TIME-TO-MARKET SILICON

By: Chad Nikoletich
Senior Engineer
JTA Research, Inc.

We've heard it all before. ASIC design cycles are shortening due to time-to-market and product life-cycle demands. Add to the mix shrinking geometries, increased gate counts, higher clock frequencies and reduced power requirements, and engineers are constantly searching for solutions to these formidable challenges. Every ASIC designer is searching for the holy grail of product design: the identification of proven design methodologies that support these challenges. Therefore, the intent of this article is to demonstrate an innovative methodology that uses the Xilinx Virtex FPGA technology for ASIC emulation. By the end of this article, engineers should have the context to apply the methodology across a wide spectrum of applications.

In the fall of 1998, a southern California company that we'll call XYZ was tasked with the development of a million gate ASIC, yet their standard development cycle couldn't support the project's stringent schedule requirements. After much discussion, XYZ determined that significant schedule improvements could occur if the embedded software developers were engaged earlier into the design process. In order to do this, XYZ sought to emulate the ASIC, through the use of multiple Field Programmable Gate Arrays (FPGAs), allowing the software team an opportunity to test and integrate the embedded software on real hardware prior to receiving first silicon from the foundry.

XYZ engineers started by partitioning early RTL code into multiple design blocks. The ASIC was originally targeted for individual Altera FLEX 10KE parts, however, logic gate count estimates and embedded RAM requirements exceeded the capacity of the technology.

Meanwhile, Xilinx was developing a new FPGA family called Virtex. In the fall of 1998, Xilinx knew it had a promising yet unproven technology. When XYZ's engineers discussed their needs with the local sales team, it was apparent that Xilinx was staring at exactly what it needed -- a large, real-world design by which to target the new Virtex architecture and design software.

Xilinx made their pitch, describing a new technology that appeared to support XYZ's requirements: up to one million system gates, over 16K bytes of embedded dual port RAM, support for multiple IO standards, four Delay Lock Loops (DLLs), and design tools that supported both VHDL and Verilog. Based on these FPGA qualities, XYZ took a chance on the yet unproven Virtex family.

Since the majority of the project staff was already engaged in designing the ASIC, XYZ required an outsource solution to perform the ASIC translation to Virtex. Xilinx recommended its Southern California-based Xperts partner, JTA Research, Inc. to perform the task. JTA, XYZ, and Xilinx were now on a path that would exercise the breadth of the new Virtex architecture.

HOW WILL IT FIT?

JTA initially embarked on fully understanding the magnitude of the design. The ASIC contained four internal special processing nodes, three data interfaces (one of which was PCI), one external memory interface, and a timing and control block. The special processing nodes and interfaces were linked together to the external memory through the timing and control block. Each special processing node (SPN) contained an embedded 64-bit sequencer along with input/output FIFOs, embedded RAM, and unique data manipulation logic.

JTA partitioned the design and mapped the ASIC architecture to Virtex. For example, the ASIC used a separate data bus for each SPN and each data interface (IF). In order to conserve IO pins, these busses were converted to a single, tri-state bus, which was shared among the nodes and interfaces. The embedded RAM was mapped to Virtex RAM while the IF and SPN functions were combined based on Virtex gate capacity.

The final partitioning is shown in Figure 1. First, each SPN was partitioned out to reside on a single Virtex FPGA. Second, the interfaces were likewise mapped to individual Virtex parts. Lastly, the timing and control block was combined with one of interface units.

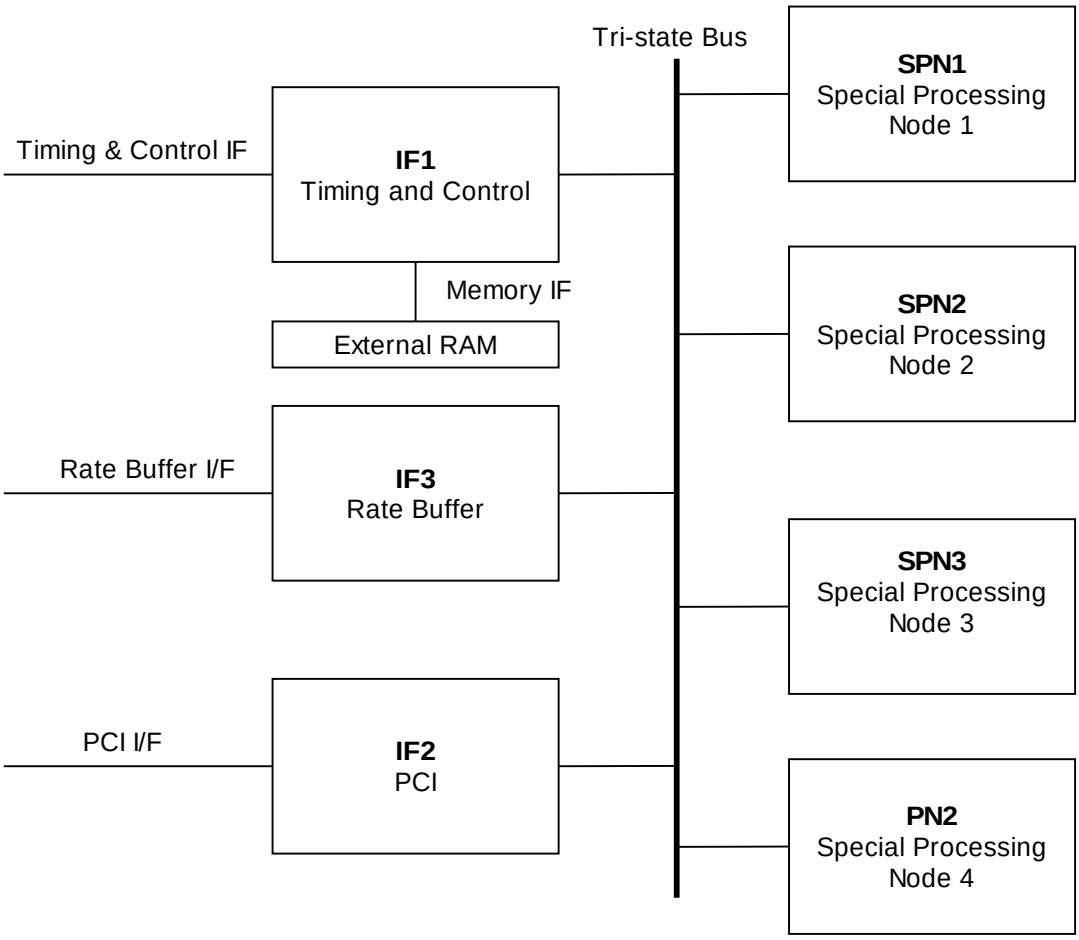

Figure 1: Final Partitioning

TURN THE DESIGN CRANK

JTA is well versed in ASIC design and methodology flow. By combining this design experience with its extensive use of FPGAs, JTA took little time coming up to speed. XYZ supplied JTA with VHDL RTL and supporting testbenches and expected FPGAs in return. To deliver working FPGAs, JTA needed to answer the following two questions: 1) What steps are required to convert the ASIC design to Virtex FPGAs and 2) How does one validate the correctness of such a conversion?

JTA developed a methodology based on prior design knowledge to tackle the ASIC to FPGA conversion. Design activities provided by JTA engineers included five areas: 1) replacing RAM blocks using CORE Generator (Xilinx's tool used to generate user defined BlockRAM and other IP), 2) DLL insertion, 3) modifying the internal ASIC data bus, 4) mapping ASIC IO pads to Virtex IOs, and 5) creating RTL source for custom instantiated components.

JTA's RTL changes were to be verified through simulation of each Virtex device. In addition, a system level testbench would be created to check all the Virtex devices together. Once completed, this methodology would validate three things: 1) that the original ASIC had been partitioned correctly, 2) that the connections between all Virtex devices were correct, and 3) the interoperability of the integrated Virtex devices.

The JTA ASIC design flow was modified to support the ASIC conversion to Virtex FPGAs. What emerged was a hybrid methodology that combined the best of two worlds: standard ASIC and FPGA flows. Figure 2 shows the combination of Unix and PC tools used to perform the ASIC conversion to Virtex.

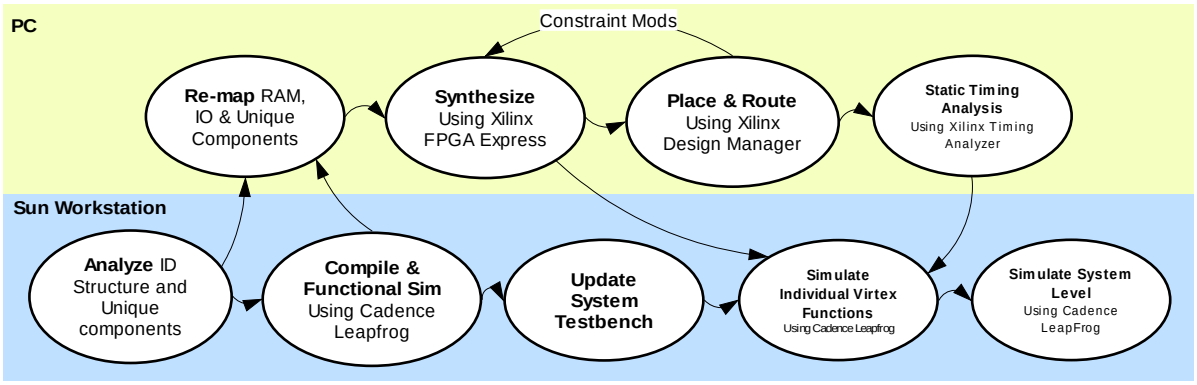

Figure 2: Tool/Conversion Flow

The flow initiated with each ASIC entity's code. It was apparent at the onset that some analysis of the delivered source code was required, identifying such things as size, complexity missing source, and any unique or library component source requirements. Multiple simultaneous deliveries of various ASIC modules were commonplace; and a manual inspection of the source threatened to take weeks. It was clear that some form of automation was required to accelerate task execution while reducing the effort of this portion of the design flow.

JTA eliminated the manual checking bottleneck by creating a VHDL source analyzer, written in TCL, to perform the analysis functions. The tool did not require compilation and only the VHDL top module file needed to be identified. It identified all components of the module including hierarchy. In addition, all missing library components were identified automatically. As a secondary benefit, the tool used hierarchical information to create scripts which were used to compile the source for Cadence's VHDL simulator, LeapFrog. Figure 3 illustrates a sample output produced by this custom tool.

```
-------------------------------------------------------
.top
.top.pn1
.top.pn1.ram1wrap
.top.pn1.ram1wrap.ram64x1k    **library component**
.top.pn1.ram2wrap
.top.pn1.ram1wrap.ram32x256    **library component**
.top.pn1.sequnit
.top.pn1.sequnit.dec
.top.pn1.sequnit.wr
.top.pn1.sequnit.rd
.top.pn2
.top.pn2.ram1wrap
.top.pn2.ram1wrap.ram64x1k    **library component**
...
...
-------------------------------------------------------
Number of HDL source lines: 53682
Number of Components: 131
Number of Components without an associated entity: 46
Number of Instances:  2822
Number of Instances without an associated entity: 1990
Libraries used: ieee
                work
                gtech
```

Figure 3. VHDL Source Analyzer output

Once this report was generated, JTA engineers could create RTL concurrently for the unique library components while remapping the RAM and IO components. VHDL was then compiled and functionally verified using LeapFrog and XYZ's testbench.

RTL was synthesized using Xilinx's Foundation Series software, FPGA Express. The synthesized source was simulated and the results compared with the outputs of XYZ's RTL simulations. In addition, the system level testbench was incrementally developed to verify the concerted actions of multiple FPGAs.

After synthesis, it was on to place-and-route, using Xilinx's Design Manager software. Upon completion, a post layout netlist was created and the place & route results were analyzed using Design Manager's Timing Analyzer tool. The post layout netlist was functionally simulated using Leapfrog and compared with all prior simulation results.

The hybrid flow provided two distinct benefits: 1)Overlapping conversion activities, allowing fast turnaround upon delivery of RTL and 2) Verifiable snapshots using simulation throughout the design flow, thus validating the correctness of the conversion.

It's Size and Speed that Matters

In the end, the ASIC's million gates were converted to eight Xilinx Virtex FPGAs (XCV1000-5 parts in BG560 packages) FPGAs. Figure 4 summarizes the design attributes for each device.

Module	Original Gate Count	Virtex Gate Count	CLB Capacity (%)	Block RAM Capacity	IO Capacity (%)	Speed-5 (MHz)
SPN1	45K	94K	41	0	92	30.4
SPN2A	389K*	262K	99	12	98	28.8
SPN2B		548K	90	0	70	27.8
SPN3	157K	260K	98	12	45	27
SPN4	71K	112K	52	0	45	25.7
IF1	74K	215K	44	25	61	25.3
IF2	146K	276K	66	25	73	25.6
IF3	92K	301K	31	43	54	30.6

Figure 4. ASIC Stats

* original SPN2 gate count

The final partitioning required deviation from the original plan in two areas. First, SPN2 was divided into two devices SPN2A and SPN2B because the original module would not fit in a

single XCV1000 Virtex part. Second, the Timing and Control module was combined with SPN1 instead of IF1, primarily because SPN1 had achieved the highest speed after place & route of any ASIC module and could afford the speed and area impact.

Originally, each converted ASIC module was targeted to run at 33MHz. After several synthesis and place-and-route cycles on each module, the Timing Analyzer tool helped determine that the maximum, worst case speed was 25MHz (the worst case minimum acceptable speed).

NOT ACCORDING TO PLAN

As in any new technology, minor technical hurdles needed to be overcome during first deployment. Sometimes, kinks need to be worked out and other times, the best approach is to either rethink a process or develop a workaround until a fix is in place. Despite hurdles such as preliminary tools, a modified methodology and a new technology, JTA intended to meet its commitments. In order to do so, JTA addressed problems in three main areas: tool related, RTL source code problems, and size/speed issues. Tool problems were directly attributed to those typically found in beta code. JTA provided feedback to Xilix's R&D team to help them transform prereleased code into robust, released code. The majority of RTL source code problems were related to delivery of preliminary RTL in order to trial build each SPN and IF function. Size/speed estimates were validated with trial place & routes.

To mitigate further risk, JTA verified every step of the design flow. Tools resided on both Unix and PC platforms, requiring data to be passed between the platforms using JTA's heterogeneous network. A trial build was performed and passed through each step of the design flow, revealing several minor data problems that were easily corrected. An example would include simply identifying the proper output of the place-and-route tool such that the postlayout netlist could be simulated with Leapfrog. Another would be the need to compile the Xilinx SIMPRIM and UNISIM libraries in order to simulate the Xilinx Design Manger tools output.

The first major problem occurred when attempting to insert the DLL module into one of the PNs. FPGA Express produced a constraints file that was incompatible with the Design Manager software. Further exploration into the problem revealed that the new DLL feature was only available within the Virtex architecture, and subsequently, the synthesis tool had not yet implemented this required feature. Again, Xilinx's R&D team responded to our request and the next release of FPGA Express accounted for the DLLs, allowing us to continue moving forward.

It wasn’t long before we hit the next problem: a timing problem surfaced in all the SPN sequencers. The problem was attributed to the RTL that produced over 50 layers of CLB logic between flip-flops. Since Virtex performance is heavily predicated on the layers of CLBs between flip-flops, XYZ responded to JTA’s request and delivered optimized source code to reduce the logic to less than 20 layers, ultimately meeting the required timing constraints, but more importantly, illustrating a side benefit of ASIC emulation. By running into the upper limit of the emulation implementation technology, potential ASIC timing problems due to non-optimal RTL were identified and corrected very early in the process.

Two of the modules delivered by XYZ reported port problems during synthesis by FPGA Express, yet Cadence's LeapFrog reported no error during simulation. Finding the source of the discrepancy was proving to be difficult, so JTA Engineers downloaded an evaluation copy of Synplify from the Synplicity website. Using Synplify, JTA engineers found two problems: a port mismatch and a multiple driven net. Using the Synplify RTL Viewer, JTA screen captured the offending net and emailed the picture to XYZ. Again, the conversion flow was robust enough to identify long-term RTL issues that XYZ used to rectify.

By far the stickiest problems occurred after synthesizing the PN2 module. Design Manager could not place-and-route this module because it used 137% of the Virtex CLB and 212% of the BlockRAM resources. Using Synplify for synthesis did reduce the CLB usage by fifty percent, but the resulting 119% was still too much to implement. Therefore, the only option left was to split PN2 into two XCV1000 parts – a challenging task considering that the data paths were 64 bits wide. In the end, PN2A was CLB and IO bound with 99% and 98% usage respectively. PN2B turned out to use 90% of the CLBs since SelectRAM (CLB RAM) was utilized because PN2B required 200% of BlockRAM availability. The problems did not end there. The split uncovered a bug in the placer tool, which would not iterate enough times and increase the router’s effort to the point where high-density designs would not complete. Xilinx responded with an update of the Foundation Series software that resolved the place-and-route problem of PN2A and PN2B. The troublesome PN2 did uncover one gem for the JTA engineers, though. High-density designs requiring a high percentage of Virtex resources would route 100%!

Overall, the JTA engineers were impressed with Virtex's capability to aid in ASIC emulation/rapid prototyping. As the Virtex design tools mature, the future only looks brighter for its use as "time-to-market silicon.”

Chad Nikoletich

Chad Nikoletich has a BSEE from California State University, Long Beach. As a 21 year industry veteran, Chad has performed numerous ASIC and FPGA designs for such companies as JPL, Hughes Aircraft Company, and General Dynamics. An avid digital video enthusiast, Chad enjoys spending his spare time in his home video studio. He can be reached at cniko@jta.com.