Centro Científico Rio **Relatório Técnico CCR-109**

Mapeamento de um Processo de Projeto de Circuitos VLSI para o Modelo de Dados do Ambiente GARDEN

por

Flávio R. Wagner

IBM Rio Scientific Center
P.O. Box 4624
20001 Rio de Janeiro - RJ
Brazil

Vnet Ids
GARDEN at RIOVMSC

Sumário

Este relatório descreve um exercício de simulação de uma seqüência de passos de projeto de circuitos VLSI que resulta na criação de objetos no ambiente GARDEN. É definida uma metodologia de projeto de sistemas digitais típica, e os passos desta metodologia são mapeados para operações sobre os objetos definidos no modelo de dados do GARDEN. Este mapeamento atribui aos objetos GARDEN uma determinada semântica. Os objetivos deste trabalho são ilustrar a utilização do modelo de dados do GARDEN, e mostrar que o modelo permite a definição de diferentes esquemas conceituais, de acordo com critérios definidos pelo usuário.

Abstract

This report describes an exercise in simulating a sequence of design steps for VLSI circuits which creates design objects in the GARDEN framework. A typical digital systems design methodology is defined, and the methodology steps are mapped onto operations on the GARDEN data model objects. This mapping assigns to the GARDEN objects a given semantics. Main goals of this work are to illustrate the use of the GARDEN data model and to show that the model allows the definition of various conceptual schemes, according to user-defined criteria.

O autor está em licença da Universidade Federal do Rio Grande do Sul, Instituto de Informática e Curso de Pós-Graduação em Ciência da Computação

Conteúdo

1 Introdução

Ambientes de projeto de circuitos integrados VLSI têm sido propostos e implementados nos últimos anos para suportar a integração de conjuntos de ferramentas orientadas a diferentes aplicações, arquiteturas e níveis de abstração. Estes ambientes deveriam idealmente garantir consistência automática entre diferentes representações de um mesmo circuito, oferecer mecanismos para permitir uma integração uniforme de novas ferramentas, e suportar a gerência de dados e de projeto. Podem ser citados em especial os ambientes ADAM, da USC [1], FACE, da GE [2], OCT, de Berkeley [3], a estação de trabalho CWS do CADLAB [4] e o sistema do IMEC [5]. No Brasil, a UFRGS está desenvolvendo o sistema AMPLO [6].

GARDEN [7] é um sistema em especificação no Centro Científico Rio da IBM, que procura responder a todos os principais requisitos de ambientes de projeto. Até o momento, o trabalho concentrou-se na especificação de um modelo de dados que contempla os aspectos de representação e gerência dos dados de projeto.

Estes aspectos envolvem a maneira pela qual um circuito integrado é representado em GARDEN como um objeto complexo, e os mecanismos que permitem controlar as diversas representações deste objeto, geradas em função de três diferentes dimensões da evolução do projeto: descrições em níveis de abstração distintos, implementações alternativas do circuito, e revisões de projeto de uma mesma implementação.

O modelo de dados do GARDEN é bastante flexível. Ele define objetos complexos, organizados segundo uma hierarquia composta por Design, ViewGroup, View, Modificação e Iteração. Esta hierarquia pode ser interpretada de inúmeras maneiras, que implementam as dimensões do processo de evolução de projeto segundo critérios variados.

O objetivo deste relatório é ilustrar o uso do modelo de dados do GARDEN. Para isto, define-se um mapeamento entre os passos de uma determinada metodologia de projeto de circuitos integrados e os objetos GARDEN, criando-se com isto implicitamente um esquema conceitual para a aplicação. É então simulada uma seqüência de criação de objetos GARDEN que corresponde a uma possivel seqüência de passos de projeto segundo a metodologia estabelecida. Este exercício de simulação permite que se verifique a consistência do mapeamento sugerido, e dá muitas indicações de outros mapeamentos possíveis.

O modelo GARDEN é bastante rico e complexo. Sua estrutura básica [7], ilustrada na Figura 1, oferece uma hierarquia para a representação e gerência de versões de objetos complexos, composta pelos conceitos **Design**, **ViewGroup**, **View**, **Modificações** e **Iterações** (estes dois últimos formando **ViewStates** e sendo referidos como M-I's neste relatório). ViewGroups podem conter outros ViewGroups ou Views, podendo ser criada assim uma representação hierárquica com profundidade qualquer.

Views podem ser de três tipos: **HDL**, **Layout** e **MMHD** (Mixed-Mode Hie-

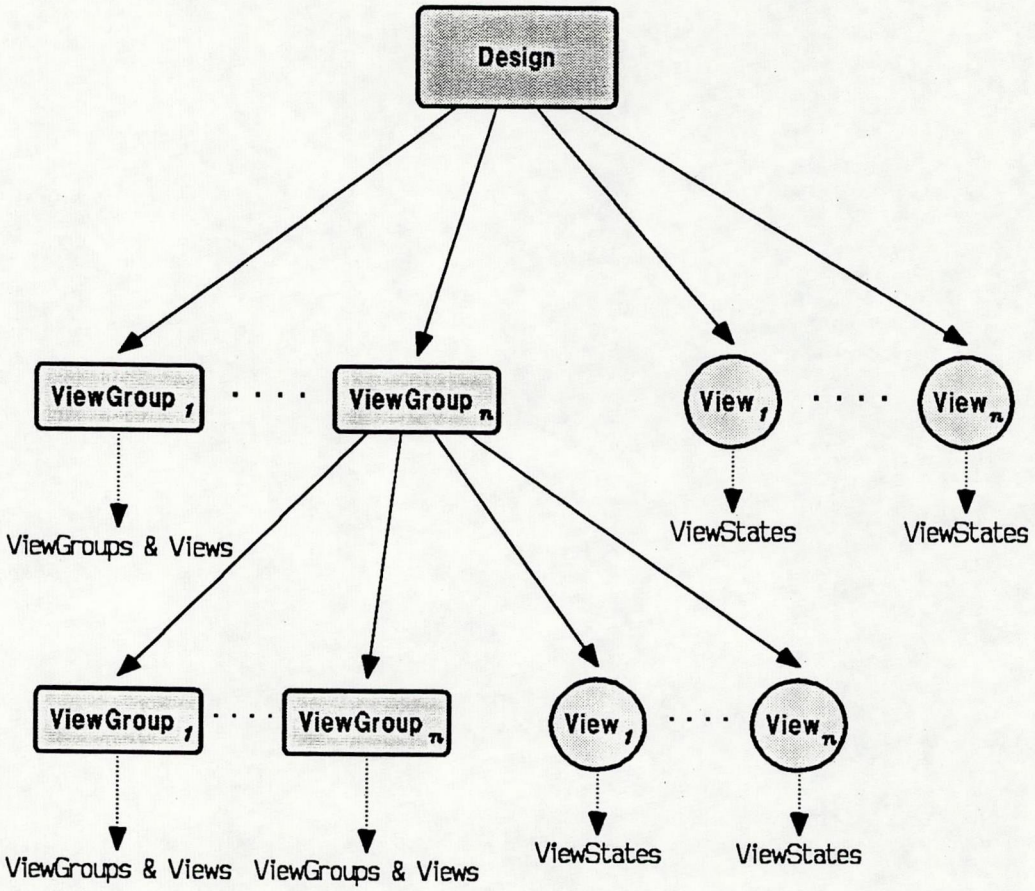

Figura 1: Objetos básicos do modelo de dados do GARDEN

rarchical Description). Todos os tipos de Views podem conter componentes, que fazem referência a outros Designs. A referência pode ser feita a qualquer nível da hierarquia de descrição destes outros Designs.

Conforme ilustrado na Figura 2, Modificações são descrições alternativas para uma determinada View, enquanto Iterações são refinamentos sucessivos de uma mesma Modificação.

Sinais de interface (**Ports**) podem ser definidos nos níveis de Design, ViewGroup e View, sendo herdados pelos níveis inferiores da hierarquia. **Correlações** podem ser estabelecidas, relacionando objetos quaisquer, em qualquer nível de suas respectivas hierarquias.

O modelo de dados do GARDEN oferece diversos outros recursos adicionais (Libraries, Processes, TimeStamps). Este relatório irá se restringir ao uso e análise do conjunto de conceitos relacionados nos parágrafos anteriores, no entanto, já que eles formam a estrutura básica para a representação e gerência de dados.

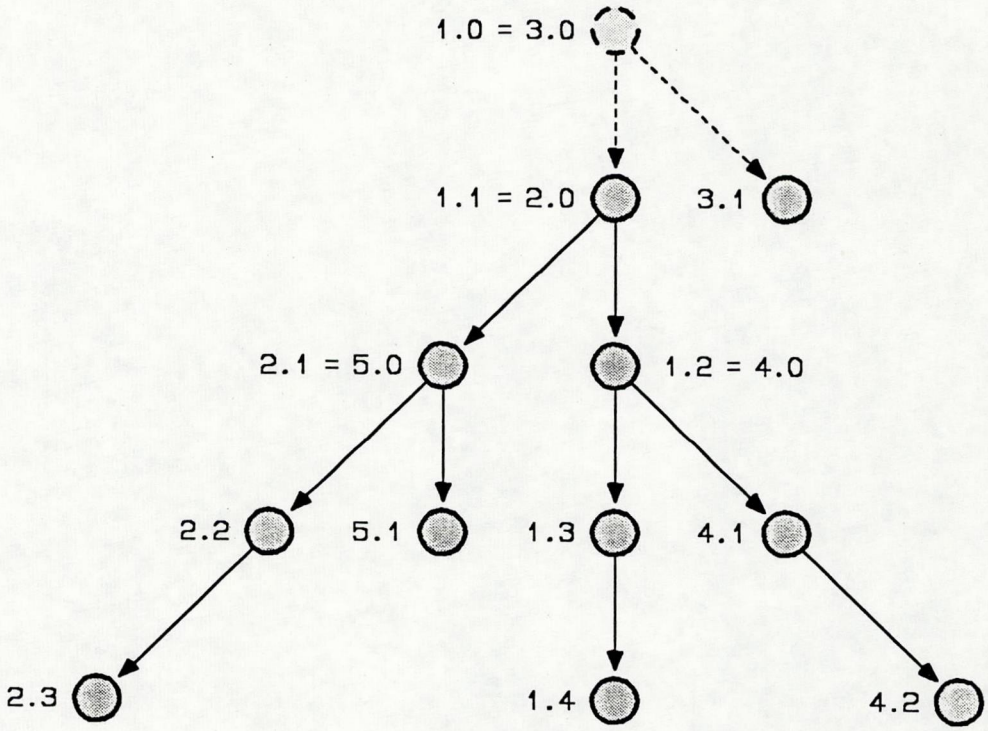

Figura 2: Modelo de dados do GARDEN – modificações e iterações

2 Metodologia de projeto

A metodologia proposta para este exercício de simulação prevê o projeto de um sistema digital (denominado genericamente de X) segundo a divisão clássica em blocos operacional e de controle (designados genericamente de XBO e XBC). Estes blocos são projetados com o auxílio de ferramentas distintas. Estratégias de implementação diversas são testadas em cada um destes blocos, para que se avalie alternativas de implementação. Para o bloco operacional são tentadas implementações com barramentos e com multiplexadores e, no nível de layout, são testadas estratégias standard-cell e gate-array. Para o bloco de controle são avaliadas soluções com PLA e com lógica aleatória.

Na descrição a seguir, as tarefas da metodologia proposta são designadas por Tn. Apenas a especificação inicial (tarefa T0) não será mapeada posteriormente para nenhum objeto GARDEN, pois imagina-se que ela seja apenas uma idéia na cabeça do projetista.

Não se entra aqui na discussão de qual (ou quais) HDL(s) será(ão) utilizada(s). Eventualmente uma única HDL, como VHDL [8], poderia ser utilizada em todas as descrições, desde o comportamento inicial de X até a estrutura RT de XBO e as equações lógicas de XBC. Linguagens diferentes poderiam ser usadas para estas situações distintas.

Apenas por questão didática, a metodologia foi dividida em grupos de tarefas mais fortemente relacionadas. As tarefas são executadas seqüencialmente. Após as avaliações (simulações, avaliações topológicas, de timing ou de testabilidade) podem ser feitos desvios nesta seqüência, repetindo-se o processo desde alguma tarefa anterior.

Projeto de alto nível

`T0`. Especificação inicial

`T1`. Descrição do comportamento de X com HDL

`T2`. Simulação do comportamento

`T3`. Geração de casos de teste a partir do comportamento

Síntese de alto nível do bloco operacional

`T4`. Síntese do bloco operacional: etapas de scheduling e alocação de operadores, registradores e interconexões - gera descrição RT com HDL
Observação: podem ser testadas duas alternativas:

1. BO com barramentos
2. BO com multiplexadores

T5. Síntese do bloco operacional: etapa de mapeamento tecnológico - escolha dos tipos de registradores e esquema de fases - gera descrição RT com HDL
Observação: podem ser testadas duas alternativas:

1. relógio de 4 fases (registradores dinâmicos)
2. relógio de 2 fases (registradores dinâmicos)

Síntese de alto nível do bloco de controle

T6. Síntese do bloco de controle: etapa de geração da máquina de estados - gera descrição com HDL

Avaliação de alto nível

T7. Composição de X com BO descrito como uma estrutura RT e BC descrito como uma máquina de estados - gera descrição com HDL

T8. Simulação de X

T9. Avaliação topológica preliminar

T10. Análise de testabilidade

T11. Geração de vetores de teste (combinando resultados das tarefas T3 e T7)

Síntese lógica do bloco de controle

T12. Síntese do bloco de controle: etapa de atribuição de estados e minimização lógica - gera conjunto de equações booleanas.
Observação: podem ser adotadas duas alternativas:

1. BC com PLA
2. BC com lógica aleatória

T13. Síntese do bloco de controle: etapa de otimização tecnológica - gera conjunto otimizado de equações booleanas

Síntese do layout do bloco operacional

T14. Geração de partes operativas, usando biblioteca de células - gera descrição estrutural (composição de células, como portas lógicas, flip-flops, etc)
Observação: podem ser testadas duas alternativas:

1. estratégia standard-cell
2. estratégia gate-array

T15. Inserção de estruturas de teste (usando informações de testabilidade geradas em T10)

T16. Geração do layout do bloco operacional, usando biblioteca de células

Síntese do layout do bloco de controle

T17. Geração do layout para bloco de controle

Avaliação de timing

T18 . Extração de parâmetros elétricos para o bloco operacional - gera composição de transistores

T19. Simulação elétrica de partes críticas do bloco operacional

T20. *Back-annotation* de informações de timing para a descrição do BO como composição de células (saída da T14)

T21. Extração de parâmetros elétricos do bloco de controle - gera composição de transistores

T22. Simulação elétrica de partes críticas do bloco de controle

T23. *Back-annotation* de informações de timing para a descrição do BC como conjunto de equações lógicas (saída da T13)

T24. Avaliação de timing de X - feita sobre descrição composta por BO (composição de células) e BC (equações lógicas), anotada pelas informações de timing

Composição final do layout

T25. Composição do layout de BO e BC

T26. Verificação de regras de projeto (DRC)

3 Mapeamento da metodologia de projeto para o modelo de dados do GARDEN

A partir da metodologia anterior, pode-se executar inúmeras seqüências possíveis de ativações de ferramentas, dependendo de quantas alternativas de implementação são tentadas nos pontos em que há opções de projeto, e obviamente dependendo de quantas revisões de uma mesma descrição são feitas até que o projetista considere-se satisfeito com os resultados.

Antes que se possa passar a descrever a seqüência de criação de objetos GARDEN ao longo da metodologia de projeto, uma decisão muito importante deve ser tomada. GARDEN é um modelo de dados flexível. Os objetos ViewGroup, View e M-I têm uma semântica bastante aberta. Os critérios que levam ao agrupamento de determinadas descrições de um objeto como Views debaixo de um mesmo ViewGroup, ou como M-I's debaixo de uma mesma View, ou que levam à criação de um novo ViewGroup ou de uma nova View, são variáveis. Estes critérios devem ser criados previamente à execução de um projeto, e devem ser obedecidos uniformemente ao longo do processo de projeto, de modo que a estrutura hierárquica abaixo dos vários Designs mantenha uma coerência. Denominamos esta definição de critérios como "mapeamento da metodologia de projeto para o modelo de dados". Este mapeamento, num ambiente de projeto, deveria poder ser definido pelo usuário (ou talvez mais propriamente por um super-usuário), através de uma ferramenta de controle de metodologia de projeto. Não é objetivo deste relatório, no entanto, apresentar tal ferramenta.

Os critérios que definem o mapeamento adotado neste relatório reproduzem três conceitos presentes no modelo de gerenciamento de versões do sistema DAMASCUS [9], embora vá ser adotada uma hierarquia diferente para organização destes conceitos. Um objeto possui diferentes **representações**, que correspondem a descrições em níveis de abstração diversos. Cada representação pode ter diversas **alternativas**, que resultam de várias decisões de projeto. Para cada alternativa existem várias **revisões**, que correspondem a melhoramentos sucessivos do projeto.

Os objetos de GARDEN serão usados da seguinte maneira: ViewGroups serão alternativas de projeto. Views sob um mesmo ViewGroup serão representações diversas para uma mesma alternativa, e M-I's serão revisões de uma mesma representação.

Deve-se notar três diferenças fundamentais entre os modelos de gerenciamento dos sistemas DAMASCUS e GARDEN (este tal como utilizado no mapeamento aqui adotado). Enquanto em DAMASCUS representações reúnem alternativas, em GARDEN temos o contrário. Em GARDEN, ao contrário de DAMASCUS, é possível organizar as revisões na forma de uma árvore. Finalmente, em GARDEN pode-se organizar uma hierarquia de alternativas (ViewGroups), com o que pode-se mapear de forma bastante direta para o modelo a hierarquia de decisões de projeto.

Tanto em DAMASCUS como em GARDEN há um mecanismo que permite a

inclusão de Ports em qualquer nível da hierarquia, com herança em direção aos níveis inferiores da hierarquia.

O mapeamento adotado pode ser resumido pelas regras abaixo.

Para cada Design:

- na raíz da sua hierarquia estão armazenados os Ports que são comuns a todas as implementações deste Design (tipicamente sinais de endereço e dados)
- ViewGroups num mesmo nível são implementações alternativas, que acrescentam Ports em relação ao seu nodo pai, e que acrescentam detalhes de projeto num nível inferior de abstração
- Views num mesmo nível são representações do nodo pai (o Design ou um ViewGroup) em níveis diversos de abstração, baseadas nas informações de projeto típicas do nível de detalhamento deste nodo pai, ou são descrições compostas (MMHDs) geradas a partir de uma das Views irmãs
- Um ViewGroup pode reunir
  1. Views de tipos diferentes, e/ou
  2. ViewGroups que representam detalhamentos alternativos

4 Seqüência de criação dos objetos GARDEN

Nesta seção é apresentada uma seqüência possível de criação de objetos GARDEN. Os passos de projeto seguem a metodologia descrita na seção anterior. Nesta seqüência, por economia de espaço, não foram criadas várias revisões (M-I's) consecutivas para os objetos. Considerou-se, de modo geral, que a primeira descrição de cada alternativa de implementação de cada objeto já é satisfatória para o projetista.

Na descrição destes passos foi adotada a notação apresentada na Tabela 1.

VGn	ViewGroup de número n
Vn	View de número n
Mn	Modificação de número n
In	Iteração de número n
(HDL)	View de tipo HDL
(MMHD)	View de tipo MMHD
(LO)	View de tipo Layout
Pi	Passo de projeto de número i
(Tj)	Tarefa j da metodologia de projeto à qual o passo de projeto está associado

Tabela 1: Notação adotada na descrição dos objetos criados

Para maior clareza dos exemplos, os ViewGroups e Views foram numerados de forma corrida dentro de um mesmo Design, enquanto os M-I's foram numerados a partir de 1 dentro de cada View.

Na criação dos objetos, foi usada ainda uma notação simplificada para indicar um novo objeto dentro de outro já existente. Assim, por exemplo, nova V para X-VG1-VG2 significa "nova View para o ViewGroup2 do ViewGroup1 do Design X".

Projeto de alto nível

P1. (T1) Especificação inicial do comportamento.
Cria X, com V1 (HDL), M1, I1

P2. (T2) Simulação do comportamento

P3. (T1) Revisão da especificação do comportamento
Cria I2 para M1 de X-V1

P4. (T2) Simulação da revisão do comportamento

Síntese de alto nível da alternativa com barramentos

P5. (T4) Síntese do bloco operacional: scheduling e alocação
Alternativa com barramentos.
Cria XBO, com VG1, V1 (HDL), M1, I1

P6. (T5) Síntese do bloco operacional: mapeamento tecnológico
Alternativa com relógio de 4 fases.
Cria VG2 para XBO-VG1, com V2 (HDL), M1, I1

P7. (T6) Síntese do bloco de controle: geração da máquina de estados
Cria XBC, VG1, VG2, V1 (HDL), M1, I1

P8. (T7) Composição de BO com BC.
Cria VG1 para X, com VG2, V2 (MMHD), M1, I1

P9. (T8) Simulação da descrição de X composta pelas versões de XBO e XBC criadas nos passos P6 e P7, respectivamente.

Síntese de alto nível da alternativa com multiplexadores

P10. (T4) Síntese do bloco operacional: scheduling e alocação
Alternativa com multiplexadores.
Cria VG3 para XBO, com V3 (HDL), M1, I1

P11. (T5) Síntese do bloco operacional: mapeamento tecnológico.
Alternativa com relógio de 4 fases.
Cria VG4 para X-VG3, com V4 (HDL), M1, I1

P12. (T6) Síntese do bloco de controle: geração da máquina de estados.
Cria VG3 para XBC, com VG4, V2 (HDL), M1, I1

P13. (T7) Composição de BO com BC.
Cria VG3 para X, com VG4, V3 (MMHD), M1, I1

P14. (T8) Simulação da descrição de X composta pelas versões de XBO e XBC criadas nos passos P11 e P12, respectivamente.

Síntese de alto nível da alternativa com barramento: novo scheduling

P15. (T4) Síntese do bloco operacional: etapa de scheduling e alocação.
Alternativa com barramento, mas com novo scheduling de operações.
Cria I2 para XBO-VG1-V1-M1

P16. (T5) Síntese do bloco operacional: mapeamento tecnológico.
Alternativa com relógio de 4 fases.
Cria I2 para XBO-VG1-VG2-V2-M1

P17. (T6) Síntese do bloco de controle: geração da máquina de estados.
Cria I2 para XBC-VG1-VG2-V1-M1

P18. (T8) Simulação da descrição de X composta pelas versões de XBO e XBC criadas nos passos P16 e P17, respectivamente.

Síntese do layout para primeira alternativa com barramento

P19. (T14) Geração de parte operativa para alternativa de XBO com barramento, relógio com 4 fases, primeira opção de scheduling e alocação.
Alternativa com estratégia standard-cell.
Cria VG5 para XBO-VG1-VG2, com V5 (MMHD), M1, I1

P20. (T16) Geração de layout para bloco operacional.
Cria V6 (LO) para XBO-VG1-VG2-VG5, com M1, I1

P21. (T14) Geração de parte operativa para alternativa de XBO com barramento, relógio com 4 fases, primeira opção de scheduling e alocação.
Alternativa com estratégia gate-array.
Cria VG6 para XBO-VG1-VG2, com V7 (MMHD), M1, I1

P22. (T16) Geração de layout para bloco operacional.
Cria V8 (LO) para XBO-VG1-VG2-VG6, com M1, I1

P23. (T12) Síntese do bloco de controle: etapa de atribuição de estados e minimização lógica.
Para alternativa de XBC gerada no P7 (corresponde a XBO com barramento, relógio com 4 fases, primeira opção de scheduling e alocação).
Alternativa com estratégia PLA.
Cria VG5 para XBC-VG1-VG2 com V3 (HDL), M1, I1

P24. (T13) Síntese do bloco de controle: etapa de otimização tecnológica.
Cria V4 (HDL) para XBC-VG1-VG2-VG5, com M1, I1

P25. (T17) Geração de layout do bloco de controle.
Cria V5 (LO) para XBC-VG1-VG2-VG5, com M1, I1

P26. (T12) Síntese do bloco de controle: etapa de atribuição de estados e minimização lógica.
Para alternativa de XBC gerada no P7 (corresponde a XBO com barramento, relógio com 4 fases, primeira opção de scheduling e alocação).
Alternativa com estratégia lógica aleatória.
Cria VG6 para XBC-VG1-VG2, com V6 (HDL), M1, I1

P27. (T13) Síntese do bloco de controle: etapa de otimização tecnológica.
Cria V7 (HDL) para XBC-VG1-VG2-VG6, com M1, I1

P28. (T17) Geração de layout do bloco de controle.
Cria V8 (LO) para XBC-VG1-VG2-VG6, com M1, I1

P29. (T25) Composição do layout de BO e BC. Usa versão de XBO criada no P20 (com estratégia standard-cell) e de XBC criada no P25 (com estratégia PLA).
Cria V4(LO) para X-VG1-VG2, com M1, I1

P30. (T26) Verificação de regras de projeto do layout composto no P29

P31. (T25) Composição do layout de BO e BC. Usa versão de XBO criada no P20 (com estratégia standard-cell) e de XBC criada no P28 (com estratégia lógica aleatória).
Cria V5 (LO) para X-VG1-VG2, com M1, I1

P32. (T26) Verificação de regras de projeto do layout composto no P31

Síntese da alternativa com barramento, novo mapeamento tecnológico

P33. (T5) Síntese do bloco operacional: mapeamento tecnológico.
Alternativa com relógio de duas fases.
Cria VG7 para XBO-VG1, com V9 (HDL), M1, I1

P34. (T6) Síntese do bloco de controle: geração da máquina de estados.
Cria VG7 para XBC-VG1, com V9 (HDL), M1, I1

P35. (T7) Composição de BO com BC. Não altera interface externa de X.
Cria VG5 para X-VG1, com V6 (MMHD), M1, I1

P36. (T8) Simulação da descrição de X composta pelas versões de XBO e XBC criadas nos passos P33 e P34, respectivamente.

P37. (T14) Geração da parte operativa. Usa descrição de BO gerada no P28 (com barramento e relógio com 2 fases).
Alternativa com estratégia standard-cell.
Cria VG8 para XBO-VG1-VG7, com V10 (MMHD), M1, I1

P38. (T16) Geração do layout do bloco operacional.
Cria V11 (LO) para XBO-VG1-VG7-VG8, com M1, I1

P39. (T12) Síntese do bloco de controle: etapa de atribuição de estados e minimização lógica.
Alternativa com estratégia PLA.
Cria VG8 para XBC-VG1-VG7, com V10 (HDL), M1, I1

P40. (T13) Síntese do bloco de controle: etapa de otimização tecnológica.
Cria V11 (HDL) para XBC-VG1-VG7-VG8, com M1, I1

P41. (T17) Geração do layout do bloco de controle.
Cria V12 (LO) para XBC-VG1-VG7-VG8, com M1, I1

P42. (T25) Composição do layout de BO e BC. Usa versão de XBO criada no P38 (com estratégia standard-cell) e de XBC criada no P41 (com estratégia PLA). Cria V7 (LO) para X-VG1-VG5, com M1, I1

P43. (T26) Verificação de regras de projeto do layout composto no P42

Avaliação de timing da alternativa com barramento, relógio com 4 fases, BO com standard-cell e BC com PLA

P44. (T18) Extração de parâmetros elétricos para alternativa de XBO com barramento, relógio de 4 fases e estratégia standard-cell.
Cria V12 (MMHD) para XBO-VG1-VG2-VG5, com M1, I1

P45. (T19) Simulação elétrica de partes críticas de XBO.

P46. (T20) *Back-annotation* de informações de timing para a descrição de XBO como composição de células.
Cria I2 para XBO-VG1-VG2-VG5-V5-M1

P47. (T21) Extração de parâmetros elétricos para alternativa de XBC com PLA e correspondendo a XBO com barramento, e relógio com 4 fases.
Cria V13 (MMHD) para XBC-VG1-VG2-VG5, com M1, I1

P48. (T22) Simulação elétrica de partes críticas de XBC

P49. (T23) *Back-annotation* de informações de timing para a descrição de XBC com PLA.
Cria I2 para XBC-VG1-VG2-VG5-V4-M1

P50. (T24) Avaliação de timing de X. Cria V8 (MMHD) para X-VG1-VG2, com M1, I1, usando versão de XBO criada no P46 e de XBC criada no P49. A criação de V8 é opcional e depende da configuração adotada em X-VG1-VG2-V2-M1-I1, criada no P8 (ver observação 11 na seção 5.1)

5 Objetos criados

Após a conclusão de todos os passos de projeto apresentados anteriormente, tem-se no banco de dados 3 Designs (X, XBO e XBC) com inúmeros ViewGroups, Views e M-I's. As subseções a seguir apresentam a hierarquia de objetos para cada um destes Designs. Uma série de comentários para fins de esclarecimento é feita a respeito dos mesmos. Estes comentários são identificados por superscritos ao lado de cada objeto, e listados em separado, agrupados em diferentes temas (comentários sobre os Ports, sobre as Correlações, etc).

5.1 Design X

<table>
<tr><td rowspan="10">X[1]</td><td>V1 (HDL)
comport.inicial</td><td>M1</td><td colspan="3">I1</td></tr>
<tr><td></td><td colspan="3">I2</td></tr>
<tr><td rowspan="8">VG1[2,5]
implementação
c/barramentos</td><td rowspan="4">VG2[3,6]
relógio
4 fases</td><td>V2 (MMHD)[10]</td><td>M1</td><td>I1[R1]</td></tr>
<tr><td>V8 (MMHD)[11]
após back-
annotation</td><td>M1</td><td>I1[R7]</td></tr>
<tr><td>V4 (LO)[4]
standard-cell
+ PLA</td><td>M1</td><td>I1[R2]</td></tr>
<tr><td>V5 (LO)[4]
standard-cell
+lóg.aleatória</td><td>M1</td><td>I1[R3]</td></tr>
<tr><td rowspan="2">VG5[3,7]
relógio
2 fases</td><td>V6 (MMHD)[10]</td><td>M1</td><td>I1[R4]</td></tr>
<tr><td>V7 (LO)[4]
standard-cell
+ PLA</td><td>M1</td><td>I1[R5]</td></tr>
<tr><td>VG3[2,8]
implementação
c/multiplex.</td><td>VG4[3,9]
relógio
4 fases</td><td>V3 (MMHD)[10]</td><td>M1</td><td>I1[R6]</td></tr>
</table>

Figura 3: Design X

A Figura 3 contém a hierarquia de ViewGroups, Views e M-I's para o Design X. A Tabela 2 apresenta, para cada View MMHD de X, quais são as referências feitas estaticamente a XBO e XBC. O restante da referência, até o nível de M-I, é feito dinamicamente quando da ativação de alguma ferramenta de projeto. Nesta tabela, {-VGi-Vj} significa que a referência até este VGi e/ou Vj poderia ter sido feita opcionalmente quando da criação desta versão de X. Caso tenha sido feita a referência completa, apenas a seleção de uma M-I precisará ser feita quando

da chamada de uma ferramenta de projeto. A referência mais curta tambem é possível, pois o caminho adicional até o nível de View, indicado opcionalmente, não acrescenta Ports ao Design. Esta opção permite que a versão de X seja reutilizada mais livremente com diferentes versões de XBO e XBC (ver comentários adicionais a respeito na seção 6, que trata de configurações).

Referência	XBO	XBC
R1	XBO-VG1-VG2{-V2}	XBC-VG1-VG2{-V1}
R2	XBO-VG1-VG2-VG5-V6	XBC-VG1-VG2-VG5-V5
R3	XBO-VG1-VG2-VG5-V6	XBC-VG1-VG2-VG6-V8
R4	XBO-VG1-VG7{-V9}	XBC-VG1-VG7{-V9}
R5	XBO-VG1-VG7-VG8-V11	XBC-VG1-VG7-VG8-V12
R6	XBO-VG3-VG4{-V4}	XBC-VG3-VG4{-V2}
R7	XBO-VG1-VG2{-VG5-V5}	XBC-VG1-VG2{-VG5-V4}

Tabela 2: Referências a XBO e XBC nas versões compostas de X

X: observações sobre as portas

1. Sinais de dados e endereço, que são comuns às diversas implementações de X, ficam armazenados na raíz da hierarquia
2. Este VG adiciona portas para sinais de comando e teste
3. Este VG adiciona portas para sinais de fase do clock
4. Esta V adiciona portas para layout

X: observações sobre as correlações

5. Correlação com XBO-VG1 - alternativa com barramentos
6. Correlação com XBO-VG1-VG2 - alternativa com barramentos e relógio de 4 fases
7. Correlação com XBO-VG1-VG7 - alternativa com barramentos e relógio de 2 fases
8. Correlação com XBO-VG3 - alternativa com multiplexadores
9. Correlação com XBO-VG3-VG4 - alternativa com multiplexadores e relógio de 4 fases

X: observações sobre alternativas de mapeamento para os objetos GARDEN

10. Poderia ser HDL, com referências a XBO e XBC

11. Há duas opções aqui:

    1. se no P8, quando da criação de X-VG1-VG2-V2(MMHD)-M1-I1, foram feitas referências a XBO-VG1-VG2 (deixando a View em aberto) e a XBC-VG1-VG2 (também deixando a View em aberto), então não é necessário criar a V8 de X, já que a avaliação do timing poderá ser feita sobre a mesma V2-M1-I1 de X, mas selecionando-se VG5-V5-M1-I2 para XBO-VG1-VG2 e VG5-V4-M1-I2 para XBC-VG1-VG2 (estas novas I's contêm as informações de timing geradas por *back-annotation*)
    2. se no P8 foram no entanto feitas referências estáticas a XBO-VG1-VG2-V2 e a XBC-VG1-VG2-V1, então é necessária a criação de V8, que referencia outros caminhos em XBO e XBC

5.2 Design XBO

A Figura 4 contém a hierarquia de ViewGroups, Views e M-I's criada para o Design XBO.

XBO: observações sobre as portas

1. Sinais de dados e endereço, que são comuns às diversas implementações de X, ficam armazenados na raíz da hierarquia
2. Este VG adiciona portas para sinais de comando e teste
3. Este VG adiciona portas para sinais de fase do clock
4. Esta V adiciona portas para layout

XBO: observações sobre as correlações

5. VG1-V1-M1-I1 sintetizada de X-V1-M1-I2
6. VG1-VG2-V1-M1-I1 sintetizada de XBO-VG1-V1-M1-I1
7. VG3-V3-M1-I1 sintetizada de X-V1-M1-I2
8. VG3-VG4-V4-M1-I1 sintetizada de XBO-VG3-V3-M1-I1
9. VG1-V1-M1-I2 sintetizada de X-V1-M1-I2
10. VG1-VG2-V2-M1-I2 sintetizada de XBO-VG1-V1-M1-I2
11. VG1-VG2-VG5-V5-M1-I1 sintetizada de XBO-VG1-VG2-V2-M1-I1

XBO[1]	VG1[2] implem. c/barr.	V1 (HDL)[20] c/schedul. e alocação	M1	I1[5]		
				I2[9,22]		
		VG2[3] relógio 4 fases	V2 (HDL)[21]	M1	I1[6]	
					I2[10]	
			VG5 standard-cell	V5(MMHD) composição células	M1	I1[11]
						I2[19,23]
				V6 (LO)[4]	M1	I1[12]
				V12(MMHD) extração elétrica	M1	I1[18]
			VG6 gate-array	V7(MMHD) composição células	M1	I1[13]
				V8 (LO)[4]	M1	I1[14]
		VG7[3] relógio 2 fases	V9 (HDL)	M1	I1[15]	
			VG8 standard-cell	V10(MMHD) composição células	M1	I1[16]
				V11 (LO)[4]	M1	I1[17]
	VG3[2] implem. c/mplex	V3 (HDL) c/schedul. e alocação	M1	I1[7]		
		VG4[3] relógio 4 fases	V4 (HDL)	M1	I1[8]	

Figura 4: Design XBO

12. VG1-VG2-VG5-V6-M1-I1 sintetizada de XBO-VG1-VG2-VG5-V5-M1-I1
13. VG1-VG2-VG6-V7-M1-I1 sintetizada de XBO-VG1-VG2-V2-M1-I1
14. VG1-VG2-VG6-V8-M1-I1 sintetizada de XBO-VG1-VG2-VG6-V7-M1-I1
15. VG1-VG7-V9-M1-I1 sintetizada de XBO-VG1-V1-M1-I1
16. VG1-VG7-VG8-V10-M1-I1 sintetizada de XBO-VG1-VG7-V9-M1-I1
17. VG1-VG7-VG8-V11-M1-I1 sintetizada de XBO-VG1-VG7-VG8-V10-M1-I1
18. VG1-VG2-VG5-V12-M1-I1 extraída de XBO-VG1-VG2-VG5-V6-M1-I1
19. VG1-VG2-VG5-V5-M1-I2 obtida por *back-annotation* de XBO-VG1-VG2-VG5-V12-M1-I1

XBO: observações sobre alternativas de mapeamento para os objetos GARDEN

20. Poderia ser uma View ainda HDL, mas já com referências a alguns Designs (unidades funcionais)
21. Poderia ser
    1. uma View ainda HDL, mas com referências a unidades funcionais e registradores
    2. uma View MMHD
22. Há três opções onde colocar esta nova versão debaixo de VG1. A decisão deve atender um critério subjetivo do usuário.
    1. como nova V
    2. como nova M de V1
    3. como nova I de M1 de V1 (esta foi escolhida)
23. A decisão de se criar uma nova I foi tomada por coerência com a decisão do passo 15 (ver nota anterior)

5.3 Design XBC

A Figura 5 contém a hierarquia de ViewGroups, Views e M-I's criada para o Design XBC.

<table>
<tr><td rowspan="15">XBC[1]</td><td rowspan="13">VG1[2]
implement.
c/barram.</td><td rowspan="9">VG2[3]
relógio
4 fases</td><td>V1(HDL)
máquina
estados</td><td>M1</td><td colspan="2">I1[5]
I2[7,20]</td></tr>
<tr><td rowspan="4">VG5
estratégia
PLA</td><td>V3 (HDL)
c/minim.lóg.</td><td>M1</td><td>I1[8]</td></tr>
<tr><td>V4 (HDL)
c/otim.tecn.</td><td>M1</td><td>I1[9]
I2[19]</td></tr>
<tr><td>V5 (LO)[4]</td><td>M1</td><td>I1[10]</td></tr>
<tr><td>V13(MMHD)
extração
elétrica</td><td>M1</td><td>I1[18]</td></tr>
<tr><td rowspan="3">VG6
estratégia
lógica
aleatória</td><td>V6 (HDL)
c/minim.lóg</td><td>M1</td><td>I1[11]</td></tr>
<tr><td>V7 (HDL)
c/otim.tecn.</td><td>M1</td><td>I1[12]</td></tr>
<tr><td>V8 (LO)[4]</td><td>M1</td><td>I1[13]</td></tr>
<tr><td rowspan="4">VG7[3]
relógio
2 fases</td><td>V9(HDL)
máquina
estados</td><td>M1</td><td colspan="2">I1[14]</td></tr>
<tr><td rowspan="3">VG8
estratégia
PLA</td><td>V10 (HDL)
c/minim.lóg.</td><td>M1</td><td>I1[15]</td></tr>
<tr><td>V11 (HDL)
c/otim.tecn.</td><td>M1</td><td>I1[16]</td></tr>
<tr><td>V12 (LO)[4]</td><td>M1</td><td>I1[17]</td></tr>
<tr><td>VG3[2]
implement.
c/multiplex.</td><td>VG4[3]
relógio
4 fases</td><td>V2(HDL)
máquina
estados</td><td>M1</td><td colspan="2">I1[6]</td></tr>
</table>

Figura 5: Design XBC

XBC: observações sobre as portas

1. Sinais de dados e endereço, que são comuns às diversas implementações de X, ficam armazenados na raíz da hierarquia
2. Este VG adiciona portas para sinais de comando e teste
3. Este VG adiciona portas para sinais de fase do clock

4. Esta V adiciona portas para layout

XBC: observações sobre as correlações

5. VG1-VG2-V1-M1-I1 sintetizada de X-V1-M1-I2 e XBO-VG1-VG2-V2-M1-I1
6. VG3-VG4-V2-M1-I1 sintetizada de X-V1-M1-I2 e XBO-VG3-VG4-V4-M1-I1
7. VG1-VG2-V1-M1-I2 sintetizada de X-V1-M1-I2 e XBO-VG1-VG2-V2-M1-I2
8. VG1-VG2-VG5-V3-M1-I1 sintetizada de XBC-VG1-VG2-V1-M1-I1
9. VG1-VG2-VG5-V4-M1-I1 sintetizada de XBC-VG1-VG2-VG5-V3-M1-I1
10. VG1-VG2-VG5-V5-M1-I1 sintetizada de XBC-VG1-VG2-VG5-V4-M1-I1
11. VG1-VG2-VG6-V6-M1-I1 sinteizada de XBC-VG1-VG2-V1-M1-I1
12. VG1-VG2-VG6-V7-M1-I1 sintetizada de XBC-VG1-VG2-VG6-V6-M1-I1
13. VG1-VG2-VG6-V8-M1-I1 sintetizada de XBC-VG1-VG2-VG6-V7-M1-I1
14. VG1-VG7-V9-M1-I1 sintetizada de X-V1-M1-I2 e XBO-VG1-VG7-V9-M1-I1
15. VG1-VG7-VG8-V10-M1-I1 sintetizada de XBC-VG1-VG7-V9-M1-I1
16. VG1-VG7-VG8-V11-M1-I1 sintetizada de XBC-VG1-VG7-VG8-V10-M1-I1
17. VG1-VG7-VG8-V12-M1-I1 sintetizada de XBC-VG1-VG7-VG8-V11-M1-I1
18. VG1-VG2-VG5-V13-M1-I1 extraída de XBC-VG1-VG2-VG5-V5-M1-I1
19. VG1-VG2-VG5-V4-M1-I2 obtida por *back-annotation* de XBC-VG1-VG2-VG5-V13-M1-I1

XBC: observações sobre alternativas de mapeamento para os objetos GARDEN

20. Há três opções onde colocar esta nova versão abaixo de VG2. A decisão deve atender um critério subjetivo do usuário.
    1. como nova View
    2. como nova M de V1
    3. como nova I de M1 de V1 (esta foi a escolhida)

5.4 Considerações sobre o mapeamento adotado

Foram criadas ao longo do processo de projeto várias composições para X, cada uma delas constando de diferentes versões de XBO e XBC. Nos passos 8, 13 e 35 foram criadas composições com diferentes pares de Views HDL de XBO e XBC, enquanto nos passos 29, 31 e 42 foram criadas composições com diferentes pares de Views Layout de XBO e XBC. No entanto, como estas Views de XBO e XBC, em cada um destes casos, apresentava uma interface diferente, torna-se impossível para o GARDEN armazenar a informação "X é composta por XBO e XBC" num único lugar. (Na realidade há mais em comum a estas Views de X, pois todas as versões referidas de XBO e XBC mantêm constante grande parte dos sinais de interface, e assim muitas das interconexões dentro de X são idênticas). A informação está portanto duplicada em todas as Views de X. Cada uma destas Views tem informações internas adicionais distintas, correspondendo aos sinais de interface de XBO e XBC que são diferentes em cada caso.

No nível RT, existe uma decisão importante a ser tomada no mapeamento. Deve-se escolher entre Views HDL e MMHD para representar o resultado da síntese de alto nível. A descrição resultante já contém muita informação estrutural após as fases de "scheduling" e alocação (quais unidades funcionais serão utilizadas), e após o mapeamento tecnológico já se tem todos os registradores como ocorrências de tipos de registradores definidos numa biblioteca de células. Embora ambos os tipos de Views permitam anotar os objetos referenciados, no caso da View MMHD já se tem a representação, via modelo de dados, de todas as interconexões entre estes objetos, o que permite estabelecer, mais tarde, relações entre os objetos desta View MMHD e os objetos da descrição do bloco operacional obtida pelo gerador de partes operativas, que também será uma View MMHD, mas mais refinada (composição de células de uma biblioteca, tais como portas e flip-flops). (Ver [10] para uma discussão deste "trade-off").

6 Configurações

De acordo com Katz [11], uma *configuração* associa versões particulares aos componentes de uma descrição composta de um objeto. Configurações podem ser *estáticas*, quando estas versões são selecionadas já dentro desta descrição composta, ou *dinâmicas*, quando esta associação é feita posteriormente, por exemplo quando uma ferramenta de projeto deve efetuar algum tratamento sobre esta descrição composta.

GARDEN oferece um mecanismo de configuração dinâmica bastante flexível. Em uma View de qualquer tipo (MMHD, HDL ou Layout) podem ser feitas referências completas a um caminho que vá da raíz de um Design até uma determinada M-I, ou podem ser feitas referências dinâmicas a qualquer ponto intermediário deste caminho (só o Design, ou Design - View Group, etc).

Ao longo do processo de projeto ilustrado neste relatório, configurações dinâmicas foram criadas para descrições compostas de X em vários momentos. Na tabela 2 estão anotadas (R1-R7) estas configurações. Elas são repetidas abaixo, com comentários adicionais.

`P8.` Cria descrição composta de X para alternativas de XBO e XBC com barramentos e relógio com 4 fases. Objeto X-VG1-VG2-V2(MMHD)-M1-I1 contém referências a XBO-VG1-VG2 e a XBC-VG1-VG2

`P13.` Cria descrição composta de X para alternativas de XBO e XBC com multiplexadores e relógio com 4 fases. Objeto X-VG3-VG4-V3(MMHD)-M1-I1 contém referências a XBO-VG3-VG4 e a XBC-VG3-VG4

`P29.` Cria descrição composta do layout de X para alternativa com barramentos, relógio com 4 fases, XBO com standard-cell e XBC com PLA. Objeto X-VG1-VG2-V4(LO)-M1-I1 contém referências a XBO-VG1-VG2-VG5-V6 e a XBC-VG1-VG2-VG5-V5

`P31.` Cria descrição composta do layout de X para alternativa com barramentos, relógio com 4 fases, XBO com standard-cell e XBC com lógica aleatória. Objeto X-VG1-VG2-V5(LO)-M1-I1 contém referências a XBO-VG1-VG2-VG5-V6 e a XBC-VG1-VG2-VG6-V8

`P35.` Cria descrição composta de X para alternativas de XBO e XBC com barramentos e relógio com 2 fases. Objeto X-VG1-VG5-V6(MMHD)-M1-I1 contém referências a XBO-VG1-VG7 e a XBC-VG1-VG7

`P42.` Cria descrição composta do layout de X para alternativa com barramentos, relógio com 2 fases, XBO com standard-cell e XBC com PLA. Objeto X-VG1-VG5-V7(LO)-M1-I1 contém referências a XBO-VG1-VG7-VG8-V11 e a XBC-VG1-VG7-VG8-V12

Quando ferramentas são aplicadas sobre as descrições que contêm estas configurações, as configurações devem ser completadas com referências até M-I's. Estas situações estão relacionadas abaixo.

P2. Simulação do comportamento de X. X-V1 é um objeto primitivo, não estruturado internamente. Basta selecionar a versão (M1-I1 neste caso)

P4. Como em P2. É selecionada M1-I2 para X-V1

P9. Simulação de X, com uma descrição HDL no nível RT para XBO, que já contém escolha de registradores, e com uma descrição na forma de máquina de estados para XBC. Usada descrição de X criada no P8. Selecionadas versões V2-M1-I1 para XBO-VG1-VG2 e V1-M1-I1 para XBC-VG1-VG2

P14. Simulação de X, com uma descrição HDL no nível RT para XBO, que já contém escolha de registradores, e com uma descrição na forma de máquina de estados para XBC. Usada descrição de X criada no P13. Selecionadas versões V4-M1-I1 para XBO-VG3-VG4 e V2-M1-I1 para XBC-VG3-VG4

P18. Simulação da mesma descrição de X criada no P8, mas escolhendo-se, em relação ao P9, novas versões para XBO e XBC, que correspondem ao novo scheduling para a alternativa com barramentos e relógio com 4 fases. Selecionadas versões V2-M1-I2 para XBO-VG1-VG2 e V1-M1-I2 para XBC-VG1-VG2

P30. DRC aplicado sobre layout de X criado no P29. Selecionadas versões M1-I1 para XBO-VG1-VG2-VG5-V6 e M1-I1 para XBC-VG1-VG2-VG5-V5

P32. DRC aplicado sobre layout de X criado no P31. Selecionadas versões M1-I1 para XBO-VG1-VG2-VG5-V6 e M1-I1 para XBC-VG1-VG2-VG6-V8

P36. Simulação de X, com uma descrição HDL no nível RT para XBO, que já contém escolha de registradores, e com uma descrição na forma de máquina de estados para XBC. Usada descrição de X criada no P35. Selecionadas versões V9-M1-I1 para XBO-VG1-VG7 e V9-M1-I1 para XBC-VG1-VG7

P43. DRC aplicado sobre layout de X criado no P42. Selecionadas versões M1-I1 para XBO-VG1-VG7-VG8-V11 e M1-I1 para XBC-VG1-VG7-VG8-V12

P50. Avaliação de timing de X, após *back-annotation*. É usada a mesma descrição de X gerada no P8. Selecionadas versões VG5-V5-M1-I2 para XBO-VG1-VG2 e VG5-V4-M1-I2 para XBC-VG1-VG2

Em função da abordagem de mapeamento do processo de projeto para o modelo de dados, dois tipos de configurações estáticas foram criadas ao longo do projeto:

- configurações com referências até o nível de View, criadas sempre que se tem de composição de layouts. A referência estática até a View é necessária em função do armazenamento dos Ports próprios do nível de abstração "layout"

no nível da View (ver passos P29, P31 e P42). Nestes casos, quando da execução do DRC, é necessário selecionar-se dinamicamente apenas as M-I's.

- configurações com referências até um nível de ViewGroup, criadas sempre que se tem composições de mais alto nível. Nota-se em XBO, por exemplo, que Ports são definidas no nível do Design, do VG1 e do VG2, mas não dos VG5 e VG6, onde a decisão de projeto (escolha de estratégia standard-cell ou gate-array) não afeta a interface do circuito. Uma mesma descrição de X referenciando XBO-VG1-VG2, portanto, pode ser utilizada em dois momentos bem distintos. No passo P9 é feita uma simulação de uma especificação inicial, ainda no processo descendente de projeto, onde para XBO é selecionada uma descrição comportamental HDL, antes da escolha da estratégia de implementação. No passo P50, já na avaliação de timing final do layout gerado, é selecionada para XBO uma descrição MMHD (composição de células na estratégia standard-cell, anotada pelo timing extraído do layout).

Constata-se aqui um compromisso entre duas abordagens no tratamento de configurações. Por um lado, é conveniente armazenar-se os Ports nos níveis mais altos possíveis da hierarquia de ViewGroups e Views de um Design Y. Isto permitirá, numa View X_i de outro objeto X, uma referência estática a Y com a maior flexibilidade possível de escolha de ViewGroups e Views no momento da ativação de uma ferramenta de avaliação. Com isto são economizadas descrições alternativas do objeto X. Um problema pode ocorrer se os ViewGroups e Views a serem selecionados acrescentarem Ports a Y. VHDL [8], por exemplo, permite tal situação. Os Ports adicionais da View escolhida são simplesmente desconsiderados.

Deve-se atentar aqui para o tipo de análise de X que vai ser realizada. Se ela exige que todos os Ports de Y sejam considerados, então a View X_i não poderá ser utilizada, e uma nova descrição terá que ser feita, fazendo referência à View de Y que contém todos os Ports. Seria útil que GARDEN oferecesse um mecanismo, acionado opcionalmente pela aplicação, que automaticamente verificasse se há um casamento perfeito entre os Ports do componente em X_i e os Ports da View selecionada para Y. Na especificação até aqui feita do ambiente GARDEN, esta questão não foi considerada.

Por outro lado, esta abordagem de modelagem na qual se dá preferência ao armazenamento de Ports nos níveis mais altos da hierarquia de ViewGroups aumenta o número de configurações a serem geradas dinamicamente pelas ferramentas. O modelo de dados do GARDEN, no entanto, não prevê o armazenamento de *descrições de configurações* de objetos definidas dinamicamente. Em VHDL, por exemplo, pode-se definir e armazenar "corpos configuracionais", que podem ser reutilizados posteriormente. Da mesma forma que os objetos auxiliares necessários às tarefas de geração de vetores de teste, análise de testabilidade e simulação, conforme explicado na seção 7, configurações só poderiam ser armazenadas em GARDEN como objetos auxiliares em diretórios particulares dos projetistas, caso as ferramentas oferecessem algum apoio específico.

7 Considerações sobre vetores de teste e análise de testabilidade

As tarefas de projeto relacionadas à geração de vetores de teste e à análise de testabilidade, com a conseqüente inserção de estruturas de teste no circuito, não são apoiadas pelo modelo de dados do GARDEN, que não permite a criação de objetos auxiliares e a manutenção de relações entre estes objetos e os objetos primários do modelo (Designs, ViewGroups, Views).

A tarefa T3 gera casos de teste a partir da descrição comportamental inicial do circuito. Estes casos de teste deveriam ser armazenados em algum objeto auxiliar relacionado ao objeto em questão (uma M-I de X-V1). Posteriormente à síntese de alto nível do circuito, quando XBO está descrito como uma estrutura RT (uma M-I do objeto XBO-VG1-VG2-V2, por exemplo) e XBC como uma máquina de estados (uma M-I do objeto XBC-VG1-VG2-V1, por exemplo), parte-se de uma composição destes objetos e dos casos de teste gerados anteriormente para, na tarefa T11, gerar os vetores de teste para esta implementação particular.

Estruturas de teste (T15) são acrescentadas a uma descrição do bloco operacional já na forma de uma composição de células numa estratégia particular (uma M-I do objeto XBO-VG1-VG2-VG5-V5, por exemplo). Estas estruturas de teste são geradas a partir de informações de testabilidade que foram extraídas, na T10, da mesma composição de XBO e XBC citada no parágrafo anterior. Também estas informações de testabilidade deveriam ser armazenadas num objeto auxiliar relacionado àquela descrição composta de X.

A solução parcial para estes problemas é o armazenamento destes objetos auxiliares em diretórios livremente administrados pelos projetistas. O sistema GARDEN não apoiará diretamente o projetista na manutenção das relações entre estes objetos auxiliares e os objetos primários, mas as ferramentas poderão ser implementadas para tirar proveito de alguma organização inteligente adotada para o armazenamento de objetos nos diretórios.

Esta mesma solução deve ser adotada no tratamento de objetos auxiliares necessários à gerência do processo de simulação, tais como estímulos externos, estruturas de dados simuláveis ("modelos de simulação") e estados iniciais de modelos.

8 Conclusões e trabalhos futuros

Este trabalho estabeleceu um método para análise de mapeamentos entre metodologias de projeto de circuitos VLSI e o modelo de dados do GARDEN. Este método deve ser utilizado na análise de outras metodologias de projeto e de outros mapeamentos possíveis, assim como na análise de várias seqüências de criação de objetos para uma mesma metodologia de projeto e um mesmo mapeamento. Em [12], por exemplo, é sugerido de que forma poderia ser feito um mapeamento dos modelos de gerenciamento de versões dos sistemas OCT [3] e DAMASCUS [9] e da linguagem VHDL [8] para o GARDEN. Em particular, devem ser analisadas metodologias de projeto que explorem mais as decisões de projeto nos níveis inferiores de abstração e que manipulem informações de natureza geométrica, aspectos estes não considerados na metodologia de projeto sugerida neste relatório.

Destas análises deve-se obter um conhecimento aprofundado das vantagens e desvantagens do modelo de dados do GARDEN no aspecto do controle da evolução de projeto. Apesar de sua flexibilidade, o modelo GARDEN estabelece determinadas relações entre os objetos e determinadas restrições de integridade, e está baseado num mecanismo de herança de atributos através da hierarquia de representação de um objeto complexo. Estas características do modelo devem favorecer determinadas abordagens de mapeamento em detrimento de outras. Provavelmente, este fato é ainda influenciado pela metodologia de projeto de CIs utilizada. Espera-se que novos estudos baseados no método de análise aqui esboçado contribuam para o entendimento destes fatores.

A partir da compreensão das vantagens e desvantagens do modelo GARDEN deve-se poder estabelecer uma metodologia genérica para a modelagem de dados, que oriente futuros usuários na definição de mapeamentos particulares.

Este mapeamento entre os passos da metodologia de projeto adotada e as operações sobre os objetos do modelo de dados do GARDEN cria implicitamente um esquema conceitual da aplicação. Não foi objetivo deste trabalho formalizar o mecanismo pelo qual este mapeamento pode ser definido pelo usuário. A existência deste mecanismo é no entanto essencial para que ambientes para aplicações particulares sejam implementados sobre o GARDEN.

Duas direções podem ser investigadas para a definição deste mecanismo. Em primeiro lugar, podem ser estudados os **gerenciadores de projeto** [13,14], ferramentas que permitem ao usuário controlar a execução de ferramentas de projeto dentro de um ambiente e que podem implementar um mapeamento entre as entidades manipuladas pelas ferramentas de projeto e os objetos de um modelo de dados. Em segundo lugar, e talvez adicionalmente ao mecanismo anterior, deve-se estudar a possibilidade de oferecimento de uma linguagem de alto nível para definição de esquemas conceituais, orientada para aplicações de projeto de circuitos VLSI e, em particular, para o modelo de dados suportado pelo GARDEN.

Referências

[1] J.Granacki, D.Knapp e A.Parker. The ADAM advanced design automation system: overview, planner, and natural language interface. 22nd DAC, 1985

[2] W.D.Smith et al. FACE core environment: the model and its application in CAE/CAD tool development. 26th DAC, 1989

[3] D.S.Harrison et al. Data management and graphics editing in the Berkeley design environment. ICCAD 1986

[4] K.Gottheil et al. The CADLAB workstation CWS - an open, generic system for tool integration. In F.J.Rammig (ed.). Tool Integration and Design Environments. North-Holland, 1988

[5] L.Claesen et al. Open framework of interactive and communicating CAD tools. Mesma referência de [4]

[6] F.R.Wagner, C.Freitas e L.G.Golendziner. The AMPLO system - an integrated environment for digital systems design. Mesma referência de [4]

[7] E.de la Quintana, G.Annarumma e P.Molinari Neto. GARDEN - the design data interface. Technical Report CCR-107. IBM, Centro Científico Rio, outubro 1990

[8] IEEE Standard VHDL Language Reference Manual. IEEE, New York, 1988

[9] J.A.Mülle, K.R.Dittrich e A.M.Kotz. Design management support by advanced database facilities. Mesma referência de [4]

[10] F.R.Wagner. Integrating a VHDL dialect into a design framework. Artigo submetido ao CHDL 91.

[11] R.H.Katz et al. Design version management. IEEE Design & Test, fevereiro 1987

[12] F.R.Wagner, A.H.V.de Lima, G.O.Annarumma, E.B.de la Quintana e P.Molinari Neto. Design version management in the GARDEN framework. Artigo submetido à 28th DAC

[13] D.W.Knapp e A.C.Parker. The ADAM planning engine. 23rd DAC, 1986

[14] M.L.Bushnell e S.W.Director. VLSI CAD tool integration using the Ulysses environment. 23rd DAC, 1986