REDES DE PETRI E VHDL NA PROTOTIPAGEM RÁPIDA DE SISTEMAS DIGITAIS

Ricardo J. Machado João M. Fernandes Alberto J. Proença
Dep. Informática, Escola de Engenharia, Universidade do Minho, 4710 Braga, Portugal
Email: {rmac,miguel,aproenca}@di.uminho.pt

Sumário

O objectivo principal deste artigo é exemplificar a utilização de uma metodologia de especificação de sistemas digitais, baseada em Redes de Petri orientadas por objectos, para obter de uma forma rápida e simplificada um protótipo em VHDL do sistema pretendido. É considerado para exemplificação um sistema digital, para o qual se efectua a especificação no modelo RdP-shobi e a geração automática de código VHDL. Este exemplo permite concluir àcerca da utilidade desta metodologia no projecto de sistemas digitais, suportado por princípios de orientação por objectos e por uma ferramenta de EDA desenvolvida propositadamente para o efeito.

1 INTRODUÇÃO

Nos sistemas digitais de maior complexidade e dimensão é possível identificar dois componentes distintos do sistema global: o controlador e o sistema controlado. Por vezes, a unidade de controlo possui um comportamento marcadamente paralelo, situação que tem sido modelada recorrendo a Redes de Petri (RdP) Síncronas e Interpretadas. Esta abordagem apresenta-se vantajosa relativamente à tradicional utilização de várias máquinas de estados locais a funcionar simultaneamente e com sinais comuns para sincronismo entre elas. No que diz respeito à modelação do sistema global, as poucas soluções existentes actualmente recorrem a técnicas em que ambas as partes são especificadas quase independentemente, não conseguindo integrar num único formalismo a representação do sistema global.

Foi desenvolvida uma extensão para as RdP Síncronas e Interpretadas, RdP-shobi, que possibilita a utilização de hierarquia nos modelos e o recurso a objectos para modelar o sistema controlado. Desta maneira, torna-se possível especificar o sistema digital global de uma forma estruturada e incremental. Basicamente, o modelo RdP-shobi substitui as tradicionais marcas por objectos. A invocação de métodos corresponde à leitura de sinais de entrada e à geração de sinais de controlo, que permite, simultaneamente, a modelação do comportamento do sistema controlado. A modelação de três controladores paralelos, bem como dos respectivos sistemas controlados, permitiu verificar que o modelo desenvolvido se apresenta adequado na especificação de sistemas digitais de uma forma hierárquica, modular e incremental.

A maior parte das ferramentas de CAD electrónico existente, para suporte ao projecto de sistemas digitais, disponibiliza linguagens de especificação de hardware, do tipo HDL, no entanto podem não permitir, directamente, a modelação de actividades concorrentes e cooperativas.

Foi definida a arquitectura de uma aplicação computacional, ferramenta SOFHIA, que suporta directamente o modelo proposto para as RdP e que possibilita uma especificação gráfica do sistema. Como solução para implementação da arquitectura definida, utiliza-se o ambiente SCBA [1], o que exigiu o mapeamento do RdP-shobi num modelo de computação baseado em agentes, o MCBA [2]. Uma vez que esta aplicação utiliza, igualmente, o ambiente ConPar [3], torna-se possível obter uma descrição em VHDL da unidade de controlo do sistema especificado.

2 O MODELO RdP-shobi

VHDL é uma linguagem standard IEEE para descrição de hardware que explora a concorrência na

especificação de sistemas digitais, permitindo a sua simulação e síntese. Por outro lado, as RdP são apropriadas para modelar e analisar formamente sistemas discretos de grande complexidade [4]. A abordagem, seguida neste trabalho, para implementar, em VHDL, controladores paralelos e síncronos, a partir das suas especificações baseadas em RdPs de alto-nível, recorre à única metodologia que modela eficientemente RdPs em VHDL [5].

O modelo de especificação, baseado em RdP Síncronas e Interpretadas (RdP-SI) [3], desenvolvido com o objectivo de suportar a utilização de hierarquia generalizada, bem como para possibilitar a modelação conjunta, quer da parte de controlo quer do sistema controlado, deu origem a um tipo de RdP. Este modelo foi chamado de RdP-shobi (RdP Síncronas, Hierárquicas, Orientadas por Objectos e Interpretadas) e incorpora todos os conceitos de sincronismo e interpretação básica do modelo RdP-SI, juntamente com novos conceitos de hierarquia no sistema global e com conceitos da modelação orientada por objectos, pelo que acrescenta funcionalidades ao modelo RdP-SI. Uma das consequências deste novo modelo consiste no suporte directo a estruturas hierárquicas quer na unidade de controlo quer no sistema controlado.

Desta maneira, algumas das características básicas deste novo modelo revelam a fusão numa única extensão de RdP de conceitos primeiramente definidos no âmbito das RdP Síncronas, RdP Hierárquicas, RdP Coloridas e RdP Orientadas por Objectos [6, 7].

As características mais importantes do modelo RdP-shobi são as seguintes:

- As marcas (tokens) representam objectos que modelam estruturas do sistema controlado. Assim, as variáveis de instância representam a informação que flui e é transformada ao longo do sistema controlado e os métodos dos objectos são a interface entre a unidade de controlo e o sistema controlado. As marcas podem, desta maneira, ser consideradas marcas coloridas em oposição às marcas incolores do modelo RdP-SI. Cada marca, colorida, modela uma estrutura, básica ou não, do sistema controlado.

- Os nodos (transições e lugares) invocam métodos às marcas (objectos), quando estas lá chegam. Contudo, só são invocados aqueles métodos que tem uma relação directa com os sinais de controlo de hardware, uma vez que existem métodos adicionais disponíveis na interface dos objectos que não são utilizados pela RdP. Estes métodos são invocados pelo software de simulação para ler e alterar o conteúdo da cada estrutura do sistema controlado num estado bem definido da RdP (estado do controlador).

- Cada arco possui uma ou mais cores que o associa aos tipos de objectos que por lá podem passar. Isto implica que passem a existir caminhos bem definidos na RdP que estão associados aos percursos que cada estrutura do sistema controlado (marca/objecto) efectua ao longo da mesma. Esta imposição simplifica a leitura estrutural da RdP, para além de limitar a capacidade de alguns lugares, uma vez que não é necessário que objectos que não são invocados num determinado caminho da rede o percorram sem necessidade.

- Podem ser utilizados macronodos e macromarcas para implementar a hierarquia no modelo. Os macronodos representam sub-redes de Petri e as macromarcas representam hierarquias de submarcas. Desta maneira, é possível encapsular macro-actividades num macronodo, assim como macro-estruturas do sistema controlado em macromarcas.

Seguem-se alguns conceitos necessários para compreender o exemplo apresentadas mais à frente: (1) rede de controlo: conjunto de nodos e arcos contíguos (esqueleto) da RdP-shobi que corresponde estruturalmente à RdP-SI, sem reiniciações, para especificar a unidade de controlo; (2) caminho de controlo: percurso definido por uma marca na rede de controlo; (3) nodos de controlo: nodos da rede de controlo; (4) arcos de controlo: arcos da rede de controlo; (5) caminho de fecho: percurso definido por uma marca fora da rede de controlo (podem surgir situações em que este caminho não necessite de existir); (6) nodos de fecho: nodos pertencentes a um caminho de fecho; (7) arcos de fecho: arcos pertencentes a um caminho de fecho; (8) ciclo de fecho: percurso definido por uma marca ao longo da RdP-shobi, que é composto pelo caminho de controlo e pelo caminho de fecho (caso este exista) e pode ser identificado através do rasto (tracking) da cor da marca associada a todos os arcos do ciclo; (9) rede global: RdP-shobi que especifica o sistema digital global, é composta pela rede de controlo, pelos caminhos de fecho e pelas marcas; (10) rede associada: RdP-SI estruturalmente correspondente à rede de controlo da rede global após as reiniciações para marcas incolores.

Neste artigo, o modelo RdP-shobi não é caracterizado de forma detalhada, pelo que a consulta de [8] poderá complementar algumas questões não abordadas.

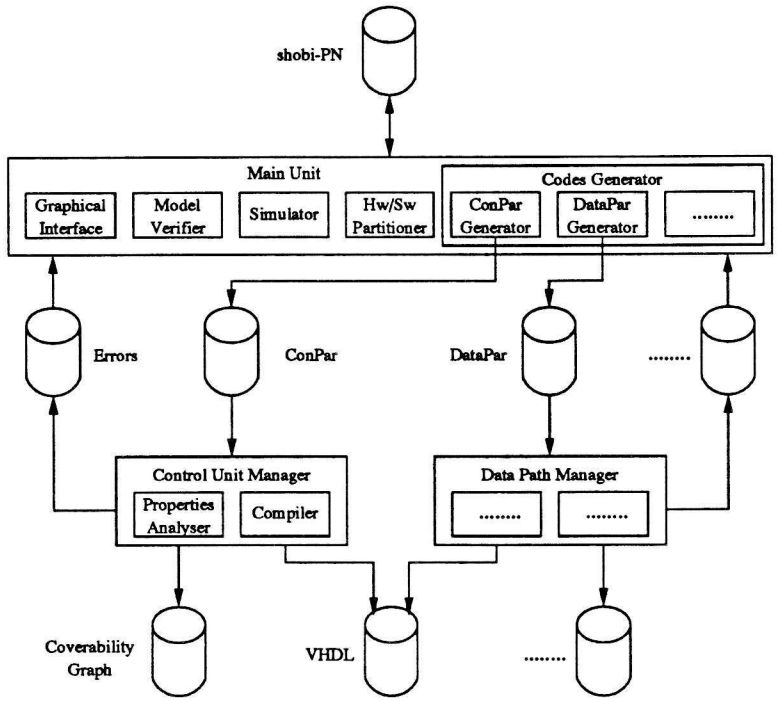

Figura 1: A ferramenta SOFHIA.

3 A FERRAMENTA SOFHIA

A utilização do modelo teórico RdP-shobi para especificar sistemas digitais, depende fortemente da existência de uma ferramenta de CAD que auxilie em muitos dos passos necessários para, a partir da especificação inicial do sistema, obter-se o código final para a sua síntese.

A ferramenta SOFHIA (Software for Hierarchical Architectures) (fig. 1) é adequada para especificar sistemas digitais de controlo usando o modelo RdP-shobi [9]. A partir de especificações hierárquicas em RdP-shobi, é gerado código VHDL, o que permite utilizar, em fases posteriores do projecto (simulação e síntese), ferramentas de CAD que aceitem como entrada especificações em VHDL.

Quando um sistema é constituido principalmente por componentes já modelados e reutilizáveis, a atenção do engenheiro de projecto pode centrar-se essencialmente no desenvolvimento das partes novas do projecto. Desta forma, os sistemas podem ser desenvolvidos mais rápida e facilmente. Esta abordagem conduz a uma nova disciplina conhecida como Prototipagem Rápida, muito vantajosa em situações em que os objectivos e requisitos do sistema não são completamente conhecidos na fase inicial do projecto. A ferramenta proposta suporta completamente todas as tarefas para a prototipagem rápida de sistemas digitais, a partir de especificações em RdP, incluindo: (1) as verificações formais de propriedades e consistência do modelo; (2) a possibilidade de simulação do funcionamento do sistema global; (3) a obtenção de código VHDL para síntese final do controlador.

4 EXEMPLO

Para verificar a adequabilidade do modelo proposto escolheram-se alguns casos práticos de áreas suficientemente diversificadas (interfaces de comunicações, microarquitectura de computadores e controlo industrial) para poder concluir da versatilidade do modelo na especificação de qualquer tipo de controlador digital paralelo [10]. Seleccionaram-se exemplos perfeitamente caracterizados e já analisados por outros autores, para validar inquestionavelmente o modelo RdP-shobi. Neste artigo é apresentado o controlador do Transputer Link Adaptor (circuito integrado IMS-C011 da INMOS) [11]. O IMS-C011 efectua a conversão bidireccional série-paralelo. Este circuito pode ser utilizado para interligar totalmente Transputers, controladores de periféricos INMOS, subsistemas de I/O e microprocessadores de diferentes famílias.

O Transputer Link Adaptor foi originalmente concebido para servir de interface entre uma rede de Transputers e um driver de um sistema de barramentos. As comunicações entre a rede de Transputers e o sistema externo baseado em barramentos exigem a realização de conversões série-paralelo

e paralelo-série na periferia da rede. O Transputer Link Adaptor (fig. 2) oferece uma interface full-duplex entre uma ligação série de Transputer e dois barramentos unidireccionais de 8 bits. Os dados provenientes do barramento de entrada (I0-I7) são multiplexados em LinkOut e os dados presentes em LinkIn são armazenados (latched) num registo de 8 bits para serem escritos no barramento de saída (Q0-Q7). Existe um protocolo de comunicações (handshake protocol) completo, baseado nos sinais IValid, IAck, QValid e QAck, que controla a transferência de dados de e para os barramentos [12].

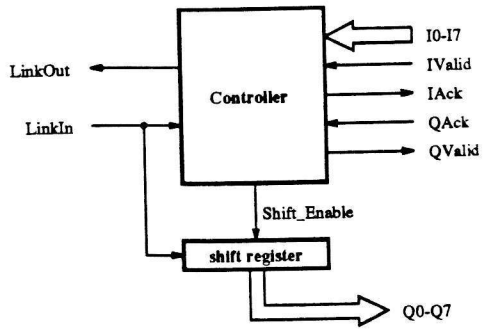

Figura 2: O Transputer Link Adaptor.

Uma rede de Transputers utiliza ligações (links) série para implementar as comunicações entre Transputers. São utilizadas duas linhas por cada ligação série, para permitir o fluxo de dados em full-duplex e os dados transmitidos pelas linhas de saída são sincronizados com sinais de confirmação (acknowledgment) enviados pelo Transputer destinatário. Desta maneira, dados e sinais de acknowledgment surgem em alternância (interleaved) em cada uma das linhas série.

Cada pacote de comunicações é composto por dois start bits, um octeto (byte) de dados e um stop bit. O transmissor espera por um pacote de acknowledgment, que é composto por um start bit seguido de um stop bit. Para permitir uma transferência contínua de dados, o receptor pode transmitir um pacote de acknowledgment logo que detecte o cabeçalho (header) de um pacote de dados, desde que não esteja ele também a transmitir um pacote de dados.

Na conversão paralelo-série (fig. 3.a), o driver do barramento coloca um octeto no barramento de entrada (I0-I7) e activa IValid. O link adaptor armazena os dados, multiplexa-os em LinkOut e espera por um pacote de acknowledgment em LinkIn. Quando este pacote de acknowledgment for recebido, o link adaptor activa IAck. O driver do barramento responde desactivando IValid e espera que o link adaptor desactive IAck.

Na conversão série-paralelo (fig. 3.b), o Transputer envia um pacote de dados por LinkIn. Quando o link adaptor detecta o cabeçalho do pacote transmitido, armazena os dados no shift register, cujas saídas estão ligadas ao barramento de saída (Q0-Q7), e activa QValid. Quando o sinal QAck for activado como resposta, pelo driver do barramento, o link adaptor confirma enviando ao Transputer um pacote de acknowledgment por LinkOut. Nessa altura, o link adaptor desactiva QValid e espera que QAck seja igualmente desactivado pelo driver do barramento.

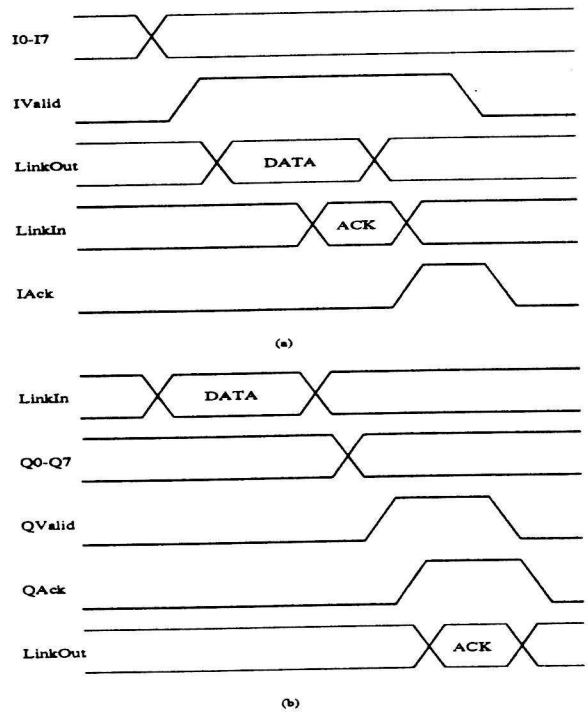

Figura 3: Diagramas Temporais: (a) conversão paralelo-série, (b) conversão série-paralelo.

Antes de construir a RdP-shobi para especificar as sequências de controlo, é necessário modelar o sistema controlado associado ao Transputer Link Adaptor, identificando os objectos existentes e definindo as suas variáveis e métodos de instância.

Analisando pormenorizadamente o sistema representado na fig. 2, é possível identificar três grupos distintos de objectos no sistema controlado, eles são: as ligações série (links), os barramentos de dados (buses) e o registo de deslocamento (shift register). De acordo com esta selecção é, então, necessário declarar e codificar as classes correspondentes aos objectos identificados.

Só depois da modelação do sistema controlado é que se torna possível especificar o controlador, através do desenho da RdP-shobi. Na RdP-shobi são utilizadas instâncias das classes descritas anteriormente, que definem ciclos de fecho ao longo da rede e onde cada instância é invocada com métodos existentes na sua interface.

Neste caso, vai ser necessário dispor de: (1) duas instâncias da classe link (Lin e Lout) para repre-

sentar os objectos ligação série de entrada (input link) e ligação série de saída (output link); (2) duas instâncias da classe bus (Bin e Bout) para representar os objectos barramento de entrada (input bus) e barramento de saída (output bus); (3) uma instância da classe shift_register (Reg) para representar o objecto registo de deslocamento (shift register). As classes estão codificadas na fig. 4.

```
CLASSE: link
VAR. INST.: BOOL: bit
METDS. INST.:
bool RD_BIT (BOOL level)
{ if (level == HIGH)
  then return (bit)
  else return (NOT bit)
}

void WR_BIT (BOOL level)
{ if (level == HIGH)
  then bit = HIGH
  else bit = LOW
}

CLASSE: bus
VAR. INST.: BOOL: bit[8], valid, ack
METDS. INST.:
bool RD_BIT (BOOL level, INT index)
{ if (level == HIGH)
  then return (bit[index])
  else return (NOT bit[index])
}
bool RD_VALID (BOOL level)
{ if (level == HIGH)
  then return (valid)
  else return (NOT valid)
}
bool RD_ACK (BOOL level)
{ if (level == HIGH)
  then return (ack)
  else return (NOT ack)
}

void WR_BIT (BOOL level, INT index)
{ if (level == HIGH)
  then bit[index] = HIGH
  else bit[index] = LOW
}
void WR_VALID (BOOL level)
{ if (level == HIGH)
  then valid = HIGH
  else valid = LOW
}
void WR_ACK (BOOL level)
{ if (level == HIGH)
  then ack = HIGH
  else ack = LOW
}

CLASSE: register
VAR. INST.: BOOL: bit[8], shift
METDS. INST.:
bool RD_BIT (BOOL level, INT index)
{ if (level == HIGH)
  then return (bit[index])
  else return (NOT bit[index])
}
bool RD_SHIFT (BOOL level)
{ if (level == HIGH)
  then return (shift)
  else return (NOT shift)
}

void WR_BIT (BOOL level, INT index)
{ if (level == HIGH)
  then bit[index] = HIGH
  else bit[index] = LOW
}
void WR_SHIFT (BOOL level, value)
{ if (level == LOW)
  then shift = LOW
  else shift = HIGH
       for (i=7;i=1;i--)
         bit[i] = bit[i-1]
       bit[0] = value
}
```

Figura 4: Classes para o Transputer Link Adaptor.

A RdP-shobi (fig. 5) para especificar o Transputer Link Adaptor consiste numa máquina de Mealy com 30 lugares, 35 transições e dois macrolugares, um para especificar a conversão paralelo-série (ParSer) e outro para especificar a conversão série-paralelo (SerPar).

Esta RdP pode ser dividida em quatro partes funcionalmente distintas, correspondendo três delas a ciclos de fecho completos: (1) o troço de rede (p1,t1,p2,t2,SerPar,t3) que executa a detecção do pacote de dados na porta série de entrada e efectua a conversão série-paralelo, (2) o ciclo de fecho de Bout (p29,t3,p13,t4,p14,t5,p15,t6,p16,t7) que gera o sinal de controlo de saída QValid e espera pelo sinal de acknowledgement QAck, de modo a completar a conversão série-paralelo; (3) o ciclo de fecho de Bin (p12,t8,ParSer,t9,p28,t10,p11,t11) que executa a conversão paralelo-série, bem como a sinalização para início de transmissão onde estão envolvidos os sinais IAck e Ivalid; (4) o ciclo de fecho de Lout (p17,t8,ParSer,t9,t5,p15,t6) que gere o acesso à porta série de saída. O único troço de rede que, dos quatro descritos, não define um ciclo de fecho completo, constitui parte do ciclo de fecho de Lin,

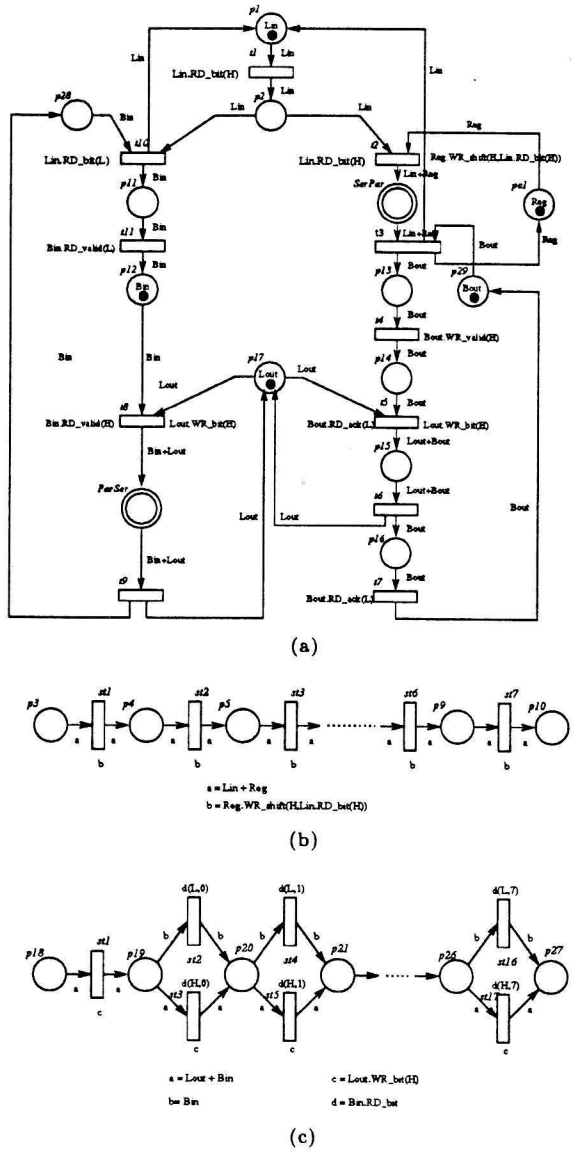

Figura 5: RdP-shobi do Transputer Link Adaptor: (a) RdP principal. (b) Macronodo de conversão série-paralelo. (c) Macronodo de conversão paralelo-série.

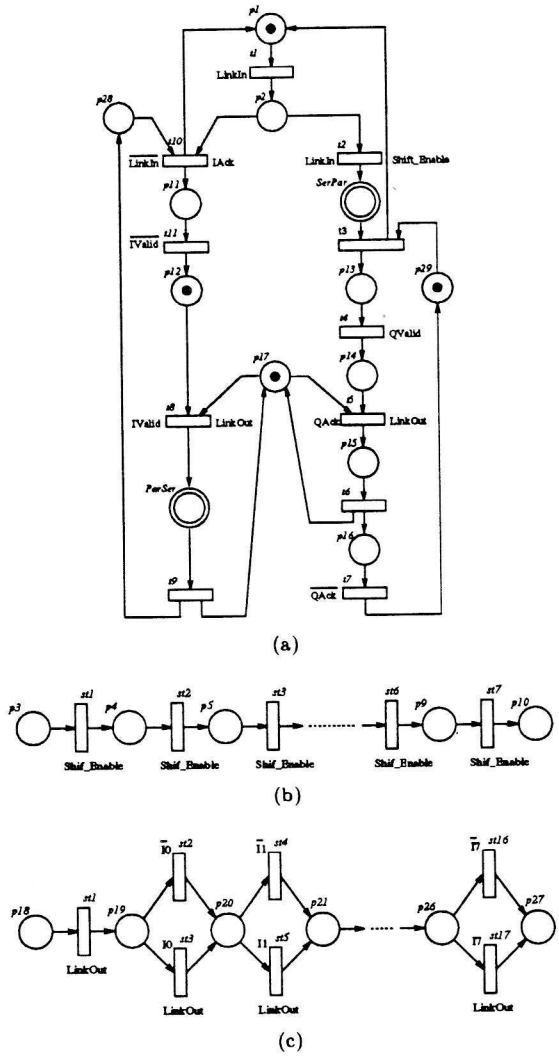

Figura 6: RdP-SI do Transputer Link Adaptor: (a) RdP principal. (b) Macronodo de conversão série-paralelo. (c) Macronodo de conversão paralelo-série.

uma vez que a outra parte do ciclo (p1,t1,p2,t10) possibilita a comutação para o modo de conversão paralelo-série. O objecto Reg é o único que possui um ciclo de fecho (pf1,t2,SerPar,t3) que é composto pelo caminho de controlo e pelo caminho de fecho, uma vez que, para todos os outros objectos presentes na RdP-shobi, os ciclo de fecho são somente constituídos pelos respectivos caminhos de controlo.

A RdP-shobi não possui declarações de nodos de reiniciação, pelo que, a rede de controlo é reiniciada com a marcação inicial de alguns dos lugares de controlo (p1,p12,p17,p29). Para obter a RdP-SI (fig. 6) do Transputer Link Adaptor, a partir da respectiva RdP-shobi, é necessário remover o único lugar de fecho existente (pf1), bem como os arcos que o ligam às transições de controlo t2 e t3, para obter a estrutura da rede associada. O lugar pf1 é somente utilizado para simulação do sistema controlado e não possui qualquer informação relativa à unidade de controlo. Em termos de interpretação da rede associada, só é necessário retirar todas as referências a cores e transformar as invocações dos métodos em sinais de hardware.

A especificação textual do controlador do sistema em notação CONPAR encontra-se listada na fig. 7. Um ficheiro, com código VHDL ao nível fluxo de dados, é gerado pelo módulo Compilador (Fig. 8). Foi usada uma opção de compilação, de modo a criar um comando BLOCK; alternativamente poder-se-ia ter criado um comando PROCESS.

5 CONCLUSÕES

Este artigo mostra que o modelo RdP-shobi é extremamente útil e eficiente na especificação de sistemas digitais de controlo. Este modelo é o único conhecido que utiliza RdP orientados por objectos para especificar a unidade de controlo paralela e o sistema controlado de uma forma integrada e modular. O modelo RdP-shobi apresenta um comportamento síncrono, técnicas de modelação orientadas por objectos e mecanismos hier'arquicos. Como resultado, esta nova metodologia suporta directamente estruturas hierárquicas quer na unidade de controlo quer no sistema controlado.

A análise de alguns exemplos de aplicação, incluindo o exemplo considerado neste artigo, demonstra que existe uma relação entre a estrutura das RdP e o tipo de abordagem seguida na especificação do sistema considerado. No caso do TLA, a RdP obtida reflecte uma abordagem "data-driven", porque s'o existe um caminho de fecho. Este tipo de RdP incorpora na rede de controlo as reiniciações dos objectos, sem a necessidade de incluir, total ou parcialmente, caminhos de fecho. As redes de controlo

```
.CLOCK clock
.INPUT I0 I1 I2 I3 I4 I5 I6 I7 LinkIn QACK IValid
.OUTPUT LinkOut IAck QValid ShiftEnable
<••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••>
.MACROPLACE PS(in0 in1 in2 in3 in4 in5 in6 in7, out)
.INTERFACE p18, p27
.PLACE p19 p20 p21 p22 p23 p24 p25 p26
.TRANSITION st1 st2 st3 st4 st5 st6 st7 st8 st9
        st10 st11 st12 st13 st14 st15 st16 st17
.NET
   st1: p18          |- p19 • out;
   st2: p19 • !in0   |- p20;
   st3: p19 •  in0   |- p20 • out;
   st4: p20 • !in1   |- p21;
   st5: p20 •  in1   |- p21 • out;
   st6: p21 • !in2   |- p22;
   st7: p21 •  in2   |- p22 • out;
   st8: p22 • !in3   |- p23;
   st9: p22 •  in3   |- p23 • out;
  st10: p23 • !in4   |- p24;
  st11: p23 •  in4   |- p24 • out;
  st12: p24 • !in5   |- p25;
  st13: p24 •  in5   |- p25 • out;
  st14: p25 • !in6   |- p26;
  st15: p25 •  in6   |- p26 • out;
  st16: p26 • !in7   |- p27;
  st17: p26 •  in7   |- p27 • out;
<••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••>
.MACROPLACE SP(in, out)
.INTERFACE p3, p10
.PLACE p4 p5 p6 p7 p8 p9
.TRANSITION st1 st2 st3 st4 st5 st6 st7
.NET
   st1: p3 |-  p4 • out;
   st2: p4 |-  p5 • out;
   st3: p5 |-  p6 • out;
   st4: p6 |-  p7 • out;
   st5: p7 |-  p8 • out;
   st6: p8 |-  p9 • out;
   st7: p9 |- p10 • out;
<••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••>
.PART macronet
.PLACE p1 p2 p11 p12 p13 p14 p15 p16 p17 p28 p29
        SerPar=SP(IO,ShiftEnable)
        ParSer=PS(IO I1 I2 I3 I4 I5 I6 I7, LinkOut)
.TRANSITION t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11
.PREDICATE predt8
.NET
   t1: p1 • LinkIn          |- p2;
   t2: p2 • LinkIn          |- SerPar • ShiftEnable;
   t3: SerPar • p29         |- p1 • p13;
   t4: p13                  |- p14 • QValid;
   t5: p14 • p17 • QACK     |- p15 • LinkOut;
   t6: p15                  |- p16 • p17;
   t7: p16 • !QACK          |- p29;
   t8: p12 • p17 • predt8   |- ParSer • LinkOut;
   t9: ParSer               |- p28 • p17;
  t10: p2 • p28 • !LinkIn   |- p1 • p11 • IAck;
  t11: p11 • !IValid        |- p12;
.PREDICATEDESCRIPTION
  predt8 • IValid • !p14;
.MARKING
   p1 p12 p17 p29
<••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••>
.E
```

Figura 7: Código CONPAR para o Transputer Link Adaptor.

```
-- Place Signals
  SIGNAL p1  : REG_BIT REGISTER;
  SIGNAL Np1 : BIT;
  ...
  SIGNAL p29  : REG_BIT REGISTER;
  SIGNAL Np29 : BIT;
  SIGNAL serpar_p3  : REG_BIT REGISTER;
  SIGNAL Nserpar_p3 : BIT;
  ...
  SIGNAL serpar_p10  : REG_BIT REGISTER;
  SIGNAL Nserpar_p10 : BIT;
  SIGNAL parser_p18  : REG_BIT REGISTER;
  SIGNAL Nparser_p18 : BIT;
  ...
  SIGNAL parser_p27  : REG_BIT REGISTER;
  SIGNAL Nparser_p27 : BIT;
-- Transition Signals
  SIGNAL t1 : BIT;
  ...
  SIGNAL t10 : BIT;
  SIGNAL serpar_st7 : BIT;
  ...
  SIGNAL serpar_st1 : BIT;
  SIGNAL parser_st17 : BIT;
  ...
  SIGNAL parser_st1 : BIT;

BEGIN
  PART : BLOCK (clock='1' AND NOT clock'STABLE)
  BEGIN
    p1 <= GUARDED Np1 WHEN reset='0' ELSE '1';
    p2 <= GUARDED Np2 WHEN reset='0' ELSE '0';
    ...
    p28 <= GUARDED Np28 WHEN reset='0' ELSE '0';
    p29 <= GUARDED Np29 WHEN reset='0' ELSE '1';
    serpar_p3 <= GUARDED Nserpar_p3 WHEN reset='0' ELSE '0';
    ...
    serpar_p10 <= GUARDED Nserpar_p10 WHEN reset='0' ELSE '0';
    parser_p19 <= GUARDED Nparser_p19 WHEN reset='0' ELSE '0';
    ...
    parser_p18 <= GUARDED Nparser_p18 WHEN reset='0' ELSE '0';
  END BLOCK;
-- Dataflow description for transitions
  t1 <= NOT p2 AND linkin AND p1;
  ...
  t8 <= NOT parser_p18 AND p17 AND p12 AND (ivalid AND NOT p14);
  ...
  serpar_st7 <= serpar_p9 AND NOT serpar_p10;
  ...
  serpar_st1 <= serpar_p3 AND NOT serpar_p4;
  parser_st17 <= parser_p26 AND i7 AND NOT parser_p27;
  ...
  parser_st1 <= parser_p18 AND NOT parser_p19;
-- Dataflow description for next place markings
  Np1 <= t10 OR t9 OR (p1 AND NOT t1);
  Np2 <= t1 OR (p2 AND NOT t10 AND NOT t2);
  ...
  Np29 <= t7 OR (p29 AND NOT t9);
  Nserpar_p3 <= t2 OR (serpar_p3 AND NOT serpar_st1);
  ...
  Nserpar_p10 <= serpar_st7 OR (serpar_p10 AND NOT t3);
  Nparser_p18 <= t8 OR (parser_p18 AND NOT parser_st1);
  ...
  Nparser_p27 <= parser_st16 OR parser_st17 OR (parser_p27 AND NOT t9);
-- Output Signals Equations
  linkout <= t8 OR t5 OR parser_st1 OR parser_st3 OR ... OR parser_st17;
  iack <= t10;
  qvalid <= t4;
  shiftenable <= t2 OR serpar_st1 OR ... OR serpar_st7;
---- ASSERT Commands
-- Transitions in conflict
  ASSERT NOT (t10 AND t3)
    REPORT "output place p1 overflows (transitions t10,t3)"
    SEVERITY ERROR;
  ASSERT NOT (t9 AND t6)
    REPORT "output place p17 overflows (transitions t9,t6)"
    SEVERITY ERROR;
  ASSERT NOT (t8 AND t5)
    REPORT "t8 and t5 are in conflict (input place p17)"
    SEVERITY ERROR;
-- No Enabled Transitions
  ASSERT NOT (t1='0' AND .. AND serpar_st7='0' AND ... )
    REPORT "Petri Net may be deadlocked"
    SEVERITY WARNING;
END dataflow;
```

Figura 8: Código VHDL gerado para o Transputer Link Adaptor.

"data-driven" podem ser totalmente fechadas, ou seja, não possuem nodos de reiniciação. Por outro lado, as redes de controlo "control-driven" têm de ser abertas. Esta análise mostra que as RdP-shobi seguem directamente uma abordagem do tipo "data-driven" na especificação de sistemas digitais paralelos e suporta adequadamente a especificação, quer na unidade de controlo quer no sistema controlado, para a prototipagem rápida de sistemas.

Referências

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

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

[3] João Miguel Fernandes and Alberto José Proença. Redes de Petri na Especificação e Validação de Controladores Paralelos. In *1o. Encontro Nacional do Colégio de Engenharia Electrotécnica*, pp. 113–8, Ordem dos Engenheiros, Lisboa, Portugal, Maio 1994.

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

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

[6] Charles Lakos. The Object Orientation in Object Petri Nets. In *1st Workshop on Object-Oriented Programming and Models of Concurrency*, Turim, Itália, Junho 1995.

[7] Kurt Jensen. An Introduction to the Theoretical Aspects of Coloured Petri Nets. Relatório Técnico, Comp. Science Dept, Aarhus University, Dinamarca, Agosto 1994.

[8] Ricardo Jorge Machado. Hierarquia em Redes de Petri Orientadas por Objectos na Especificação de Sistemas Digitais. Tese de Mestrado, Dep. Informática, Universidade do Minho, Braga, Portugal, Novembro 1996.

[9] Ricardo J. Machado, João M. Fernandes, and Alberto J. Proença. SOFHIA: A CAD Environment to Design Digital Control Systems. In *Proceedings of the XIII IFIP Conference on Computer Hardware Description Languages and Their Applications (CHDL'97)*, Toledo, Espanha, Abril 1997. Chapman & Hall.

[10] Ricardo J. Machado, João M. Fernandes, and Alberto J. Proença. Specification of Industrial Digital Controllers with Object-Oriented Petri Nets. In *IEEE International Symposium on Industrial Electronics (ISIE'97)*, Guimarães, Portugal, Julho 1997.

[11] R. Taylor. Transputer Communication Link. *Microprocessors and Microsystems*, 10(4):211–5, 1986.

[12] João M. Fernandes, Marian Adamski, and Alberto J. Proença. VHDL Generation from Hierarchical Petri Net Specifications of Parallel Controller. *IEE Proceedings: Computers and Digital Techniques*, 1997. A publicar.