LINGUAGENS DE DESCRIÇÃO DE HARDWARE PARA SUPORTE À INTEGRAÇÃO DO PROCESSO DE PROJETO EM AMPLO

por

Flávio Rech Wagner
Carla M. Dal Sasso Freitas
Lia Goldstein Golendziner*

RP nº 65 MARÇO/1987

Nota técnica do projeto "Banco de Dados e Ferramentas para CAD de Sistemas Digitais"

*trabalho realizado com o apoio do CNPq

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha 99
90.210-Porto Alegre-RS-Brasil
Telex (051) 2680 Tel. (0512) 21.8499

Endereço para Correspondência:
UFRGS/CPGCC/Biblioteca
Caixa Postal 1501
90.001-Porto Alegre-RS-Brasil

Linguagens: Descrição; Hardware
Projeto: Sistemas digitais

RESUMO

Este relatório apresenta as características das linguagens de descrição de hardware existentes no sistema AMPLO que suportam a integridade do processo de projeto de sistemas digitais. Em especial, é completamente descrita a linguagem REDES, que permite a descrição de sistemas como redes de agências hierarquizadas, contendo agências descritas em diferentes níveis de abstração.

PALAVRAS-CHAVE: projeto de sistemas digitais, linguagens de descrição de hardware, redes de agências, descrição estrutural de sistemas digitais

ABSTRACT

This report presents the features of the hardware description languages implemented in the AMPLO system, that support the integrity of the digital systems design process. In particular, the REDES language is completely described. It allows the description of systems as nets of hierarchical agencies, which can contain agencies described at different abstraction levels.

KEYWORDS: digital systems design, hardware description languages, nets of agencies, structural description of digital systems

SUMARIO

1. INTRODUÇÃO

O sistema AMPLO [WAG86c,WAG87c] é um ambiente integrado para o projeto lógico de sistemas digitais em desenvolvimento na UFRGS. Este ambiente possui ferramentas que permitem a descrição e validação de sistemas digitais em três níveis de projeto: sistema, transferência entre registradores (RT) e portas lógicas. Para cada um destes níveis existe uma linguagem de descrição de sistemas, respectivamente LASSO [BOR79], KAPA [WAG87a] e NILO [WAG87b].

Sistemas são descritos de forma modular e hierarquizada como redes de agências [WAG84a,b], através da linguagem REDES, que é objeto deste relatório. Uma agência pode ser descrita de forma puramente estrutural, como uma composição de outras agências, utilizando-se a linguagem REDES, ou através de um comportamento especificado com uma das linguagens LASSO, KAPA ou NILO. Esta metodologia de projeto é apresentada em [WAG86a,c,WAG87c].

Todas as quatro linguagens suportadas pelo ambiente possuem formas gráfica e textual que são completamente equivalentes entre si [WAG86b]. Isto permite que a descrição de uma agência possa ser feita de forma textual, via editor de textos convencional e compilador, ou de forma gráfica, via um editor especializado para a linguagem. Tanto o compilador como o editor gráfico geram uma mesma estrutura de dados interna, que é independente da forma de entrada, exceto pelas informações geométricas manipuladas pelo editor. Esta representação interna é armazenada em uma base de dados unificada, de onde pode ser recuperada para a construção de modelos simuláveis.

O sistema AMPLO garante uma efetiva integração do processo de projeto através de todos os níveis de descrição graças às seguintes características:

- integração de todos os dados de projeto numa base de dados única de forma íntegra e não redundante;
- uniformidade na interface do usuário, obtida via uma linguagem de comandos unificada para todo o sistema e via a padronização da sintaxe e da semântica das operações disponíveis em todos os editores gráficos;
- separação explícita entre a descrição da estrutura, representada como uma rede de agências através da linguagem REDES, e do comportamento, representado numa das linguagens LASSO, KAPA ou NILO;
- uniformidade na sintaxe e na semântica das construções, existentes em todas as linguagens, destinadas á descrição da interface das agências.

Este relatório deve explorar estas duas últimas características. No capítulo 2 será apresentado de que forma a separação entre estrutura e comportamento suporta a integridade do processo de projeto, enquanto o capítulo 3 mostrará como esta separação se reflete nas linguagens comportamentais (LASSO, KAPA e NILO). O capítulo 4 introduz os conceitos de alternativas e versões de projeto, que se refletem na linguagem REDES, que está completamente descrita no capítulo 5. O capítulo 6, finalmente, abordará questões relativas aos tipos de dados que podem ser associados aos sinais de interface de agências descritas em diferentes níveis.

2. SEPARAÇÃO ENTRE ESTRUTURA E COMPORTAMENTO

Todo sistema digital em AMPLO é representado como uma agência. Uma agência pode ser descrita de uma de duas formas, puramente estrutural ou puramente comportamental. Uma descrição de uma agência pode criar um novo tipo de agência a ser armazenado na base de dados. Ocorrências deste tipo podem posteriormente serem utilizadas dentro da descrição de novos tipos. Toda definição de um novo tipo, seja a descrição feita de forma estrutural ou comportamental, inclui a descrição da interface da agência.

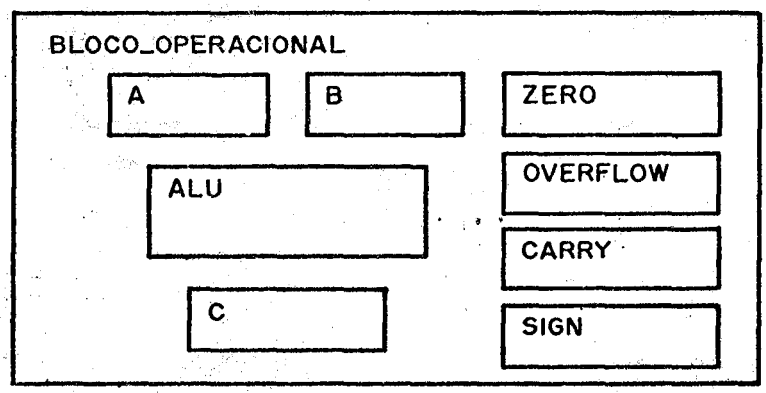
Numa descrição puramente estrutural, um tipo de agência é definido como uma rede de agências, na qual são apenas indicadas as interconexões entre os sinais de interface das agências. Cada agência desta rede é uma ocorrência de um tipo definido na base de dados. Nenhuma indicação é dada a respeito da função executada pelas agências. Descrições estruturais são feitas através da linguagem REDES. Esta forma de representação de agências permite definições hierarquizadas sem limitação de níveis para a hierarquia.

Numa descrição puramente comportamental, um tipo de agência é definido através da sua função interna, sendo esta descrita numa das linguagens comportamentais (LASSO, KAPA ou NILO). Apenas primitivas da linguagem escolhida podem ser utilizadas para a descrição da função da agência, não sendo possível a utilização de uma ocorrência de um tipo de agência definido na base de dados.

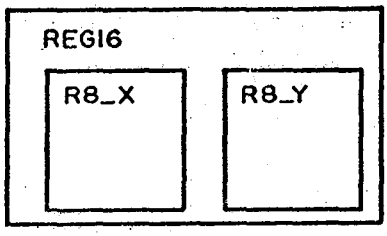
O exemplo abaixo esclarece a metodologia acima descrita. O sistema digital a ser descrito é um bloco operacional de um microprocessador, que na primeira etapa do projeto é definido como uma agência BLOCO_OPERACIONAL. Esta agência é descrita estruturalmente como sendo uma rede composta por ocorrências dos tipos de agências REG16, ARITH_LOGIC_UNIT e REGØ1:

- ocorrências A, B e C de REG16;
- ocorrência ALU de ARITH_LOGIC_UNIT; e
- ocorrências ZERO, OVERFLOW, CARRY e SIGN de REGØ1.

Nesta descrição inicial de BLOCO_OPERACIONAL, nada é dito acerca da função de suas sub-agências. Apenas são indicadas as interconexões entre seus sinais de interface, assim como as ligações destes sinais com os sinais definidos para a interface de BLOCO_OPERACIONAL.

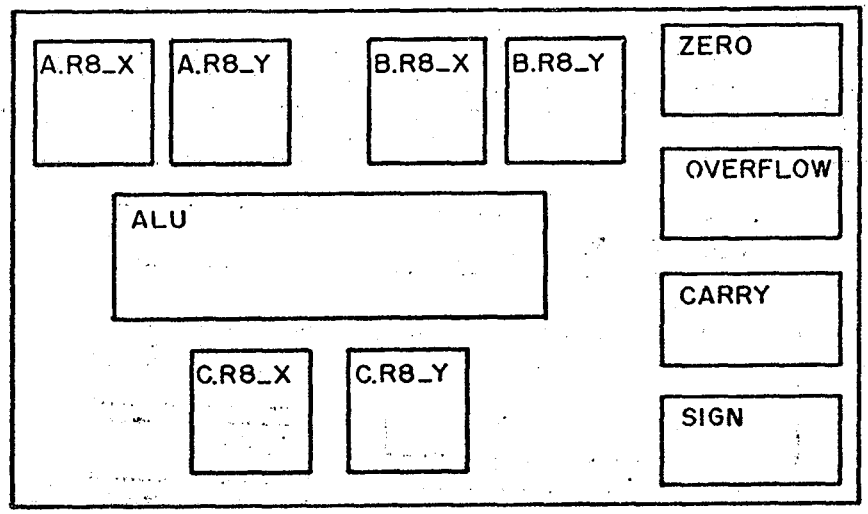


Numa segunda etapa, o projetista passa à descrição dos tipos de agências REG_16, ARITH_LOGIC_UNIT e REGØ1. Os tipos ARITH_LOGIC_UNIT e REGØ1 são descritos comportamentalmente. ARITH_LOGIC_UNIT è descrita no nivel de sistema, através da linguagem LASSO, enquanto REGØ1 è descrita no nivel de transferência entre registradores, através da linguagem KAPA. Nestas descrições, apenas primitivas das linguagens LASSO e KAPA são utilizadas, e elas não podem conter nenhuma ocorrência de outros tipos de agências definidos na base de dados. A agência REG16, por sua vez, è descrita estruturalmente como uma rede composta por duas ocorrências do tipo de agência REG8, denominadas R8_X e R8_Y.



Num último passo do projeto, o tipo de agência REG8 è descrito comportamentalmente no nivel RT, através da linguagem KAPA. Esta descrição não contem nenhuma ocorrência de outros tipos de agências definidos na base de dados.

O projetista poderia agora construir um modelo de simulação para a agência BLOCO_OPERACIONAL. Um modelo simulável deve ser uma rede de agências, que contenha apenas ocorrências de tipos de agências descritos comportamentalmente [WAG86a,87c]. De acordo com os passos de projeto acima descritos, o modelo seria uma rede contendo seis ocorrências de REG8 (que seriam denominadas hierarquicamente de A.R8_X, A.R8_Y, B.R8_X, B.R8_Y, C.R8_X e C.R8_Y), além das ocorrências ALU, ZERO, OVERFLOW, CARRY e SIGN. Como neste modelo uma agência (ALU) está descrita em LASSO, enquanto as demais estão descritas em KAPA, deve ser utilizado um simulador multi-nível [WAG86a].



Neste exemplo, nada foi dito a respeito de como declarar os sinais de interface das agências, nem sobre a forma de expressar as interconexões entre sinais de interface de duas agências conectadas entre si. Isto será visto nos capitulos seguintes.

A integridade do processo de projeto è garantida pela base de dados e pelas ferramentas de projeto. No exemplo acima, que descreveu um processo "top-down" de projeto, ocorrências de agências são utilizadas antes que os tipos respectivos estejam definidos. Como será visto mais adiante, a interface destes tipos de agências è definida simultaneamente com o uso das ocorrências. Na etapa posterior do projeto, quando os tipos são definidos através de uma estrutura (caso de REG16) ou de um cpomportamento (demais casos), o editor gràfico garante que a interface do tipo permanecerá consistente com a interface da ocorrência. AMPLO permite também um processo "bottom-up" de projeto no qual a integridade seja assegurada automaticamente.

3. ESTRUTURA DAS LINGUAGENS

Todas as linguagens do sistema AMPLO (REDES, LASSO, KAPA e NILO) seguem uma mesma sintaxe e semântica para as partes comuns a elas, que são:

1) Designação da descrição da agência, identificada pela construção

agency <designação_descrição>.

2) Declaração do nivel no qual a agência está descrita, identificada pela construção

level = <nome_linguagem>,

onde <nome_linguagem> è LASSO, KAPA ou NILO. Mesmo no caso de uma descrição estrutural em REDES, è necessàrio declarar um destes três niveis para a agência. Este aspecto será abordado nos capitulos 5 e 6.

3) Declaração da interface da agência, introduzida pela palavra-chave interface.

4) Introdução à declaração do comportamento da agência, identificada pela palavra-chave behavior (para o caso das linguagens comportamentais LASSO, KAPA e NILO), ou à declaração da estrutura da agência, identificada pela palavra-chave structure (para a linguagem REDES).

A declaração da interface de uma agência segue o modelo abaixo:

```
in  <desig_sinal>, ..., <desig_sinal> : <tipo_sinal> ;
    |
    <desig_sinal>, ..., <desig_sinal> : <tipo_sinal> ;
out <desig_sinal>, ..., <desig_sinal> : <tipo_sinal> ;
    |
    <desig_sinal>, ..., <desig_sinal> : <tipo_sinal> ;
inout <desig_sinal>, ..., <desig_sinal> : <tipo_sinal> ;
```

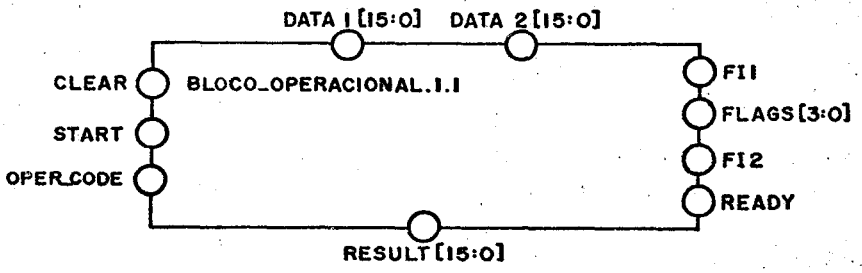
No modelo acima, <desig_sinal> é
<nome_sinal> <dimensão_sinal>,
onde <dimensão_sinal> pode ser omitida, se o sinal tiver uma largura de 1 bit, ou pode ser
[n1 : n2],
caso o sinal tiver uma largura de n bits, sendo n1 a designação do bit mais significativo e n2 a do bit menos significativo.

As 3 listas de sinais, iniciadas pelas palavras-chaves in, out e inout, identificam respectivamente os sinais de entrada, saída e bidirecionais da interface, e podem ser declaradas em qualquer ordem. É também possível a declaração de qualquer número de listas iniciadas por cada uma destas palavras-chave. É ainda possível a declaração de qualquer número de sub-listas de sinais com um mesmo <tipo_sinal>.

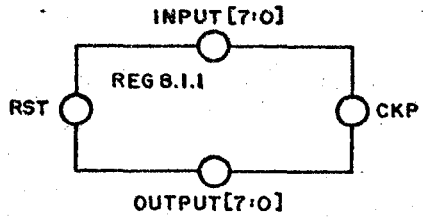
Para cada uma das linguagens LASSO, KAPA e NILO há um repertório de tipos de dados que podem ser associados aos sinais de interface. Este assunto será abordado no capítulo 6.

Voltando ao exemplo do bloco operacional do capítulo anterior, as figuras abaixo ilustram a estrutura geral das linguagens. Vemos parte da declaração estrutural da agência BLOCO_OPERACIONAL e parte da declaração comportamental da agência REG8.

```
agency  BLOCO_OPERACIONAL.1.1
level = LASSO
interface
     in  CLEAR, START : control;
         DATA1 [15:0], DATA2 [15:0] : terminal;
         OPER_CODE : integer;
         FI1, FI2 : clock;
     out  READY : control;
         RESULT [15:0], FLAGS [3:0] : terminal;
structure
     |
end
```




```
agency  REG8.1.1
level = KAPA
interface
     in  INPUT [7:0], RST : terminal;
         CKP : clock;
     out  OUTPUT [7:0] : terminal;
behavior
     |
end
```



Os algarismos após o nome da agência indicam os números da alternativa e versão de projeto destas descrições (ver capítulo seguinte).

4. ALTERNATIVAS E VERSÕES DE PROJETO

AMPLO permite ao projetista de hardware a criação de diversas alternativas e versões de projeto de um mesmo sistema ou de uma mesma agência.

Alternativas de uma agência diferem entre si por terem diferentes definições para a interface. Alterações na interface que criam uma nova alternativa de projeto para uma agência são:

- inclusão ou remoção de sinais de interface;
- mudança no nome de um sinal de interface;
- mudança na largura em bits de um sinal de interface;
- mudança do tipo de dados de um sinal de interface;
- mudança dos atributos geométricos da interface (proporção entre largura e altura, posição dos sinais na envoltória).

Uma mesma alternativa de uma agência pode estar definida em diversos níveis de descrição. Estas definições devem ter mesmo número de sinais e mesmos atributos geométricos para a interface. Cada um dos sinais da interface deve ter, em todas as definições, mesmo nome e mesma largura em bits. Os tipos de dados associados a um sinal nas diversas definições de uma mesma alternativa devem ser compatíveis entre si. A questão da compatibilidade entre tipos de dados de níveis diversos é abordada no capítulo 6.

A cada alternativa de uma agência podem estar associadas diferentes **versões** de projeto. Versões de uma mesma alternativa têm portanto em comum uma mesma definição para a interface (ou definições compatíveis, se feitas em níveis diversos). Cada alternativa de uma agência pode ter qualquer número de versões, tanto estruturais como comportamentais, e estas versões podem ter sido definidas em quaisquer níveis de projeto.

Alternativas e versões de projeto são identificadas por números inteiros. Uma alternativa de uma agência é portanto designada por

<nome_agência> . <número_alternativa>,

enquanto uma versão é identificada por

<nome_agência> . <número_alternativa> . <número_versão>.

Cada tipo de agência criado na base de dados corresponde a uma alternativa de projeto de uma agência. Qualquer versão desta alternativa pode ser utilizada no lugar de uma ocorrência deste tipo, já que todas as versões de uma mesma alternativa têm a mesma interface, o mesmo não ocorrendo para versões de alternativas diferentes.

5. APRESENTAÇÃO DA LINGUAGEM REDES

5.1 Estrutura geral da descrição

Uma descrição da estrutura de uma agência na linguagem REDES contèm as 4 partes jà apresentadas no capitulo 3:

- designação da descrição da agência;
- declaração do nivel no qual a agência està descrita;
- declaração da interface da agência; e
- declaração da estrutura interna da agência.

As 3 primeiras partes seguem a mesma sintaxe e semântica utilizadas em todas as linguagens do sistema AMPLO.

A designação da descrição da agência tem a forma

<nome_agência> . <nro.alternativa>. <nro.versão>

Todo tipo de agência armazenado na base de dados è uma alternativa de uma certa agência, identificada por

<nome_agência> . <nro.alternativa>.

Se o número de alternativa for um número jà existente, o compilador e o editor gràfico conferem se a definição da interface desta descrição coincide com a da alternativa indicada, armazenada na base de dados. Se esta for a primeira versão para esta alternativa, um novo tipo de agência è criado. Se o número de versão for um número jà existente, esta versão substitui a anterior.

Embora em principio REDES seja uma linguagem para a descrição de estrutura, sem qualquer referência ao comportamento interno das agências, e portanto ao nivel de descrição a ser adotado para cada agência da rede, è necessàrio declarar o nivel no qual a rede està descrita (construção "level"). Esta declaração permite ao sistema AMPLO a verificação da integridade dos tipos de dados declarados para os sinais de interface, na construção "interface", e para os sinais internos da agência, na construção "declare" (veja seção 5.3 adiante).

A declaração da estrutura interna de uma agência corresponde a uma descrição de uma rede de agências. Cada agência nesta rede è uma ocorrência de um tipo, jà definido na base de dados ou a definir simultaneamente com a descrição que usa sua ocorrência. A descrição da estrutura interna da agência, introduzida pela palavra-chave structure, utiliza apenas 3 construções: external, declare e use. Esta estrutura da linguagem REDES è fortemente baseada na linguagem CASCADE [BOR85].

Como todas as demais linguagens de AMPLO, REDES tem formas textual e gràfica que são completamente equivalentes entre si. A apresentação a seguir se concentrarà mais nos aspectos formais das descrições textuais. A forma gràfica serà mostrada de maneira apenas informal. Outro relatòrio tècnico apresentarà as operações do editor gràfico REDES que devem ser utilizadas para a descrição de agências.

5.2 External

A construção external serve à declaração dos tipos de agências cujas ocorrências serão utilizadas na descrição corrente. Ela é uma lista

external <declar_tipo>, ... , <declar_tipo> ;

onde <declar_tipo> pode ser

<designação_tipo>,

ou

<designação_tipo> (<declaração_nível> ,
<declaração_interface>).

No primeiro caso, no qual o tipo de agência a ser utilizado é declarado sem a interface, supõe-se que este tipo já estava previamente definido na base de dados. A declaração da interface do tipo da agência será então buscada na base para verificação de consistência com a interface utilizada pelas ocorrências deste tipo. Esta designação de tipo é portanto apropriada a um processo "bottom-up" de projeto, no qual os tipos de agências são sempre definidos e testados antes do uso de suas ocorrências.

No segundo caso, a interface de um tipo de agência é declarada simultaneamente com a designação deste tipo dentro de uma descrição REDES. A declaração da interface, colocada entre parênteses, segue exatamente a mesma sintaxe apresentada no capítulo anterior, comum a todas as linguagens de AMPLO, utilizando as construções in, out e inout. Este caso é portanto mais adequado para um processo "top-down" de projeto, no qual tipos de agências são utilizados antes que se detalhe a sua estrutura ou comportamento interior.

As duas formas de designação de tipos de agências (com ou sem a declaração da interface) podem ser utilizadas simultaneamente numa construção "external".

Abaixo segue uma descrição da agência BLOCO_OPERACIONAL antes introduzida, na qual os tipos REGØ1.1 e ARITH_LOGIC_UNIT.1 são considerados como já conhecidos previamente, enquanto a interface do tipo REG16.1 é declarada dentro da própria descrição que utiliza sua ocorrência.

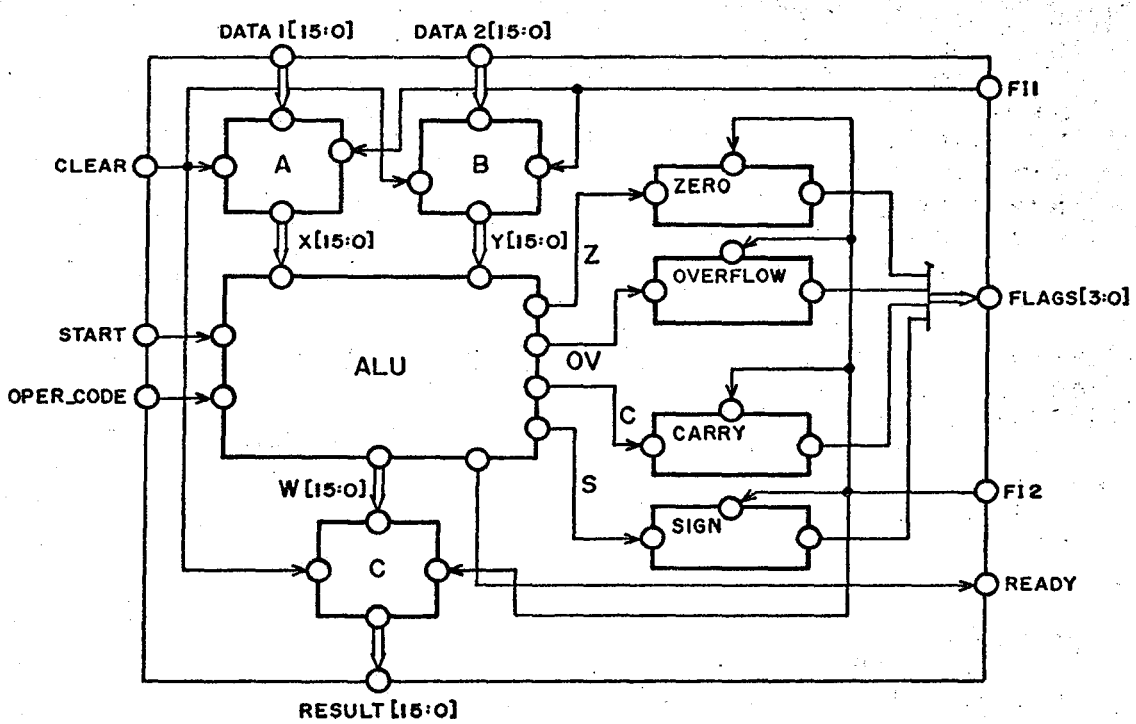
```
agency  BLOCO_OPERACIONAL.1.1
    |
structure
external  ARITH_LOGIC_UNIT.1, REGØ1.1,
   REG16.1 ( level = KAPA
             interface
               in DATA_IN [15:Ø], RESET : terminal;
                    CK_PULSE : clock;
               out DATA_OUT [15:Ø] : terminal );
```

5.3 Declare

A construção "declare" serve à declaração dos sinais que são internos a uma agência X descrita estruturalmente. Sinais internos são aqueles que não estão conectados a nenhum sinal de interface da agência X, correspondendo à ligação entre sinais de interface das sub-agências que compõem a rede que descreve X.

No exemplo abaixo, que mostra a agência BLOCO_OPERACIONAL.1 como sendo composta por ocorrências dos tipos REG16.1, ARITH_LOGIC_UNIT.1 e REGØ1.1, existem vários sinais internos. Na forma gráfica de REDES, fica claro quais sinais de interface das ocorrências A, B, C, ALU, ZERO, OVERFLOW, CARRY e SIGN estão ligados entre si através destes sinais internos.

```
agency BLOCO_OPERACIONAL.1.1
   |
structure
      external REG16.1, REGØ1.1, ARITH_LOGIC_UNIT.1;
      declare
        X [15:Ø], Y [15:Ø], W [15:Ø], Z, C, OV, S : terminal;
   |
end
```



5.4 Use

A construção "use" permite simultaneamente

- a instanciação dos tipos de agências declarados previamente na construção "external", e

- a conexão dos sinais de interface de cada uma das ocorrências criadas com os sinais de interface e sinais internos da agência englobante.

A construção "use" tem a forma

use <instanciação>, ..., <instanciação> ;

onde <instanciação> é

<tipo_agência> <nome_ocorrência> (<lista_conexões>).

A designação do tipo da agência deve seguir a mesma forma " <nome_agência> . <número_alternativa> ", também utilizada na construção "external". Opcionalmente, no entanto, o projetista pode escolher, já no momento da descrição estrutural, qual das versões da alternativa declarada deverá ser usada, seguindo neste caso o formato

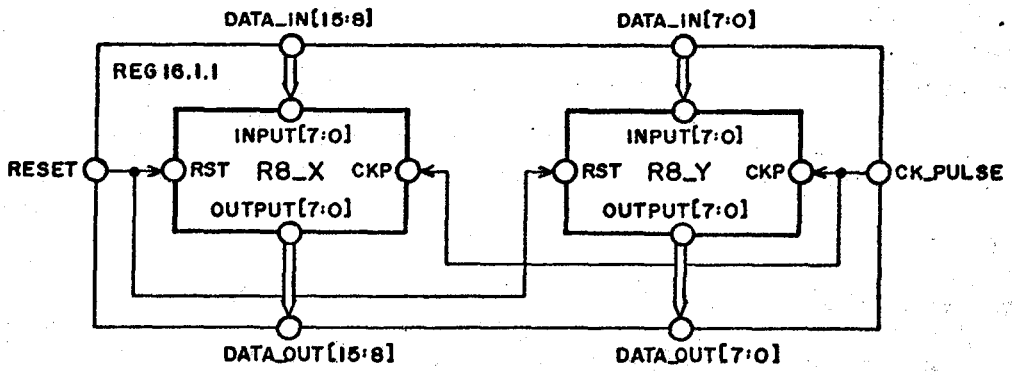
<nome_agência> . <número_alternativa> . <número_versão>.

Esta escolha da versão não tem nenhuma influência sobre o processo de consistência efetuado pelo compilador ou pelo editor gráfico, já que todas as versões de uma mesma alternativa têm interfaces idênticas ou equivalentes. A escolha pode ser feita apenas para simplificar o processo de construção de um modelo simulável, durante o qual normalmente o usuário de AMPLO deve optar por uma versão de cada ocorrência utilizada em descrições estruturais [WAG86c,87c].

A lista de conexões é uma lista de sinais declarados na agência englobante, sejam eles sinais internos ou da interface. Estes sinais devem ser listados na mesma ordem dos sinais de interface da ocorrência aos quais cada um deles está conectado.

O exemplo abaixo mostra a descrição estrutural completa da versão REG16.1.1. A comparação entre as formas textual e gráfica da descrição deixa claro o uso conjunto das construções "external", "declare" e "use". Para que o exemplo fique completo, a interface de REG8 foi declarada na própria construção "external".

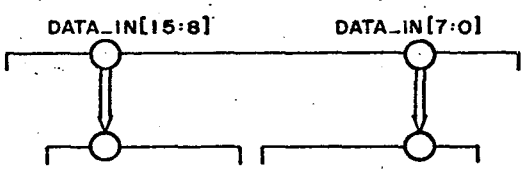
```
agency REG16.1.1
level = KAPA
interface
     in DATA_IN [15:0], RESET : terminal;
        CK_PULSE : clock;
     out DATA_OUT [15:0] : terminal;
structure
     external REG8.1 (
              level = KAPA
              interface
                   in INPUT [7:0], RST : terminal;
                      CKP : clock;
                   out OUTPUT [7:0] : terminal )
     use REG8.1 R8_X ( DATA_IN [15:8], RESET,
                       CK_PULSE, DATA_OUT [15:8] ),
         REG8.1 R8_Y ( DATA_IN [7:0], RESET,
                       CK_PULSE, DATA_OUT [7:0] );
end;
```



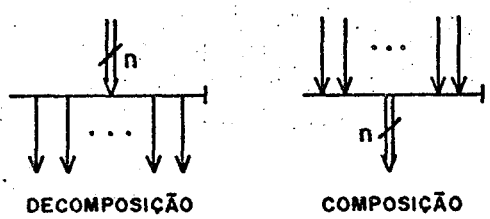
Os sinais de interface de uma ocorrência não necessariamente precisam ser conectados a sinais da agência que utiliza esta ocorrência. Tal situação ocorre frequentemente no projeto de sistemas digitais, quando se usa circuitos ou módulos já existentes na biblioteca de projeto, sem se aproveitar no entanto todas as potencialidades destes circuitos ou módulos. Um exemplo típico é a não conexão das entradas de PRESET e CLEAR de um flip-flop. Na declaração da instanciação, a posição dos sinais de interface não conectados é notada pela presença de delimitadores (vírgulas) consecutivos:

```
use FLIP_FLOP FF1 ( DATA, CLOCK, , , OUT );
```

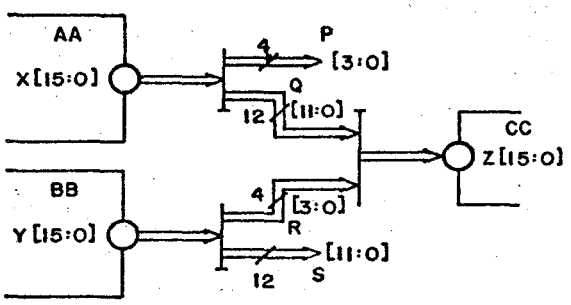
Graficamente, a linguagem REDES possui duas características adicionais para facilitar a especificação da interconexão de sinais. Em primeiro lugar, é possível decompor um sinal de interface de n bits de largura em vários sinais, cada um deles correspondendo a um diferente sub-conjunto destes n bits. Estes sinais têm o mesmo nome, mas diferentes intervalos de bits. No exemplo da agência REG16, esta característica foi utilizada, como salientado abaixo.



A segunda característica é a existência das operações gráficas de **composição** e **decomposição** de sinais, também existentes nas linguagens NILO e KAPA, e que permitem conexões separadas a bits ou sub-conjuntos de bits de um sinal de n bits. A operação de composição foi utilizada no exemplo do bloco operacional na seção anterior.



As operações de composição e decomposição também existem na forma textual de REDES, e são representadas pelo símbolo ":", como na linguagem KAPA. O exemplo abaixo mostra a utilização deste operador.




```
declare P[3:Ø], Q[11:Ø], R[3:Ø], S[11:Ø];
use A AA ( ..., P:Q, ... );
    B BB ( ..., R:S, ... );
    C CC ( ..., Q:R, ... );
```

Neste exemplo, o sinal X da interface da ocorrência AA, que é uma agência de tipo A, é decomposto em dois sub-vetores de 4 e 12 bits respectivamente. Estes sub-vetores são declarados como sinais internos P e Q. Da mesma forma, o sinal Y da interface da ocorrência BB (agência de tipo B) é decomposto em dois sub-vetores declarados como sinais internos R[3:Ø] e S[11:Ø]. Na construção use estas operações de decomposição são denotadas P:Q e R:S, respectivamente.

Já o sinal Z[15:Ø] da interface da ocorrência CC (agência de tipo C) é formado pela composição dos sinais internos Q[11:Ø] e R[3:Ø], o que é declarado por Q:R na construção use correspondente á instanciação de CC.

6. TIPOS DE DADOS

6.1 Repertório de tipos de dados

A cada sinal de interface ou interno de uma agência está associado um tipo de dados. Cada uma das linguagens comportamentais de AMPLO (LASSO, KAPA e NILO) possui um repertório particular de tipos de dados, o que pode ser visto nos relatórios técnicos correspondentes. Nem todos estes tipos de dados, no entanto, podem ser associados aos sinais de interface das agências. A tabela abaixo mostra, para cada uma das 3 linguagens, quais tipos de dados podem ser declarados para sinais de entrada, saída e bidirecionais.

	LASSO	KAPA	NILO
sinais de entrada	terminal integer control clock bus	terminal clock bus	terminal bus
sinais de saída	terminal integer control	terminal clock	terminal clock
sinais bidirecionais	bus	bus	bus

Como REDES é uma linguagem para especificar interconexões entre ocorrências de tipos de agências definidos numa das 3 linguagens comportamentais, apenas os tipos de dados acima podem ser utilizados na declaração de sinais de interface e sinais internos de agências descritas estruturalmente em REDES. Além disto, é necessário, numa descrição estrutural, que se declare o nível da agência, com o que podem ser utilizados apenas os tipos de dados disponíveis para os sinais de interface deste nível.

Os tipos de dados da tabela acima possuem a seguinte semântica:

terminal - sinais de saída de elementos combinacionais, que alteram seu valor imediatamente em função de alterações de valor em sinais de entrada destes elementos.

bus - sinais combinacionais especiais, aos quais podem ser feitas diversas atribuições ao longo de uma descrição, todas elas no entanto condicionadas por um sinal de controle.

clock - sinais combinacionais especiais, utilizados para controlar a carga de registradores, sub-registradores e registradores concatenados.

integer - sinais que podem assumir valores inteiros quaisquer, como variáveis de uma linguagem programação, e que modelam sinais cuja implementação não se quer ainda detalhar.

control - sinais booleanos, cujos valores indicam a ocorrência ou não de um evento. Correspondem aos lugares de uma

rede de Petri.

6.2 Compatibilidade de tipos de dados

Tipos de dados definidos em níveis diferentes podem ser compatíveis entre si. A compatibilidade de tipos de dados se reflete em AMPLO em duas situações diferentes:

- Ocorrências de tipos de agências declarados em níveis diferentes podem ser ligadas entre si, desde que os sinais interconectados sejam de tipos de dados compatíveis.
- Uma alternativa de projeto (ver capítulo 4) pode ser definida em diversos níveis, desde que os tipos de dados utilizados para um determinado sinal em cada uma destas definições sejam compatíveis entre si.

A tabela abaixo define a compatibilidade entre os tipos de dados que podem ocorrer nas interfaces de agências declaradas em cada um dos níveis de projeto.

tipo	nível	é compatível com tipo	nível
terminal	NILO	terminal	qualquer
		clock	NILO, KAPA (1,2,4)
		integer	LASSO
		control	LASSO (1)
terminal	KAPA	terminal	qualquer
		integer	LASSO
		control	LASSO (1)
terminal	LASSO	terminal	qualquer
bus	qualquer	bus	qualquer
		terminal	qualquer (5)
clock	NILO	clock	qualquer (3)
		terminal	NILO (1,4)
		control	LASSO (2)
clock	KAPA	clock	qualquer (3)
		terminal	NILO (1,2)
		control	LASSO (2)
clock	LASSO	clock	qualquer
integer	LASSO	terminal	NILO, KAPA
control	LASSO	terminal	NILO, KAPA (1)
		clock	NILO, KAPA (2)

Observações:
(1) "terminal" deve ter largura de um bit
(2) "clock" deve ser monofásico
(3) "clocks" devem ter mesmo número de fases
(4) "terminal" de NILO não é no entanto compatível com "clock" de NILO para efeito de determinação de consistência entre definições de uma mesma alternativa
(5) apenas quando "terminal" corresponde a um sinal de entrada

Além da compatibilidade entre os tipos de dados, uma outra regra deve ser seguida para se assegurar a integridade das interconexões entre sinais de interface de agências diversas:

- um sinal de saída pode estar conectado a vários sinais de entrada, mas a nenhum outro sinal de saída ou a algum sinal bidirecional;
- um sinal de entrada não pode estar ligado unicamente a outros sinais de entrada;
- um sinal bidirecional pode estar ligado a outros sinais bidirecionais e a sinais de entrada.

6.3 Interconexão de sinais do tipo "clock"

Se uma ocorrência A de um certo tipo de agência tem na sua interface um sinal de entrada de tipo "clock", este sinal pode ser conectado de uma de duas formas:

a) Ele é ligado a um sinal de saída de outra agência B da rede, também declarado como "clock". Neste caso, trata-se de um clock secundário [WAG87a], necessariamente monofásico, gerado no interior desta agência B.

b) Ele é ligado a um sinal de entrada da agência X que contém a ocorrência A. Neste caso, trata-se de um clock primário, gerado externamente a X, que pode ser multifásico e estar conectado a entradas (de tipo "clock") de outras ocorrências no interior de X.

REFERENCIAS BIBLIOGRAFICAS

[BOR79] BORRIONE,D. e J.F.GRABOWIECKI. "Informal Introduction to LASSO: A Language for Asynchronous Systems Specification and Simulation". In: P.A.Samet (ed.). EURO-IFIP 79. North-Holland Publ. Co., Amsterdam, 1979. pp 419-426

[BOR85] BORRIONE,D. e C.LeFAOU "Overview of the CASCADE Multi-level Hardware Description Language and its Mixed-mode Simulation Mechanisms". In: Koomen,C.J. e T.Moto-oka (editores), Computer Hardware Description Languages and their Applications, North-Holland, 1985. pp 239-260

[WAG84a] WAGNER,F.R. "Modelamento de Processos Digitais com Redes de Instâncias". Porto Alegre, CPGCC-UFRGS, Mar. 1984. 20p. (Relatório Técnico 006)

[WAG84b] WAGNER,F.R. "Modelamento e Simulação de Processos Digitais Usando Redes de Instâncias". In: Seminário Integrado de Software e Hardware, XI. Viçosa, MG, 21-28 julho 1984. Anais, UFV, 1984. pp 309-318

[WAG86a] WAGNER,F.R. "A Multi-level Digital Systems Simulator based on Nets of Agencies:. In: JSST Conference on Recent Advances in Simulation of Complex Systems. Toquio, Japão, 15-17 julho 1986. Proceedings, 1986. pp 125-130

[WAG86b] WAGNER,F.R., C.M.D.S.-FREITAS e L.G.GOLENDZINER. "Equivalência de Descrições Textuais e Gráficas de Sistemas Digitais num Ambiente de CAD". In: Seminário Integrado de Software e Hardware, XIII. Recife, PE, 19-25 julho 1986. Anais, UFPe, 1986. pp 486-493

[WAG86c] WAGNER,F.R. et al. "Ambiente Integrado para Projeto de Sistemas Digitais Auxiliado por Computador". In: Congresso Nacional de Informática, XIX. Rio de Janeiro, RJ, 18-25 agosto 1986. Anais, Vol. 2, SUCESU, 1986. pp 111-116

[WAG87a] WAGNER,F.R. "KAPA - Uma Linguagem para a Descrição de Hardware no Nível de Transferência Entre Registradores". Porto Alegre, CPGCC - UFRGS, 1987. (Relatório Técnico, a ser publicado)

[WAG87b] WAGNER,F.R. e C.M.D.S.-FREITAS. "NILO - Uma Linguagem para a Descrição de Hardware no Nível de Portas Lógicas". Porto Alegre, CPGCC - UFRGS, 1987. (Relatório Técnico, a ser publicado)

[WAG87c] WAGNER,F.R., C.M.D.S.-FREITAS e L.G.GOLENDZINER. "A Digital Systems Design Methodology based on Nets of Agencies". Aceito para publicação em: 8th International Symposium on Computer Hardware Description Languages and their Applications. Amsterdam, Holanda, 27-29 abril 1987

APÊNDICE. GRAMATICA REDES

Nota: a primeira versão desta gramática foi especificada pelos alunos Geraldo Arnt, Marco Cortinovi e Vanderlei Rodrigues como trabalho de aula da disciplina Compiladores I da UFRGS, sob orientação da Profa. Ana Price.

Convenções

< >	simbolos não terminais
!	alternativas
negrito	símbolos terminais
* *	delimitadores de comentários
&	símbolo vazio

```
<descricao-agencia> ::= agency <designacao-agencia> <nivel>
                        <interface> ; <estrutura-interna> end ;

* identificação da agência e do nível de descrição *

<designacao-agencia> ::= <designacao-tipo-agencia> .
                         <numero-versao>
<designacao-tipo-agencia> ::= <nome-tipo-agencia> .
                              <numero-alternativa>
<nome-tipo-agencia> ::= <identificador>
<numero-alternativa> ::= <inteiro>
<numero-versao> ::= <inteiro>
<nivel> ::= level = <identificador-nivel>
<identificador-nivel> ::= LASSO ! KAPA ! NILO

* declaração da interface *

<interface> ::= interface <declaracao-interface>
<declaracao-interface> ::= <direcao> <lista-declar-sinais> !
      <direcao> <lista-declar-sinais> ; <declaracao-interface>
<direcao> ::= in ! out ! inout
<lista-declar-sinais> ::= <declaracao-sinais> !
                    <declaracao-sinais> ; <lista-declar-sinais>
<declaracao-sinais> ::= <lista-sinais> : <nome-tipo-sinal>
<lista-sinais> ::= <sinal> ! <sinal> , <lista-sinais>
<sinal> ::= <identificador> !
            <identificador> [ <inteiro> : <inteiro> ]
<nome-tipo-sinal> ::= terminal ! clock ! bus ! integer !
                      control

* declaração da estrutura interna da agência *

<estrutura-interna> ::= structure <lista-declar-tipos> <use>
<lista-declar-tipos> ::= <declaracao-tipos> !
                         <declaracao-tipos> <lista-declar-tipos>
<declaracao-tipos> ::= <declare> ! <external>
<declare> ::= declare <lista-declar-sinais>
<external> ::= external <lista-declar-externas> ;
<lista-declar-externas> ::= <declar-external> !
```

```
                        <declar-external> , <lista-declar-externas>
<declar-external> ::= <designacao-tipo-agencia> |
          <designacao-tipo-agencia> ( <nivel> <interface> )

* declaracao das instanciacoes de tipos de agencias *

<use> ::= <lista-instanciacoes> | <lista-instanciacoes> <use>
<lista-instanciacoes> ::= use <lista-itens-instanc> ;
<lista-itens-instanc> ::= <instanciacao> |
                          <instanciacao> , <lista-itens-instanc>
<instanciacao> ::= <designacao-tipo > <nome-ocorrencia>
                   ( <lista-conexoes> )
<designacao-tipo> ::= <designacao-tipo-agencia> |
                      <designacao-agencia>
<nome-ocorrencia> ::= <identificador>
<lista-conexoes> ::= <conexao> | <conexao> , <lista-conexoes>
<conexao> ::= <sinal> | <bit-sinal> | <concatenacao>
<concatenacao> ::= <termo> : <termo>
<termo> ::= <sinal> | <bit-sinal>
<bit-sinal> ::= <identificador> [ <inteiro> ] | &


* definições básicas *

<inteiro> ::= <digito> | <digito> <inteiro>
<digito> ::= 0 | 1 | 2 | ... | 9

<identificador> ::= <letra> | <letra> <texto>
<texto> ::= <simbolo> | <simbolo> <texto>
<simbolo> ::= <letra> | <digito> | _
<letra> ::= a | b | c | ... | z |
            A | B | C | ... | Z
```

Relatórios de Pesquisa

RP-66 WAGNER, F. R. & DAL SASSO-FREITAS, C. M., "NILO - uma linguagem para a descrição de hardware no nivel de portas lógicas" , março 87.

RP-65 WAGNER, F. R. , DAL SASSO-FREITAS, C. M. & GOLENDZINER, L. G. , "Linguagens de descrição de hardware para suporte à integração do processo de projeto em AMPLO" , março 87.

RP-64 ROCHA COSTA, A. C. , "Prolog como linguagem de implementação de sistemas: oito programas ilustrativos" , fev 87.

RP-63 POLANCZYK, C. A. , CLAUDIO, D. M. & LOPEZ, J. D. , "Software elementar para intervalos" , fev 87.

RP-62 WESTPHALL, C. B. , "Sinópse da especificação MAP", dez 86.

RP-61 ROCHA COSTA, A. C. , "Sobre os fundamentos da inteligência artificial" , nov 86.

RP-060 WALTER, C., "A method for the specification of manufacturing systems and their controllers" , out 86.

RP-059 PALAZZO OLIVEIRA, J.P. & RUIZ, D.D.A., "Formulários Eletrônicos: definição e manipulação automática da interface de usuário" , set 86.

RP-058 TOSCANI, L.V. & SZWARCFITER, J.L., "Algorítmos aproximativos" , set 86.

RP-057 JANSCH-PORTO, I. & COURTOIS, B., "Design and assembling of SDC checkers based on analytical fault hypotesis" , set 86.

RP-056 TAZZA, M., "Fundamentals of a Net Theory Based Performance Evaluation Model" , ago 86.

RP-055 FREITAS, C.M.D.S & OLIVEIRA, F.M, "Editor gráfico para sistemas digitais descritos no nivel lógico" , ago 86.

RP-054 ROCHA COSTA, A.C., "Para uma revisão epistemológica da inteligência artificial" , jul 86.

RP-053 TAZZA, M., "Quantitative analysis of a resource allocation problem: a net theory based proposal" , jul 86.

RP-052 LONGHI, M.T., "Representação de objetos sólidos por octrees" , julho 86.

RP-051 NASCIMENTO, F.R. do, Kit MCP-68000: Descrição do projeto de hardware., julho 86.

RP-050 MURR, A.L.D. & CASTILHO, J.M.V. de, Tradução de sentenças em lógica para sentenças em forma clausal: uma implementação em PROLOG.", julho 86.

RP-049: TOSCANI L.V. & VELOSO P.A.S., Especificação formal e análise da complexidade da programação dinâmica., mai 86.

RP-048: HURTADO, J.O.H.; GOMES, R.F. & SEELING, M., Documentação da concepção e testes de um circuito integrado NMOS. jun/86.

RP-047: PALAZZO OLIVEIRA, J., Electronic forms: a local area microcomputer system - Project description. (também disponivel em português.), mai/86.

RP-046: TOSCANI, L.V. Guia de estudo da complexidade de algorítmos de procura. nov/85.

RP-045: NASCIMENTO, F.R. & OLIVEIRA, R.S., Programa monitor para um microcomputador educacional implementado com o MC68000. dez/85.

RP-044: WAGNER, F.R., Simulação de sistemas modelados como redes de agencias. nov/85.

RP-043: FREITAS, C.M.D.S. & OLIVEIRA, F.M., Editor gráfico para sistemas modelados como redes de agencias. nov/85.

RP-042: TAZZA, M., Sistemas-C/E como ferramenta de modelagem. out/85.

RP-041: WAGNER, F. R.; FREITAS, C. M. D. S. & GOLENDZINER, L. G., O processo de projeto de sistemas digitais num ambiente integrado de CAD. out/85.

RP-040: TAZZA, M., Performance evaluation using a net theory based model. set/85.

RP-039: WAGNER, F. R., On the properties of event oriented logic simulation according to significant timing models. set/85.

RP-038: WAGNER, F. R., Algorítmos de simulação de hardware no nivel RT. set/85.

RP-037: COSTA, A. C. R., Processando linguagens naturais em PROLOG - Parte 1: Formalismo gramatical básico. jul/85.