

Um Modelo de Ensino de Ciência da Computação Seguindo Normas Brasileiras

Raymond Westwater¹, Alexandre de Queiroz²

¹Future Ware, Inc.

475 Wall Street, Princeton, NJ USA 08540

² Departamento de Licenciatura em Computação – Faculdades de Dracena (UniFaDra)

Rua. Bahia, 332 – Metrópole, Dracena, SP CEP 17.900-000

rwestwater@hmlware.com.br, alexqueiroz@ig.com.br

Abstract. *In keeping with current trends in Brazilian education, the work described in this paper proposes to supplement the currently textbook-based mode of presentation with interactive tutorials. This paper will focus on the presentation of the Computer Architecture course in conjunction with FPGA hardware technology and the Hardware Modeling Language (HML) Web-based development tool.*

Resumo. *Estamos vendo, no Brasil, uma revolução silenciosa em pedagogia, a introdução de uma forma de ensino com foco na interação. A proposta deste trabalho é amplificar a maneira de ensinar o curso de Arquitetura de Computadores com exercícios baseados na tecnologia de hardware FPGA e a Linguagem de Modelagem Hardware (HML) disponível na Web.*

1. Educação em Brasil

“Todo mundo fala sobre o tempo, mas ninguém faz nada com respeito a ele” – *Mark Twain*. Às vezes, sente-se frustrado com as limitações e as contra-produções do sistema de educação no Brasil – parece que todo mundo fala sobre ele, mas sem êxito – nada de bom acontece [Economist 2009]. Enquanto o governo investe em educação de altos níveis sem precedente, os resultados parecem pior que nunca [Hebmüller 2007].

Se a necessidade é a mãe da invenção, o desespero deve ser o pai da ação. Em todas as partes do Brasil, exige-se uma mudança geral, até mesmo um modelo novo de ensino – ensino interativo. Fizemos um estudo informal sobre essa atividade, e compartilhamos os resultados aqui.

Visitamos várias escolas de níveis diversos. Reparamos que é bem comum ensinar de forma interativa os níveis elementares. Por exemplo, encontramos diversos trabalhos manuais, como: trabalhos de cortar com tesouras, desenhar com lápis de cera, e colar com cola nos primeiros graus, aceitamos isso como um fato normal. Vemos inovações em educação começando pelo uso de ferramentas, mais ou menos habilitando a aprendizagem manual. Por exemplo, a Escola Anglo e o SENAI/SESI de Curitiba compartilham uma seqüência de ensino, desta forma, se ensinam jovens no primeiro grau o jogo de programação “Lego Mindstorms”, e jovens na idade de formação com uma máquina de controle numérico computadorizado (CNC). Aqui, o conceito de

operação de máquinas por meio de programação foi introduzido bem antes que o aluno precisasse aprender o processo de programação da ferramenta de moagem, por exemplo.

Seguindo os escritos de Jean Piaget [Piaget 1961], o desenvolvimento da capacidade de pensamentos abstratos segue alguns ciclos de crescimento, cada ciclo compreendido dos passos:

- Experimento com ações sobre objetos, e a observação dos efeitos/resultados,
- Internalização de um modelo abstrato de propriedades dos objetos abaixo das ações,
- Elaboração e generalização do modelo mental resultando na construção de uma nova etapa de cognição.

A maior parte do tempo é gasta no processo de experimentação, desde que se chegue a uma gestalt de conhecimento representando um nível novo de abstração, habilitando um ciclo novo. Estas etapas, logicamente precisam ser seguidas em seqüência.

As teorias pedagógico-didáticas emergindo no Brasil, começando nos anos 1990 [Ghiraldelli 2006] mostram etapas de desenvolvimento bem-parecidos como o trabalho de Piaget:

- Apresentação dos problemas.
- Articulação entre os problemas apresentados.
- Associação e assimilação de conceitos por comparação
- Generalização
- Aplicação

Poder-se-ia perguntar por que o sistema de didática dos Estados Unidos serve tão bem no país, mas com resultados mais complicados aqui no Brasil. Os autores pensam que os Estados Unidos historicamente têm investido mais na infra-estrutura da educação, criando uma cultura de aprendizagem. Por exemplo, bibliotecas com acesso grátis têm sido providas pelos órgãos de governos a nível federal até municipal desde a época colonial. Também, os alunos nos Estados Unidos gastam quase duas vezes mais tempo na escola do que os do Brasil [Cardoso, 2006]. Por qualquer razão, acreditamos que há uma diferença fundamental entre o nível de preparação para estudar a nível universitário, e as metodologias Norte Americanas não servirão aos brasileiros.

1.1. Implicações para Ciência de Computação

A literatura, as estruturas das aulas, até mesmo as linguagens de programação, todas ferramentas de ensino no campo de computação vêm dos Estados Unidos. O aluno é esforçado a aprender dentro do sistema Americano, não importa seu nível de preparação. Pior ainda, os pensamentos aprendidos, por causa das barreiras da educação, da distância e da linguagem, ficarão atrás do trabalho dos cientistas e trabalhadores Americanos, permanecendo o aluno (e o trabalhador) brasileiro em desvantagem competitiva.

No momento de iniciar o estudo no campo da computação a nível universitário surge uma oportunidade quase única – a oportunidade de recomeçar do zero. O estudante de computação precisa aprender uma forma de analisar e trabalhar porque ele não tem preparação. Em teoria, há uma oportunidade em aplicar os princípios do ensino de Piaget à forma de didática, seguindo o desenvolvimento das normas espalhando pelo Brasil.

Como apresentar materiais nesta forma de didática? O problema não é fácil de solucionar – o curso de computação, no prazo de quatro curtos anos, pretende ensinar ao aluno do básico da programação até o estudo do campo de soluções, por exemplo, a teoria do desenho. A teoria do desenho segue as pesquisas no campo de estudos das propriedades de problemas gerais [Altshuller 1975], estrutura de programas [Yourdan 1979], e a Linguagem Unificada de Modelagem (UML) [Booch 2000]. Então, o aluno precisa se elevar do nível de racionamento ineficaz até o nível de solucionamento de problemas, e depois, até mesmo ao nível de racionamento sobre a natureza dos racionamentos – uma missão formidável.

O sistema Norte-americano tem tentado desenvolver uma forma de educação auxiliada por computador baseado nos princípios da inteligência artificial [Minsky 1977]. Na atualidade, o que têm sido desenvolvidos são primariamente aplicações de ensino à distância, e mesmo os cursos tipo ensino auxiliado por computador (CAI), simplesmente enrolando a mesma forma de didática como antes [Valente 1997]. Este fracasso ao desenvolver um sistema novo de ensino pode ter raízes bem básicas: o sistema norte-americano funciona muito bem para os norte-americanos.

É sobre o desafio de desenvolver um sistema de ensino baseado nos princípios de Piaget que este trabalho fala. Os passos que os autores estão seguindo são:

- Especificação de um curso de dois anos que vai ensinar todos os básicos da computação aos alunos, na forma brasileira (baseados em partes por Piaget). Ao fim deste prazo de dois anos, o aluno deve ter alcançado o nível de pensamento abstrato igual ao dos norte-americanos com dois anos de ensino de computação.
- Escrevendo os livros de ensino, divididos em temas convencionais, somando a didática nova.
- Desenvolvimento de materiais interativos embutindo o conteúdo dos cursos na forma desejada.
- Aplicando foco nas áreas de pesquisa novas, abrindo oportunidades únicas para os brasileiros.

Apresentado nos capítulos seguinte, encontra-se um resumo do curso de dois anos planejado, e detalhes das matérias no processo de desenvolvimento para o curso de arquitetura de computadores.

2. Forma da Didática

A didática proposta deveria satisfazer os constrangimentos seguintes:

- Organiza a pedagogia em ciclos de aprendizagem, introduzindo com complementação de cada ciclo uma nova gestalt para ser aprendido pelo aluno.

- Habilita os alunos para prosseguir a didática norte-americana.
- Compatível nos temas convencionais, capaz de uso em currículos preexistentes.
- Idealmente, abre novas oportunidades unicamente brasileiras.

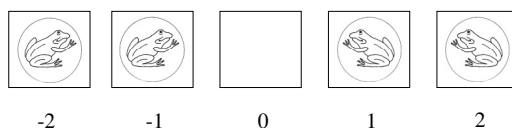
Então, a seqüência de cursos previstos segue a forma das aulas dadas atualmente. Está sendo desenvolvido a seqüência de quatro semestres (Figura 1).

<p>Semestre I: Introdução à Programação</p> <ul style="list-style-type: none"> • Solução de Problemas • Representação Abstrata das Soluções • Lógica Simbólica • Organograma • Java 	<p>Semestre II: Programação Avançada</p> <ul style="list-style-type: none"> • C++ • Estruturas de Dados • Solução de Problemas • Desenho Orientado a Objeto (OOD) • Análise Orientada a Objeto (OOA) • Direção de Projetos
<p>Semestre III: Ambiente de Operações</p> <ul style="list-style-type: none"> • Linguagem Assembler • Sistemas Operantes • Comunicações • Base de Dados • Interface de Usuário Gráfico (GUI) • Analisando Texto • Linguagem Aumentar Extensível (XML) 	<p>Semestre IV: Arquitetura de Computadores</p> <ul style="list-style-type: none"> • Introdução a Firmware • OOD para Firmware • Hardware para Computadores • Comunicações • Comunicações sem Fio

Figura 1. Seqüência de Ensino Proposta

O curso “Introdução à Programação” serve como exemplo da didática proposta. A primeira etapa, “Solução de Problemas”, pretende habilitar alunos na prática de solucionar problemas. Uma variedade de problemas está implementado em um site da Web, e os alunos jogam com os problemas interativos até se acomodarem ao método necessário.

Início:



Final:

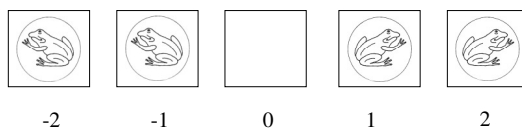


Figura 2. Problema Típico, “Sapos Saltando”

Como está ilustrado na Figura 2, um problema típico tem uma condição inicial, um estado de terminação desejada, e regras de mudar de um estado até o próximo. Esta forma de jogar é bem conhecida por jovens nesta idade em videogames, e os alunos deveriam ser capazes de entender e solucionar problemas deste tipo. Neste caso, os movimentos de um sapo do jogo são andar em frente um passo, quando tocado por um clique do mouse (Figura 3).

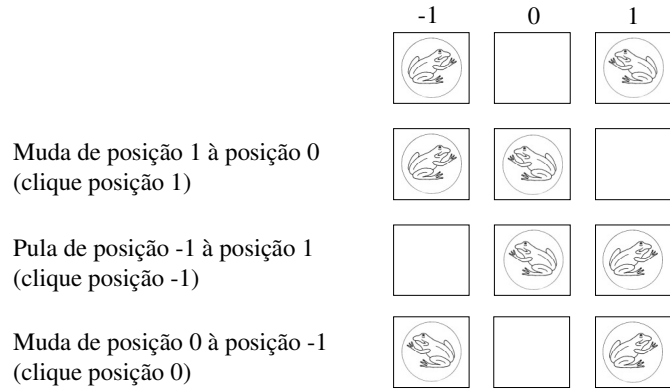


Figura 3. Animação dos Sapos

Desde aprender a estrutura da solução, o aluno experimenta com o mesmo problema uma representação mais abstrata (Figura 4). O aluno pode encontrar as soluções em uma linguagem bem simples, e testa ao clicar o botão. Os sapos animam-se, e se for encontrado corretamente os dados, irá chegar ao estado final.

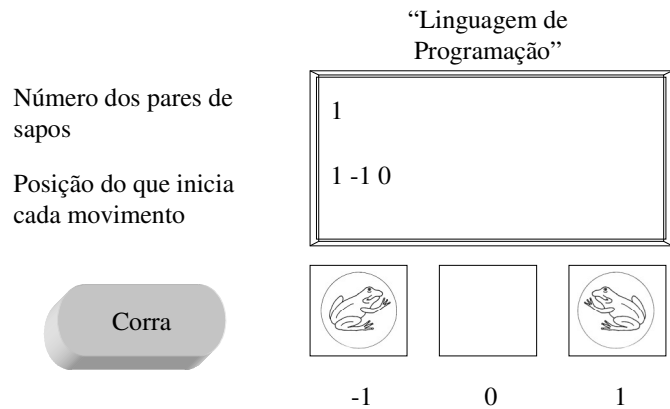


Figura 4. Representação Abstrata do Problema

Este problema segue a forma proposta da didática. Uma coleção de problemas é apresentada, cada um precisa de uma formulação de seqüência de etapas para solucionar. As habilitações necessárias são as típicas dos jovens de hoje: conceito de jogo de computador e facilidade com o mouse. Após solucionar uma coleção de problemas deste tipo, o estudante conseguirá um modelo mental solucionando

O aluno escreve seu sistema operante em Java, como um objeto implementando as interfaces dos componentes de simulador: geração de tarefas, simulação de memória, simulação de disco duro e simulação de processador. Cada componente provê as capacidades que modela as capacidades do hardware, importante para desenvolver uma abstração de um sistema de operação:

- Geração de Tarefas. Uma tarefa tem requisito de memória exposto ao sistema operante, um horário do uso de processador e disco duro.
- Memória. Disponibilidade de memória esta limitada e o sistema de operação precisa compartilhar a memória entre tarefas, mandando guardar o contexto de uma tarefa ao disco duro quando é necessário.
- Disco duro. O disco duro é usado por cada tarefa em sua operação e também pelo sistema operante. O sistema operante implementa uma estratégia: dirigir transferências ao disco.
- Processador. O contexto de uma tarefa está impresso no processador, portanto a tarefa começa a ser executada. A tarefa pode interromper-se com um pedido de acesso ao disco duro ou terminação, ou por ações dos outros componentes: uma tarefa nova pode surgir, um acesso do disco pode terminar, ou um cronômetro pode escoar.

Assim o aluno pode ganhar conhecimento de muitos problemas encontrados no desenho de um sistema operante, gera soluções e verifica os resultados. O tipo de metodologia o aluno pode desenvolver inclui a priorização das tarefas e acesso ao disco duro (sistema de tempo real), compartilhar o computador entre tarefas (sistema de tempo compartilhado), ou outros algoritmos.

Terminando este ciclo de aprendizagem, o aluno está preparado a encontrar desafios na área lidando com uma memória em hardware. Esta etapa é feita com um sistema operante de realidade – Linux. O aluno escreve um “device driver” usando interfaces simplificadas. Assim o aluno terá experiência concreta dentro de um sistema operante atual.

3. A Didática Abre Novos Campos de Pesquisa

Propõe-se uma didática para adaptar uma metodologia de ensino na tecnologia de computação para melhorar resultados obtidos por alunos brasileiros. Quando se analisa as implicações da didática, resultados inesperados e bons surgem. Estes resultados chegam do campo de desenvolvimento das ferramentas de ensino para a arquitetura de computadores.

A seqüência de ensino para arquitetura de computadores é proposta:

- Linguagem assembler. Uma linguagem bem simples é especificada para ensinar os princípios de assembler. Esta linguagem e um simulador são providos para executar programas escritos em assembler. O aluno aprende como fazer, simpaticamente à forma das didáticas proposta.
- A arquitetura dentro do computador. As ferramentas de simulação são providas para desenhar e desenvolver as unidades de construção de um computador

(unidades de lógica, unidades de interpretação de ordens escritas em linguagem assembler, unidades de interface de memória, unidades de seqüência do contador de programa). O computador vai executar as instruções da linguagem de assembler já ensinadas.

- Interface de hardware externo ao computador. As ferramentas de construção do hardware próprio são providas; o aluno vai construir a placa de hardware interface pelo sistema Linux, que já possui experiência no uso.

Propõe-se que a plataforma de implementação de hardware será o FPGA (array de portões lógicos programáveis no campo). Esta tecnologia é muito poderosa e flexível para implementar soluções desta classe, no entanto as linguagens de implementação são bem complicadas, obscuras e inacessíveis aos programadores. Mesmo para os engenheiros, as linguagens são complicadas porque pretendem guardar toda funcionalidade disponível teoricamente em hardware (pense em assembler contra Java ou C++).

A solução que proverá capacidade de programação dos FPGAs para programadores é vista nos padrões de UML (linguagem de modelagem unificada). A linguagem visual da UML parece um bom fundamento para o desenvolvimento de ferramentas de ensino de uso de hardware.

3.1 Uso dos FPGAs no Campo de Ensino

Enquanto o poder de se montar em um chip integrado cresce, a forma de utilizar este poder também está aumentando. Está acontecendo uma evolução não só em poder disponibilizado de utilização, mas em flexibilidade na forma de usar.

Chips de função fixada, chamados circuitos integrados por aplicações específicos (ASIC), têm capacidade de implementar processadores, interfaces, memórias, e outras funções importantes no campo de computação. As conexões são impressas sobre o substrato do chip com “fios” ou modelos de metal. Assim, mesmo como o chip tem capacidade de programação, a funcionalidade dele não muda (Figura 7).

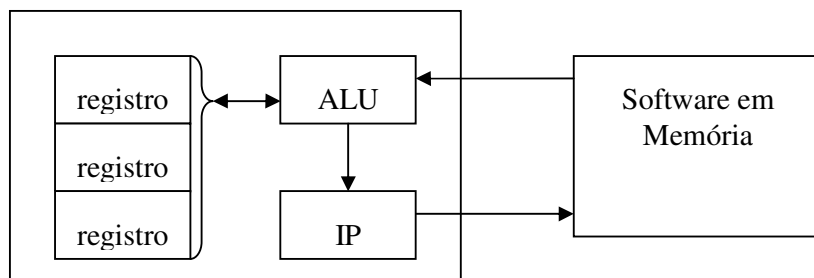


Figura 7. Modelo de programação dos ASICs.

Quase em paralelo da história de chips de funções fixadas, iniciava a história de chips com capacidade de programação por usuário (no sentido de trocar conexões de hardware). Até mesmo nos anos 70 existiram memórias semi-permanentes onde os

valores do conteúdo de memória podem ser escritos uma vez, mas persistem por anos. A tecnologia de escrever e apagar este tipo de memória avançou por anos até a forma moderna onde bilhões de bytes podem ser gravados sobre um chip do tipo flash (USB pen-drive, iPod, etc.).

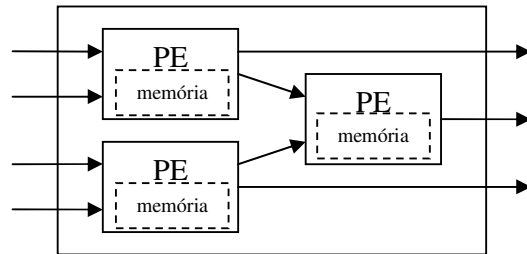


Figura 8. As conexões são fixadas dentro de PLDs

Os dispositivos de lógica programável (PLD) originaram na mesma época. Esta tecnologia provê os elementos programáveis (PEs) conectados por fios fixos, permitindo trocar equações de lógica dentro do chip (Figura 8). No ano 1985, a empresa Xilinx inventou a array dos portões lógicos tipo programável em campo (FPGA), tipo de chip onde não são programáveis só PEs, mas também as conexões entre os PEs. O FPGA pode ser usado para fabricar aparelhos de hardware, ou desenvolver desenhos para ASICs [Westwater 2009].

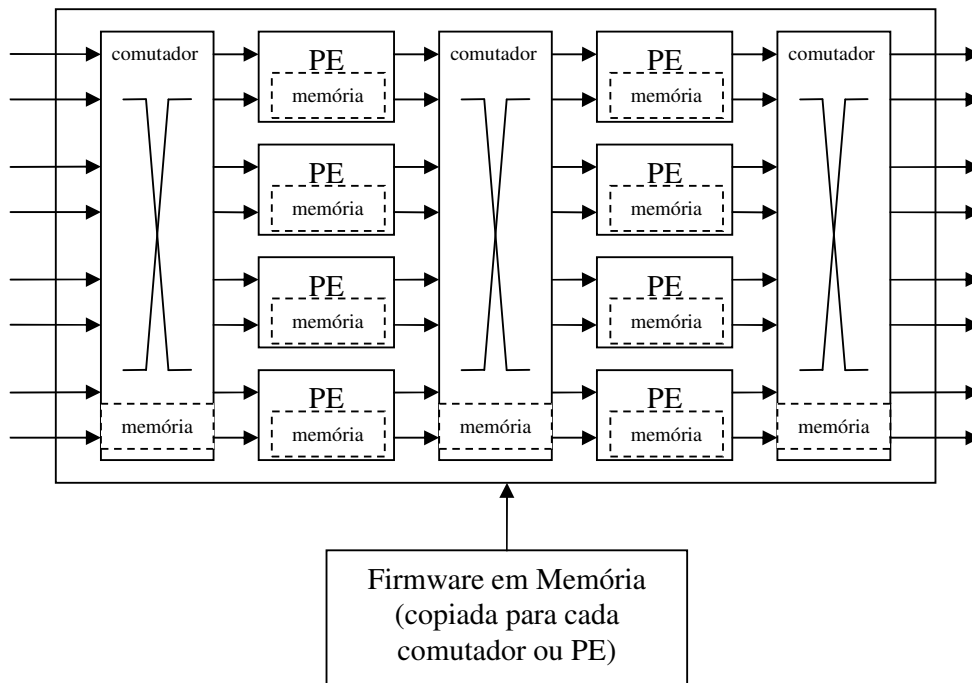


Figura 9. Modelo de programação dos FPGAs.

Programas escritos para esta classe de chip (Figura 9) são tão poderosos que merecem ser reconhecidos como uma classe nova de produção. Este tipo de programação está sendo chamada firmware. Na forma proposta de didática, os conceitos de arquitetura vão ser apresentados em conjunção com realizações em FPGA.

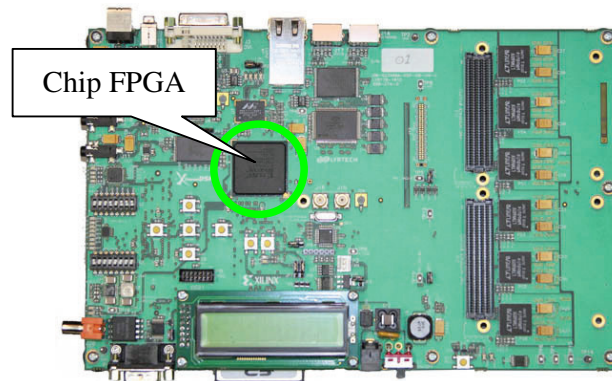


Figura 10. Placa de FPGA.

Placas de FPGA estão disponíveis e são baratas (Figura 10), ainda mais baratas com descontos especialmente para as universidades. Também, há ferramentas de graça para comunicar com a placa de FPGA e para desenvolver projetos. No entanto, a complexidade das ferramentas proíbe que a maioria dos programadores possa usá-las efetivamente.

3.2 Linguagem de Modelagem Hardware (HML)

A linguagem visual “Linguagem de Modelagem Universal” (UML) foi selecionada como a linguagem preferida para o ensino de desenvolvimento das soluções em hardware. A UML foi selecionada por essas qualidades:

- A UML é uma ferramenta padrão para programadores e conhecido pelos alunos seguindo o modelo pedagógico proposto.
- A UML tem natureza visual simples para comunicar as comunicações (ligações) entre instâncias de componentes do desenho de hardware e níveis de abstração.
- A UML tem uma modelagem do tipo máquina de estado de ser que pode ser utilizada nas soluções e construções em hardware [LeBeux 2007], e um padrão de representação que pode ser usado para gerar soluções por processo de compilação [Kovse 2002].
- O comitê da UML pretende desenvolver um padrão utilizável nas implementações completas.

Infelizmente, seguindo avaliação de algumas implementações e pesquisando o trabalho de outros, parece que a UML não pode servir por geração nem representação de firmware. Em resumo, os resultados encontrados incluem:

- As combinações de diagramas da UML estão mal-definidas e não servem para geração de soluções em hardware [Björklund 2002]

- A linguagem de compilar selecionado por UML tem problemas na precisão da especificação [Richters 1999], na complexidade de uso, e na especificação de semânticas por hardware.
- O padrão de configuração de UML, como é desenhado por união das especificações das linguagens, não serve para aplicação aos desenhos de hardware, que usam um vocabulário de configuração bem mais rico.
- O estado de implementações da UML não é maduro em muitos aspectos, especialmente na implementação da especificação do Intercâmbio de Meta-Modelos em XML (XMI).

Levando em consideração, a decisão a ser realizada é fazer uma linguagem de modelagem nova, a Linguagem de Modelagem Hardware (HML), baseada em conceitos de Facilidade de Meta-Objeto (MOF) em que a UML está escrita. Então, embora a HML pode ser descrita na UML, deve-se ser novamente reescrita.

A HML está baseada em princípios [Futureware 2009]:

- A HML personifica um modelo novo de desenvolvimento de hardware - não é nada semelhante a uma unificação das linguagens correntes.
- A HML propicia um modelo simplificado, porém flexível e poderoso, para desenvolvimento de hardware.
- A HML coopera com linguagens modernas para fazer uma solução completa.
- A HML provê uma representação gráfica dos conceitos abstratos de firmware.
- A HML provê uma representação escrita em linguagem semelhante a Java, para dar mais poder ao programador.
- A HML gera automaticamente a solução em firmware.
- A HML gera automaticamente o provador de verificação.

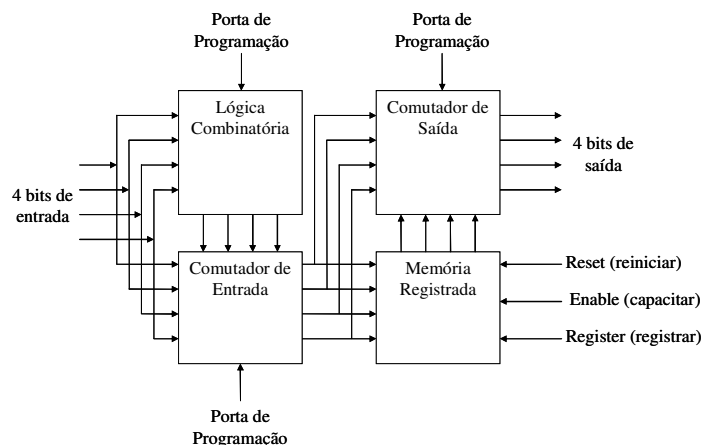


Figura 11. Célula de Lógica (FPGA).

A HML tem capacidades de representar graficamente, verificar e gerar soluções usando tecnologia FPGA. A tecnologia FPGA provê células de lógica (Figura 11) que podem ser programadas com um formulário lógico. O processo de programação está abstrata em HML com tabelas de entrada ou outros tipos de interação (Figura 12).

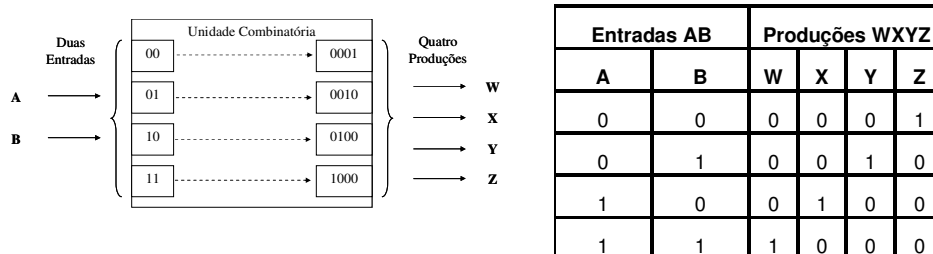


Figura 12. Representação de uma Solução Combinatória.

3.3 Benefícios Inesperados

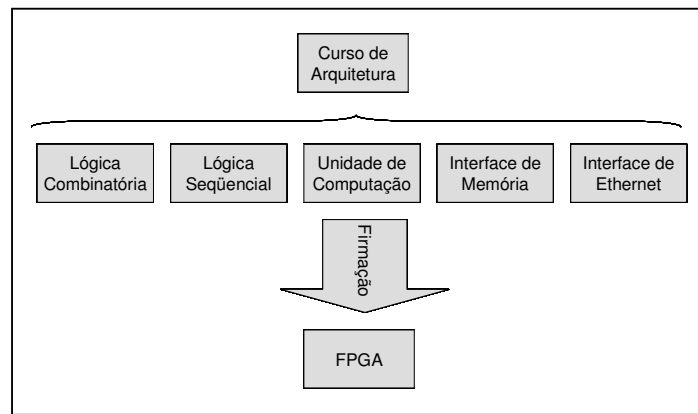


Figura 13. Uso de HML em Ensino de Arquitetura.

Além de uso como ferramenta pedagógica (Figura 13), HML tem aplicação no campo de desenvolvimento de novos aparelhos. Como o Brasil vem seguindo os Estados Unidos no campo de desenvolvimento de software, aberto ao campo novo de desenho de hardware por programadores, uma oportunidade existe de saltar adiante. Com a habilidade de inovar aparelhos rapidamente sem dependência dos especialistas de campos múltiplos, novas idéias podem ser incorporadas mais rapidamente e de forma mais barata com esta tecnologia, do que pode ser feito com tecnologias convencionais.

Se for adotado como parte da didática de ensino, estão previstos bens de consumo seguindo o uso desta tecnologia, incluindo:

- Desenvolvimento e lançamento dos produtos novos brasileiros. Exemplos incluem aparelhos de vídeo (Figura 14), laptops de segurança nível militar, e idéias novas ainda não previstas,
- Venda dos aparelhos aos outros países, gerando renda e criando mercados e empresas,

- Crescimento e valorização do trabalho dos programadores que facilitam este mercado, melhor distribuição de renda aos trabalhadores.

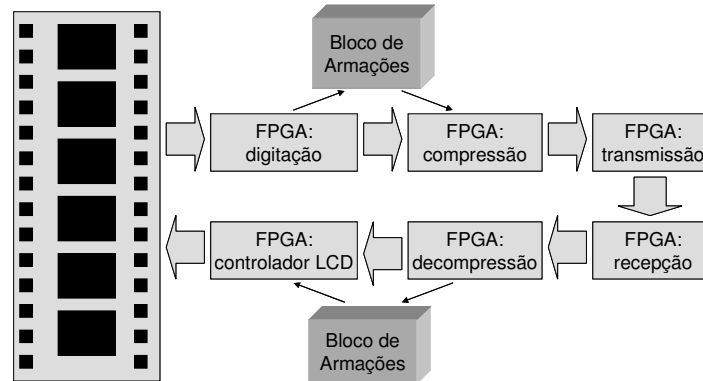


Figura 14. Exemplo de Aparelho baseado em HML: Compressão de Vídeo.

4. Conclusões, Trabalho em Frente:

O processo pedagógico proposto segue os padrões de educação emergentes do Brasil. A evolução na forma de ensino até a que funcione melhor dentro da cultura brasileira possui o objetivo de melhorar a formação dos estudantes. No campo de aprendizagem e ensino de tecnologia, há oportunidade de implementação desta forma de ensino em coordenação com o currículo existente. Implementação destes padrões de educação terá conseqüências bem importantes por todo o país, melhor educando os alunos, aumento e crescimento de TI e do PIB, e melhor distribuição de renda.

É necessário progresso no desenvolvimento do curso de Arquitetura de Computadores. A linguagem de HML está disponível para avaliação no site www.hmlware.com.br, a versão do livro preliminar deste curso será lançada em tempo para o uso efetivo nos anos 2010. Cooperação entre universidades interessadas na realização e implementação desta forma pedagógica de ensino é solicitada.

Outros lançamentos mantendo o atual processo pedagógico serão contínuos e seqüenciais, durante os anos 2010 e 2011, mantendo a força e o objetivo final. Assim sendo, a linguagem HML estará madura, viva e dinâmica para alcançar toda uma geração de cientistas e profissionais na área da tecnologia de computação.

References

- [Altshuller 1973] Altshuller, Genrich (1973). "Innovation Algorithm." Worcester, MA: Technical Innovation Center
- [Björklund 2002] D. Björklund and J. Lilius, "From UML behavioral descriptions to efficient synthesizable VHDL", Proc. of 20th IEEE NORCHIP Conference, 11-12 November 2002
- [Booch 2000] Grady Booch, Ivar Jacobson & Jim Rumbaugh (March 2000) OMG "Unified Modeling Language Specification, Version 1.3 First Edition". Object Modeling Group.

- [Cardoso 2006] Cardoso, Ana Rute e Verner, Dorte (December 2006). "School Drop-Out and Push-Out Factors in Brazil: The Role of Early Parenthood, Child Labor, and Poverty" Institute for the Study of Labor (IZA)
- [Economist 2009] Economist, staff (June 4, 2009). "Brazil's poor schools: Still a lot to learn" The Economist.
- [Futureware 2009] Future Ware, Inc. (2009). "Liguagem de Modelagem Hardware". www.hmlware.com.br
- [Ghiraldelli 2006] Ghiraldelli, Paulo (2006) "História da Educação Brasileira"
- [Hebmüller 2007] Hebmüller, Paulo. (26 mar./1 abr. 2007) "As novas metas do ensino brasileiro." Jornal da USP n.795. Universidade de São Paulo, São Paulo
- [Kovse 2002] Jernej Kovse, Theo Härder (2002) "Generic XMI-Based UML Model Transformations". Proceedings of the 8th International Conference on Object-Oriented. Information Systems Pages: 192 - 198
- [LeBeux 2007] Sébastien Le Beux, Philippe Marquet, Antoine Honoré, Jean-Luc Dekeyser "A Model Driven Engineering Design Flow to Generate VHDL" in: International ModEasy'07 Workshop, Barcelona, Spain, September 2007
- [Minsky 1977] Minsky, M. (1977). "Applying Artificial Intelligence to Education". Em R.J. Seidel & M.L. Rubin (ed.) "Computers and Communications: implications for education". New York: Academic Pres
- [Piaget 1961] Piaget, J. (1961). "La psychologie de l'intelligence." Paris: Armand Colin
- [Richters 1999] Mark Richters and Martin Gogolla (1999) "On the Need for a Precise OCL Semantics". Colorado State University, Fort Collins
- [Valente 1997] Valente, José Armando e de Almeida, Fernando José (1997). "Visão Analítica da Informatica na Educação na Brasil: a questão da formação do professor" Informática Jurídica
- [Westwater 2009] Westwater, R. (2009). "Arquitetura de Computadores". Ainda não publicado.
- [Yourdan 1979] Yourdan, Edward and Constantine, Larry L (1979). "Structured Design: Fundamentals of a Discipline of Computer Program and System Design" Prentice-Hall.