Abstract Page

On the Synthesis-Oriented characteristics of high performance, deep-submicron CMOS VLSI cell libraries.

Abstract

A method to evaluate the “synthesis-oriented” quality of cell libraries, as well as their ability to be efficiently inferred and used during the technology mapping step of DesignCompiler is presented. The definition of both the functional variety of synthesis-oriented advanced cell libraries and the low power physical design methodology for HCMOS6 (0.35 µm minimum channel length) forms part of an investigatory phase, prior to the creation of an ultra-low power cell library based on HCMOS7 (0.25 µm minimum channel length) at low supply voltage.

On the Synthesis-Oriented characteristics of high performance, deep-submicron CMOS VLSI cell libraries.

Roberto Zafalon, Pascal Meier[1]

Central R&D, SGS-THOMSON Microelectronics
20041 Agrate, Italy

1 Introduction

This document describes an evaluation of three different advanced CMOS cell libraries, measuring to what extent they are "synthesis-oriented" and can be inferred and efficiently used during the technology mapping step of DesignWare/DesignCompiler [1].

The three cell libraries are production implementations in two deep-submicron CMOS VLSI technologies (HCMOS6 and HCMOS7.) Two of these libraries are targeted for general purpose applications, the third one is for custom specific designs.

The present work is a small part of a more general aim to establish rules on the definition of both the functional variety of synthesis-oriented advanced cell libraries and the low power physical design methodology for HCMOS6 (0.35 µm minimum channel length,) prior to the creation of a low-power/low-voltage cell library based on HCMOS7 (0.25 µm minimum channel length) at low supply voltage (1 volt and below.)

An understanding of design goals is necessary to establish development criteria for any digital library. Issues such as finding an optimal choice of logic functions to integrate in the library, the high routing density of certain designs, timing/power budgets of special signals (e.g., clock), availability of levels of metal for signal routing, CAD tool limitations, etc., all impact the design of cells which are required to be low power while still be highly performant [3].

2 A method for library benchmarking

The three cell libraries are listed below, along with a brief description of some basic properties. All libraries have been fully laid out in their respective processes.

LIB1: A general purpose standard-cell library, implemented in HCMOS6 technology, 0.35 µm minimum gate length, 5 metal interconnection layers. Total of 392 cells of which 90 are sequential. A large amount of combinational logic is available, with many complementary versions of basic cells. Nominal supply is at 3.3V.

LIB2: A specific high speed addendum to LIB1, quite area-expensive. Implemented in HCMOS6 technology. With a total of 221 cells of which 32 are sequential, this

1. Currently employed as intern at SGS-Thomson, on leave from Carnegie Mellon University, Pittsburgh, PA.

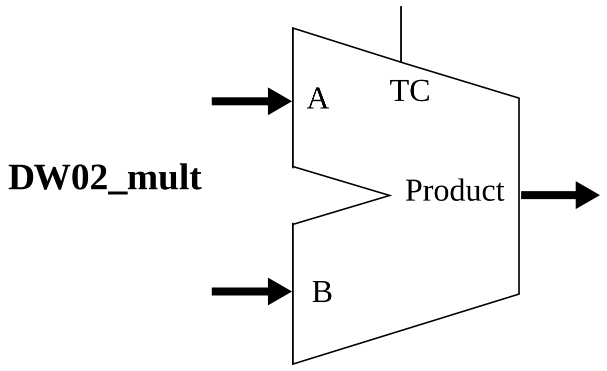
library shows 3 drive strengths (X1, X2, X4) for each available logic function. Nominal supply is at 3.3V.

LIB3: A custom/design-specific high speed cell library, implemented in HCMOS7 technology, 0.25µm minimum gate length, 6 metal interconnection layers. Total of 300 cells of which 20 are sequential. Each function has 4 drive strengths (X1, X2, X3, X4.) Nominal supply is at 2.5V.

The evaluation method for the three cell libraries described above targeted their suitability for the design of fast mathematic units. A 24 bit multiplier from the DesignWare Advanced Math family of the Foundation Library [1] was synthesized by Design Compiler [2]. Specifically, the DW02_mult component was instantiated through a simple piece of Verilog code (see "Fig.1",) with the design modified to force 2's complement implementation. To stress the high speed aspects of the library, the selected architecture is a Booth recoded Wallace tree, with appropriate final addition stage [5][6].

Instantiated components are subject to implementation selection but are not subject to resource sharing or arithmetic optimization. Component instantiations require the declaration of actual parameters and port connections. One can use name-based, explicit mapping or position-based, implicit map-

Fig.1 ***DW02_mult multiplies the operands A by B to produce PRODUCT. The control signal TC determines whether the input and output data is interpreted as unsigned (TC is low) or signed (TC is high) numbers. The Verilog code for instantiating the 24-bit multiplier is reported below.***



```
module multiplier(in1,in2,product);
  parameter wordlength1 = 24,wordlength2 = 24;
  input [wordlength1-1:0] in1;
  input [wordlength2-1:0] in2;
  wire control;
  output [wordlength1+wordlength2-1:0] product;

/* synopsys dc_script_begin
   set_implementation wall U1 */

  // instantiate DW02_mult

  assign control = 1'b1;              // Two's complement
  DW02_mult #(wordlength1,wordlength2) U1(in1,in2,control,product);
endmodule
```

ping, as illustrated in our case using Verilog (see “Fig.1”.)

The choice of Wallace tree architecture was taken because this architecture becomes faster than the classical Carry-Save Array (CSA) as the number of bits increases.

We observed how the computational effort spent in synthesis is dependent on the number of functions implemented in the library. A larger choice of cells in the library usually leads to an easier and faster synthesis process. Should the set of cells be too restricted in function/drive-strengths, experiments with Design Compiler have shown that compilation may not be possible in a reasonable amount of time. Of course, a too-large library can also hinder the process, as the program spends an inordinate amount of time trying out all combinations.

Note: According to the principles published in [4], the following synthesis and timing estimation are based on pre-layout implementation only. Timing analysis has been done with estimated wire loads (conservative wire load models) using the “enclosed” mode when nets belong to sub-designs. The compiled design size has been observed to be, on the average, about 8K nand2-equivalent gates.

3 Design goals and Synthesis result

Once the testbench architecture of a 24-bit 2's complement multiplier had been chosen, a very aggressive timing constrain was set. We tried to achieve an operational frequency well in excess of 100 MHz under worst case conditions, with the intention of stressing the capabilities of the 0.35 µm process. An output load of 0.23pF was set on all the primary outputs.

max “combinational” delay ≤ $\mathbf{t_{H6}}$

Several DesignCompiler runs were performed, varying the target_library configuration in order to determine the potential advantages and drawbacks of each of the libraries presented in section 2 above[1].

A - Target library: LIB1 HCMOS6

1st run: We attempted to map the design using only HCMOS6 LIB1. The design mapped successfully onto LIB1, although the timing goal was not met.

Compilation was performed using “high” timing effort. Mapping used a “low” effort.

Final delay: t=1.59*$\mathbf{t_{H6}}$ Execution time: ~5.0 hrs.

2nd run: We tried remapping the design using a “high” mapping effort. This resulted in some minor improvement in delay.

Final delay: t=1.56*$\mathbf{t_{H6}}$ Execution time: ~6.5 hrs.

B - Target library: LIB1 + LIB2 (HCMOS6)

3rd run: We next attempted to improve the design by adding the high speed library, LIB2. We performed an incremental compile from the 2nd result with a ‘high’ mapping effort.

Final delay: t=1.5*$\mathbf{t_{H6}}$ Execution time: ~2.5 hrs.

1. All the listed experiments have been run on a SPARCstation 20 based on SunOS 4.1, with 128 MB RAM.

4th run: Finally, by fully compiling from scratch the verilog HDL with a ‘high’ mapping effort, we managed to get a substantially better timing, albeit, employing a large execution time. However, the timing goal was not met:

Final delay: $t=1.2*\mathbf{t_{H6}}$ Execution time: ~21.0 hrs.

C - Target library: LIB2 (HCMOS6)

Following the above results, we attempted to vary various parameters as reported hereafter. However, we were not able to lower the delay beyond that achieved by the 4th run.

Synthesis using only LIB2: this never succeeded. Messages from DesignCompiler indicate that the mapping time/memory required would be exponential in the number of inputs. Normally, mapping takes 10 minutes using LIB1. For this experiment the multiplier had not been mapped after 1.5 hours, so we ended the run.

Trying to ‘upsize’ cells by hand: we created a script which manually upsizes cells, and we also attempted to replace cells on the critical path by hand. This was not successful, as we were not able to consider the great number of system level trade-offs between input capacitance and drive strengths. Given the large number of signal paths in a multiplier, many paths will exist which have delays similar to the critical path. Therefore, we conclude that hand tuning is impractical for multipliers of size larger than a few bits.

At this point, we have in the 4th run the best result achievable using the HCMOS6 process. Note that the result obtained using an aggressive synthesis mapping effort did not achieve the targeted specification. Consequently, in this phase we conclude that we cannot achieve the specification using the HCMOS6 libraries available.

D - Target library: LIB3 (HCMOS7)

Moving to HCMOS7, a synthesis run was performed with the Wallace architecture as the target implementation. Noting that the HCMOS7 technology is faster, we decided to further stress the test case. In particular, the timing constraints have been reduced in half with respect to HCMOS6-based experiments, under the same worst case conditions. Our timing goal is,

$$\mathbf{t_{H7} = t_{H6} /2}$$

5th run: Compilation performed using “high” timing effort. Mapping uses a “low” effort. Synthesis was able to produce a design at this more aggressive delay target.

Final delay: $t=\mathbf{t_{H7}}$ (exactly) Execution time: ~2.5 hrs.

By comparison, runs performed using an array-based CSA implementation, under the same conditions give a delay of $t=0.88*\mathbf{t_{H6}}$ ($=1.76*\mathbf{t_{H7}}$). Therefore the above Wallace tree architecture, while requiring a 12% larger estimated area, yields a 45% improvement in speed over the best HCMOS6 result.

Note that both array and Wallace tree designs in HCMOS7 meet the original $\mathbf{t_{H6}}$ goal.

3.1 Technology differentiation

These initial experiments show that we should target implementation of this design in HCMOS7. However, a crucial question is whether the inability to meet the timing goal in HCMOS6 represents limitations of the process, or rather reveals an inherent problem in the libraries.

This question is very important for cost reasons. We prefer to implement designs in HCMOS6, a process which is relatively stable from a manufacturing standpoint. The 0.25 μm HCMOS7 process is not as mature as the 0.35 μm process, meaning that yields are lower and cost higher.

An attempt was made to extrapolate the performance of the LIB3 design in HCMOS6, by performing an "expansion" of the fast HCMOS7 multiplier design (de-shrink from 0.25 μm to 0.35 μm.) Allowing for the higher supply voltage (3.3V instead of 2.5V,) which would mostly balance the slowing effect of longer transistors, this method predicted that this design could be implemented in HCMOS6 and meet timing goals. This estimate led us to reconsider conclusions drawn from the results of the above runs.

4 Analysis of the results

The inability to map a 24 bit multiplier onto HCMOS6 with a delay of $\mathbf{t_{H6}}$ can lead to questions about the libraries' suitability. We were successful in mapping the multiplier onto LIB1 and LIB2. The addition of LIB2 improved the performance of the design. However closer examination of the libraries revealed several potentially limiting characteristics:

a) LIB2 is a "large area" library. In some cases, cell height is twice that of LIB1. Furthermore, cells implementing a certain function take up a larger area than corresponding cells in LIB1. When Design Compiler attempts to upsize the driving strength of a cell to meet timing goals, area has an impact on the way the cell is chosen. This occurs in 2 ways:

- during synthesis and timing optimization, DesignCompiler concurrently optimizes for area and therefore penalizes larger-area cells. I.e., in some cases a large-area cell will not be chosen, even if it improves timing.

- DesignCompiler chooses the wire model based on the design's area of implementation. Therefore, the choice of larger cells will cause the program to move to a larger-area wire model. This means that when high-drive cells are chosen, the estimated wire load increases. This acts as negative feedback and hinders the algorithm from choosing high-drive cells. This effect is particularly problematic in LIB2, where cells are much larger than their corresponding LIB2 versions.

Therefore we conclude that LIB2, though compatible with LIB1, does not complement LIB2 well as a "high-drive" option.

b) The functionality of LIB2 (number of implemented functions) is less than LIB1. Therefore, it is not clear whether it is *possible* to implement a version of our design using only LIB2, much less fulfil the aggressive timing constraints.

c) Two cells which are critical to the design of multipliers are Full Adders and Half Adders. These cells exist in LIB1, but there are no high-drive versions in the LIB2. Furthermore, in LIB1 there is only one high-drive version of the Full Adder and no such version for the Half Adder. Examination of the critical path verified that the Full and Half Adders have a major impact on the delay. Six cells out of 31 on the critical path are of this type, and their delay have a overall contribution of more than $\mathbf{t_{H6}}/2$ to the total path delay.

The HCMOS7 LIB3 appears to be very promising. It may be possible to leverage this library's architectural style by porting it back to the HCMOS6 technology. In this manner, we could take advantage of latter's higher production yields and lower manufacturing cost.

4.1 Prospects for implementation

Results obtained with Design Compiler show that we will be able to create a multiplier to meet our timing target, especially if HCMOS7 is used; certain modifications to existing HCMOS6 libraries may allow us implement the multiplier in this less aggressive technology. Yet we should be careful with the Synopsys estimates of wire load for the Wallace tree; interconnections for this design tend to be much larger than for the CSA-array case. In this comparison, physical layout confirmation of wire loads is important when determining relative performance [5]. However, given that we are able to synthesize a design which meets specifications using the shorter-wire array architecture, we can probably safely conclude that we will be able to implement this design.

Note that the time spent in synthesis is dependent on the number of functions implemented in the library. Currently, it is not known whether LIB2 alone can be used to successfully implement our design. Indications from Design Compiler show that it may not be possible to do so in a reasonable amount of time. Note also that "discontinuities" between libraries are a problem for synthesis. That is, while higher-drive versions of cells are available for a great many functions, their area-of-implementation may be too large; if implementing a function as a high-drive version results in more-than-proportional increase in overall layout area, the cell may never be used by synthesis.

5 Technological trends and future investigation

As previously mentioned, this work should be seen as an investigative step in the design of a low-power/low-voltage library (LPLV) in HCMOS7 (0.25 µm minimum channel length.) We need to understand what have been the limitations in system design using previous generations of cells in order to derive design criteria for the next generation of cell libraries.

Low power design in HCMOS6 and future technologies benefit from the fact that for a given function, moving to 0.35 µm and beyond results in a decrease in implementation size (silicon "footprint"). This causes average net length for this function to decrease, which means that the average switched capacitance decreases.

Furthermore, while a chip's power supply voltage has traditionally been fixed at the system-level (board-level) power supply, this voltage is now seen as a design parameter. Lowering the supply voltage allows for a roughly quadratic saving in power; indeed, supply voltages have seen a steady decrease in recent years to take advantage of this effect.

However, the fact that in succeeding technologies a given function takes up a smaller amount of area, has led not to smaller chips but rather to more functions being implemented in later generations of a family of designs (e.g., see 386, 486, etc.) Therefore, while power per function decreases, newer chips' overall power may well remain constant, or even increase.

This trend points to the need for high-level power analysis tools to be able to analyse power trade-offs at the macroblock level. One must be careful about applying low power techniques to a particular macroblock; it is important to identify whether the targeted macroblock dissipates a substantial amount of a design's power budget. For example, if 80% of the power in a DSP design is dissipated in the control logic, it is useless to reduce power consumption in the ALU. To determine which sections of the design are important, a front-end floorplanner is needed to allow timing/power-budgeting from the very beginning of the HDL synthesis flow [7]. Such a tool allows the exploration of design decisions at a high level. Its analysis must be detailed enough to give accurate estimates, while flexible enough to analyse design trade-offs rapidly.

This type of tool is also relevant to the development of low power libraries. The ultimate validation of a cell library is its ability to implement designs; obviously, this metric of cell library quality can only be established by generating full designs, down to the physical layout level. The use of a floorplanning tool would greatly speed up this process, by quickly generating testcases which would allow cell design philosophies to be rapidly evaluated. For example, an important consideration in low power cell design is the question of how densely individual cells should be laid out. Highly dense layouts mean that capacitances between a cell's elements (i.e., transistors, internal routing) will increase, causing power and delay penalties. However, dense cells have small cell footprints, meaning that system level interconnect between cells would be reduced, lowering capacitance. Such trade-offs can only be considered at the system level.

A related issue is that of cell porosity. Previous technologies have been system-interconnect limited, requiring cell layouts to minimize their use of metal for internal connections. An open question is whether this requirement can be relaxed given: 1) the trend towards more wiring layers, i.e., HCMOS6's five wiring layers and HCMOS7's six layers, versus 2) the larger function count of future chips, which require more system-level interconnect.

6 Conclusion

The above analysis suggests specific and general areas of importance for designing cell libraries which allow maximization of system performance within a given process.

1) Implementations of high-drive versions for Full Adders and Half Adders are necessary when implementing high performance mathematical functions.

2) Area must be minimized in high-drive versions of functions. While a growth in cell footprint for high-drive cells is unavoidable, very large increases in area are unacceptable for synthesis.

3) By suitably leveraging the HCMOS7 cell design's architecture, we can effective exploit the 0.35 μm HCMOS6 technology, allowing creation of high performance designs while saving cost by using the more stable and mature process.

While this analysis has concentrated on synthesis aspects of library design, it is clear that system-level and physical design issues have an important role to play. Future work should be directed at judging a cell library's low power qualities when integrated into a system design methodology.

References

[1] DesignWare Databook, Synopsys v3.5a, Oct 1996

[2] Design Compiler Reference Manual, Synopsys v3.5a, Oct 1996

[3] C. Fisher et al., "Optimization of Standard Cell Libraries for Low Power, High Speed or minimal Area Design", CICC '96, May 1996, S. Diego (CA).

[4] K. Scott, K. Keutzer, "Improving Cell Libraries for Synthesis", CICC '94.

[5] P. Meier, R. Rutenbar, L. Carley, "Exploring Multiplier Architecture and Layout for Low Power", CICC '96, May 1996, S. Diego (CA).

[6] J.J.F. Cavanaugh, Computer Science Series: Digital Computer Arithmetic. New York, McGraw-Hill, 1984.

[7] R. Zafalon, C. Guardiani, M. Casale-Rossi et al, "Forward Power Annotation on Physical Layout Floor-Plan", CICC '96, May 1996, S. Diego (CA).