A Hybrid ASIC and FPGA Architecture

Paul S. Zuchowski, Christopher B. Reynolds, Richard J. Grupp[1]
Shelly G. Davis[2], Brendan Cremen[3], Bill Troxel[4]

1. IBM Microelectronics Division
Essex Junction, Vermont 05452, USA
2. Xilinx Corporation
San Jose, California, USA
3. Xilinx Corporation
Dublin, Ireland
4. Xilinx Corporation
Boulder, Colorado, USA

Introduction

This paper introduces a new hybrid ASIC/FPGA chip architecture that is being developed in collaboration between IBM and Xilinx, and highlights some of the design challenges this offers for designers and CAD developers. We will review recent data from both the ASIC and FPGA industries, including technology features, and trends in usage and costs. This background data indicates that there are advantages to using standard ASICs and FPGAs for many applications, but technical and financial considerations are increasingly driving the need for a hybrid ASIC/FPGA architecture at specific volume tiers and technology nodes.

As we describe the hybrid chip architecture we will point out evolving tool and methodology issues that will need to be addressed to enable customers to effectively design hybrid ASIC/FPGAs. The discussion will highlight specific automation issues in the areas of logic partitioning, logic simulation, verification, timing, layout and test.

Background

Design teams today must choose to implement logic either in ASIC or FPGA technology. Each of these offerings has distinct advantages: performance and density for ASICs, vs. Turn-Around-Time (TAT) and flexibility for FPGAs. Figure 1 illustrates the average gate delay for both ASIC and FPGA products, as a function of technology node [1-10]. Note the significant difference in scale for the two technologies. Figure 2 shows the average power per gate for each circuit technology. These two figures show the substantial differences in performance and power between the typical ASIC and FPGA approaches. Similarly, figure 3 shows the typical density tradeoff that must be made when choosing to implement a design in an FPGA versus an ASIC technology.

Applications Emerge for Hybrid Devices

As can be seen in figures 1 through 3, implementation using an ASIC approach typically yields a faster, smaller, and lower power design than implementation in FPGA technology. The growing requirements in the marketplace for design flexibility however, are driving the need for hybrid ASIC/FPGA devices. The potential to change hardware configuration in real time, to support multiple design options with a single mask set, and to prolong a product's usable life, all compel designers to look for a blending of high density ASIC circuits along with the inherent FPGA circuit flexibility.

The ability to create a “base design” and then reuse the base with minimal changes for subsequent devices helps reduce design time and encourages standardization. Since many consumer and office products are offered with a range of low to high-end options, this base design concept can be effectively used - with features added to each successive model. Printers, fax machines, PC's and digital imaging equipment are examples where this concept can be useful.

DSP applications are also well suited to FPGA because of the FPGAs fast multiply and accumulate (MAC) processing capability. When building a DSP system, the design can take advantage of parallel structures and arithmetic algorithms to minimize resources and exceed performance of single or multiple purpose DSP devices [11]. DSP designers using both ASIC and FPGA within the same design can optimize a system for performance beyond the capabilities of either separate circuit technology.

Other applications that lend themselves to the hybrid ASIC/FPGA approach are designs that support multiple standards such as USB, FireWire and CameraLink, in a single device. Similarly, designs that are finalized, with the exception of any undefined features or emerging standards (PCI Express, for example), are excellent candidates for this technology. Without the benefit of programmable

0-7803-7607-2/02/$17.00 ©2002 IEEE

logic, the designer must decide between taping-out the chip knowing that the PCI logic has a high probability for change, or waiting until the design requirements are firm – potentially impacting the end product's schedule. With both programmable logic and ASIC working together on a single device, some situations like these can be accommodated. Other similar issues like differing geographic or I/O standards could also be incorporated within the FPGA cores, without requiring mask and fabrication updates for each change.

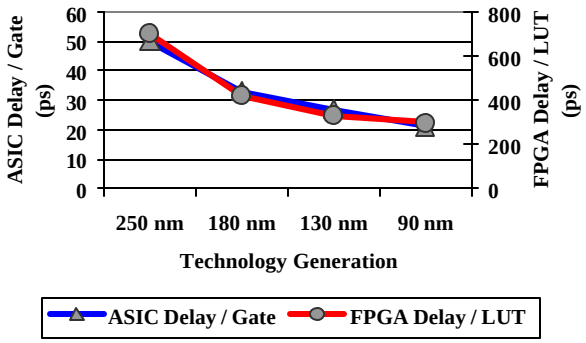

Fig. 1. Performance by Technology Generation

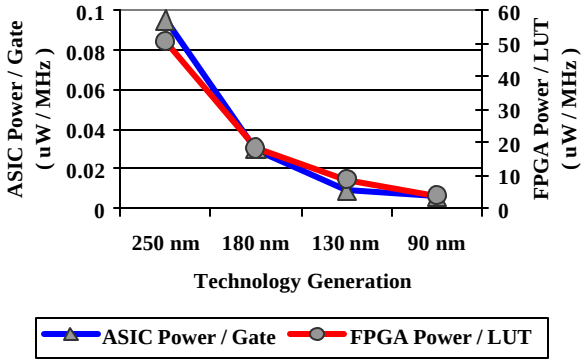

Fig. 2. Power by Technology Generation

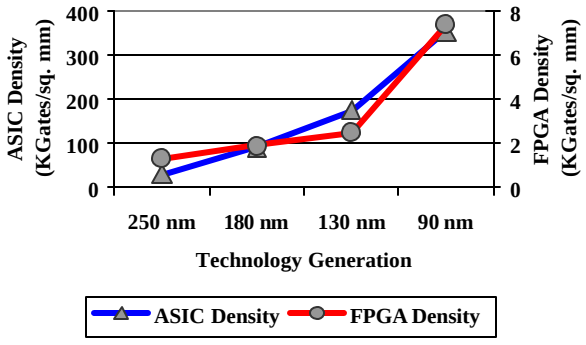

Fig. 3. Density by Technology Generation

Economics Play a Role in Using Hybrid Devices

While technical applications are emerging for the hybrid architecture, it is unlikely that design teams would utilize this new capability unless it is also economically viable. We will now explore the economics behind this new architecture.

To realize the performance and density advantages of an ASIC, design teams must accept higher NREs and longer TATs than FPGAs. Unlike off-the-shelf FPGAs, each ASIC design requires a custom set of masks for silicon fabrication. The custom mask set allows circuitry and interconnections to be tailored to the requirements of each unique application - yielding high performance and density. However, the cost of the mask sets is rapidly increasing (nearly doubling with each successive technology node). As a result, mask costs are becoming a significant portion of the per-die cost in many cases.

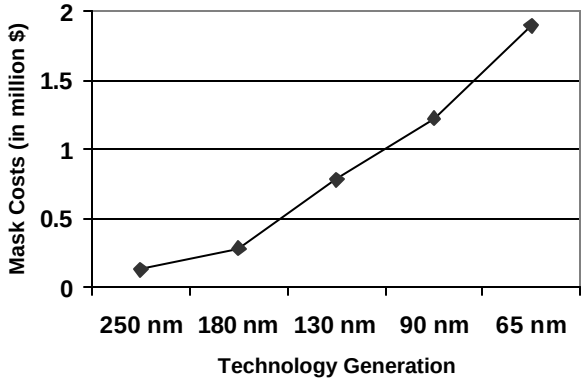

Fig. 4. Mask Cost vs. Technology Generation

Figure 4 shows the estimated trend in ASIC mask set costs from the 250 nm technology node through the 65 nm technology node [12]. The complexities of sub-wavelength lithography are causing mask costs to increase significantly with each technology generation. This cost escalation has a direct impact on the price-competitiveness of ASICs and other designs requiring unique mask sets per design pass. This can be a major issue for low-volume applications.

For example, consider the case where a mask set costs $1M. For applications where only 1,000 chips are required, each chip will cost well over $1000, since the mask cost (plus many other expenses) must be amortized over the volume of chips sold. As the volume requirements for this same ASIC rise, the effective cost of each die decreases.

Conversely, FPGAs are standard products, where the mask charges for a small number of design passes are amortized over a large number of customers and chips, so the mask cost per chip sold is minimal. As a result, for each technology node there is a volume threshold, below which it's more cost-effective to buy an FPGA chip vs. a smaller ASIC chip.

TAT is another primary economic driver, having a direct impact on time-to-market for many applications. The time required for ASIC layout and fabrication is typically in the range 2-5 months - much longer than FPGAs, which generally require 1-4 weeks once a customer's RTL is firm.

These NRE and TAT issues are compounded by customers' needs for multiple design passes. Since each ASIC design requires a unique mask set, if a customer discovers logic errors or needs to add features after tape out, they must initiate another ASIC design pass, requiring additional NRE charges and silicon fabrication time. As silicon technologies progress and chip designs become more complex, design verification becomes increasingly difficult, and the chance for logic errors grows. In many cases, time to market pressures drive design teams to continue verification well into layout and sometimes beyond chip tape out. This increases the risk that logic updates will be required, and therefore cost per chip will increase.

So to review: ASICs to date have offered higher performance in smaller chip sizes than FPGAs. However, the NRE for current technology nodes has rendered them very expensive for applications that require low quantities of chips - particularly when multiple designs or design passes are required.

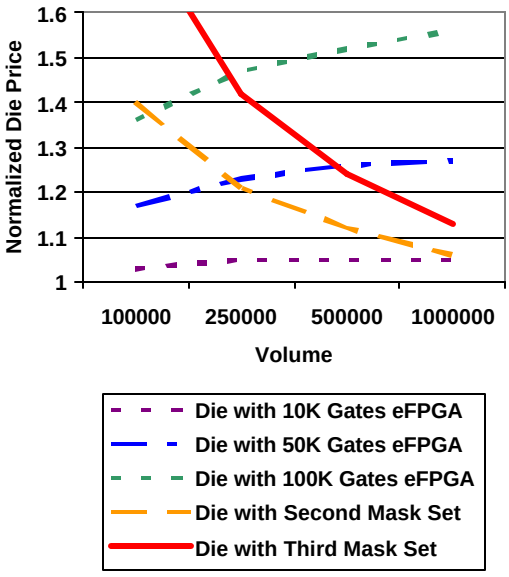

Fig. 5. Normalized Die Price vs. Volume

The Hybrid ASIC/FPGA Solution

Enter the hybrid ASIC/FPGA. Like an ASIC, the initial mask set must be purchased. But with the incorporation of FPGA cores into the ASIC, it is now possible to use the programmable circuitry to enable a single physical chip design to satisfy several different applications. This has the potential to eliminate multiple designs and in some cases, avoid costly respins. In the case where a customer requires several similar ASICs for a family of products, FPGA circuitry can be added to the base ASIC logic and be configured as needed to satisfy the multiple applications. Similarly, logic updates required to correct bugs discovered late in the verification process, or to accommodate changing market needs, can be handled with appropriately placed FPGA cores.

The question must be asked; why embed FPGA into an ASIC if a two chip solution could achieve the same results? The answer is both technical and economic. Technically, for a certain class of applications, the embedded solution offers greater performance with lower power dissipation. By embedding the FPGA into the ASIC, signals that must propagate from the ASIC through the FPGA, then back to the ASIC can avoid four chip boundary delays, two card crossings, and the associated power dissipation. By keeping the ASIC to FPGA interconnections on the die, valuable ASIC I/O pins are also conserved.

Economically, the embedded solution can be the less expensive option. As we will discuss, the FPGA fabric does not require any unique semiconductor processing above and beyond the base ASIC (unlike embedded flash or embedded DRAM). The resulting increase in ASIC cost is associated with the area occupied by the embedded FPGA core. In addition, the cost of assembly, test and packaging of a second chip are eliminated.

Figure 5 shows normalized die price data for the 90 nm technology node. The table is normalized to the approximate price of an ASIC die containing 10M gates. Each data series represents a unique die configuration where the total gates on the die equal 10M. The prices shown include only the mask and die fabrication estimates, and do not include packaging, test or other costs. The data is meant only to show the general trends in prices for hybrid designs compared to ASIC designs. The prices of the various options were obtained by calculating the total die size that would result from the chosen configuration. Areas were calculated using IBM Cu-08 gate areas and Xilinx 90 nm FPGA gate areas. Foundry average wafer prices and the mask costs from Figure 4 were used. Chips per wafer and yield were estimated.

The figure shows that it can be advantageous in certain cases to include embedded FPGA on an ASIC if that FPGA eliminates the need for additional design passes. For example, at volumes of up to 250,000 pieces, 50K gates of embedded FPGA are cost effective. (At that volume, the per-chip cost of a single design with 50k gates of FPGA logic is roughly equal to the cost of a 2-pass ASIC design.) Similarly, 10K gates of embedded FPGA are cost effective versus a 2-pass ASIC design at volume of up to 1M pieces. In general, if mask costs rise, volumes decrease, or more design passes are avoided, then the embedded FPGA approach becomes progressively more cost-effective compared to the ASIC approach. This is because at low volumes, the mask costs (and NRE) for additional design passes becomes a significant adder to per-chip cost, and this can outweigh the cost impact of the larger die area required by the embedded FPGA circuitry.

This analysis leads us to conclude that technology and market trends have created a need for the development of the hybrid ASIC/FPGA product. Mask costs for advanced technologies are

growing - making multiple design passes too costly for many applications. Fortunately, the technology advancements that have driven this trend have also opened up the potential to embed significant amounts of FPGA gates onto an ASIC die - enough to handle some of the design updates that would otherwise require additional design passes.

Hybrid Offering Overview

The IBM/Xilinx hybrid will first be available in IBM's Cu-08 90nm ASIC offering [7], and will consist of three FPGA block sizes. Multiple blocks can be used on the same die and the sizes of blocks used can be mixed and matched. Figure 6 shows the features of the various blocks.

Estimated Equivalent ASIC Gates	Estimated Size	Signal IO
10K	3 mm^2	384
20K	5 mm^2	512
40K	7 mm^2	640

Fig. 6. Hybrid Offering

Physically, the FPGA cores are being ported to the same semiconductor process that the ASIC product uses. The issues encountered in doing this porting are similar to those of other 3rd party IP ports. One of the largest challenges is full chip physical verification. Common design rules and transistor design points are critical in blending of IP between suppliers. Minor differences in design rules can be accommodated, assuming that checking decks and other verification software are able to handle the mixture of design rules. Designing these tools for increased flexibility will likely be needed as more companies share IP.

To ensure that the FPGA can be integrated with the rest of the ASIC, agreements must be reached on metal stack options. In the case of the Cu-08 hybrid offering, 5 levels of metal were allocated to the FPGA blocks. This requires a re-layout of the FPGA cores, which were originally designed for a standard product with 9 levels of metal.

As part of the re-layout, the power distribution of the FPGA blocks will be designed to integrate easily into the ASIC power distribution methodology. Care needs to be taken to ensure the power density required by the FPGA blocks are within the capability of the ASIC power supply routing. Due to extensive use of pass-gate structures, the FPGA blocks require standard 1.2V power supply levels, and are not operable below 1.0 Volt. For low-power applications, the FPGA blocks will make use of IBM's Voltage Island capability [13], allowing them to operate at typical 1.2V levels, while the bulk of the chip operates at lower levels.

The embedded FPGA blocks consist of programmable logic blocks, configuration logic, test interface logic, and simplified IO buffers for use in driving and receiving on-chip nets. Multiple end user configuration modes are supported including JTAG, serial and parallel modes. Individual cores can be configured asynchronously, allowing for "on-the-fly" reconfiguration.

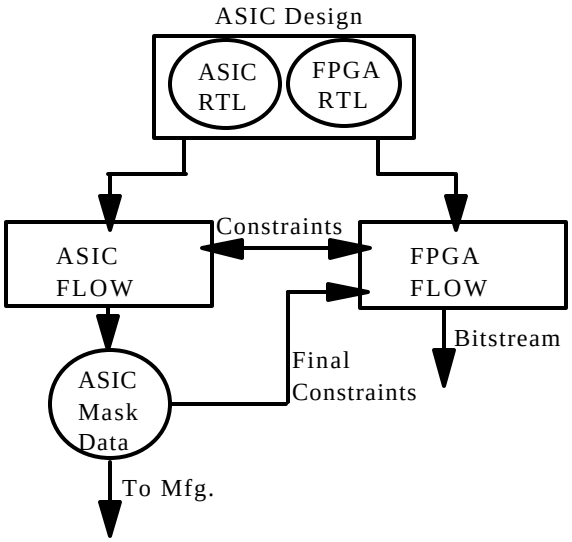

Fig. 7. Hybrid ASIC/FPGA Flow

To design the new hybrid chips, a modified design methodology is being developed as shown in figure 7. This hybrid design flow incorporates two proven design methodologies, the IBM ASIC flow and the XILINX FPGA flow, including several third party vendor synthesis options. The ASIC methodology integrates the embedded FPGA as a hard core with appropriate ASIC level models. The FPGA flow, including timing closure of the FPGA configuration, is done using XILINX tools. The designer has the choice of using constraints or detailed timing from the XILINX tool flow to close the ASIC timing at the FPGA core interfaces. If an FPGA configuration is known prior to the design of the ASIC, actual timing information can be passed to the ASIC tools from the FPGA tools. If the logic content of the embedded FPGA is unknown, the ASIC design can be completed using timing assertions and the embedded FPGA design can be completed later. If the embedded FPGA design is being reconfigured after the ASIC is in manufacturing, the final timing constraints from the completed ASIC can be passed to the FPGA tools for timing closure of the new FPGA design.

The logical design of the chip must be partitioned prior to final synthesis. The logic destined for an FPGA block is processed independently of the logic destined for ASIC logic. When multiple FPGA logic blocks are used, each must be designed and optimized independently.

The ASIC physical design process treats the FPGA macro similarly to other large placeable objects, except for port assignment. During the initial ASIC design, the port assignment of each embedded FPGA block can be modified to accommodate floor planning or timing requirements. Once the final ASIC design is taped-out, the port assignments are fixed for subsequent FPGA configurations.

The IBM ASIC methodology has been described in references [14-16], and the Xilinx FPGA methodology is described in reference [17]. As to be expected, most of the issues in creating the hybrid methodology occur at the boundary between the two methodologies. The mechanics of the communications between the two systems can be accomplished by creating data translators, however, optimization between the two systems can be difficult, due to the significant architectural differences between traditional ASIC flows and traditional FPGA flows.

CAD Challenges / Design Challenges

There are several significant challenges posed by this new architecture. The FPGA gate counts that can be embedded are still a relatively small percentage of the total ASIC gates on today's designs. Efficient design planning and logic partitioning will be crucial to successfully use this scarce resource. Timing and clocking will need to be optimized across terrains, and detailed floor planning will be critical. Finally, a variety of synthesis and simulation model issues need to be resolved to enable customers to design with confidence.

One of the key design challenges of this hybrid technology is how to efficiently partition the logic design [18]. The partitioning problem takes on several flavors, including partitioning between the ASIC and FPGA domains, and between individual FPGA cores.

The initial partitioning of logic functions between FPGA and fixed ASIC gates is particularly critical. As figures 1, 2 and 3 illustrate, there is a clear tradeoff between the flexibility of the FPGA circuitry, and the area, power and performance advantages of standard ASIC gates. The large differences in these circuit metrics require the designer to carefully evaluate which portions of the logic to implement in embedded FPGA.

As previously mentioned, candidates for FPGA implementation include the logic associated with changing standards, and logic required for families of similar products. These applications require significant planning, since the FPGA circuitry must satisfy the complete set of design requirements for multiple configurations. Fortunately, these requirements are known up-front, so the scope of this planning is narrowed.

On the other hand, cases where a design team uses FPGA cores to help prevent redesigns due to logic bugs can be much more difficult. For many general applications, using embedded FPGA in this way is impractical, since it requires successfully predicting where logic bugs will occur. It also requires incorporating the correct interconnections between the FPGA and standard cell logic for potential fixes.

There are some specific applications however, where 'buggy' portions of the design can be identified up-front, making them good candidates for FPGA use. Data processing applications for instance, often require the majority of the chip be dedicated to dataflow. This dataflow logic is commonly implemented with repetitive logic blocks, which are less prone to errors than the more random control logic. Cases like this can narrow the scope of locations for logic bugs, and help in identifying the best partitions for FPGA circuitry. Similarly, cases where significant portions of the design are reused can help indicate the best functions to implement in FPGA cores.

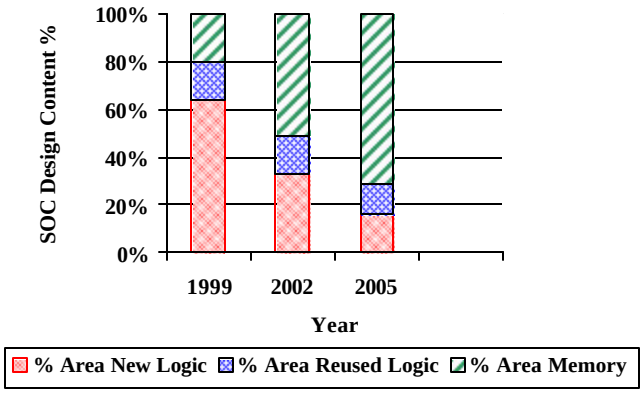

Fig. 8. SOC Design Content by Year

Figure 8 above shows the trend in the division of SOC design content [19]. This indicates that the percentage of SOC chip area dedicated to newly designed logic is decreasing. While this trend favors a hybrid approach, the amount of new logic on SOC designs still outstrips the capability to target all of this logic into embedded FPGA cores. Choosing which logic is implemented in FPGA will remain a challenging issue.

We expect fabrication technology to continue to advance, allowing design complexity to continue to grow. The resulting smaller feature sizes will also allow hybrid technologies to integrate greater numbers of FPGA gates, easing the task of identifying the best partitions.

Another partitioning challenge arises when the size of the logic targeted for an embedded FPGA block exceeds the capacity of the largest available FPGA core. The logic must then be split and implemented in more than one FPGA core. Today, a one-to-one mapping between the logical and physical partitions is needed for these cases. Separating the physical implementation hierarchy from the logical hierarchy would leave the original logical partitions unchanged and easily recognizable to the designer. Earlier research in this area has shown that large designs can be mapped between discrete FPGA devices [20], and potentially these techniques can be applied to the automatic partitioning of logic into multiple FPGA macros in an ASIC.

Optimizing logic that has been split in this manner also introduces additional interconnect complexity between the FPGA blocks on the ASIC die. This can be further complicated by the possibility that these connections may need to traverse a portion of the ASIC circuitry as well.

These issues of design partitioning leads to the requirement for the design tools to simultaneously consider the logic paths in both the ASIC and FPGA portion of the design. The ideal solution would be optimization of cross-architecture paths. This would allow design of tightly integrated logic, with timing paths crossing multiple boundaries between fixed and reconfigurable circuitry. Current tools can only handle this piecewise. As a result, the design team must anticipate all the required connectivity up-front. Alternatively, defining some form of structured design (such as implementing flexible bus architectures) could facilitate integration of the fixed and reconfigurable logic.

In summary, there are number of issues surrounding partitioning of logic for implementation in the hybrid architecture. Logic design teams already face many of the same issues with today's ASIC SOCs and are solving these problems through a combination of manual effort and emerging design tools [21-23]. For tomorrow's complex hybrid designs, enhanced partitioning tools and methods will be even more critical to achieve optimum results across multiple circuit architecture terrains.

Planning for future reconfiguration

In addition to partitioning, designers will face several other challenges in using embedded FPGAs. The basic question of how many FPGA gates to include is fundamental. Not only must the FPGA be sized sufficiently for the initial application, but enough unused FPGA resources must be left to support future logic configurations. This is a critical design-planning consideration, since once the hybrid chip has been implemented in silicon, a second (costly) mask set is required if the FPGA capacity is insufficient to handle the future configurations. To prevent this unfortunate situation, the design team must anticipate the potential growth in the logic which is to be implemented in the FPGA, as well as correctly estimate the embedded FPGA utilization that can be achieved. In addition, because the interconnect between the embedded FPGA and the ASIC is fixed in the mask set, any future interconnect requirements must be accounted for during the initial ASIC design. These are difficult architectural and design planning challenges that will require enhanced CAD tools to help in the design of tomorrow's hybrid SOCs.

For optimization tools to effectively partition hybrid designs, they must be able to correctly model the area, power and performance capabilities of both ASIC and FPGA circuit architectures. Since the architectures are so different in these characteristics, tools that are capable of efficiently and quickly assessing these tradeoffs will be needed to help the designers choose the best logic partition and specific circuit options for each portion of the design.

Floorplanning and Physical Design

Once the initial design is partitioned, the next step is to plan the physical layout of the chip. The hybrid architecture presents the design tools with some interesting challenges in this area. First, by their nature, the embedded FPGA cores are very metal-intensive. The floorplan of the ASIC design must consider the global chip interconnect requirements when choosing the location for each core, to prevent chip wiring congestion. Similarly, the size of the FPGAs can have an impact on signal routing over the core itself, due to RC delays and noise considerations. The large cores may also interfere with pad buffer placement and routing in flip chip architectures. These present additional dimensions that floorplanning tools and designers need to consider and optimize.

Next, the problem of port assignment must be solved. In traditional hierarchical design, the port assignment of a block involves simultaneously solving an optimization problem between two levels of hierarchy within the same circuit architecture. In the hybrid architecture, this optimization problem is more complex; spanning two tool sets and two circuit architectures.

Proper port assignment is necessary at the ASIC level to remove routing congestion and also to aid in timing closure. However, this port assignment can have a significant impact on the optimal configuration of the FPGA. In today's environment, this leads to a "chicken and egg problem". Is the ASIC optimized first and then the resulting port assignment used in the FPGA, or is the FPGA optimized first, resulting in a potential impact to the ASIC design? A third option is to perform multiple iterations through each tool set, evaluating the results and identifying the best solution. What is needed is a port assignment algorithm that can find an optimum solution, by considering the congestion and timing issues at the ASIC design level as well as the configuration complexities of the FPGA.

Synthesis

Following port assignment, the various portions of the design are implemented through synthesis. Unfortunately, the existing synthesis tools are optimized for solving the ASIC synthesis problem or the FPGA synthesis problem - but not both. Differences in the architectures between ASICs and FPGAs have caused synthesis tool vendors to create independent solutions for the different targeted circuit architectures.

When synthesizing the ASIC, the existence of embedded cores can present optimization challenges at the core-to-ASIC interfaces. In cases where non-programmable cores are present, the timing relationships and constraints at the boundaries are well understood and detailed optimization can be performed. FPGA cores are not initially characterized for the timing of the final configured logic function and present the ASIC synthesis process with only general constraints. Therefore, the ASIC designer must conservatively budget for the interface timing of the programmable core(s) and represent this to the ASIC synthesis tools as user-defined constraints. The reverse is also true. The designer must represent the ASIC timing constraints to the FPGA synthesis tool in the form of timing budgets.

A more general solution is needed, where the synthesis tool comprehends the detailed timing interface between the ASIC and the embedded FPGA macro. It could then optimize timing across the ASIC/FPGA boundary and handle multiple technology targets without multiple passes through the tools. Once the initial synthesis and timing closure are complete on the entire design, the timing constraints surrounding the FPGA block must be characterized. Then when future FPGA configurations are required, the timing constraints for the FPGA are known and the design team does not need to modify the chip-level ASIC timings.

"Uniquification"

Simulating several identically named FPGA macros in an ASIC environment is a significant issue to be solved. Each macro must have a unique functional representation in the design for correct verification. Current levels of HDL languages do not easily support multiple function and timing references in a single "module". Automatic instance-based selection of the appropriate functional representation is desired.

Synthesis of the ASIC can handle the FPGA macro as a "black box" with timing assertions provided by the designer, but it would be preferable to use detailed timing information from the FPGA flow to influence the ASIC synthesis results for each macro boundary. Since each instance of the macro can have differing timing arcs and timing checks, annotation of this information from the FPGA flow into a single FPGA model can be troublesome.

ASIC static timing would also benefit from detailed timing information about each design implementation in the multiple FPGA cores in the design. However, the timing model for one FPGA core is typically not going to be applicable to another (differently configured) instance of the same core on the ASIC. The overall problem to be solved is supporting one physical block that can have many unique logical configurations for simulation, synthesis and static timing, with automatic selection of pertinent information based on some type of "key" mechanism.

The tactical approach we have chosen for this problem is through "uniquification" of the FPGA core name. That is, to create a uniquely named set of models that allows function, timing and synthesis independence for each FPGA core instance in the ASIC design. However, the cost for replicating all of the physical design data and creating a new cell name for each FPGA core instance is prohibitive. This is because the new models and physical data must be created, added to the ASIC library, and then supported for the life of the ASIC design. The data volume associated with these physical models is large, and can be costly to maintain.

Uniquifying the FPGA core name in only the logical design models is a more manageable approach, since there is much less data to manipulate than physical design data. However the resulting disparity in names between the logical and physical core models must be resolved in the ASIC physical design space for successful physical design results. Software that can recognize the uniquified logical FPGA core names is used to resolve the appropriate physical model to use for each instance of the FPGA cores.

Ideally, what is needed in the future is a new set of tools and algorithms to support embedded reconfigurable logic blocks. These tools need to support one physical macro with many different logical and timing "views", eliminating the need for uniquification.

Test

The hybrid architecture presents an interesting test problem. Since the final configurations of the FPGA blocks may be unknown to the ASIC supplier, the test program that verifies these cores must be very robust and test all FPGA resources that could be used. Ideally, the test would exhaustively verify all of the circuitry in multiple configurations and every part of the routing and switching fabric. This type of exhaustive test is routinely done for standalone FPGA products, executing dozens of configurations and exercising the chips at-speed.

This test strategy is feasible when test equipment has direct access to all FPGA I/Os. The embedded FPGA situation is much more restrictive. If a similar test scheme is to be used, a method of isolating the FPGA core is required - like multiplexing (if sufficient chip pins are available), or scan isolation. Either of these methods adds complexity both in the boundary logic between the FPGA and ASIC circuitry, and in the test procedures. Test time and data volume are added concerns, since now the manufacturing test must be able to verify both the ASIC and FPGA circuitry – each of which ordinarily requires significant test data volumes. Performance testing requirements can compound this issue, and have the potential to be very difficult in the embedded environment. Careful thought must be given to the test methodology, and modifications will be required to the FPGA and ASIC circuitry, as well as the test software (and potentially hardware).

Apart from these chip test concerns, the introduction of configurable logic within an ASIC design gives system designers new opportunities for testing at the application level. In addition to functional system operations, FPGA cores can be configured to provide specific self-test or diagnostic functions for the ASIC or the end system [24]. This ability to add 'optional' circuitry could provide new avenues for system function and quality improvements in a number of applications.

Conclusions

We have briefly reviewed the basic features of the recently announced hybrid ASIC/FPGA technology, and shown that certain applications can benefit from the integration of these two technologies. This combination has significant financial and business implications, due to the potential for combining multiple designs and avoiding costly redesigns. The economics of this hybrid technology rest strongly on the fabrication process complexity – due to the increasing cost for masks, as well as the

amount of FPGA circuitry that can be cost-effectively integrated. We have shown some preliminary analysis which indicates that this marriage is economically viable at the 90nm technology node. While technically possible, the blending of these technologies introduces new requirements on many of the design optimization tools that are currently in use. The physical integration of large metal-intensive FPGA cores is challenging for floorplanning and chip physical design. Wide differences in power and performance specifications for the two technologies create unique challenges for design planning, logic partitioning, synthesis, and timing. The reconfigurable nature of the FPGA cores introduces complexity in modeling, simulation, and manufacturing test.

We view the incorporation of FPGA circuitry into the ASIC products as a logical step in the progression of technology integration. The offering of reconfigurable circuitry on the same die with high-performance ASIC logic can open new opportunities for system design features and quality. For specific applications, the hybrid approach also offers the potential for significant cost and TAT savings.

The realization of these benefits relies on design tools. We have identified some of the major areas for new development, and look forward to continuing work to help customers take full advantage of this new technology offering.

References

[1] IBM Microelectronics, *SA-12E Databook*, April 2002.

[2] IBM Microelectronics, *ASIC SA-12E Standard Cell/Gate Array Product Brief*, June 2002.

[3] IBM Microelectronics, *SA-27E Databook: Base Library and I/Os*, July 2002.

[4] IBM Microelectronics, *Blue Logic SA-27E ASIC Product Brief*, June 2002.

[5] IBM Microelectronics, *Cu-11 Databook: 12-Track Base Library*, June 2002.

[6] IBM Microelectronics, *Blue Logic Cu-11 ASIC Product Brief*, June 2002.

[7] IBM Microelectronics, Blue *Logic Cu-08 ASIC Product Brief*, April 2002.

[8] Xilinx Corporation, *XC4000E and XC4000X Series Field-Programmable Gate Arrays*, Version 1.6, May 1999.

[9] Xilinx Corporation, *Virtex ™ 2.5V Field Programmable Gate Arrays DS003-1*, Version 2.5, April 2001.

[10] Xilinx Corporation, *Virtex-II 1.5V Field Programmable Gate Arrays DS031-1*, Version 2.5, April 2001.

[11] G.R.Goslin, "A guide to using FPGAS for Application Specific Digital System Processing Performance", Xilinx Corporation, 1995.

[12] Semico, "Foundry Wafer Pricing: Fourth Quarter 2001"

[13] David E. Lackey et al "Managing Power and Performance for System-on-Chip Designs using Voltage Islands," *Proc. ICCAD-2002*, Nov. 2002.

[14] A.M.Rincon, M.Trick, T.Guzowski, "A proven Methodology for Designing One-Million-Gate ASICs," *Proc. IEEE Custom Integrated Circuits Conference*, pp. 44-52, May 1996.

[15] A.M. Rincon et al, "Design Methodology for IBM ASIC Products," *IBM Journal of Research and Development*, Volume 40, Number 4, July 1996.

[16] K.M. Carrig, N.T.Gargiulo, R.P.Gregor, D.R.Menard, H.E.Reindel, "A New Direction in ASIC High-Performance Clock Methodology," *Proc. IEEE Custom Integrated Circuits Conference*, May 1998, pp. 593-596.

[17] Karen Parnell and Nick Mehta, "Programmable Logic Design Quick Start Hand Book," Second Edition, Jan. 2002.

[18] W. E. Donath "Logic Partitioning in Physical Design Automation of VLSI Systems," The Benjamin/Cummings Publisher Company, 1988.

[19] International Technology Roadmap for Semiconductors (ITRS) 2000.

[20] H. Krupnova, A. Abbara, G. Saucier, "A Hierarchy-Driven FPGA Partitioning Method," *Proc. Design Automation Conference*, June 1997.

[21] "IC Wizard - The Hierarchical Design Planning Tool," © 2002 Monterey Design Systems, Inc., http://www.mondes.com/prod_icw.html

[22] "TeraForm® RTL Design Planner for Deep Submicron SOCs," © 2002 Tera Systems, Inc., http://www.terasystems.com/ products/datasheet.htm

[23] "First Encounter," © 2002 Cadence Design Systems, Inc., http://www.cadence.com/products/first_encounter.html

[24] S. Wilton and R. Saleh, "Programmable Logic IP Cores in SoC Design: Opportunities and Challenges," *IEEE Custom Integrated Circuits Conference*, May 2001.