VERIFICATION OF PORTABLE INTELLECTUAL PROPERTY BLOCKS FOR FPGAS

Mark E. Kelly, and Donald W. Bouldin
Department of Electrical and Computer Engineering
University of Tennessee
Knoxville, TN 37996-2100
mekelly@utk.edu and dbouldin@utk.edu

ABSTRACT

In the world of digital electronics, the use of Intellectual Property (IP) is becoming increasingly more popular in the design of Field-Programmable Gate Arrays (FPGAs). Reuse of IP cuts down on the time-to-market for a product and the overall cost for producing that product. Verified IP blocks can also serve as examples, which speed up the learning curve for a beginning engineer because a learn-by-example format is generally easier to comprehend. It is the point of this paper to present several methods for synthesizing a design, and then place and route the design with several commercially available tool suites and in the process to present scripting methods for automating the process.

1. INTRODUCTION

The goal of the work described here was to establish an environment for the verification of IP blocks using multiple logic synthesis and physical place/route tools. Any IP block that can be verified via multiple paths can then be considered portable. This approach increases the probability that the IP block can be reused in a subsequent design with minimal effort. Furthermore, implementing an IP block via multiple paths provides a means to compare the results. This allows the paths to be compared for such features as logic and routing resources used, their maximum speed, and the time required for execution of the path.

The computer-aided design tools available for this effort included the following:

Logic Synthesis Tools	**Physical Place/Route Tools**
Viewlogic HDL Synthesis	Xilinx Alliance Series
Synopsys FPGA Compiler	Xilinx Xact Series
Synopsys FPGA Express	Altera Max+Plus II Place/Route
Xilinx Synthesis Tools	
Altera Max+Plus II Synthesis	

The principal task involved in establishing this environment consisted of the development of scripts corresponding to the desired tool paths as shown in Figure 1.

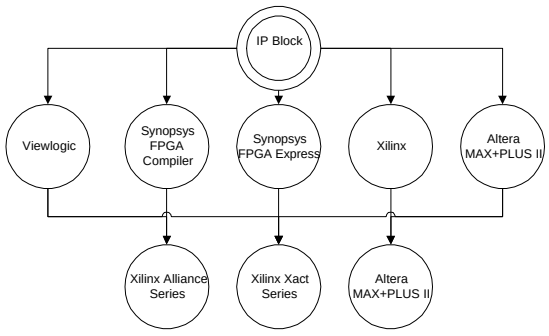

Figure 1. IP Block Flow Diagram

Each script consists of a list of executable commands to the various computer-aided design tools. Instead of invoking the graphical user interfaces of each individual tool and requiring manual selection of the desired functions, all of the options desired are specified in the scripts, which can then be executed in a command line format. For example, to produce a Xilinx FPGA configuration bit file from a Viewlogic net-list in wir format using the Xilinx Alliance tools, a script called wir2bit was written. Its contents include:

```
#!/bin/csh –f
wir2xnf –p 4010EPC84-3 $1
ngdbuild $1.xnf
map $1.ngd
par $1.ncd –w $1_r.ncd
trce $1_r.ncd
bitgen $1_r.ncd –w $1.bit
```

The file was made executable and is invoked by typing wir2bit ipblock, where ipblock is the name of the net-list file in the wir subdirectory. The script then produces routed files with the name ipblock_r and the appropriate extension. In this case, the wir2xnf command converts from the net-list into the Xilinx Net-list Format (xnf) and targets a particular Xilinx part. The remaining steps build the NeoCAD Database (ngd), perform a mapping (map) into combinational logic, place and route (par) the design, trace (trce) the delay paths and generate (bitgen) the configuration bit file to be downloaded to the part.

2. VIEWLOGIC SYNTHESIS

In this section, an IP block is synthesized and simulated for Xilinx and Altera technologies using the Viewlogic suite of tools. The circuit to be simulated is the classic traffic light signal problem commonly presented in digital design courses.

The following description of a traffic light controller represents a relatively complex control function: "A busy highway is intersected by a little used farmroad, as shown in Figure 2.

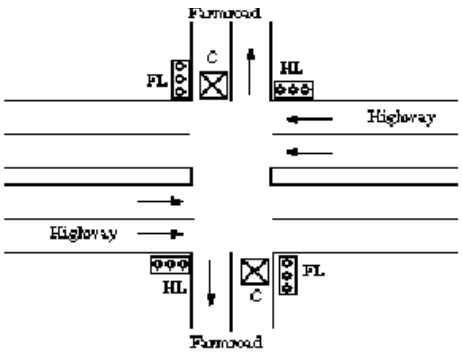

Figure 2. Traffic Light Controller Diagram.

Detectors are placed along the farmroad to raise the signal as long as a vehicle is waiting to cross the highway. The traffic light controller should operate as follows. As long as no vehicle is detected on the farmroad, the lights should remain green in the highway direction. If a vehicle is detected on the farmroad, the highway lights should change from green to yellow to red, allowing the farmroad lights to become green. The farmroad lights stay green only as long as a vehicle is detected on the farmroad and never longer than a set interval to allow the traffic to flow along the highway. If these conditions are met, the farmroad lights change from green to yellow to red, allowing the highway lights to return to green. Even if vehicles are waiting to cross the highway, the highway should remain green for a set interval.

In order to simulate this controller, the VHDL file was synthesized. A csh script was designed to automate the entire process. In this script, the following steps are performed:

- All necessary files are copied to a new directory
- The VHDL file is synthesized using Viewlogic vhdldes
- Symbol is generated for the circuit using viewgen
- Simulation file is created using vsm
- Command file is created by the script
- File is simulated in viewsim with the command file

More exactly the contents of the script include:

```
mkdir hw3
cd hw3
cp /usr/cad/course/viewdraw.ini .
mkdir behv
cp /usr/cad/course/decoder551.vhd behv
view_tools
vhdldes -tech=xc4000 tlc_enc1
viewgen tlc_enc1 -makesym
viewdraw tlc_enc11 &
vsm decoder551 &
viewsim -vsm tlc_enc1.vsm -cmd tlc_enc1.cmd &
```

The Viewlogic software produces the following output:

```
        ********************************************
                    Gate Usage Summary
        ********************************************
------------------------------------------------------------
--
Cell  Name        Cell  Count          FMAPS          HMAPS
REGISTERS
------------------------------------------------------------
--
XC4000:AND2               11           2.75           0.00
0.00
XC4000:FDCE                3           0.00           0.00
3.00
XC4000:INV                11           0.00           0.00
0.00
XC4000:NAND2              12           3.00           0.00
0.00
XC4000:NOR3                2           1.00           0.00
0.00
XC4000:OR2                 3           0.75           0.00
0.00
XC4000:OR3                 8           4.00           0.00
0.00
XC4000:OR4                 3           2.25           0.00
0.00
------------------------------------------------------------
--
Total :                   53          13.75           0.00
3.00
       ********************************************
                    Netlist Statistics
       ********************************************
Maximum level of gates = 8    Total number of nets   = 60
Maximum pins per net    = 11    Average pins per net    =
3.17
```

The last step of the script produces a timing diagram which is used to determine if the appropriate results are obtained. Figure 3 shows a typical output timing diagram for a given IP block. Complete results of this process can be seen on http://microsys6.engr.utk.edu/~mekelly/ee501/ee501.htm. [4]

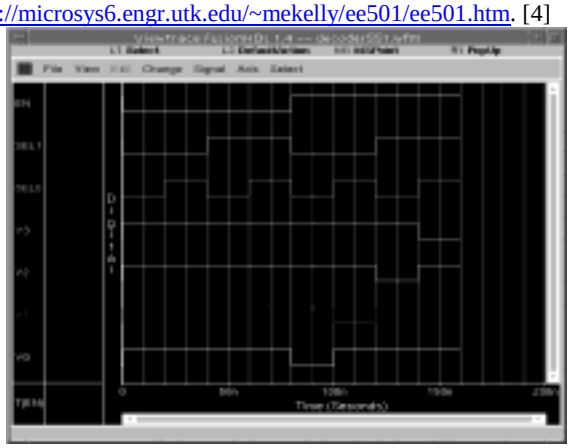

Figure 3. Viewlogic Viewsim Timing Diagram.

3. ALTERA SYNTHESIS

The Max+Plus II development software is a fully integrated programmable logic design environment. This easy-to-use tool supports the Altera® FLEX® and MAX programmable device families and works in both PC and UNIX environments. The Max+Plus II software offers flexibility and performance and allows for seamless integration with industry-standard design entry, synthesis, and verification tools. [2]

Figure 4. Altera MAX+PLUS II GUI Environment.

The default usage for the MAX+PLUS II environment is graphical user interface shown in Figure 4, but the Compiler, Timing Analyzer, and Simulator can be operated from the command prompt. To run MAX+PLUS II from a command prompt, type:

maxplus2 -h | -v | { <batch option(s)> [<I/O option(s)>] <project name> }

Multiple batch and I/O options can be used for a single project; multiple projects can be processed with the same command line. The <project name> indicates the end of the options for that project. In order to automate the synthesis of the IP blocks, a script was developed based on the command line operation of MAX+PLUS II. Its contents include:

mkdir hw6
cd hw6
cp ~cad/course/fa.tdf .
cp ~mekelly/ee501/fa.acf .
maxplus2 -c fa.tdf
maxplus2 -s -vec "fa.vec" -tbl "fa" fa.tdf

Using the above script format, the traffic light controller was synthesized with MAX+PLUS II giving the following results for estimated device usage:

```
Device: EPF10K20RC240-4

Total dedicated input pins used:       1/6      ( 16%)
Total I/O pins used:                  13/183    (  7%)
Total logic cells used:               20/1152   (  1%)
Total embedded cells used:             0/48     (  0%)
Total EABs used:                       0/6      (  0%)
Average fan-in:                       3.35/4    ( 83%)
Total fan-in:                        67/4608    (  1%)
```

4. XILINX PLACE AND ROUTE

This section uses the traffic light controller example to perform placement and routing using Xilinx's PPR software. The design is placed and routed with both xmake and m1make. M1make is the newer tool, but xmake is also used to see if any performance is gained with the newer toolset. One net is tagged as a critical net and differences (if any) in the delay times in the post-layout simulation are examined.

There are many small programs that must be run to accomplish this task, but Xilinx has automated most of the process with a program called xmake which calls all of these programs in the proper sequence and with the correct options. The general format for placing and routing a design is as follows:

xmake -g -p 4005PC84-5 tlc_pad

This will generate a make file to the design, but since the default settings for the Xilinx software cause it to ignore the net weighting command some changes have to be made to the file.

ppr tlc_pad.xtf Path_timing=false parttype=4005PC84-5
xdelay -w -x -o xdelay.out tlc_pad.lca

In order to verifiy that the design works correctly, end result was back annotated in viewlogic and resimulated for the actual timing delays.

lca2xnf -g tlc_pad.lca tlc_back.xnf
xnfba tlc_pad.xff tlc_back.xnf
xnf2wir xnfba.xnf wir/tlc_back
vsm tlc_back
cp tlc_back.cmd tlc_pad.cmd
viewsim -vsm tlc_back -cmd tlc_pad.cmd &

From the back annotation, the longest delay achieved in the design according to the xmake report was 37.3 ns. [5] In order to gauge this process, the newer M1make toolset is used to place and route the design. The M1make toolset consists of using the previously described wir2bit script to automate the process.

xilinx_tools
wir2bit tlc_pad

This produces the layout shown in Figure 5.

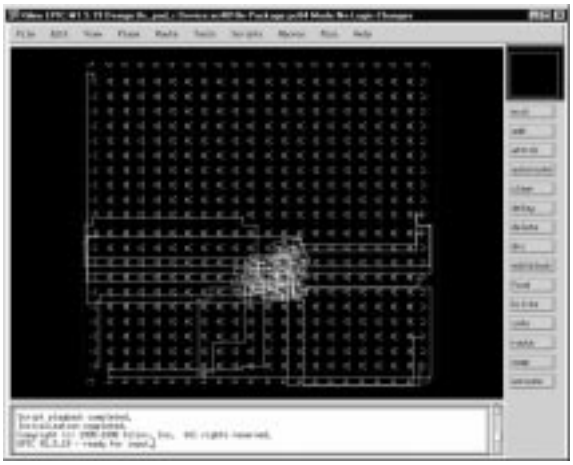

Figure 5. Xilinx M1make layout

Using the results of both tools allow the following timing comparison to be made.

Output	M1make Delay	Xmake Delay
PAD-STATEOUT	37 ns	26 ns
PAD-LTIME	42 ns	40 ns
PAD-STIME	47 ns	45 ns
PAD-FL	37 ns	38 ns
PAD-HL	37 ns	34 ns

5. ALTERA PLACE AND ROUTE

This section is a continuation of the placement and routing from the previous Xilinx section. It also uses the traffic light controller example to perform placement and routing using Altera's Max+Plus II software.

The goal of this section is to automate the place and route process for the Altera Max+Plus II software. In order to get an idea of the required steps for automation, a step by step process will be presented. The following steps are executed from the command line to achieve the desired results.

altera_tools
mkdir hw8
cd hw8
cp /usr/cad/course/viewdraw_flex10k.ini viewdraw.ini
mkdir behv
cp /usr/cad/course/tlc_enc1.vhd behv
vhdldes -tech=flex10k tlc_enc1
viewgen tlc_enc1 -makesym
viewdraw tlc_enc1 &

6. SYNOPSYS SYNTHESIS FOR A XILINX FPGA

This section is intend to present the synthesis process using the Synopsys Design Analyzer and place and route the synthesized design using Xilinx tools.

This section of the project is slightly different from the previous sections. Developing a shell script for this is not applicable because all commands executed are performed inside the Design Analyzer environment, but fortunately, there is a scripting language available in Design Analyzer that will allow us to automate the process. The following is an excerpt from the script shows the format of the macro language:

```
analyze -format vhdl TOP + ".vhd"
elaborate TOP
current_design TOP
set_port_is_pad "*"
set_pad_type -slewrate HIGH all_outputs()
.
.
.
replace_fpga
set_attribute TOP "part" -type string PART
set_attribute find(design,"*") "xnfout_use_blknames" -type boolean FALSE
write -format xnf -hierarchy -output TOP + ".sxnf"
exit
```

Executing the script, produces the following output:

```
    Partitioned Design Utilization Using Part 4005PC84-5

-----------------------------------------------------------------------
                                  No. Used   Max Available  %Used
-----------------------------------------------------------------------
      Occupied CLBs                  4           196          2%
      Bonded I/O Pins               11           61          18%
      F and G Function Generators (*)  8          392          2%
      H  Function Generators         0           196          0%
      CLB Flip Flops                 8           392          2%
      IOB Input Flip Flops           0           112          0%
      IOB Output Flip Flops          0           112          0%
      3-State Buffers                0           448          0%
      3-State Half Longlines         0           56           0%
      Edge Decode Inputs             0           168          0%
      Edge Decode Half Longlines     0           32           0%
      CLB Fast Carry Logic           4           196          2%
```

7. SYNOPSYS FPGA EXPRESS AND M1MAKE

This section is intended to present the synthesis process using the Synopsys FPGA Express package and place and route the synthesized design using Xilinx tools.

As with the FPGA Compiler, FPGA Express uses its own macro language to automate the design process. FPGA Scripting Tool (FST) implements a Tcl-based command-line interface for FPGA Express. The data models defined by the FPGA Express GUI are preserved in command line form by FST.

The script designed for this stage includes:

```
set proj hw3
set top tutor
set target XC4000E
set chip tutor
set export_dir export_dir
set device 4013EPQ208
set speed –1
.
.
.
analyze_file –progress
create_chip -progress -target $target -device $device -speed $speed -frequency 50 -name
```

The script produced the following results:

```
Design Summary
--------------
   Number of errors:        0
   Number of warnings:      1
   Number of CLBs:              8 out of   400    2%
      CLB Flip Flops:       3
      4 input LUTs:        15
      3 input LUTs:         4
   Number of bonded IOBs:      14 out of    61   22%
   Number of clock IOB pads:    1 out of     8   12%
   Number of primary CLKs:      1 out of     4   25%
Total equivalent gate count for design: 126
Additional JTAG gate count for IOBs:    672
```

8. XILINX WEBFITTER

The WebFITTER is a free web-based, CPLD design fitting software tool that allows system designers to test their designs using the XC9500 series of CPLDs, on the latest version of Xilinx software and get fitting results in minutes. In order to get a baseline for how webfitter works, the tlc_enc1.vhd file used in previous sections will again be used for webfitter. The performance of the design was very impressive which can be seen in the following excerpt from the design report file.

```
Pad to Pad (tPD):          4.0ns (1 macrocell levels)
Pad 'reset_b' to Pad 'start'
Minimum Clock Period: 5.0ns
Maximum Internal Clock Speed: 200.0Mhz (Limited by Clock
Pulse Width)
```

8. CONCLUSIONS

The traffic light example was used on to test all the paths shown in Figure 1. This was a relatively small IP block and using it in all of these paths allows us to truly call this block portable and also allows us to make a comparison between the toolsets.

For the synthesis, we were able to take several different routes and make some comparisons. For generating FPGAs, the Altera MAX+PLUS II and Synopsys FPGA Express are by far the easiest to use of the toolsets and achieve the best and fastest results. The other tools are not specifically designed for FPGAs which makes their use a little more cumbersome, but they are all very useable.

For the place and route we needed to output to the same chip to make fair comparisons and the two Xilinx place utilities are the only ones use that will output to the same chip. The table presented earlier shows that the achieved speeds are relatively the same, and the area used by this example is also very close for both tools.

Output	M1make Delay	Xmake Delay
PAD-STATEOUT	37 ns	26 ns
PAD-LTIME	42 ns	40 ns
PAD-STIME	47 ns	45 ns
PAD-FL	37 ns	38 ns
PAD-HL	37 ns	34 ns

REFERENCES

[1] *Electrical and Computer Engineering Program*, University of Tennessee, http://www.ece.utk.edu.

[2] "MAX+PLUS II - Programmable Logic Development System, Getting Started", 1999, Altera Corporation, pp 277-280

[3] *EE 551 Course Web Site*, University of Tennessee, http://microsys6.engr.utk.edu/ece/bouldin_courses/551/overview.html.

[4] *EE 552 Course Web Site*, University of Tennessee, http://microsys6.engr.utk.edu/ece/bouldin_courses/552/overview.html.

[5] Mark Kelly, *Verification of Intellectual Property Blocks for FPGAs and ASICs*, EE501 Project, University of Tennessee, http://microsys6.engr.utk.edu/~mekelly/ee501/ee501.htm