Comparative Study Between Synthesis Tools for Digital Systems

Ohira M. I., Zanella, F. P., Luca H.P., Silva A.C.R.

ikeguchi@dee.feis.unesp.br, zanella@dee.feis.unesp.br, helder@dee.feis.unesp.br, acrsilva@dee.feis.unesp.br

Universidade Estadual Paulista Júlio de Mesquita Filho
Faculdade de Engenharia de Ilha Solteira
Departamento de Engenharia Elétrica
Av Brasil 56 , 15385-000 Ilha Solteira, - SP - Brasil

This paper presents the evaluation of performance of the project tools called *Tabela* and *Tab2VHDL*, developed for synthesis of machines of finite states synchronous. The synthesis tools used for evaluation were Leonardo Spectrum, by Exemplar Logic Inc, environment Max+plus II, by Altera and the own software *Tabela* and *Tab2VHDL* developed in language C.

From the state diagram describing the operation of the machine, the program *Tabela* obtains all the functions of element control of memories and all the output functions. It can be chosen as element of memory the flip-flops type D and JK. The created combinational functions are in their minimum form, whose minimization algorithm used is Quine-McCluskey.

The program *Tab2VHDL* create from the file generated by the program *Tabela*, a RTL model (Registers of Transfer Logic) in description language VHDL (Very high speed integration Hardware Description Language).

The software Leoonardo Spectrum besides a logical synthesizer, is an analysis tool and a developed optimizator FPGA (Field Programmable Gate Array) , a cPLD (complexing Programmable Logic Device), a ASIC (Application Specific Integrated Circuit), etc. It was used different techniques of state allocation (Simplest, One-Hot e Almost-One-Hot), whose codes were implemented in the description language AHDL (Altera Hardware Description Language).

It was chosen for study of synthesis tool the codes AMI (Alternated Mark Inversion), HDB1 (High-Density Bipolar First-Order Coding) and HDB3 (High-Density Bipolar Third-Order Coding) due to their importance in transmission of data in base band through a pair of wire (phone pair), that is the simplest, the most practical and the most economical way of data communication in urban perimeters or distances of some kilometers. Nowadays the analogical voice channels of the phone system, whose maximum capacity is about 28,8Kbit/s, they have been substituted by digital channels of highspeed of IDN (Integrated Digital Net), which allow multiple rates from 64Kbit/s to 2,048Mbit/s. These changes are creating the necessity of equipments which allow the access to these digital channels through the subscriber line, known as digital modems or band base, also called CSU/DSU (Channel Service Unit / Data Service Unit). The unit called CSU accomplishes the connection from terminal equipment of data (TED) to the subscriber line of the digital net. The CSU should be able to accomplish the conditioning of the line as well as the filtering and equalization of the signs. It also should be able to detect long sequences of zeros to guarantee the timing information, and to correct bipolar violations [1].

The codes were all simulated in the environment Max+puls II and for that input files were used, for the same circuit, with different treatments, in other words, using different tools for the code synthesis and different state allocations. Those input files were: the circuit in language VHDL generated by program *Tab2VHDL*, the same circuit, however, optimized by the software Leonardo Spectrum and the circuit in language AHDL (in which the state allocations were tested).

Although these input files describe the same circuit, they have differences each other and in the simulation these differences are described in a control file called *report file*, were it's described the logical functions cost (defined as the number of input ports of the logical gates in their primitive format), the number of used cells (for animplementation in FPGA), quantity of memory and synthesis time. Such files were used to make the comparison among the implementation and the output file of the program *Tabela* was also used to compare logical costs.

All the codes were implemented in the programmable component MAX7000E, Device: EPM 7128ELC84-7 and the simulations prove their perfect operation.

The simulations are regarding each code are shown in the Figure 1.

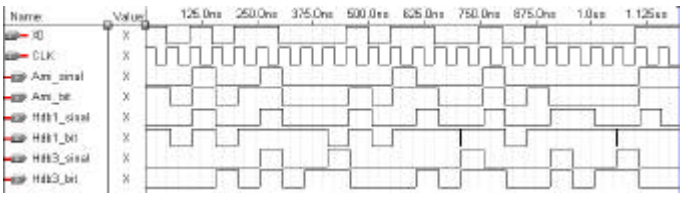

Figure 1 – Simulation of the codes for a certain input.

The Figure 2 shows the block diagram of the project to be accomplished, since the input (state diagram) until the final result, that is a simulable and synthesizable digital circuit.

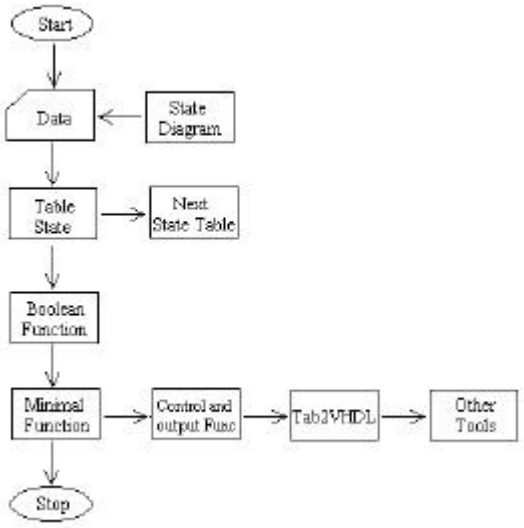

Figure 2 - Block diagram of the program *TABELA* and next steps.

Initially generate a states diagram of a certain state machine. To exemplify the process of Figure 2, had used the HDB1 code, whose states diagram is shown in the Figure 3.

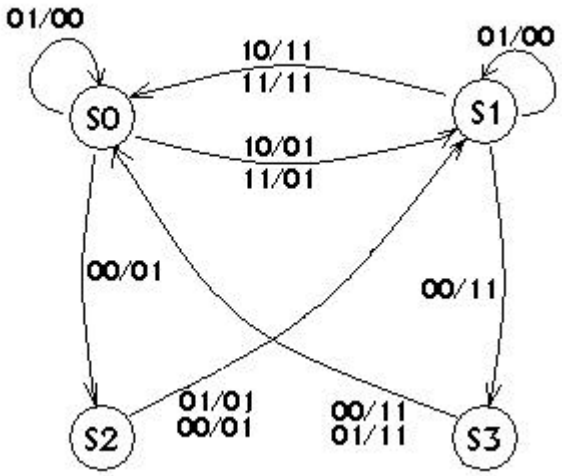

Figure 3 – States Diagram of HDB1 code.

Starting from the states diagram, make the states declaration in format demanded by the program *Tabela*, that should follow this order: flip-flops number, type of each flip-flops (D or JK), number of inputs, number of outputs and transitions of states (current state, next state, input and output, in this order) and all the declarations in decimal representation.

The Figure 4 shows the use of the program *Tabela* in the description of the state diagram shown in the Figure 3.

```
2              <- Nº of Flip-Flops
D              <- Type of Flip-Flops
D              <- Type of Flip-Flops
2 2            <- Nº of inputs and outputs respectively
0 1 1 1        <- Current State/ Next State/ Input/ Output
0 1 3 1
0 0 2 0
0 2 0 1
1 1 2 0
1 0 1 3
1 0 3 3
1 3 0 3
2 1 0 1
2 1 2 1
3 0 0 3
3 0 2 3
-100           <- end file.
```

Figure 4 – States Diagram of HDB1 code, in program TABELA.

After the file compilation in the program *Tabela*, this generates an output file, as shown in the Figure 5.

```
!DE!P/!ENTRADA  !MINT!!Q1!Q1+!D1!!Q0!Q0+!D0!!Z1!Z0!
! 3 ! 0!      2 (10) ! 11   !!  1!   0! 0!!  1!   0!  0!! 1!  1!
! 3 ! 0!      0 (00) ! 3    !!  1!   0! 0!!  1!   0!  0!! 1!  1!
! 2 ! 1!      2 (10) ! 10   !!  1!   0! 0!!  0!   1!  1!! 0!  1!
! 2 ! 1!      0 (00) ! 2    !!  1!   0! 0!!  0!   1!  1!! 0!  1!
! 1 ! 3!      0 (00) ! 1    !!  0!   1! 1!!  1!   1!  1!! 1!  1!
! 1 ! 0!      3 (11) ! 13   !!  0!   0! 0!!  1!   0!  0!! 1!  1!
! 1 ! 0!      1 (01) ! 5    !!  0!   0! 0!!  1!   0!  0!! 1!  1!
! 1 ! 1!      2 (10) ! 9    !!  0!   0! 0!!  1!   1!  1!! 0!  0!
! 0 ! 2!      0 (00) ! 0    !!  0!   1! 1!!  0!   0!  0!! 0!  1!
! 0 ! 0!      2 (10) ! 8    !!  0!   0! 0!!  0!   0!  0!! 0!  0!
! 0 ! 1!      3 (11) ! 12   !!  0!   0! 0!!  0!   1!  1!! 0!  1!
! 0 ! 1!      1 (01) ! 4    !!  0!   0! 0!!  0!   1!  1!! 0!  1!

FUNCAO D1

MINTERMOS :  0;  1;

IMPLICANTES PRIMOS ESSENCIAIS :

ESSENCIAL:  0    REDUNDANCIA:     1 ->  000X

CUSTO FINAL DE D1 = 3

FUNCAO D0

MINTERMOS :  4; 12;  9;  1;  2; 10;

IMPLICANTES PRIMOS ESSENCIAIS :

ESSENCIAL:  2    REDUNDANCIA:    12 ->  XX10
ESSENCIAL:  1    REDUNDANCIA:     8 ->  X001
ESSENCIAL:  4    REDUNDANCIA:    10 ->  X1X0

CUSTO FINAL DE D0 = 10

FUNCAO Z1

MINTERMOS :  5; 13;  1;  3; 11;

IMPLICANTES PRIMOS ESSENCIAIS :

ESSENCIAL:  3    REDUNDANCIA:    12 ->  XX11
ESSENCIAL:  1    REDUNDANCIA:     6 ->  0XX1
ESSENCIAL:  5    REDUNDANCIA:    10 ->  X1X1

CUSTO FINAL DE Z1 = 9

FUNCAO Z0

MINTERMOS :  4; 12;  0;  5; 13;  1;  2; 10;  3; 11;

IMPLICANTES PRIMOS ESSENCIAIS :

ESSENCIAL:  2    REDUNDANCIA:    13 ->  XX1X
ESSENCIAL:  0    REDUNDANCIA:     7 ->  0XXX
ESSENCIAL:  4    REDUNDANCIA:    11 ->  X1XX

CUSTO FINAL DE Z0 = 6

CUSTO TOTAL DAS 4 FUNCOES = 28
```

Figure 5 – Program TABELA output file.

The software *Tab2VHDL* generates the VHDL code directly from the output file of program *Tabela* (Figure 5).

The Figure 6 shows the VHDL model of the line code projected, generated by the *TAB2VHDL* program from the program *Tabela* output file.

```
-- Inferindo flip-flop tipo D
PROCESS(CLK, CLR)
BEGIN
   IF CLR = '0' THEN
     VE0 <= '0';
   ELSIF CLK'EVENT and CLK = '1' THEN
     VE0 <= D0;
   END IF;
   Q0 <= VE0;
END PROCESS;

-- Inferindo flip-flop tipo D
PROCESS(CLK, CLR)
BEGIN
   IF CLR = '0' THEN
     VE1 <= '0';
   ELSIF CLK'EVENT and CLK = '1' THEN
     VE1 <= D1;
   END IF;
   Q1 <= VE1;
END PROCESS;

 -- Processos que implementam as funcoes combinacionais de controle
  -- dos Elementos de memoria e de Saidas.

  D1 <= ( NOT(X1) AND NOT(X0) AND NOT(VE1));
  D0 <= ( (VE1) AND NOT(VE0)) OR (NOT(X0) AND NOT(VE1) AND (VE0)) OR ((X0) AND NOT(VE0));
  Z1 <= ( (VE1) AND (VE0)) OR (NOT(X1) AND (VE0)) OR ((X0) AND (VE0));
  Z0 <= ( (VE1)) OR (NOT(X1)) OR ((X0));

END RTL;
```

Figure 6 – HDB1 in VHDL language.

From the circuit description in VHDL language, has made the compilation of the file in the software MAX+plus II, whose were obtained the datas costs by the report file. The most interest datas are shown in the Figure 7.

```
** DEVICE SUMMARY **
Device EPM7128ELC84-7

Chip/          Input  Output  Bidir      Shareable
POF            Pins   Pins    Pins   LCs Expanders  % Utilized

hdb1_s2        4      4       0      4   0           3 %

User Pins:     4      4       0

** EQUATIONS **

CLK     : INPUT;
CLR     : INPUT;
X0      : INPUT;
X1      : INPUT;

-- Node name is 'Q0' = 'VE0'
-- Equation name is 'Q0', location is LC115, type is output.
 Q0      = TFFE(!_EQ001, GLOBAL( CLK), GLOBAL( CLR), VCC, VCC);
 _EQ001 = !Q1 & !X0;

-- Node name is 'Q1' = 'VE1'
-- Equation name is 'Q1', location is LC117, type is output.
 Q1      = DFFE( _EQ002 $ GND, GLOBAL( CLK), GLOBAL( CLR), VCC, VCC);
 _EQ002 = !Q1 & !X0 & !X1;

-- Node name is 'Z0'
-- Equation name is 'Z0', location is LC120, type is output.
 Z0      = LCELL( _EQ003 $ VCC);
 _EQ003 = !Q1 & !X0 & X1;

-- Node name is 'Z1'
-- Equation name is 'Z1', location is LC118, type is output.
 Z1      = LCELL( _EQ004 $ Q0);
 _EQ004 = Q0 & !Q1 & !X0 & X1;

Compilation Times
-----------------

  Compiler Netlist Extractor      00:00:02
  Database Builder                00:00:00
  Logic Synthesizer               00:00:00
  Partitioner                     00:00:01
  Fitter                          00:00:01
  Timing SNF Extractor            00:00:00
  Assembler                       00:00:00
  --------------------------      --------
  Total Time                      00:00:04


Memory Allocated
----------------

Peak memory allocated during compilation  = 5,325K
```

Figure 7 – Report File from MAX+plus II software.

The costs were obtained by the report files through the boolean equations.

For the results obtained from the program *Tabela*, the costs are shown in the table 1.

TABLE 1 – Total costs obtained, for each code, by the program *Tabela*.

Code	Total Cost
AMI	09
HDB1	28
HDB3	88

From the report file created by the software Max+puls II and from the allocations, the total costs for each code were obtained. The results are presented in table 2:

TABLE 2 – Costs obtained for each code, being varied the allocation type.

Code	Assign	Logic Cost	Used Cell	Used Memory	Compilation Time
AMI	Simplest	012	03	3,622 K	00:00:07
	One-Hot	012	04	3,670 K	00:00:03
	Almost One-Hot	012	03	3,499 K	00:00:04
HDB1	Simplest	028	04	3,688 K	00:00:04
	One-Hot	057	06	3,786 K	00:00:04
	Almost One_hot	058	06	6,197 K	00:00:02
HDB3	Simplest	340	14	3,687 K	00:00:09
	One-Hot	268	34	5,598 K	00:00:10
	Almost One-Hot	159	34	3,888 K	00:00:06

From the VHDL model, created by Leonardo Spectrum and implemented by the software Max+plus II, was obtained the total costs, shown in the table 3:

TABLE 3 – Total costs obtained, for each code, by Leonardo Spectrum.

Code	Logic Cost	Used Cell	Used Memory	Compilation Time
AMI	**08**	**3**	**3,509 K**	**00:00:04**
HDB1	**12**	**4**	**3,487 K**	**00:00:13**
HDB3	**65**	**7**	**3,646 K**	**00:00:15**

From the model VHDL, created by program *Tab2VHDL* and implemented by the software Max+plus II, is obtained the total costs, shown in the table 4.

TABLE 4 – Total costs obtained, for each code, by the program *Tab2VHDL*.

Code	Logic Cost	Used Cell	Used Memory	Compilation Time
AMI	**08**	**3**	**3,492 K**	**00:00:08**
HDB1	**12**	**4**	**5,325 K**	**00:00:04**
HDB3	**65**	**7**	**5,325 K**	**00:00:05**

Conclusion

When analyzing the presented results it can be concluded that the tool *Tab2VHDL* has a great performance in relation to the costs for implementation time. However the software Leonardo Spectrum presented the great advantage of reducing the quantity of used memory, what is a very important for the circuit of large complexibility. However the program Leonardo Spectrum is a commercial software which is always in development and updating. On the other hand, the programs *Tabela* and *Tab2VHDL* are programs originating from scientific productions in which there isn't the main preoccupation about the continuity in its development. So it's possible to conclude that the relation cost – benefit among the analyzed tools is better for the programs *Tabela* and *Tab2VHDL*, because they don't incur in high costs end they present results that are very similar to the commercial programs.

In relation to the methods of state allocations, it's possible to notice that their effect is not very significant in cases in which tools that optimize the circuit are used, however, when there isn't availability of any tool of this level, as the ones presented in this paper, the change of state allocation becomes very valuable, mainly for circuits with a certain complexity degree.

References

[1] A. M. Parisoto, *et all*, *Implementação em EPLDs de módulos integráveis de um modem banda base*, UFRGS – Porto Alegre, RS – Brasil

[2] I. S. Bonatti, M. C. G. *Madureira, Introdução à Análise e Síntese de Circuitos Lógicos*; Editora Unicamp, Campinas, 1990.

[3] "Max + Plus II Getting Started" Version 9.1, Copyright © 1995 - 2002 Altera Corporation, 101 Innovation Drive, San Jose, California 95134, USA.

[4] D. J. Comer, "Digital Logic and State Machine Design." 3th ed.; New York: Oxford University Press, 1995.

[5] J.F. Wakerly, "Digital Design Principles and Practices", *State Assignment*, Chap. 5, p.387-390, ?

[6] "Flex 8000 - State Machine Encoding". San Jose, CA: Altera Corp., ver 1, p.187-189, May 1994.

[7] "Max + Plus II AHDL" Version 6.0, Copyright © 1995 – November 1995, Altera Corporation, 101 Innovation Drive, San Jose, California 95134, USA.

[8] *Manual Digital do Programa TABELA*; version 2.7; July 1991

[9] www.altera.com

[10] www.exemplar.com

[11]www.ieee.org/pubs/authors.html