

# Software Livre no Ensino de Sistemas Digitais e Arquitetura de Computadores

Ricardo S Ferreira<sup>1\*</sup>, Ulisses Chieppe<sup>1</sup>, Giliardo C Freitas, Cristiano Biancardi

<sup>1</sup>Universidade Federal de Viçosa, Departamento de Ciência da Computação  
Viçosa, MG Brasil CEP 36570 000

(cacau, uchieppe, gfreitas, cbianc)@dpi.ufv.br

**Abstract.** *This work presents the state-of-art of public domain tools designed as aids for teaching undergraduate digital logic system and computer architecture. The main focus is on portability, remote-access, open source, simulation performance and feasibility. A subset of tools was chosen, and teaching material and practical exercises have been developed. New features have been added to improve the educational and didactic resources. The chosen tools were Applets, Diglog and Hades (digital logic), Misim and Sashimi (Microcontroller), DLX (Architecture), SIS (Logic Synthesis). Most of tools have been written on Java, which is multi-platform.*

**Resumen.** *Este trabalho apresenta o estado da arte das ferramentas de domínio público para o ensino de lógica digital, de organização e arquitetura de computadores. A ênfase foi dada a portabilidade, ao acesso remoto, ao código aberto e ao desempenho e robustez da simulação. Um sub-conjunto de ferramentas foi selecionado e uma série de exemplos didáticos foi elaborada. Recursos complementares foram desenvolvidos para melhorar algumas ferramentas existentes. As ferramentas selecionadas foram Applets, Diglog e Hades (nível lógico e organização de computadores), Misim e Sashimi (Microcontroladores), DLX (Arquitetura), SIS (Síntese Lógica). A maioria das ferramentas tem o código aberto e foram implementadas em Java, ou seja, são independentes da plataforma.*

## 1. Introdução

Um sistema digital pode ser especificado pelo seu comportamento ou pela sua estrutura. As formas mais usadas são as descrições esquemáticas, os diagramas de estado, as linguagens de descrição de hardware (HDL) e as linguagens de programação. Vários softwares vem sendo utilizados no ensino das disciplinas de hardware nos cursos de graduação permitindo aos estudantes o desenvolvimento de projetos de pequeno e médio porte.

Um estudo recente [Bormida et al., 1997] aponta quatro metodologias aplicadas ao ensino de lógica digital e arquitetura: expositiva, demonstrativa, interativa e prática. A metodologia expositiva pode ser implementada através de um hipertexto com recursos de animações, introduzindo os conceitos fundamentais. Na metodologia demonstrativa, os conceitos introduzidos podem ser reforçadas mostrando exemplos práticos de uso. Um circuito pode ser carregado dentro de um simulador para que o estudante altere os sinais de entrada e observe o comportamento das saídas.

Na metodologia interativa, o estudante tem que fazer escolhas, elaborar uma descrição ou realizar algum cálculo. Nas abordagens demonstrativas e expositivas, muitas vezes o estudante percorre o material tendo apenas uma visão superficial. A interação obriga a tomada de decisões por parte dos estudantes. O uso de testes e exercícios com auxílio de simuladores é uma boa metodologia para avaliar se o conhecimento foi realmente transmitido. A última metodologia é a prática, onde o estudante deve gerar um projeto. Foi observado por [Bormida et al., 1997] que o uso de ferramentas mais simples é mais adequado. A ferramenta pode estar restrita a um problema local, como a síntese em dois níveis usando método de karnaugh. Os resultados experimentais do estudo [Bormida et al., 1997], mostram que a maioria dos estudantes preferem a prática de experimentos que a leitura de hipertexto ou a visualização de animações, como comprovam os questionários aplicados aos próprios estudantes.

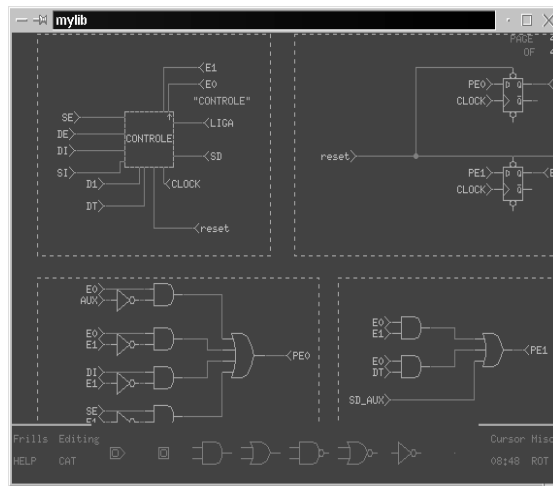
O objetivo deste trabalho foi o estudo das ferramentas de domínio público (software livre) dando prioridade a portabilidade e eficiência da simulação. A ênfase foi dada ao suporte as metodologias interativa e prática. O software livre oferece oportunidades para um público maior (na maioria das vezes sem custo), e permite ao estudante contribuir para melhorar o software (código aberto), entendendo o funcionamento interno e desmistificando as caixas pretas. A economia gerada na compra do software pode ser utilizada para aquisição de equipamentos ou alguns softwares fundamentais que não tenham equivalentes no domínio público. Além disso, incentiva o desenvolvimento da tecnologia local e reduz a dependência externa. A plataforma Cliente/Servidor e acesso remoto via internet também vem sendo alvo do estudo para ampliar o uso (educação a distância) e facilitar a instalação local. Uma tendência recente é o uso de ambientes integradas com browsers, como o sistema DEEDS da Universidade de Genova [Donzellini and Ponta, 2001] e o sistema PUNCH [Figueiredo et al., 2001] (Universidades: Purdue, Chicago e Northwestern).

Dentre as várias ferramentas analisadas, a ênfase foi dada as implementações em Java, tendo como principal atrativo a independência de plataforma (Windows ou Linux/Unix) e acesso remoto (Applets, etc.). Para avaliar as ferramentas vários projetos didáticos foram implementados. Dentre as ferramentas selecionadas, alguns recursos foram implementados e uma integração entre elas foi desenvolvida. Pode-se destacar as seguintes ferramentas: o Hades (editor e simulador), o SIS (síntese e simulação) e o simulador MISIM. Para o Hades [Hendrich, 1998b], os recursos implementados foram: a integração com o SIS [Sentovich et al., 1992] e com o Misim, um gerador de código para criação de componentes, a migração de componentes do Misim [Technology, 2000] e o componente porta paralela. Para o SIS foi desenvolvido um gerenciador de componentes do Misim e um gerenciador dinâmico de interface de comandos dentro de um ambiente cliente/servidor. Para o simulador Misim de microcontroladores, alguns componentes didáticos foram desenvolvidos como um motor de passo, um elevador e um semáforo, todos com interface gráfica e animações. Além disso, encontra-se em desenvolvimento a integração do Hades com a ferramenta Sashimi [Ito, 2000].

Nas seções seguintes, o estado da arte das ferramentas será apresentado abordando desde do ensino introdutório de sistemas lógicos até o nível de arquitetura.

## **2. Ensino de Lógica Digital**

O uso de simuladores esquemáticos é a abordagem mais utilizada para a introdução a eletrônica digital. Existem muitos editores de esquemáticos, porém poucos possuem uma simulação eficiente e confiável. As ferramentas educacionais comerciais mais populares são o Logicworks e WorkBench. Nosso interesse foi focalizado nas ferramentas de

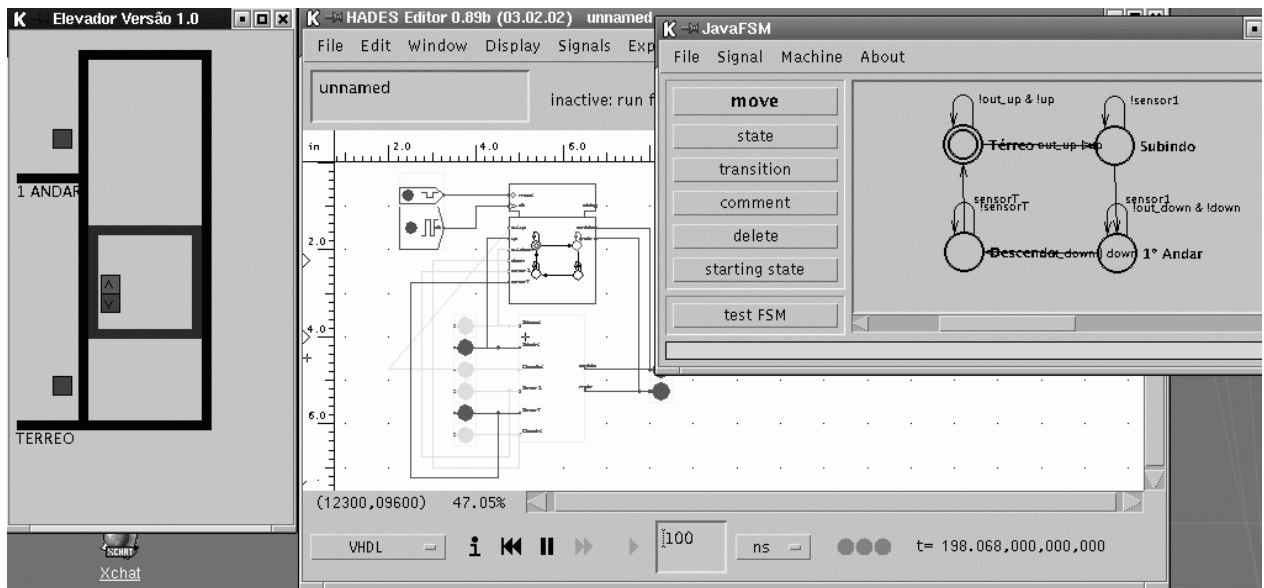


**Figure 1: Diglog - Tela de Edição/Simulação com o exemplo de controle de elevador**

domínio público. Várias ferramentas foram avaliadas, podendo-se destacar as seguintes: Diglog [Gillespie and Lazzaro, 1996], Digitcl [Craig, 1996], Retro [Bräunl, 1998], Hades [Hendrich, 1998b], Tkgate [Hasen, 2000]. Os critérios para avaliação foram: biblioteca de componentes, hierarquia, simulação, interface, desempenho e exemplos.

O Diglog[Gillespie and Lazzaro, 1996], do projeto Chipmunk da Universidade de Berkeley, se destaca pela simplicidade da interface. Possui uma biblioteca de componentes bem completa. O usuário pode ainda desenvolver novos componentes através do software Loged. É um software gratuito e aberto, desenvolvido em C, disponível tanto para Unix, Linux como também para Dos/Windows. A grande vantagem é sua eficiência e robustez na simulação, mesmo sendo executado em máquinas com baixo poder de processamento como um PC 486. Possui o conceito de hierarquia, porém tem algumas limitações (máximo de oito definições). O Diglog é adequado ao ensino de portas, decodificadores, multiplexadores, circuitos aritméticos, registradores e máquinas de estado. Além disso, pode ser usado para o projeto de pequenos processadores, porém apresenta algumas limitações, como a representação explícita dos barramentos. Este problema pode ser contornado com o uso de rótulos. Pode também ser utilizado para ilustrar como as portas lógicas são implementadas com chaves ou transistores, ensino de conversores A/D e D/A combinando componentes analógicos e digitais. A eficiência do Diglog foi comprovada através da implementação de uma grande variedade de circuitos descritos em vários livros textos de eletrônica digital [Mano, 1991, Hayes, 1993, Katz, 1994] e do seu uso intensivo nos 3 últimos anos nos cursos de Organização de Computadores da Universidade Federal de Viçosa [DPI/UFV, 2003a]. A Figura 1 mostra uma tela do Diglog, onde foi projetado um circuito de controle para um elevador baseado em máquina de estados. A implementação usa flip-flops e portas lógicas. Este exemplo será trabalhado ao longo do texto, sendo também implementado em outras ferramentas como o Hades e Misim, ilustrando outras abordagens.

Um outro software, de simples instalação, é o Digitcl [Craig, 1996], possui uma interface em Tcl/Tk e implementação em C++. Entretanto, possui poucos recursos e fica restrito a circuitos introdutórios. No mesmo nível temos o Digsim [Rienen, 1996] implementado como uma Applet em Java. A grande vantagem é a sua simplicidade. Entretanto, não possui hierarquia e também fica limitado a pequenos circuitos. Possui vários exemplos, podendo ser usado com introdução aos circuitos básicos: portas, flip-flop, registros e somadores. Tem a vantagem de possibilitar acesso remoto via browser sem a necessidade de nenhuma instalação local, o que não é possível com o Diglog nem o Digitcl. Também desenvolvido em Java, temos a ferramenta



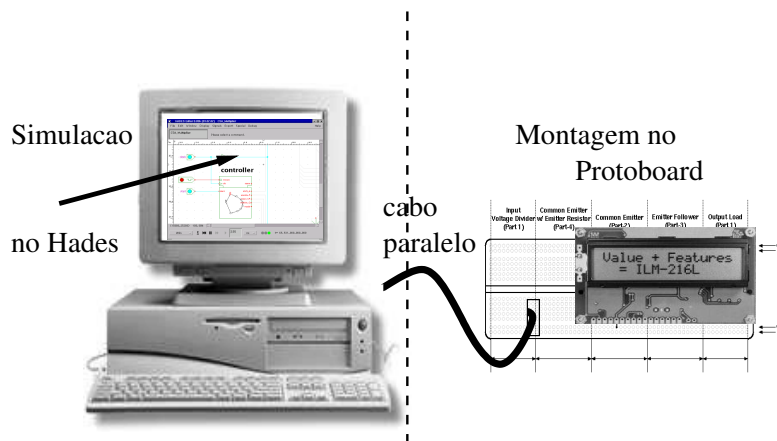
**Figure 2: Editor/Simulador Hades com exemplo de Máquina de estados e o novo componente Elevador**

Retro [Bräunl, 1998], porém é mais adequado para o nível de registro (RTL: Register Transfer Level). Além da portabilidade, o Retro possui uma biblioteca de componentes específica para o projeto de pequenos processadores e unidades aritméticas. É um software aberto, permite ao usuário definir novos componentes (a descrição é um código Java), porém o código é longo e a definição de uma simples porta AND fica complexa. Sua grande vantagem é o uso da orientação a objeto para configurar as propriedades dos componentes (ex: número de bits de um somador) ou conexões. Entretanto a simulação é lenta e não é confiável.

Outra possibilidade é o software Tkgate [Hasen, 2000] que pode ser usado tanto para o nível lógico como para o nível RT. Possui uma biblioteca de componentes e através do conceito de hierarquia possibilita ao usuário a criação de módulos. A interface é simples, assim como o Retro, possui o conceito de propriedades para configurar componentes. Além disso, possui comandos para gerar macro e micro instruções, o que facilita o projeto de processadores. A distribuição ainda está limitada à Linux/Unix e sua interface é baseada em Tcl/Tk.

O Hades [Hendrich, 1998b], que recebeu o prêmio European Academic Software Award em 1998 [Hendrich, 1998a], se situa na mesma categoria do Tkgate, porém foi desenvolvido em Java e oferece mais recursos: uma maior portabilidade, a definição aberta componentes (código Java), o suporte para máquina de estado e para alguns processadores populares. Para máquina de estado, possui um editor gráfico que permite especificar o diagrama de estados, que por sua vez se torna um módulo no circuito.

A Figura 2 ilustra um circuito para controlar um elevador implementado com uma máquina de estados. Se a máquina for selecionada, uma janela de edição do diagrama de estados se abre (janela à direita). Uma das contribuições deste trabalho, foi o desenvolvimento de circuitos como este, que são exemplos clássicos usados em vários livros textos. Além disso, o componente elevador foi desenvolvido para ilustrar ao estudante como criar seus próprios componentes (interface animada, janela à esquerda) e acrescentá-los ao simulador. O conceito de associar uma interface a um componente dá uma excelente flexibilidade ao Hades. Vários componentes de entrada e saída (E/S), como por exemplo, o elevador ou um terminal para protocolo serial estão disponíveis. A simulação conjunta de hardware e software também está



**Figure 3: Componente Porta Paralela ligando o simulador a um protoboard externo (circuito real)**

disponível para o desenvolvimento de sistemas embutidos. Um exemplo é a distribuição do microcontrolador PIC da Microchip, onde é permitido a edição do assembler. Na seção 4, graças a outra contribuição deste trabalho, a ferramenta Misim foi acoplada ao ambiente, dando mais robustez e eficiência. O Hades exporta código em linguagem VHDL, facilitando a introdução a ferramentas mais avançadas.

Componentes como memória, microcontroladores, entre outros, podem se comunicar com arquivos e ter sua configuração alterada facilmente. Como já mencionado, neste trabalho, várias contribuições foram adicionadas ao Hades. Nesta seção 2.1 iremos descrever o componente porta paralela, alguns exemplos de componentes didáticos e um gerador de código para criação de componentes. Na seção 4, iremos descrever a conexão do Hades e Misim. Finalmente, na seção 6 descreveremos a integração do Hades, Sis e componentes do Misim.

## 2.1. Contribuições ao Hades

Qualquer componente no Hades é derivado de uma classe base. Podemos destacar três métodos importantes: construtor, evaluate e wake-up. Quando o objeto é criado, o constructor define as entradas e saídas e sua inicialização, em suma define a interface externa do componente. Já o método evaluate define o comportamento do objeto ao receber qualquer estímulo nas entradas, as novas saídas podem ser re-calculadas e escalonadas para serem ativadas em um instante preciso da simulação. A grande vantagem é que a descrição é feita em Java. Todos os recursos de orientação a objeto estão disponíveis, além de uma rica biblioteca de métodos para manipulação de sinais ou barramentos. O método wake-up permite ao componente injetar um sinal, sem depender de estímulos externos no momento que ele desejar. Um exemplo de componente é um gerador de clock. Um componente pode ter sinais binários ou barramentos, que podem ser entradas, saídas ou bi-direcionais.

Para facilitar a criação de um componente, uma das contribuições deste trabalho foi a criação de um gerador de código. A documentação original do Hades não ilustra exemplos com barramentos ou sinais bi-direcionais. Já o gerador de código proposto neste trabalho, ilustra estes casos. O estudante define apenas a interface (número e configuração das entradas e saídas) e o esqueleto do código Java é criado. Depois basta escrever o método evaluate, com a descrição comportamental. O código abaixo define o comportamento de um somador/multiplicador com dois barramentos de entrada e um bit de controle.

```
// The evaluate() method defines the individual behavior
```

```

    public void evaluate( Object arg ) {
    ...
    if( value_port_PIN.is_0( ) ) {
        value_port_OUT = value_port_BIN1.add(value_port_BIN2);
    } // final do if
    else //
    {
        value_port_OUT = value_port_BIN1.mult(value_port_BIN2);
    }
    EnviarBarramento(port_OUT, value_port_SOMA);
    ...
    }

```

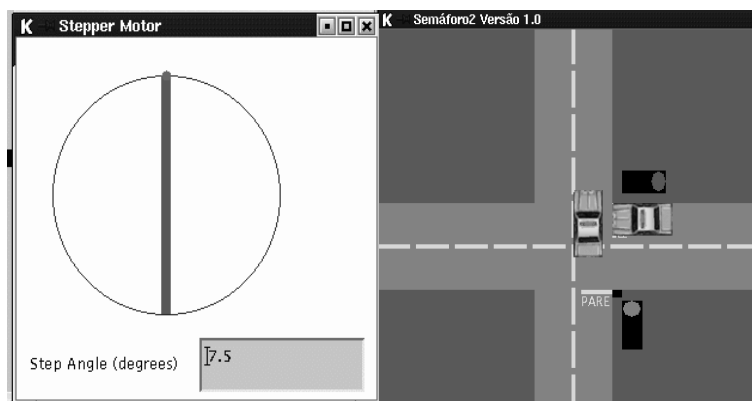
Outra contribuição foi a criação de um componente que permite comunicar o simulador com qualquer componente real, externo ao computador. O circuito descrito no simulador é conectado ao mundo externo através dos pinos da porta paralela. Um método nativo em C++, encapsulado em Java, foi implementado com esta finalidade. Assim pode-se montar parte do circuito no protoboard e parte no simulador e ambos se comunicam. Como exemplo foi realizada a montagem de um display LCD conectado a porta paralela, onde o display é controlado pelos sinais injetados no simulador. Além disso, o simulador dispõe de um display LCD virtual, assim pode-se testar ao mesmo tempo, um componente real e um virtual. Esta funcionalidade aumenta bastante os recursos que podem ser utilizados durante aulas práticas de circuitos digitais. A figura 3 ilustra o exemplo do LCD real conectado via porta paralela ao simulador.

Para gerar componentes com interface gráfica, como será descrito na seção 4, alguns componentes didáticos com interface como um sinal de trânsito, um elevador e um motor de passo, que foram criados inicialmente para o Misim, foram adaptados para o Hades. Estes componentes ilustram ao estudante como criar um objeto com interface gráfica animada. Outro componente didático foi o circuito serial para medida de temperatura da Dallas, o DS1620, que é um exemplo de como implementar um dispositivo com interface serial, semelhantes aos protocolos mais recentes como 1-wire e I2C. A figura 4 ilustra dois componentes criados: o semáforo e o motor de passo. O motor de passo segue o protocolo de comando de um motor unipolar. Outros modelos de comportamento, como por exemplo o bipolar e o multifase, podem ser implementados e a interface gráfica do motor ser re-utilizada. O exemplo do semáforo pode ser estendido também. Nesta implementação, além dos sinais de trânsito e de um gerador de carros, existe um sensor que detecta a presença de um carro nas proximidades do sinal.

### 3. Linguagens de Alto Nível

Dada a complexidade dos circuitos atuais, a especificação por meio de esquemático pode ser ineficiente (trabalhosa) e sujeita a erros. As linguagens de hardware oferecem recursos para modelagem em alto ou baixo nível. Uma pequena descrição em alto nível pode ser mapeada em uma complexa e eficiente implementação em hardware, tudo depende da ferramenta de síntese (equivalente ao compilador para software).

VHDL e Verilog são as linguagens mais usadas, sendo padrões definidos e aceitos por várias ferramentas. Entretanto, as ferramentas comerciais para circuitos integrados tem um alto custo. Algumas versões públicas estão disponíveis, como o software Alliance [Greiner et al., 1994]. Além disso, os softwares são complexos, voltados para profissionais que atuam na área e tem como maior preocupação a funcionalidade e o desempenho, sendo pouco



**Figure 4: Alguns dos Novos componentes: Motor de Passo, Semáforo**

didáticos e mais adequados a cursos avançados [Hendrich, 1998b]. O tempo necessário para o domínio do software pode ser longo para ser aplicado em uma disciplina.

Na década de 90, os dispositivos reprogramáveis (Field Programmable Gate Arrays-FPGA ou Complex Programmable Logic Devices-CPLD) se popularizaram [Venkateswaran and Mazumder, 1994, Compton and Hauck, 1999, Cong and Xu, 2000]. Kits didáticos tem um baixo custo e permitem ao estudante projetar e implementar em hardware programável, sistemas simples (alguns dezenas de portas) ou complexos como um processador de 32 bits (milhares de portas). O mercado é dominado por duas companhias: Altera [Altera, 2003] e Xilinx [Xilinx, 2003]. Ambas oferecem programas universitários com doação ou venda subsidiada.

A grande vantagem desta solução é o poder de descrição das linguagens de alto nível, a simulação virtual e a verificação real com a geração no hardware reprogramável. Os dispositivos reprogramáveis são uma tendência atual tanto para ensino como para pesquisa. No ensino, uma desvantagem do uso de ferramentas comerciais fechadas, além da dependência dos fabricantes (renovação das licenças), é o fato do processo de síntese se tornar uma caixa preta, desta forma o estudante não compreende bem como a especificação é transformada em hardware, quais são as formas eficientes de descrição e quais são as limitações dos compiladores de hardware. A compreensão é importante porque novas tecnologias reprogramáveis vêm surgindo, gerando uma grande demanda de novos métodos de síntese [Cong and Xu, 2000].

É importante ainda, destacar que VHDL é uma linguagem muito grande e nem todas as descrições são sintetizáveis pelas ferramentas, sendo que algumas usam subconjuntos ou versões proprietárias da linguagem. Na seção 6, o uso de ferramentas de síntese será abordado.

Outra alternativa para descrição de hardware é a utilização de linguagens de programação como C, C++, Java, etc. A vantagem é que o estudante já tem o domínio da linguagem e dos compiladores. Algumas ferramentas de CAD fazem a conversão para linguagens de hardware como VHDL ou Verilog [Ito, 2000, Michelli, 2000], o código em geral fica bem menor e mais legível que em VHDL.

A descrição de um componente no Hades pode ser tanto estrutural como comportamental, introduzindo os dois conceitos. Se o estudante usa o editor para criar um circuito interligando os componentes, uma descrição estrutural é realizada, como já foi mencionado, e pode ser hierárquica. Além disso, como descrito na seção 2.1, um componente pode ser criado a partir de uma descrição comportamental em Java, usando todos os recursos da linguagem e as diversas classes do Hades que facilitam tanto a descrição como a criação de uma interface

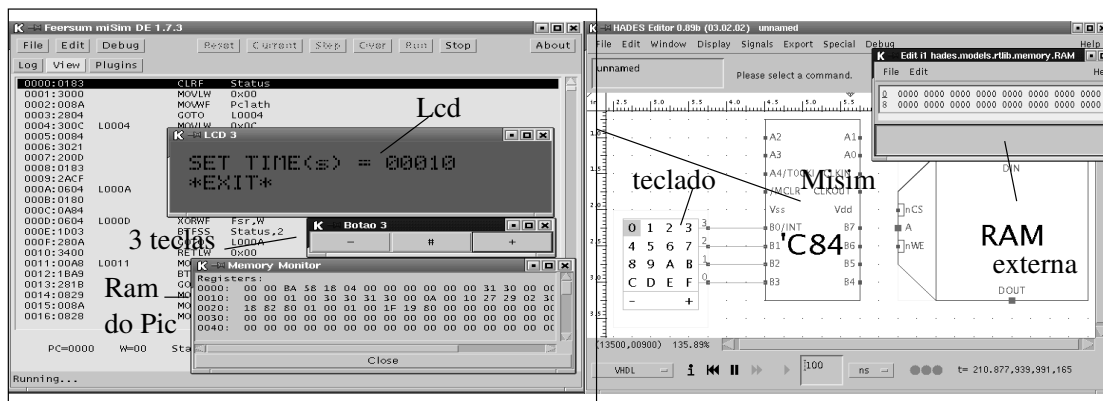


Figure 5: Misim integrado ao Hades

gráfica.

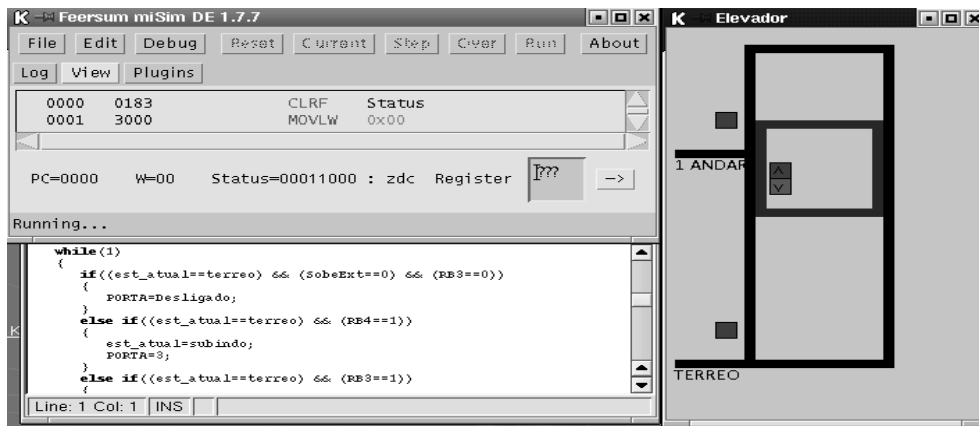
#### 4. Microcontroladores

Outro ponto importante, é o ensino de linguagens assembler nos cursos de organização de computadores. Existem diversos simuladores. Uma abordagem interessante é a utilização de microcontroladores. Por exemplo, os microcontroladores da família PIC da Microchip, além de um baixo custo, permitem ao estudante gravar seu programa no microcontrolador e implementar equipamentos simples que podem comunicar com um PC via porta serial ou paralela, ler teclas, piscar leds, controlar um display de LCD, conectar-se a memórias externas, fazer interface com conversores Analógicos/Digitais. Existem diversos simuladores de microcontroladores, dentre os quais, foi selecionado o Misim [Technology, 2000] que foi implementado em Java e possui uma biblioteca aberta de componentes externos.

O MiSim é um simulador dedicado a alguns microcontroladores da família PIC, sem editor de esquemáticos. Os módulos de entrada e saídas, como um teclado, definidos para o Misim podem ser adaptados para outros simuladores. Além dos módulos existentes na versão 1.7.3, outra contribuição desde trabalho foi o desenvolvimento de novos módulos didáticos. Entre eles podemos citar o display LCD padrão Hitachi, o elevador, o sinal trânsito e o motor de passo. A Figura 4 ilustra alguns dos módulos desenvolvidos para o Misim e depois adaptados para o padrão do Hades, como já mencionado na seção 2.1.

Uma grande vantagem do Misim é a simulação em tempo real. O Hades também possui o componente microcontrolador PIC, porém devido a sobrecarga da definição, a simulação de código PIC no Hades é muito lenta. Outra contribuição ao Hades/Misim, foi o desenvolvimento de uma comunicação entre os dois simuladores, assim o estudante carrega o programa assembler no Misim, e configura a conexão com outros componentes no Hades usando a interface gráfica de edição. Este novo ambiente é simples e útil no ensino de sistemas embutidos. A figura 5 mostra a integração do Misim ao Hades. A janela do Misim exibe o código, alguns componentes (como a memória interna, lcd e 3 teclas) podem ser visualizados. Além disso, toda a biblioteca de componentes do Hades pode ser conectada as entradas e saídas do circuito. A figura 5 ilustra o componente C84 (que representa o Misim dentro do Hades), um teclado e uma memória RAM externa.

Para não ficar limitado ao assembler de um microcontrolador específico, programas escritos em C podem ser compilados para qualquer assembler alvo. Vários exemplos de pequenos equipamentos foram implementados, como relógios, cronômetros, controle de elevador, sinal



**Figure 6: Controle de Elevador - Fonte em C, compilado para assembler PIC no Misim**

de trânsito entre outros que foram descritos em C, compilados para o assembler PIC e simulados no ambiente Misim. Um contador de intervalos reconfigurável para disparo de uma câmera fotográfica área foi descrito, simulado e implementado, ilustrando a viabilidade da plataforma. Este equipamento foi usado em vôos na região amazônica e na Antártica sem problemas. A figura 6 ilustra o exemplo do elevador, onde o controle foi escrito em C, compilado para assembler PIC e simulado no Misim.

Uma tendência atual é o uso de Java, um exemplo é a ferramenta Sashimi [Ito, 2000], que é voltada para o projeto de sistemas embutidos. O código é transformado para byte codes. Pode ser simulado em qualquer ambiente de programação Java. Um analisador de código simplifica o código para um gerador automático de microcontrolador. A descrição do microcontrolador é gerada em VHDL que pode ser processada por ferramentas para FPGA (como Altera/Xilinx) ou VLSI (como Alliance). A grande vantagem desta abordagem é a independência do microcontrolador, que é gerado on-the-fly. Além do Sashimi, existe o simulador Sashimiul, que foi desenvolvido para a conexão com os dispositivos de E/S. O código é aberto e permite o desenvolvimento de novos componentes. A distribuição vem com alguns componentes como teclado, led e display. Encontra-se em desenvolvimento a integração do Sashimi/Sashimul com o Hades, o que amplia ainda mais o ambiente. Como vantagem para o Sashimi, tem-se uma ampliação dos recursos de simulação e a geração de componentes externos em VHDL. Já para o Hades, pode-se usar um padrão de mais alto nível, descrito em Java, para especificar o software de controle em projeto de sistemas embutidos.

## 5. Nível Arquitetura

Existem inúmeras ferramentas para simulação a nível de arquitetura. Variando desde dos primeiros processadores como a família Z80 até os processadores RISC e superescalares. O tópico é longo e esta seção apresenta apenas uma visão geral da área e algumas sugestões de softwares livres. Um ambiente de integração de ferramentas para ensino de arquitetura bem como uma avaliação dos resultados em diversas universidades pode ser encontrado em [Figueiredo et al., 2001]. Como exemplo, iremos adotar o processador DLX que possui várias ferramentas para simulação.

O livro [Patterson and Hennessy, 1996] é um dos mais utilizados para o ensino das arquiteturas RISC, das técnicas de cache e pipeline. O processador DLX Risc de 32bits é definido e ilustra os conceitos e as técnicas de otimização. Dentre os exemplos de fer-

ramentas, temos desde applets acompanhadas de tutoriais no WEB, simuladores a nível de assembler, simuladores mostrando a estrutura interna e simuladores avançados de alta performance [Patterson and Hennessy, 1999].

Todos os níveis de ensino podem ser cobertos com o auxílio dos simuladores. Os conceitos básicos de pipeline e cache podem ser ilustrados pelas applets do tutorial [Prabhu, 1999], que é um hipertexto (expositivo) com algumas Applets (demonstrativo). No nível assembler ferramentas como Dlx(Java) [Ibbett, 1998] podem ser usadas via browser ou instaladas localmente como o WinDLX [Patterson and Hennessy, 1999]. O DLXview [Adams, 1999], implementado em C++ e Tcl/Tk, pode ser usado para mostrar didaticamente alguns detalhes internos como os estágios da pipeline, a configuração das unidades funcionais e conceitos avançados das técnicas de escalonamento dinâmica como tomasulo e scoreboarding [Patterson and Hennessy, 1996].

A parte de desempenho pode ser avaliada através da execução de programas descritos em C (ou outra linguagem) que podem ser compilados para DLX e simulados no Simplecalar [Burger et al., 1996], que é um software de domínio público e código aberto, onde se pode configurar a estrutura da cache, da previsão de desvio, do escalonamento dinâmico entre outras opções. A interface é baseada em comandos do tipo Unix, como o SIS, descrito na seção 6. O sistema de janela produzido para o WebSis pode ser usado no Simplecalar para produzir uma interface gráfica interativa, ou scripts via browser como mostrado em [Figueiredo et al., 2001].

Outra contribuição deste trabalho, foi o uso do gerador de componentes feito para o Hades para a implementação do datapath básico e a unidade de controle do DLX. O datapath foi implementado usando memória ram e rom, registros e mux já disponíveis no Hades. Além disso, foi criada uma unidade de controle cujo o comportamento foi descrito em Java, usando o gerador de módulos, e um banco de registros que pode ter uma versão tanto estrutural, usando os componentes do Hades, ou comportamental descrita usando o gerador de módulos. A ferramenta Hades vem sendo utilizada para implementação de Datapath DLX no curso Organização de Computadores II [DPI/UFV, 2003b] da Universidade Federal de Viçosa.

## 6. Síntese Lógica

Como já foi mencionado, o processo de síntese é visto como uma caixa preta em muitos softwares comerciais. Porém existem ferramentas de domínio público, como o SIS [Sentovich et al., 1992], que possibilitam a interação com o processo, a execução de vários tipos de otimização, verificação e avaliação. O SIS é composto por um conjunto de comandos reunindo dezenas de trabalhos de pesquisa, sendo a maior referência na área de síntese. Alguns livros textos [Katz, 1994] utilizam o SIS para ensino. Entretanto, a interface e a simulação do SIS são pouco amigáveis.

A interface é composta por comandos de linha (estilo Unix) ou lançamento de scripts. Este trabalho propõe como solução usar uma plataforma cliente/servidor com uma interface implementada em Java, denominado Websis. O SIS é instalado em uma máquina Linux (existe também uma versão DOS), e acesso remoto é feito via TCP/IP. O cliente foi implementado em Java em código aberto. Os comandos da interface podem ser personalizados pelos usuários através de uma descrição textual (com uma gramática definida no padrão Lex/Yacc), seguindo a proposta [Ferreira and Trullemans, 1998], porém usando Java no lugar de C++ e LEDA.

A Figura 7 mostra a estrutura da interface. Uma árvore de diretório contento vários arquivos, cada um com a definição da janela de um comando, compõe o menu de opções. Para incluir/excluir um comando, basta adicionar/retirar o arquivo com a definição, não há neces-

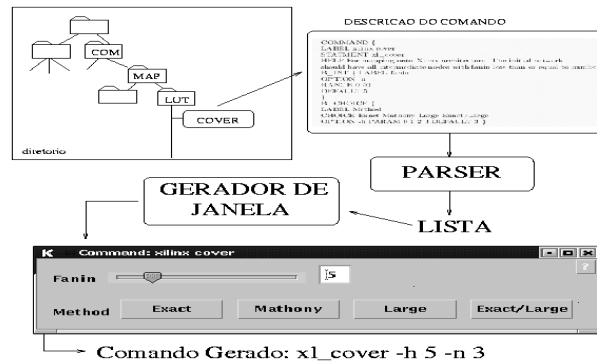


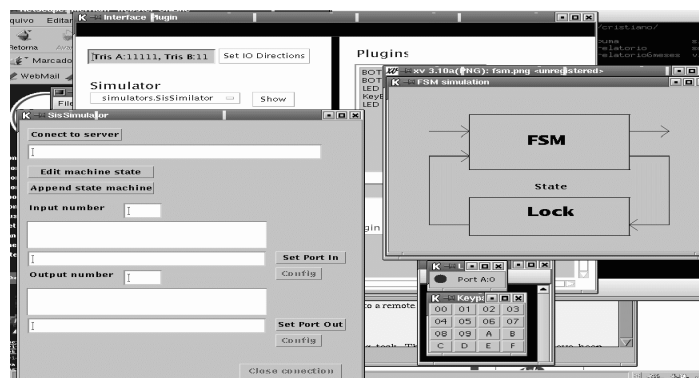
Figure 7: Estrutura do Gerenciador Dinâmico de Interface

sidade de compilar a ferramenta. A sintaxe de definição de comandos é simples, no formato texto, onde o comando pode ter botões do tipo Booleano, inteiro, string ou de opções mutuamente exclusivas. Um parser no padrão Lex/yacc processa a descrição e gera uma lista com os parâmetros selecionados para o comando. O gerador de janela processa a lista e gera a janela gráfica. Quanto o usuário interage com a janela, o comando texto é gerado e enviado para a ferramenta SIS. No exemplo da Figura 7, mostramos o comando **xl\_cover** para mapeamento em FPGA.

Diferentes versões da interface são possíveis sem necessidade de recompilação. Este recurso pode ser adequado para oferecer uma ferramenta para um estudante: iniciante, médio ou avançada. Ou oferecer apenas uma parte dos comandos do SIS para ensinar um tópico específico. A Figura 8 mostra a interface do WebSis. O acesso pode ser local ou remoto via internet. A parte que demanda processamento fica no servidor, sendo a máquina cliente responsável apenas pela gestão da interface.

O SIS aborda síntese sequencial (codificação, verificação e otimização), síntese de circuitos assíncronos, síntese lógica em 2 níveis (espresso), síntese multinível (algébrica, Booleana, Automatic Test Pattern Generation - ATPG), mapeamento em biblioteca de células ou FPGA (Multiplexador ou Lookup Table - LUT), análise (área, atraso e potência), entre outras funções. O código é aberto, sendo possível acrescentar novas operações, como vem sendo feito por diversos pesquisadores, como o Pose [Iman and Pedram, 1996] para estimação de potência e/ou Rasp [Cong et al., 1999] para mapeamento FPGA.

Dentro de um curso de sistemas digitais, o SIS pode ser usado para otimização em dois níveis, multinível, máquinas de estado e mapeamento. Otimização dois níveis é abordado, pela quase totalidade dos cursos, com o método de Karnaugh. Vários softwares podem auxiliar o ensino de Karnaugh [Dumas, 1998, Alexander, 2000], recentemente algumas applets podem ser usadas como auxílio [Hamburg, 1998, Rostock, 1998]. Métodos automatizados com Quine McCluskey [Roth, 1996, Costa, 1989, Vastianos, 1998] e espresso [Sentovich et al., 1992] também são muito utilizados. Apesar da interface texto, o espresso é bem simples e pode ser um primeiro contato com um software de síntese. O espresso é distribuído separadamente ou em conjunto com o SIS. O mesmo método usado no WebSis pode ser usado para criar um acesso remoto com interface de janelas para o espresso. Entretanto, a síntese dois níveis não é eficiente para muitos circuitos, e vem sendo substituída pela multinível desde do final da década de 80. O SIS é uma das poucas ferramentas que possibilita o ensino da síntese multinível, mostrando os processos de fatoração, decomposição, cálculo de don't care, etc. Estimativas de área, atraso e potência podem ser realizadas. Diversas técnicas de otimização e codificação de máquinas de



**Figure 8: WebSis: Ambiente gráfico para acesso remoto do SIS**

estados também podem ser interativamente avaliadas com o auxílio do SIS.

Outro ponto fraco da interface é a simulação textual, para cada ciclo de relógio, o comando *simulate* deve ser lançado especificando uma sequência binária com os valores de todas as entradas.

Uma contribuição deste trabalho foi conectar o SIS aos módulos de entrada e saída disponíveis para o Misim, no ambiente de síntese lógica WebSis, que não possuía uma interface interativa para simulação. Este recurso enriquece a simulação de máquinas de estados do SIS. Além disso, módulos de memória podem ser usados para o projeto de processadores. A primeira versão usava o padrão Misim comunicando com o WebSis. A nova versão conecta o SIS diretamente ao Hades, estendendo os recursos como: visualização de formas de ondas, estímulos programada em Java ou em scripts, toda a biblioteca do Hades, etc.

## 7. Resultados

Nesta seção, uma proposta de ensino auxiliado por software livre será apresentada. Esta proposta vem sendo testada nos dois últimos anos nos cursos de Hardware da Universidade Federal de Viçosa.

Inicialmente, o Diglog [Gillespie and Lazzaro, 1996] pode ser usado para ensino de portas lógicas, decodificadores e multiplexadores, circuitos aritméticos, flip-flops, registradores, memória, unidade lógica aritmética e bem como uma introdução a máquina de estados. Vários exemplos para ilustrar estes tópicos foram desenvolvidos. Em conjunto, a Applet de mapas de karnaugh [Hamburg, 1998] pode ser usada, por ser didática, ilustrar a PLA gerada e exportar o formato espresso.

A transição para Hades [Hendrich, 1998b] pode ser feita no nível de máquina de estados. Projetos como pequenos processadores, unidades de ponto flutuante, sistemas de controle, que ilustram a interação entre o controle e a parte de dados. Além de também permitir uma abstração na descrição da máquina de estados. Circuitos com a máquina de estado podem ser descritos parcialmente no simulador e o restante no protoboard, onde ambas as partes se comunicam graças ao componente porta paralela. Pequenos sistemas embutidos também podem ser desenvolvidos, mostrando a interação com os micro processadores comerciais disponíveis (ex: Pic16F84) ou desenvolvidos pelos próprios estudantes. No caso de sistemas embutidos, recomenda-se o uso do Misim isolado e em cooperação com o Hades, já que o PIC disponível no Hades é muito lento.

Dois pontos são importantes: a programação em HDL (Hardware Description Language) e a compreensão do processo de síntese. O primeiro ponto pode ser abordado com as ferramentas da Altera ou Xilinx, que possibilitam a descrição nos diversos níveis: esquemático, diagramas de estado e linguagens HDL. A desvantagem é o custo da ferramenta, porém as empresas mantêm programas de doação parcial/total com as universidades. Tutoriais ilustram vários exemplos de descrições VHDL bem como vários livros da área, muitos com o material disponível na Internet. No segundo ponto, a síntese, o uso do WebSis permite ao estudante interagir com o processo e acompanhar as transformações nas representações. Além disso, o WebSis pode ser usado nas fases anteriores, em conjunto com o Diglog e Hades, para ilustrar os métodos automáticos de síntese dois níveis, multinível, máquina de estados e mapeamento tecnológico.

Para os conceitos avançadas dos processadores RISC e superescalares, uma sugestão é o uso do processador DLX graças a vasta gama de ferramentas disponíveis, bem como tutoriais, exercícios e outros materiais didáticos disponíveis na Internet [Patterson and Hennessy, 1996].

## 8. Conclusão

O uso de software livre tem várias vantagens: a economia de custos, a flexibilidade e o maior domínio da tecnologia. Como foi mostrado, para a maioria dos problemas na área do ensino de sistemas digitais e arquitetura de computadores, existem vários softwares livres de código aberto com excelente qualidade. O código aberto dá oportunidade ao estudante de compreender como o software foi projetado e possibilita que ele mesmo contribua. Como exemplos, este trabalho apresenta a integração do software SIS com acesso remoto e interface em Java, sua integração à módulos externos do Misim e o seu acoplamento com o Hades. Além disso, vários recursos foram adicionados ao Hades: integração com o Misim, gerador de módulos, interface com o mundo externo (porta paralela) e exemplos didáticos para desenvolvimento de mais módulos. Muitas vezes o uso de uma ferramenta mais simples, como o Diglog para esquemáticos, é mais adequado para os conceitos básicos do que o uso de um software complexo com vários níveis de descrição e simulação, com um Altera e Xilinx. O domínio da tecnologia no uso e desenvolvimento das ferramentas gera maior independência e portabilidade, além da redução dos custos. Uma tendência são ambientes integrados [Donzellini and Ponta, 2001, Figueiredo et al., 2001] usando a interface de um browser, facilitando a instalação e manutenção, dando acesso remoto independente de plataforma e possibilitando a auto-avaliação com exercícios e exemplos interativos. O uso de software livre e implementações em Java facilita a integração com estes ambientes. Como trabalho futuro, um estudo aprofundado do estado da arte dos softwares livres (de código aberto ou não), para ensino de linguagens HDL e dos ambientes de projeto conjunto de software e hardware está em andamento. Outro ponto importante é o uso de tecnologia como Servlet e J2EE para ambientes de educação a distância.

## References

- Adams, G. (1999). Dlxview, an comprehensive pedagogical tool. Purdue University's School, Disponível em [http://www-ti.informatik.uni-tuebingen.de/~heim/lehre/ra\\_ss98/dlxview/help.html](http://www-ti.informatik.uni-tuebingen.de/~heim/lehre/ra_ss98/dlxview/help.html).
- Alexander, M. (2000). K-map windows program. <http://www.etel.dn.ua/~shurik/karnaugh/>.
- Altera (2003). Official site. <http://www.altera.com>.

- Bormida, G. D., Ponta, D., and Donzellini, G. (1997). Methodologies and tools for learning digital electronics. *IEEE Transaction in Education*, 40(4).
- Bräunl, T. (1998). Retro - register-transfer-object hardware simulator. <http://robotics.ee.uwa.edu.au/retro/>.
- Burger, D., Austin, T. M., and Bennett, S. (1996). Evaluating future microprocessors: The simplescalar tool set. Technical Report CS-TR-1996-1308, University of Wisconsin.
- Compton, K. and Hauck, S. (1999). Configurable computing: A survey of systems and software.
- Cong, J., Ding, E., Hwang, Y.-Y., Wu, J. P. C., and Xu, S. (1999). Rasp\_syn release b 1.1 lut-based fpga technology mapping package. University of California, <http://cadlab.cs.ucla.edu/~xfpga/software/raspsyn.htm>.
- Cong, J. and Xu, S. (2000). Synthesis challenges for nextgeneration high-performance and high-density plds. In *Asia and South Pacific Design Automation Conf.*, pages 157–162.
- Costa, A. (1989). Quine mccluskey and bdd educational tool. <http://www.dei.isep.ipp.pt/~acc/bfunc/>.
- Craig, D. (1996). Extensible hierarchical object-oriented logic simulation with an adaptable graphical interface. Master's thesis, Department of Computer Science, University of Newfoundland.
- Donzellini, G. and Ponta, D. (2001). Deeds - digital electronics and design suite- preliminary version. Technical report, University of Genoa.
- DPI/UFV (2003a). Disciplina: Organizacao de computadores i. [www.dpi.ufv.br](http://www.dpi.ufv.br).
- DPI/UFV (2003b). Disciplina: Organizacao de computadores ii. [www.dpi.ufv.br](http://www.dpi.ufv.br).
- Dumas, J. (1998). K-map windows program. University of Tennessee <http://www.utc.edu/~jdumas/cs250/>.
- Ferreira, R. S. and Trullemans, A.-M. (1998). Asterix: An interactive environment for logic synthesis and analysis. In *International Workshop PATMOS*.
- Figueiredo, R. J., Fortes, J. A. B., Eigenmann, R., Kapadia, N. H., Taylor, V., Choudhary, A., Vidal, L., and Chen, J.-J. (2001). On the use of simulation and parallelization tools in computer architecture and programming courses. *Computers in Education Journal*, 11(1).
- Gillespie and Lazzaro (1996). Diglog and analog. Technical report, Caltech VLSI CAD Tools, California Institute of technology.
- Greiner, A., Lucas, L., and Wajsbürt, F. (1994). Designing a high complexity microprocessor using the alliance cad system. In *Proceedings of the 7th Annual IEEE International ASIC Conference and Exhibit (ASIC'94)*, pages 223–226.
- Hamburg, U. (1998). Kmap applet. <http://tech-www.informatik.uni-hamburg.de/applets/kvd/>.
- Hasen, J. P. (2000). Tkgate - event driven digital circuit simulator. <http://www-2.cs.cmu.edu/~hansen/tkgate/>.
- Hayes, J. P. (1993). *Introduction to Digital Logic Design*. Addison-Wesley Longman, Incorporated.
- Hendrich, N. (1998a). Hades - the "hamburg design system". In *European Academic Software Award*.

- Hendrich, N. (1998b). Hades- a java-based visual simulation environment. Technical report, Fachbereich Informatik, Universität Hamburg.
- Ibbett, R. (1998). Dlx's applets. University of Edinburgh, <http://www.dcs.ed.ac.uk/home/hase/>.
- Iman, S. and Pedram, M. (1996). Pose: Power optimization and synthesis environment. In *DAC96*.
- Ito, S. e. a. (2000). System design based on single language and single-chip java asip micro-controller. In *ACM/IEEE Design Automation and Test Conference (DATE)*.
- Katz, R. (1994). *Contemporary Logic Design*. Benjamin/Cummings P. Company.
- Mano, M. M. (1991). *Digital Design*. Prentice Hall, 2nd edition.
- Michelli, G. D. (2000). Hardware synthesis from c/c++ models. In *ACM/IEEE Design Automation and Test Conference (DATE)*, pages 382–383.
- Patterson and Hennessy (1999). Additional resources for computer architecture: A quantitative approach, second edition. Morgan Kaufmann Site - [http://www.mkp.com/books\\_catalog/ca/hp2e\\_res.asp](http://www.mkp.com/books_catalog/ca/hp2e_res.asp).
- Patterson, D. and Hennessy, J. (1996). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 2nd edition.
- Prabhu, G. (1999). A computer architecture tutorial. Iowa State University, <http://www.cs.iastate.edu/~prabhu/Tutorial/>.
- Rienen, I. V. (1996). Digsim- a java applet simulator of simple digital circuits. <http://www.cs.umbc.edu/~chang/cs313.s02/digsim-info.shtml>.
- Rostock, U. (1998). Kmap applet. <http://www.informatik.uni-rostock.de/~masch/minim.html>.
- Roth, C. (1996). Sane - quine mccluskey applet, projekt schaltsysteme arbeitsblätter im netz. [http://www-ihs.theoinf.tu-ilmenau.de/~sane/projekte/qmc/embed\\_qmc.html](http://www-ihs.theoinf.tu-ilmenau.de/~sane/projekte/qmc/embed_qmc.html).
- Sentovich, E. M., Singh, K. J., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P. R., Brayton, R. K., and Sangiovanni-Vincentelli, A. (1992). SIS: A system for sequential circuit synthesis. Technical report, U.C. Berkeley.
- Technology, F. (2000). Misim - microchip simulator. <http://www.feertech.demon.co.uk/misim>.
- Vastianos, G. (1998). Quine mccluskey. <http://www.seattlerobotics.org/encoder/200106/qmccmin.htm>.
- Venkateswaran, R. and Mazumder, P. (1994). A survey of da techniques for pld and fpga based systems.
- Xilinx (2003). Official site. <http://www.xilinx.com>.