A Library of Standard Cells Suitable for Use on a Free EDA Silicon Compiler Tool

Carlos R. dos Santos, Estêvão C. Teixeira
Engineering College
Federal University of Juiz de Fora – UFJF
Juiz de Fora – MG, Brazil
carlos.rafael@engenharia.ufjf.br, estevao.teixeira@ufjf.edu.br

Abstract— **The Electric VLSI Design System is a free Electronic Design Automation package that includes different tools, as schematic and layout editors, verification tools and a Silicon Compiler, among others. This work presents a standard cells library designed for the 0.5-µm process, to be used with the Silicon Compiler tool in order to synthetize a digital circuit from its structural VHDL description. An eight-bit, up/down synchronous counter is proposed as an example, with simulation results.**

Keywords—Standard cells; silicon compiler; digital circuits; free EDA tools; structural VHDL

I. INTRODUCTION

There are several methodologies for the automation of the design of digital integrated circuits. One of them is the use of a silicon compiler, a tool that, in a general meaning, converts some description of the circuit into its actual layout [1, 2].

Several approaches for silicon compilers were proposed since the first works, that were published at the end of 1970s [3]. A great variety of approaches arose after this, as can be seen in [4,5]. Nowadays, a silicon compiler can be understood as a tool that structures the layout from a circuit description using a hardware description language (HDL). To do this, a silicon compiler typically makes use of standard cells, that are pre-designed and pre-characterized blocks implementing basic logic functions [6].

Some free or open source Electronic Design Automation (EDA) tools were investigated for use in VLSI design courses [7]. One of them is the Electric VLSI Design System, a complete suite for schematic capture, layout and verification of integrated circuits [8].

If compared with other packages, Electric has a reasonable variety of resources and a rapid learning curve. Moreover, the package runs under the Java Runtime Environment (JRE), and is portable to different operating systems. These factors make it as a good software alternative for a one-semester discipline of integrated circuits on a graduation course [9].

One of the resources available in Electric is one silicon compilation system called QUISC (Queen's University Interactive Silicon Compiler), a tool that does the placement and routing of the standard cells from a schematic or a structural VHDL description [10].

The Electric has a built-in library of standard cells, that are invoked by the QUISC if no specific library is used. However, it is explicit in the software manual that the cells contain errors, and are used just to illustrate the operation of the tool. Other libraries of standard cells were designed in Electric, by other laboratories and universities, for different CMOS technologies, and are available in [8].

A library of standard cells designed for the 0.5-µm process [8, 11] was tested with the QUISC. Although these cells are compact and free of errors themselves, design rules checking (DRC) verification of the layout obtained after silicon compilation using these cells reported a great number of errors.

This paper presents a library of standard cells developed in the Electric software, for the 0.5-µm CMOS process, suitable for use with the Silicon Compiler. The silicon compilation with this library results in a layout with no DRC errors.

To illustrate the design procedure, an eight-bit, up/down synchronous counter was synthetized. The simulation of the circuit extracted from the layout was done by using LTSpice, a free Spice engine with no node limitation or simulation time restrictions.

The paper is organized as follows: The Section II discusses briefly the design methodology used in Electric. Section III presents the library of standard cells and their main features. At the Section IV, the project of the counter is described, as well as the resulting layout and simulation results. The Section V presents the conclusions of the work.

II. THE ELECTRIC DESIGN METHODOLOGY

First to describe the designed library of standard cells, it is interesting to discuss how a cell (analog or digital) is created in Electric. Its design methodology can slightly differ from that used by commercial tools, and a brief explanation about it can illustrate better the motivations for the design of the standard cells.

The layout of a block in Electric is done by using a set of primitives called *nodes*, that can be a pre-configured MOSFET, a Poly-to-Metal1 contact, or a NWELL-to-Metal1 contact, as examples. All the structures used in a standard CMOS design are available as nodes and can be easily scaled. The connection between the nodes are done by using *arcs*, that can be lines of metal, active (n or p), well, poly or another layer.

A cell is then composed by a set of nodes and arcs. The connection between a specific cell and other ones is done by explicitly indicating the points where a metal line (or another layer) has to be connected, acting actually as a pin. These points are called *exports*, and when a cell is instantiated on another layout, the Electric does not recognize a connection at other points different from the exports.

The Fig. 1 shows the interconnection between nodes, arcs and exports. The cell contains the primitives necessary to layout a n-channel MOSFET with all the Metal connections. In practice, the Metal-to-Active nodes can be placed close to the NMOS node. However, it is necessary to them to be linked by an "Active" arc, in order to make an explicitly electrical connection.

The Electric has the MOSIS Scalable Rules (SCMOS) built-in [12]. A layout developed under these rules makes use of a parameter lambda (λ), that defines the minimum spacing and distance rules. For a 0.5-µm process, it is set λ = 300 nm. A minimum channel length in SCMOS rules is 2λ, so it has 0.6 µm for the referred process.

III. The Library of Standard Cells

The designed library was created by placing the exports in order to allow the vertical connections with Metal-2, while the power rails and the horizontal connections are made with Metal-1. The exports are intentionally misaligned, so that a vertical line that connect to an export cannot pass across another export in that cell. A distance between the exports is also needed in order to avoid spacing errors between the vertical Metal-2 lines. A generic representation of a cell is given in Fig. 2. The chosen distance between rails was of 93λ (27.9 µm), and the power rails width is of 8λ (2.4 µm).

The basic inverter is shown in Fig. 3a. For this cell, the gate capacitance, obtained by simulation, is approximately 11.5 fF. Other three cells are shown: two-input NAND gate (Fig. 3b); tristate gate (Fig. 3c), and a transmission gate (Fig. 3d).

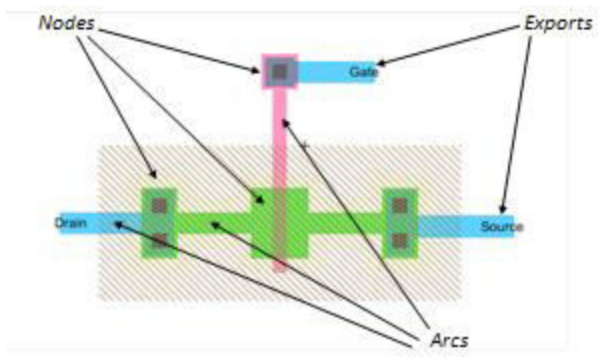

Fig. 1. A simple layout illustrating examples of nodes, arcs and exports in Electric.

The cells were designed according to the SCMOS rules. All the transistors have the minimum allowable channel length, 2λ (0.6 µm). For all the cells, the bulk contacts are present.

The cells were extracted and simulated in LTSpice, in order to achieve their parameters. The basic inverter capacitance (~11.5 fF) was used as a load. The Table I shows the list of cells and their parameters.

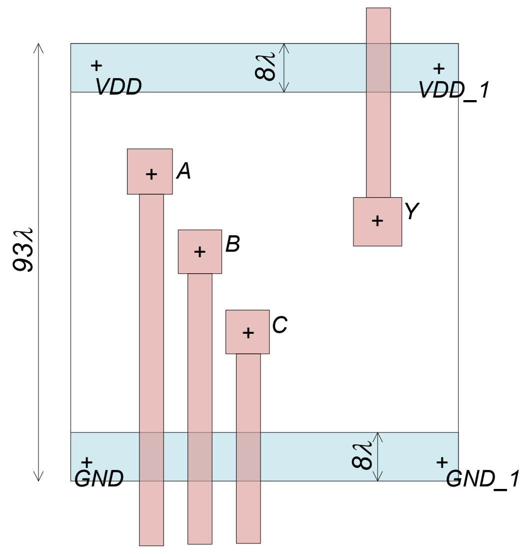

Fig. 2. A generic representation of a cell. Horizontal lines are of Metal 1, and vertical lines of Metal 2.

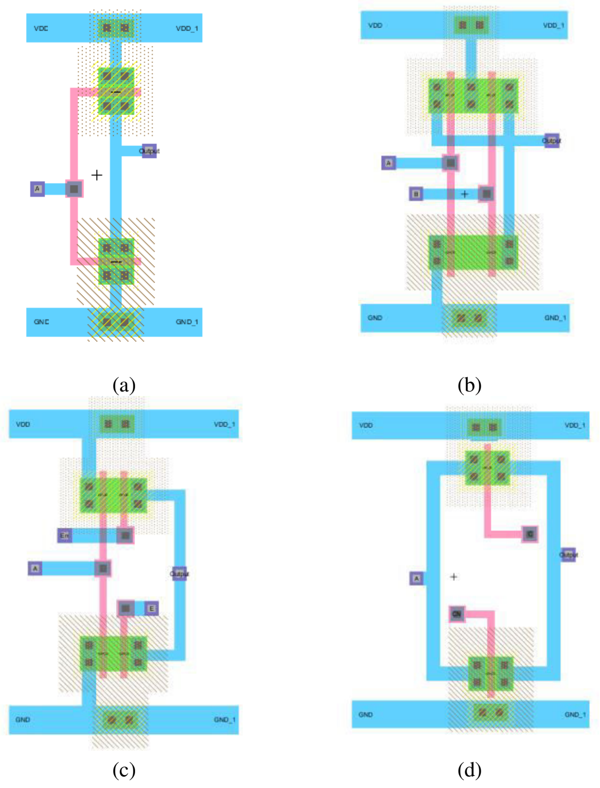

Fig. 3. Layouts of four cells: (a) Basic inverter (b) Two-input NAND; (c) Tristate gate; (d) Transmission gate.

If compared to the 0.5-µm library available in [8], these cells have, in general, a greater area. This was done in order to let an automatic routing without DRC errors. The application of this library on silicon compilation is shown in the next section.

TABLE I. PARAMETERS OF THE STANDARD CELLS.
CONDITION: C_L = ~11.5 fF. TIMES ARE EXPRESSED IN PICOSSECONDS.

Cell	Rising time	Falling time	Rising delay	Falling delay	Area (μm^2)
And2	227	169	316	422	945.5
And3	247	184	342	436	1074
Buffer1x	202	132	186	345	743.6
Buffer4x	89	85	89	284	789.5
Inv1x	335	288	283	56.5	459
Inv4x	228	220	210	34.5	395
Nand2	386	405	351	163	555.4
Nand3	481	494	368	213	734.4
Nor2	396	310	340	55.3	642.6
Nor3	825	394	530	104	784.9
Or2	214	184	252	490	945.5
Or3	221	235	260	588	1101.6
Transm	839	770	37.5	37.3	636.3
Trist	563	386	379	93.3	605.9

IV. DESIGN OF AN 8-BIT UP/DOWN COUNTER

To illustrate the application of the standard cells library using the QUISC, the design of an eight-bit synchronous up/down counter, with Preset and Clear asynchronous inputs, is described.

The Electric accepts VHDL and Verilog entries, and the design started with the structural VHDL description of the circuit. Unlike the behavioral VHDL (or other behavioral HDL), that describes what the cell does (at a higher level of abstraction), the structural VHDL specifies how a cell is composed by other cells or primitive gates [13].

The up/down counter is composed by a regular arrangement of blocks like the one illustrated in Fig. 4. It is formed by a JK Flip Flop (cell 1), two AND gates (cells 2, 3) and an OR gate (cell 4). For the first bit (*Q[0]*), *S(0)* is connected to V_{DD}, and the counting sequence is determined by setting *Up(0)* or *Down(0)* equal to 1. The outputs *Up(1)* and *Down(1)* are connected to inputs *Up(0)* and *Down(0)* of the following cell, for the next bits. The master-slave JK FF with asynchronous inputs was used, and its schematics is depicted in Fig. 5.

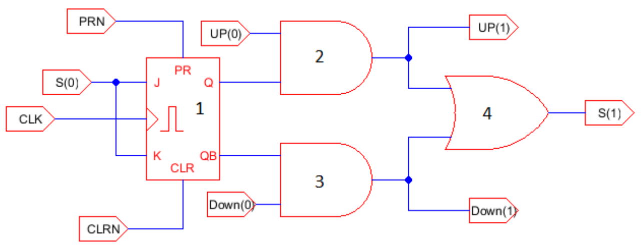

Fig. 4. Schematic of the block regularly placed to construct the 8-bit counter.

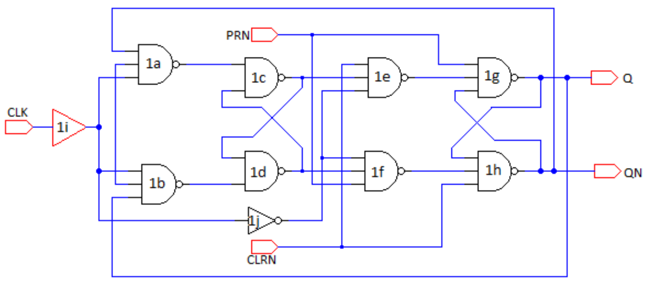

Fig. 5. Schematic of the JK FF.

By using the *GENERATE* statement, the arrangement of blocks related to outputs *Q[1]* to *Q[7]* can be described by the structural VHDL code shown in Fig. 6, which is part of the complete code that describes the counter (for the block related to output *Q[0]*, the code is slightly different). To give a better understanding of the code, the identifying number of the cells in Figs. 4 and 5 are placed as comments at each corresponding line.

Once written the structural VHDL code, the Electric converts it into a Netlist file. Then, the QUISC synthesizes the layout of the circuit. The given layout for the counter is shown in Fig. 7. A DRC verification showed that no errors were found.

The circuit was extracted in Electric and simulated in LTSpice using the 0.5-µm BSIM models. By using the nominal supply voltage (5 V), configuring the circuit as an up counter and applying a 1 MHz input clock signal, the transient simulation produced the results shown in Fig. 8. By configuring the circuit as a down counter, the results (not shown) were also according to the expected.

```
Gen: for I in 1 to 7 generate
 Gen1 : if I < 7 generate
  Bf_2: Buffer1 port map(clk, AB(I));                      -- (1i)
  In_2: Inv1 port map(AB(I),CLKN(I));                      -- (1j)
  Nm1: Nand3 port map(QN(I), AB(I), S(I-1), D(I));         -- (1a)
  Nm2: Nand3 port map(Q(I), AB(I), S(I-1), F(I));          -- (1b)
  Nm3: Nand2 port map(D(I), H(I), G(I));                   -- (1c)
  Nm4: Nand2 port map(G(I), F(I), H(I));                   -- (1d)
  Nm5: Nand3 port map(CLRN, CLKN(I),G(I),B(I));            -- (1e)
  Nm6: Nand3 port map(PRN, CLKN(I),H(I),C(I));             -- (1f)
  Nm7: Nand3 port map(B(I), PRN, QN(I), Q(I));             -- (1g)
  Nm8: Nand3 port map(C(I), CLRN, Q(I), QN(I));            -- (1h)

  Am1: And2 port map(L(I-1), Q(I), L(I));                  -- (2)
  Am2: And2 port map(M(I-1), QN(I), M(I));                 -- (3)
  O2: OR1 port map(L(I),M(I),S(I));                        -- (4)
end generate;
```

Fig. 6. Part of the structural VHDL code.

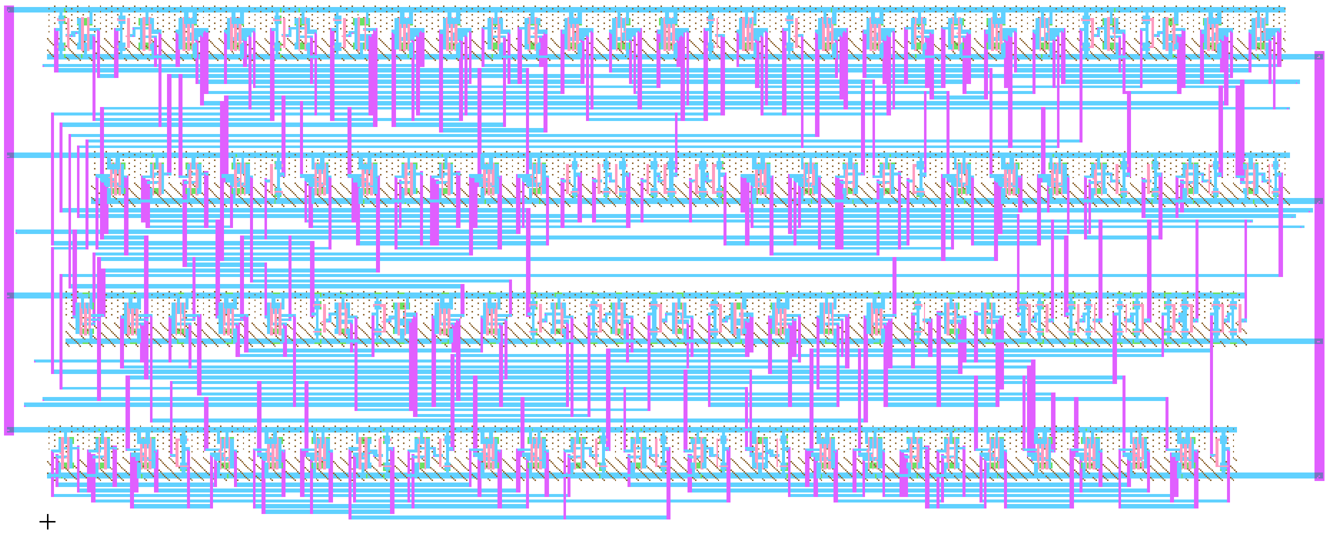

Fig. 7. Layout of the synchronous counter generated by the silicon compiler.

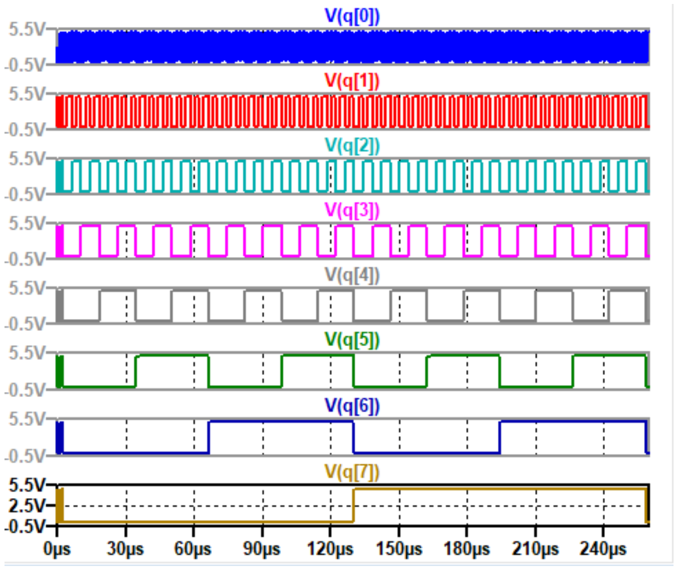

Fig. 8. Transient simulation of the circuit extracted from the layout. The counter was configured in "Up" mode.

V. CONCLUSIONS

This paper presented a library of standard cells suitable for use with QUISC, the Electric's Silicon Compiler. The cells were designed for a standard 0.5-µm technology, under the SCMOS rules. In general, the areas of the cells are larger than those obtained in another library designed for the 0.5-µm process. However, it was verified that the layout compiled using this library is free of errors.

As an example, the layout of an 8-bit up/down synchronous counter was proposed. After the well succeeded silicon compilation, the simulation results showed that the circuit presented the expected behavior.

The expansion of the library, by including new cells, is considered as a future work.

REFERENCES

[1] N. H. E. Weste, K. Eshraghian, Principles of CMOS VLSI Design – A systems Perspective. Addison-Wesley, 1985.

[2] S. Rubin, Compute Aids for VLSI Design. 2nd edition, 1994 [online]. Available in: http://www.rulabinsky.com/cavd/.

[3] D. Johannsen, "Bristle blocks: a silicon compiler". In: 16th Conference on Design Automation, San Diego, USA, 1979.

[4] W. Curtis, "Silicon compilation: the future is now: Silicon compilers liberate the integrated circuit design process". IEEE Potentials, vol. 5, no. 2, p. 27-29, 1986.

[5] M. Kahrs, "Silicon compilation of very high level language". IEEE Transactions on Computer Aided Design, vol. 11, no. 10, pp. 1227-1246, 1992.

[6] C. Nunes, L. Puricelli, L. H. Reinicke et al., "CTC06 standard cell library design". In: 6th Workshop on Circuits and Systems Design – WCAS 2016 (Chip on the Mountains). Belo Horizonte, Brazil, 2016.

[7] E. Kougianos, S. P. Mohanty, P. Patra, "Digital nano-CMOS VLSI design courses in electrical and computer engineering through open-source/free tools". In: 2010 International Symposium on Electronic System Design – ISED, Bhubaneswar, India, 2010.

[8] Static Free Software, www.staticfreesoft.com.

[9] P. P Domingues, R. V. Almeida, E. C. Teixeira, "Uso de software livre em atividades de ensino e pesquisa em microeletrônica" (in portuguese). In: XLIV Congresso Brasileiro de Educação em Engenharia – COBENGE 2016. Natal, Brazil, 2016.

[10] S. Rubin, "Electric User's Manual, version 9.07". Available in: http://www.staticfreesoft.com/jmanual/index.html.

[11] J. Baker, Electric VLSI Design System at CmosEdu [online]. Available in: cmosedu.com.

[12] The MOSIS Service, "MOSIS Scalable CMOS (SCMOS)". Revision 8.00, 2009. Available in: https://www.mosis.com/files/scmos/scmos.pdf

[13] N. H. E. Weste, D. M. Harris, CMOS VLSI Design – A Circuits and Systems Perspective. 4th edition, Addison-Wesley, 2011.