberte o utilizador da tarefa de introduzir a RdPSI via texto, usando como plataforma inicial, o trabalho de Viegas [16].

Futuramente, está também prevista a possibilidade de simular e animar graficamente as RdPSI, utilizando o sistema *SCBA* [17].

5 EXEMPLO

Como exemplo da metodologia atrás apresentada, considere-se a RdPSI, apresentada na figura 2.

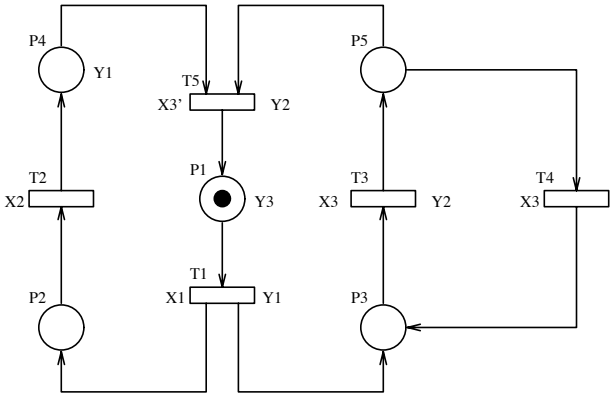

Figura 2: Controlador baseado numa RdPSI.

O respectivo código em linguagem ConPar é o apresentado de seguida:

```
.clock RELOGIO
.input X1 X2 X3
.output Y1 Y2 Y3

.part RdPSI
.place P1 P2 P3 P4 P5
.transition T1 T2 T3 T4 T5
.net
T1: P1 * X1 |- P2 * P3 * Y1;
T2: P2 * X2 |- P4;
T3: P3 * X3 |- P5 * Y2;
T4: P5 * X3 |- P3;
T5: P4 * P5 * !X3 |- P1 *Y2;
.MooreOutput
P1 |- Y3;
P4 |- Y1;
.marking P1
.e
```

Invocando o compilador ConPar, obtém-se o seguinte código VHDL:

```
-- VHDL file generated by ConPar version 1.0

ENTITY controller IS
  PORT (reset, x1, x2, x3, relogio : IN BIT;
        y1, y2, y3 : OUT BIT);
END controller;

ARCHITECTURE dataflow OF controller IS
```

```
-- Place Signals
  SIGNAL p1, Np1 : BIT;
  SIGNAL p2, Np2 : BIT;
  SIGNAL p3, Np3 : BIT;
  SIGNAL p4, Np4 : BIT;
  SIGNAL p5, Np5 : BIT;
-- Transition Signals
  SIGNAL t1 : BIT;
  SIGNAL t2 : BIT;
  SIGNAL t3 : BIT;
  SIGNAL t4 : BIT;
  SIGNAL t5 : BIT;

BEGIN
  PROCESS BEGIN
    WAIT UNTIL relogio'EVENT and relogio='1';
    IF reset='0' THEN
      p1 <= Np1;
      p2 <= Np2;
      p3 <= Np3;
      p4 <= Np4;
      p5 <= Np5;
    ELSE
      p1 <= '1';
      p2 <= '0';
      p3 <= '0';
      p4 <= '0';
      p5 <= '0';
    END IF;
  END PROCESS;

-- Dataflow description for transitions
  t1 <= p1 AND x1 AND NOT p3 AND NOT p2;
  t2 <= p2 AND x2 AND NOT p4;
  t3 <= p3 AND x3 AND NOT p5;
  t4 <= p5 AND x3 AND NOT p3;
  t5 <= p4 AND p5 AND NOT x3 AND NOT p1;

-- Dataflow description for next place markings
  Np1 <= t5 OR (p1 AND NOT t1);
  Np2 <= t1 OR (p2 AND NOT t2);
  Np3 <= t4 OR t1 OR (p3 AND NOT t3);
  Np4 <= t2 OR (p4 AND NOT t5);
  Np5 <= t3 OR (p5 AND NOT t5 AND NOT t4);

-- Ouput Signals Equations
  y1 <= t1 OR p4;
  y2 <= t5 OR t3;
  y3 <= p1;

END dataflow;
```

O compilador foi desenvolvido de forma a gerar código num dado subconjunto de especificação VHDL, que permite ser usado nas ferramentas de simulação e de síntese do pacote *Alliance* [18]. A vantagem desta possibilidade advém do facto de aquele pacote ser de domínio público. A diferença, relativamente ao código atrás apresentado, assenta basicamente na substituição do `PROCESS` por um `BLOCK`.

Referências

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

[2] Luís Gomes e A. Steiger-Garção. Especificação e Realização de Controladores utilizando Redes de Petri Coloridas e Sincronizadas integrando Lógica Imprecisa. *Revista Robótica e Automatização*, (10), Novembro 1992.

[3] Manuel Silva. Logical Controllers. In A. De Carli, editor, *IFAC Low Cost Automation: Techniques, Components and Instruments, Applications*, volume II, pp. F157–F166bis, Milão, Itália, Novembro 1989.

[4] C. A. Clare. *Design Logic Systems Using State Machines*. McGraw-Hill, 1973.

[5] Telemec. *O GRAFCET: diagrama funcional para automatismos sequenciais*. Telemec, 1982.

[6] M. Pearce. Report on Existing CAD Support for Programmable Devices. *Computer-Aided Engineering Journal*, pp. 160–6, Agosto 1991.

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

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

[9] Wolfgang Reisig. *Petri Nets - An Introduction*, *EATCS Monographs on Theoretical Computer Science*, volume 4. Springer Verlag, Heidelberg, Alemanha, 1985.

[10] Pierre Azema, Robert Valette, e 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.

[11] Luca Ferrarini e Claudio Maffezzoni. Designing Logic Controllers with Petri Nets. In H. A. Baker, editor, *5th IFAC Symposium Computer-Aided Design in Control Systems*, pp. 319–24, Julho 1991.

[12] James Pardey, Tomasz Kozlowski, Johnatan Saul, e Martin Bolton. State Assignments Algorithms for Parallel Controller Synthesis. *Proceedings of the IEEE International Conference on Computer Design*, pp. 316–9, 1992.

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

[14] João M. Fernandes. Redes de Petri e VHDL na Especificação de Controladores Paralelos. Dissertação de Mestrado, Dep. Informática, Universidade do Minho, Braga, Portugal, 1994. (em curso).

[15] Tomasz Kozlowski. Petri-net-based CAD tools for parallel controller synthesis. Dissertação de Mestrado, University of Bristol, Inglaterra, Março 1993.

[16] Paulo Viegas. Geração de Controladores de Diálogos Assíncronos Usando Redes de Petri. Dissertação de Mestrado, Dep. Informática, Universidade do Minho, Braga, Portugal, 1994. (em curso).

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

[18] Alain Greiner e François Pêcheux. ALLIANCE: A complete Set of CAD Tools for teaching VLSI Design. In *Third Eurochip Workshop on VLSI Design Training*, pp. 230–7, Grenoble, França, 1993.