DESENVOLVIMENTO DE FERRAMENTA DE AUXÍLIO DIDÁTICO: IMPLEMENTAÇÃO DE ALGORITMO PARA MINIMIZAÇÃO DE MÁQUINAS DE ESTADOS EM LUA

Bruno Martins Costa – brunomartins@telecom.uff.br
Alexandre S. de la Vega – delavega@telecom.uff.br
Grupo PET-Tele – http://www.telecom.uff.br/pet
Universidade Federal Fluminense – UFF
Escola de Engenharia – TCE
Departamento de Engenharia de Telecomunicações – TET
Rua Passo da Pátria, 156 / Bloco D / Sala 504
24.210-240 – Niterói – RJ

***Resumo**: O objetivo principal do presente trabalho foi implementar o algoritmo de minimização de máquinas de estados conhecido como Algoritmo de Paul-Unger, usando a linguagem de programação Lua. A motivação inicial para o desenvolvimento desse aplicativo foi a sua integração em um conjunto de ferramentas de auxílio didático que o grupo PET-Tele está desenvolvendo para o ensino de Circuitos Digitais. Além dos aplicativos, o grupo tem desenvolvido também diferentes tipos de interfaces com usuário, a fim de distribuí-los de diferentes formas.*

***Palavras-chave:** Ferramenta didática, Ensino à distância, Circuitos Digitais, Minimização de Máquinas de Estados, Linguagem de Programação Lua.*

1. INTRODUÇÃO

O Programa de Educação Tutorial (PET) exige que os bolsistas dos grupos PET, ao serem submetidos a uma formação complementar, desenvolvam atividades que possuam, cada uma delas, itens relativos às áreas de Pesquisa, Ensino e Extensão, bem como consigam algum tipo de penetração no curso ao qual o seu grupo pertence.

Trabalhando na linha de ferramentas de auxílio didático, o grupo PET do Curso de Engenharia de Telecomunicações da Universidade Federal Fluminense (PET-Tele) aproveitou o surgimento da tecnologia de Televisão Digital (TV Digital) e desenvolveu um aplicativo interativo multimídia (MATTOS et al., 2009), codificado na linguagem NCL (SOARES & BARBOSA, 2009) (COMUNIDADE GINGA, 2012).

O aplicativo é um manual que, utilizando recursos de áudio, texto e imagem, descreve o algoritmo de minimização de funções booleanas conhecido por Algoritmo de Quine-McCluskey (HILL & PETERSON, 1981) (RHYNE, 1973) (McCLUSKEY, 1956).

A partir do manual desenvolvido, o grupo visualizou a possibilidade de implementar um pacote de aplicativos didáticos para o ensino de Circuitos Digitais, com as opções de

interação com o usuário nas formas local, via TV Digital e via rede de computadores.

O aplicativo aqui apresentado é parte integrante do pacote de ferramentas didáticas e realiza a minimização de máquinas de estados.

Objetivando-se uma ampla e gratuita divulgação, todo o material produzido pelo grupo é colocado à disposição para *download* no *website* do grupo, após a devida publicação.

A ferramenta desenvolvida é apresentada a seguir. A Seção 2 comenta o pacote de aplicativos didáticos que o grupo está desenvolvendo, do qual a ferramenta faz parte. As máquinas de estados, que representam o contexto no qual a ferramenta teve sua origem, são discutidas na Seção 3. A Seção 4 justifica a escolha da linguagem utilizada. O aplicativo desenvolvido é apresentado na Seção 5. Na Seção 6 encontra-se o resumo de exemplos de execução. Finalmente, a Seção 7 apresenta as conclusões e os trabalhos futuros.

2. PACOTE DE APLICATIVOS DIDÁTICOS

Um pacote de aplicativos didáticos dedicado ao ensino de Circuitos Digitais, incluindo as interfaces com o usuário para tais aplicativos, encontra-se em desenvolvimento pelo grupo PET-Tele. Em relação aos integrantes do grupo, o objetivo é fazer com que eles colaborem com a melhoria das práticas de ensino. Para os alunos do curso, o objetivo é fornecer a eles ferramentas auxiliares para estudo e para a verificação de resultados.

O passo inicial (MATTOS et al., 2009) foi um manual interativo multimídia que, utilizando recursos de áudio, texto e imagem, descreve o algoritmo de minimização de funções booleanas conhecido por Algoritmo de Quine-McCluskey (HILL & PETERSON, 1981) (RHYNE, 1973) (McCLUSKEY, 1956). O manual foi implementado para TV Digital, usando a linguagem NCL (SOARES & BARBOSA, 2009) (COMUNIDADE GINGA, 2012).

Em seguida, a fim de possibilitar uma maior interação com o usuário, o algoritmo de minimização foi implementado (BEPPU et al., 2010). Dado que a linguagem NCL não possui mecanismos procedurais de processamento, foi utilizada a linguagem de programação Lua (IERUSALIMSCHY, 2006) (IERUSALIMSCHY, 2012).

A linguagem Lua foi escolhida devido a sua característica de fácil interação com a linguagem NCL e pela decisão, tomada na época, em adotá-la para os dispositivos móveis no Sistema Brasileiro de Televisão Digital (SBTVD) (COMUNIDADE GINGA, 2012).

Uma vez que a linguagem Lua também possibilita uma fácil integração com outras formas de interface com o usuário (local ou via rede de computadores) além de NCL para TV Digital, o grupo decidiu desenvolver outros aplicativos codificados em Lua, cada um com três possibilidades de distribuição e interfaces com o usuário: local, rede via TV Digital e rede via computadores (SANTOS & DE LA VEGA, 2012).

Nessa linha, foi desenvolvido um algoritmo que realiza conversões entre os formatos mais comumente utilizados para expressar funções lógicas, seguido da sua codificação em Lua (SANTOS & DE LA VEGA, 2011).

Dando continuidade à construção do pacote de ferramentas didáticas, o trabalho atual é um aplicativo que implementa o Algoritmo de Paul-Unger (HILL & PETERSON, 1981) (Paul & Unger, 1959), utilizado para a minimização do número de estados de um circuito digital sequencial, o qual pode ser modelado por uma máquina de estados finitos.

3. MÁQUINAS DE ESTADOS

No ramo de Telecomunicações, circuitos digitais são largamente utilizados, nas mais diversas aplicações. Os circuitos digitais podem ser divididos em dois grandes grupos: circuitos combinacionais e circuitos sequenciais.

Circuitos digitais combinacionais podem ser interpretados como implementações de funções lógicas booleanas, sendo classificados como sistemas instantâneos ou sistemas sem memória, uma vez que existe uma relação única entre um determinado conjunto de valores forçados na entrada e os valores calculados como saída, em qualquer momento.

Por sua vez, circuitos digitais sequenciais podem gerar valores de saída diferentes para um mesmo conjunto de valores de entrada, em momentos diferentes. Por isso, são classificados como sistemas dinâmicos ou sistemas com memória.

Uma vez que a saída de um circuito sequencial, em um dado momento, depende da entrada aplicada e do seu estado (conjunto de valores das variáveis internas), e que o número de estados diferentes que um determinado circuito pode assumir é finito, é comum associar-se o circuito sequencial a uma máquina de estados finitos.

Circuitos sequenciais cuja saída atual depende tanto da entrada atual quanto do estado atual são denominados de Máquinas de Mealy. Quando a saída atual depende apenas do estado atual, o circuito sequencial é classificado como uma Máquina de Moore (HILL & PETERSON, 1981).

Uma forma habitual de se descrever pictoricamente uma máquina de estados finitos é através de um diagrama de estados finitos. Para facilitar o trabalho computacional, os diagramas de estados finitos são traduzidos em tabelas de estados, onde são apresentados os valores possíveis para o estado atual, a entrada atual, a saída atual e o próximo estado. Dada uma entrada atual ***x***, as Tabelas 1 e 2 apresentam exemplos de tabelas de estados para Máquinas de Mealy e de Moore, respectivamente.

Tabela 1 – Exemplo de tabela de estados para Máquina de Mealy.

Estado atual	Próximo estado		Saída atual	
	x = 0	x = 1	x = 0	x = 1
a	b	d	2	5
b	d	c	1	4
c	a	b	7	0
d	c	a	3	6

Tabela 2 – Exemplo de tabela de estados para Máquina de Moore.

Estado atual	Próximo estado		Saída atual
	x = 0	x = 1	
A	b	d	4
B	d	c	3
C	a	b	1
D	c	a	7

As Tabelas 1 e 2 apresentam exemplos de máquinas classificadas como completamente especificadas. Nas máquinas ditas não completamente especificadas, alguns valores de saída atual e de próximo estado podem não ser determinados pelo comportamento da máquina.

A quantidade de estados de uma determinada descrição é um ponto importante a ser tratado no ensino de circuitos sequenciais digitais. Ao se traduzir o comportamento de uma determinada máquina de estados finitos por seu diagrama ou por sua tabela de estados, é normal que ocorra a inclusão de estados redundantes. Uma vez que a complexidade do circuito sequencial que implementa a máquina de estados, bem como o seu custo, tem relação direta com a quantidade de estados existentes, deve-se procurar descrever a máquina com a quantidade mínima de estados que atenda ao comportamento desejado. Portanto, dada uma determinada tabela de estados, deve-se procurar uma tabela equivalente, contendo o número mínimo de estados necessários. Um algoritmo que pode ser utilizado para esse fim é o denominado Algoritmo de Paul-Unger (HILL & PETERSON, 1981) (Paul & Unger, 1959), que pode ser aplicado para tabelas completamente especificadas ou não.

O aplicativo desenvolvido implementa o Algoritmo de Paul-Unger, usando a linguagem de programação Lua, visando um emprego didático para o mesmo.

4. ESCOLHA DA LINGUAGEM

A linguagem de programação Lua (IERUSALIMSCHY, 2006) (IERUSALIMSCHY, 2012) foi naturalmente escolhida por uma série de motivos.

Inicialmente, ela já havia sido utilizada na implementação do algoritmo de minimização realizada pelo grupo (BEPPU et al., 2010), devido à sua compatibilidade com a linguagem NCL e à sua adoção no SBTVD (Ginga - NCL). O objetivo foi incorporar tal implementação no manual sobre o algoritmo, também implementado pelo grupo (MATTOS et al., 2009), a fim de aumentar o grau de interatividade do mesmo.

Em seguida, foi percebido que podem ser implementados diferentes tipos de interfaces com o usuário, os quais são facilmente compatíveis com Lua. O primeiro deles é a própria linguagem NCL (SOARES & BARBOSA, 2009) (COMUNIDADE GINGA, 2012), para aplicações em TV Digital. Para uso local (*stand alone*), pode-se associar o código Lua a uma interface gráfica local (PUC-RIO, 2012). O terceiro tipo, para uso via rede de computadores, pode ser obtido ao se utilizar a linguagem de ligação CGILua (KEPLER PROJECT, 2012), que permite associar código Lua a formulários e a páginas de Internet (*webpages*).

Assim sendo, a linguagem de programação Lua foi escolhida para implementar a parte procedural dos aplicativos desenvolvidos, que compõem o pacote de ferramentas didáticas.

5. APLICATIVO DESENVOLVIDO

Fazendo parte de um pacote de ferramentas didáticas, o aplicativo aqui apresentado implementa o Algoritmo de Paul-Unger (HILL & PETERSON, 1981) (Paul & Unger, 1959), aqui utilizado para a minimização do número de estados de uma máquina de estados finitos, a qual modela um circuito digital sequencial. O algoritmo trabalha com tabelas de estados completamente especificadas, similares às Tabelas 1 e 2, ou não.

O código está separado em arquivos que contêm funções específicas: funções básicas, funções para manipulação de listas, funções do algoritmo, código principal, funções para exibição de dados e arquivo com dados de entrada. A união de todo o código é realizada sem qualquer esforço, através da facilidade de inclusão de código em Lua (função *dofile*).

A estrutura de dados de entrada para o aplicativo, que representa uma tabela de estados organizada de forma similar às Tabelas 1 e 2, é composta por três listas (tabelas), contendo valores do tipo *string* (cadeia de caracteres). A primeira tabela contém o conjunto de estados atuais. A segunda tabela contém um conjunto de tabelas de próximos estados, sendo cada conjunto associado a um estado atual e cada próximo estado associado a um valor de entrada do circuito. A terceira tabela contém um conjunto de tabelas de saídas atuais, sendo cada conjunto associado a um estado atual e cada valor de saída associado a um valor de entrada do circuito.

No terminal de execução do aplicativo, é realizada uma saída de dados durante toda a sua execução. A cada etapa do algoritmo, as informações geradas são apresentadas, de forma que o usuário possa acompanhar a evolução do processo de minimização.

A estrutura de dados de saída final depende do tipo da tabela de estados de entrada. No caso de uma tabela de estados completamente especificada, trabalha-se com o conceito de equivalência de estados e o resultado final identifica diretamente a lista de estados mínima. Nesse caso, o aplicativo fornece uma versão minimizada da tabela de estados de entrada. Porém, se a máquina for descrita por uma tabela de estados não completamente especificada, é empregado o conceito de compatibilidade e, isoladamente, o uso do algoritmo não é capaz de fornecer a lista de estados mínima. Como resultado final, ele indica apenas as denominadas “classes de máxima compatibilidade”, de onde pode ser retirada a lista de estados mínima. Para esses casos, o aplicativo fornece as classes de máxima compatibilidade, deixando a cargo do usuário a análise do problema e a tomada de decisão. Isso é interessante, do ponto de vista didático, porque o trabalho bruto é realizado pelo aplicativo enquanto a lapidação intelectual é trabalhada pelo aluno.

O algoritmo não possui uma complexidade matemática muito elevada, mas uma explicação detalhada da teoria que o fundamenta foge do escopo do documento em questão. Portanto, são descritas, a seguir, apenas as etapas básicas implementadas.

O primeiro passo na implementação do algoritmo é montar uma tabela de duplas de estados (inicialmente) compatíveis e uma tabela de duplas de estados proibidas (pares de estados não compatíveis), a partir da informação dos valores de saída.

Em seguida, é montada a tabela de duplas de estados implicadas, com base na informação de próximos estados.

Com a informação das duplas de estados proibidas, inicialmente encontradas, são eliminadas as duplas de estados incompatíveis da tabela de duplas de estados implicadas.

A partir dessa primeira eliminação, realiza-se um processo de novas eliminações que só é interrompido no momento em que mais nenhuma eliminação possa ser realizada.

Dois tipos de resultado podem surgir dessas eliminações. Primeiramente, uma lista de estados completamente isolados (não redundantes) pode ser encontrada. Além disso, pode-se obter uma lista de classes de estados maximamente compatíveis (conjuntos de estados redundantes que podem ser resumidos a um único estado para cada classe).

Até esse ponto, o algoritmo é o mesmo para ambos os tipos de tabelas de entrada.

No caso de uma descrição inicial dada por uma tabela de estados completamente especificada, a lista de classes de estados maximamente compatíveis já representa a lista de estados equivalentes da máquina. Portanto, o passo final é apresentar a tabela de estados

minimizada, formada por estados representantes das classes de equivalência e pelos estados isolados.

Por outro lado, para descrições por meio de tabelas de estados não completamente especificadas, o algoritmo chegou ao seu fim e o usuário fica responsável por encontrar o número mínimo de classes, a partir da lista de classes de estados maximamente compatíveis, e por adicionar os estados isolados para formar a tabela de estados mínima.

6. EXEMPLOS DE EXECUÇÃO

O número de padrões diferentes a serem testados e a quantidade de dados produzidos pelo aplicativo inviabilizam sua exposição completa no corrente texto. Assim, é apresentado aqui apenas um resumo de alguns exemplos de execução, os quais ilustram algumas variações em torno das possibilidades básicas enfrentadas pelo algoritmo: descrição por tabela de estados completamente especificada ou não, com possibilidade de simplificação ou não.

Exemplo 1: Tabela de estados completamente especificada, com simplificação.

Entrada de dados:

```
estados = {"1","2","3","4","5","6","7","8",}

prox_estados =
 {
  ["1"] = {"2","7","3","8",},
  ["2"] = {"8","1","7","6",},
  ["3"] = {"4","7","5","8",},
  ["4"] = {"8","3","7","2",},
  ["5"] = {"6","7","1","8",},
  ["6"] = {"8","5","7","4",},
  ["7"] = {"8","5","4","7",},
  ["8"] = {"7","6","3","8",},
 }

saidas =
 {
  ["1"] = {"4", "3", "2", "1",},
  ["2"] = {"2", "1", "4", "3",},
  ["3"] = {"4", "3", "2", "1",},
  ["4"] = {"2", "1", "4", "3",},
  ["5"] = {"4", "3", "2", "1",},
  ["6"] = {"2", "1", "4", "3",},
  ["7"] = {"1", "3", "2", "4",},
  ["8"] = {"3", "1", "4", "2",},
 }
```

Resultado final:

```
classes_max_compat = {{"1","3","5",},{"2","4","6",},{"7",},{"8",},}

estados = {"1","2","7","8",}

prox_estados =
 {
  ["1"] = {"2","7","1","8",},
  ["2"] = {"8","1","7","2",},
  ["7"] = {"8","1","2","7",},
  ["8"] = {"7","2","1","8",},
 }

saidas =
 {
  ["1"] = {"4", "3", "2", "1",},
  ["2"] = {"2", "1", "4", "3",},
  ["7"] = {"1", "3", "2", "4",},
  ["8"] = {"3", "1", "4", "2",},
 }
```

Exemplo 2: Tabela de estados completamente especificada, sem simplificação.

Entrada de dados:

```
estados = {"a","b","c","d",}

prox_estados =
 {
  ["a"] = {"b","c","a","d",},
  ["b"] = {"d","a","c","b",},
  ["c"] = {"d","a","b","c",},
  ["d"] = {"c","b","a","d",},
 }

saidas =
 {
  ["a"] = {"4", "3", "2", "1",},
  ["b"] = {"2", "1", "4", "3",},
  ["c"] = {"1", "3", "2", "4",},
  ["d"] = {"3", "1", "4", "2",},
 }
```

Resultado final:

```
classes_max_compat = {{"a",},{"b",},{"c",},{"d",},}

estados = {"a","b","c","d",}

prox_estados =
 {
  ["a"] = {"b","c","a","d",},
  ["b"] = {"d","a","c","b",},
  ["c"] = {"d","a","b","c",},
  ["d"] = {"c","b","a","d",},
 }

saidas =
 {
  ["a"] = {"4", "3", "2", "1",},
  ["b"] = {"2", "1", "4", "3",},
  ["c"] = {"1", "3", "2", "4",},
  ["d"] = {"3", "1", "4", "2",},
 }
```

Exemplo 3: Tabela de estados não completamente especificada.

Entrada de dados:

```
estados = {"R","1","2",}

prox_estados =
 {
  ["R"] = {"R","1",},
  ["1"] = {"2","R",},
  ["2"] = {"1","R",},
 }

saidas =
 {
  ["R"] = {"-","0",},
  ["1"] = {"0","0",},
  ["2"] = {"1","0",},
 }
```

Resultado final:

estados_isolados = { }

Classes maximamente compatíveis:
classes com 1 elemento: {{"R",},{"1",},{"2",},}

classes com 2 elementos: {{"R","1",},{"R","2",},}
duplas implicadas: {{"2","R",},{"1","R",},}

7. CONCLUSÃO E TRABALHOS FUTUROS

Um pacote de aplicativos didáticos dedicado ao ensino de Circuitos Digitais, incluindo as interfaces com o usuário para tais aplicativos, encontra-se em desenvolvimento pelo grupo PET-Tele. Em relação aos integrantes do grupo, o objetivo é fazer com que eles colaborem com a melhoria das práticas de ensino. Para os alunos do curso, o objetivo é fornecer a eles ferramentas auxiliares para estudo e para a verificação de resultados.

Como trabalho inicial, foi elaborado um manual interativo multimídia para o algoritmo básico de minimização de funções booleanas conhecido por Algoritmo de Quine-McCluskey, codificado em NCL e empregando diferentes mídias (áudio, texto e imagem).

A partir do manual desenvolvido, e visualizando a possibilidade de implementar um pacote de aplicativos, com diversas opções de interação (TV Digital, local ou via rede de computadores), o grupo implementou o Algoritmo de Quine-McCluskey, utilizando a linguagem de programação Lua.

Dando prosseguimento, foi desenvolvido um aplicativo que realiza conversões entre as expressões de funções lógicas mais comumente utilizadas, também utilizando a linguagem de programação Lua. Uma vez que a expressão mínima também é uma forma de expressão para uma função lógica, o aplicativo de minimização anteriormente desenvolvido foi incorporado no atual aplicativo de conversão.

Esse trabalho apresentou a implementação do algoritmo de minimização de estados conhecido como Algoritmo de Paul-Unger, empregado no ensino de Circuitos Digitais, agora incorporado ao pacote de aplicativos didáticos.

Como trabalhos futuros, o grupo PET-Tele já iniciou o desenvolvimento de interfaces com o usuário, a fim de oferecer uma versão NCL (para TV Digital), uma versão para uso local e uma versão para uso via rede de computadores para o pacote de aplicativos didáticos.

AGRADECIMENTOS

O grupo PET-Tele faz parte do Programa de Educação Tutorial (PET), financiado pelo Ministério da Educação (MEC).

O grupo agradece aos alunos da disciplina “Circuitos Digitais” que ajudaram a testar o aplicativo desenvolvido.

REFERÊNCIAS BIBLIOGRÁFICAS

BEPPU, M.M.; AMARAL, V.R.L.; DE LA VEGA, A.S. Ferramenta de Auxílio Didático: Algoritmo de Quine-McCluskey em Lua. **Anais: XXXVIII - Congresso Brasileiro de Educação em Engenharia – COBENGE**. Fortaleza, v.1, n.1, p. 1-7, 2010.

COMUNIDADE GINGA. **Ginga-NCL**. Disponível em: <http://www.gingancl.org.br> Acesso em: 01 mai. 2012.

HILL, F. J.; PETERSON, G. R. **Introduction to Switching Theory and Logical Design**. 3.ed. New York: John Wiley, 1981. 636 p, il.

IERUSALIMSCHY, R. **Programming in Lua**.2. ed. Rio de Janeiro: Lightning Source Inc, 2006. 328 p, il.

IERUSALIMSCHY, R. **The Programming Language Lua**.
Disponível em: <http://www.lua.org> Acesso em: 01 mai. 2012.

KEPLER PROJECT. **CGILua: Building Web Scripts with Lua**.
Disponível em: <http://keplerproject.github.com/cgilua> Acesso em: 01 mai. 2012.

MATTOS, H.; SOUZA, T.; DE LA VEGA, A.S.; SAADE, D.M. Ferramenta Didática Interativa Utilizando a Linguagem NCL: Algoritmo de Quine-McCluskey. **8th International Information and Telecommunication Technologies Symposium**, Florianópolis, v.1, n.1, p. 1-4, 2009.

MCCLUSKEY, E. J. Minimization of Boolean Functions. **Bell System Tech. J.**, New Jersey, v.35, n.5., p. 1417-1444, 1956.

Paul, M.C.; Unger, S.H. Minimizing de Number of States in Incompletely Specified Sequential Switching Functions. **IRE Transactions on Electronic Circuits**, v.EC-8, n.3, p. 356-357, 1959.

MINISTÉRIO DA EDUCAÇÃO. **Apresentação – PET**. Disponível em: <http://portal.mec.gov.br/index.php?option=com_content&view=article&id=12223 &ativo=481&Itemid=480> Acesso em: 01 mai. 2012.

PAUL, M.C.; UNGER, S.H. Minimizing de Number of States in Incompletely Specified Sequential Switching Functions. **IRE Transactions on Electronic Computers**, v.EC-8, n.3, p. 356-357, 1959.

PET-TELE. **PET – Engenharia de Telecomunicações da UFF.**
Disponível em: <http://www.telecom.uff.br/pet> Acesso em: 01 mai. 2012.

PUC-RIO. **Tecgraf – Tecnologia em Computação Gráfica**.
Disponível em: <http://www.tecgraf.puc-rio.br> Acesso em: 01 mai. 2012.

RHYNE, V. T. **Fundamentals of Digital Systems Design**. New York: Prentice-Hall, 1973. 560 p, il.

SANTOS, B.S.; DE LA VEGA, A.S. Ferramenta de Auxílio Didático: Conversão de Expressões de Funções Lógicas em Lua. **Anais: XXXIX - Congresso Brasileiro de Educação em Engenharia – COBENGE**. Blumenau, 2011.

SANTOS, B.S.; DE LA VEGA, A.S. Desenvolvimento de Ferramenta de Auxílio Didático: Implementação de Interfaces com o Usuário via Internet usando CGILua. **Anais: XL - Congresso Brasileiro de Educação em Engenharia – COBENGE**. Belém, 2012.

SOARES, L. F. G.; BARBOSA S. D. J. **Programando em NCL 3.0: Desenvolvimento de aplicações para o middleware Ginga**.1. ed. Rio de Janeiro: Campus, 2009. 341 p, il.

WIKIPEDIA. **ISDB-T International – Wikipedia, the free encyclopedia**. Disponível em:<http://en.wikipedia.org/wiki/ISDB-T_International>Acesso em:01 mai. 2012.

DEVELOPMENT OF A LEARNING AID TOOL: STATE MACHINE MINIMIZATION ALGORITHM CODED IN LUA

***Abstract**: The aim of this job was to code the state machine minimization algorithm known as Paul-Unger Algorithm by using Lua programming language. The initial motivation to develop such a software tool was to include it into a learning aid tools package that is under construction by the PET-Tele group. The package is intended to be used on Digital Circuits learning. The group has already been working on building some different kinds of user interfaces for this software in order to offer different ways of usage.*

***Key-words**: Learning aid tool, Distance education, Digital Circuits, State Machine Minimization, Lua Programming Language.*