HBD872/D
Rev. 1, Nov-2013

ON Semiconductor®

Energy Efficient Innovations

FPGA-to-ASIC Conversion Reference Manual

www.onsemi.com

Table of Contents

1.0 Purpose

A fundamental decision a designer must make for any new design is which hardware platform makes the most sense for the application. For engineers the technological issues of performance, power, quality, etc. are usually the first things considered. However, the business issues of cost and time-to-market (TTM) are just as critical. You can design the fastest product but if a competitor beats you to market then gaining market share will be more difficult even if your product is superior. That is why cost and TTM issues usually drive technological decisions, and why prototyping with FPGAs and converting to ASIC for production can often make sense.

This guide will help the designer create technology-independent portable designs specifically for the purpose of converting FPGA designs into ASIC designs with the best possible TTM and cost reduction solution. This guide is also useful for creating portable ASIC designs. ASIC houses often obsolete manufacturing processes after just a few years, making it necessary to port the design to another vendor.

The portability issues covered by this guide include documentation, packaging, IP, verification, synchronous design, and other issues that directly affect FPGA-to-ASIC and ASIC-to-ASIC migrations.

2.0 How to Maximize Cost Savings in FPGA Conversions

FPGA products focus heavily on a time-to-market value proposition. The field programmable nature of FPGA technology facilitates extremely fast design/debug iterations that lead to faster TTM. Even though FPGA technology can never match the performance and capacity inherent with cell-based technology, the latest high-end FPGA technologies offer enough of both for most applications. However, the per unit cost of high-end FPGAs is prohibitive for all but the lowest volume applications (less than a few thousand units per year).

ASIC technology offers the greatest technological benefit in terms of performance, power and capacity. However, technical advancements in shrinking geometries have exponentially increased reticle costs required for every new design. This increase in reticle cost translates to excessive non-recurring engineering (NRE) costs, making deep sub-micron cell-based ASICs too expensive for all but the highest volume applications. Additionally, the development span and risk associated with this technology makes it difficult to compete in terms of TTM.

The two extremes of cell-based NRE and FPGA per unit cost have left a significant gap in the mid-volume market. This gap has led to the emergence of structured ASIC products. Structured ASIC technology overcomes the two extremes by offering designers a solution with the capacity and performance required for modern applications but without the high NRE of cell-based ASIC technology and high per unit cost of FPGA technology.

Designing an application in an FPGA through the prototype stage and then converting that design into either a cell-based ASIC or structured ASIC, depending on volume, will provide the most cost effective solution for your application. Figure 1 illustrates tradeoffs between these platforms.

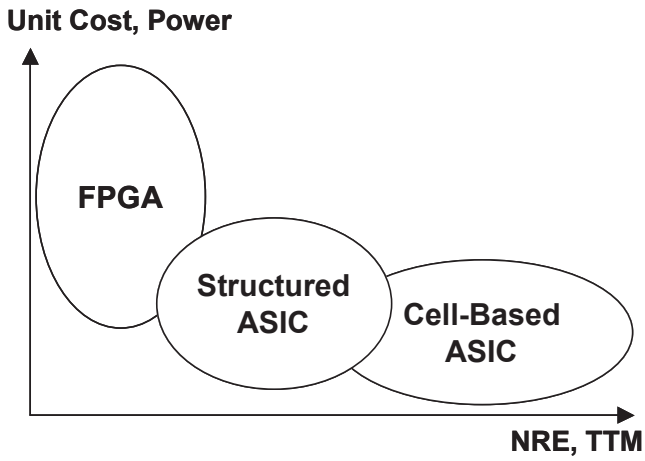


Figure 1. Design Platform Tradeoffs

In addition to choosing the appropriate ASIC platform there are other considerations that can increase cost savings during conversion from an FPGA.

2.1 Packaging Considerations

The primary reason ASICs are less expensive than FPGAs is that ASICs use less silicon area to implement a given function. However, packaging can be a significant part of the total cost. For prototyping, many designers select the largest FPGA device in the largest package just to avoid running out of gates or pins. This approach can lead to an expensive ASIC solution if converted directly. Because package costs are directly related to the number of pins, designers should be on the lookout for ways to reduce package pin count.

If there are many unused I/Os, then converting an FPGA into a smaller ASIC package can enhance cost savings significantly. In order to reduce the size of the package, the board has to either be redesigned to support the smaller ASIC package or designed to support two package

footprints, one for the FPGA and one for the ASIC. Figure 2 illustrates a package shrink approach using concentric pad rings.

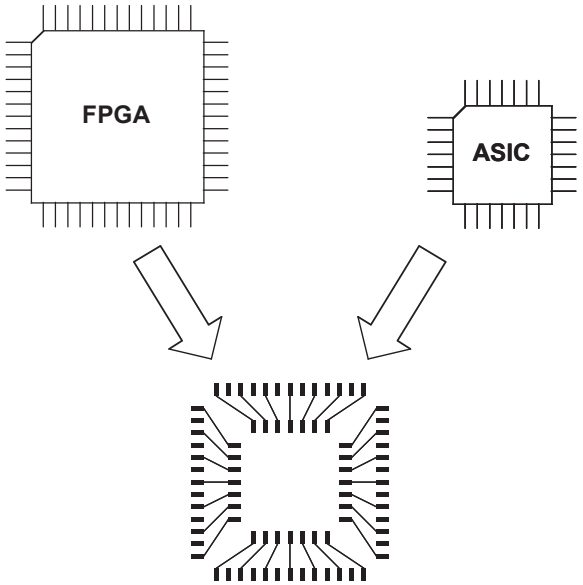


Figure 2. Package Shrink and Concentric Pad Rings

A similar strategy can be used with ball grid array packages where the outer signal balls on the FPGA are not used, making it possible to replace it with an ASIC having a smaller body size with fewer rows of solder balls.

2.2 JTAG - Match FPGA or Optimize for ASIC

Another area where significant cost savings can be achieved is with partial JTAG implementation. Unused or “no connect” pins can greatly increase die size when FPGA compatibility is maintained. In the FPGA, all unused I/O pins still have JTAG boundary scan support. Board level testers may use those pins to test interconnect on the board even though the pins are not used in normal operation. In a 100 percent drop-in replacement device it is necessary to retain these unused pins and associated I/O cells, resulting in a larger than necessary die size.

If your design has a significant number of unused I/Os you should work with the board level test group to avoid this dependency. It is still feasible to support the boundary scan register inside the ASIC to match register lengths, but connections to the outside of the chip should be avoided. For example, JTAG on all pins of a device with 30 percent “no connect” pins will double the size of the ASIC die. By using optimized JTAG on a reduced number of “no connects” the die size can be greatly reduced, leading to more per unit cost savings.

2.3 Core Power Supply Voltage

The supply voltage used to power the FPGA core has a limiting effect when selecting conversion technologies. Generally 1.8 V calls for 0.18 mm ASIC technology, 1.5 V calls for 0.15 mm, 1.2 V calls for 0.13 mm technology, etc. As the voltage drops, finer feature size technologies are required for good performance. This results in higher NRE tooling costs and potentially higher piece prices.

If the core supply voltage can be changed during the conversion, it is possible that a less expensive, older process technology ASIC solution can be used. This flexibility can be achieved by providing an independent core supply regulator on the board for any FPGAs that might be converted.

2.4 Other Savings: Power Reduction, Configuration EPROM, Board Space

There are numerous peripheral cost saving benefits at the board level generated by conversion from FPGA- to-ASIC technology. Since ASICs use considerably less power than FPGAs, power supplies can often be modified to reduce cost. Voltage regulators can be exchanged for less expensive models that handle less current, unnecessary heat sinks can be removed, etc. FPGA configuration EPROMs can be removed, saving component cost. Any removal of components or reduction in package size saves board space. Of course, being able to reduce the board size directly equates to additional cost savings.

2.5 Converting Multiple FPGAs

In addition to the savings possible through single FPGA-to-ASIC conversions, even greater savings can be achieved by either translating multiple FPGAs into a single ASIC for one product or translating multiple disjoint FPGAs into a single ASIC for use in multiple products. The single product multiple-to-one conversion benefits are fairly obvious. The more programmable devices combined into a single ASIC, the greater the cost reduction for that product in terms of both direct component cost and board real estate. The downside of a multiple-to-one conversion is the engineering resources required by both the ASIC supplier and the customer to implement the design. Multiple-to-one conversions require an in-depth understanding of the timing and interaction between all devices being integrated. Additionally, some design changes are inevitable. External tri-state busses will need to be converted to uni-directional busses. A top-level design architecture stitching together all of the devices being integrated will have to be created and fully verified.

Combining multiple disjoint FPGAs into a single ASIC is possible, due to greater capacities of ASIC technology, and can lead to conversion cost savings across multiple product lines. Disjoint FPGAs are a number of FPGAs that all have different functions. The FPGAs can be located on the same board or can be designed into several different boards. Combining these types of FPGAs into a single ASIC requires up front coordination (same footprint, different I/O, etc.) but can lead to volume cost savings.

Due to the engineering effort required, multiple-to-one conversions may not make sense for every application but should be considered due to the significant cost savings potential.

3.0 How to Reduce Time-to-Market

Time-to-market is a very important issue that has to be taken into account when starting any new project. TTM is not only critical in beating competitors to market but also impacts the profitability of the product. The longer it takes to get a product to market the greater the impact on market share. If TTM is the biggest factor for a particular application, then no other digital logic platform can match the TTM benefits of FPGAs, offering very short spans from design completion to prototypes. However, if the product ships more than a few thousand units per year then FPGAs quickly become too expensive.

Designs that are targeted directly to ASIC technology allow significant cost savings in volume but often require design cycles up to 24 months, primarily due to the verification effort. Even with the increased verification effort, full ASIC designs induce much greater risk than FPGA implementations.

Reducing TTM is one of the most important factors driving the structured ASIC industry. TTM is optimized by reducing design and manufacturing cycles. Providing pre-designed, built-in features and functions minimizes the design cycle. Pre-designed functions can include configurable IO, power grids, block RAM, timing generators, and other embedded IP. Structured ASICs reduce manufacturing cycle time as there are fewer layers to be processed.

Prototyping in an FPGA and then converting to either a structured or cell-based ASIC provides a path that reduces TTM to manageable levels while at the same time offering a cost effective solution for mid- to high- volume applications. This design methodology also provides minimal risk of silicon re-spins since the function of the design is first proven in a programmable platform.

3.1 FPGA Prototype to ASIC Production

By using the FPGA-to-ASIC conversion flow, it is possible to get the TTM benefits of FPGA technology and still achieve significant cost savings as volume increases. To accomplish this, designers first prototype and go into limited production using FPGA technology. Then, while production is ramping-up, the FPGA design is transferred into an ASIC using the FPGA-to-ASIC conversion methodology, as shown in Figure 3. Using this methodology the product is ready for the market as soon as the system design is finalized.

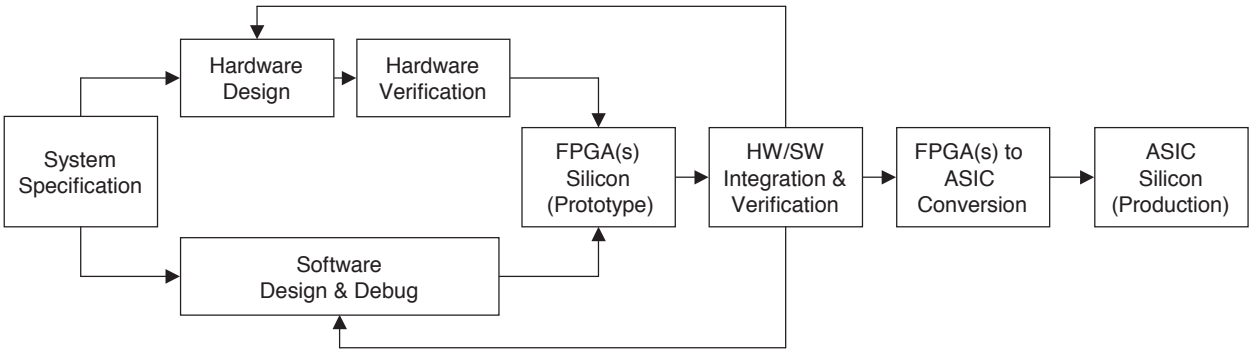


Figure 3. FPGA-to-ASIC Conversion Methodology

The FPGA-to-ASIC conversion methodology shown above enables short FPGA verification cycles while at the same time providing the cost effectiveness of an ASIC solution during production. Even multi-million gate designs can be prototyped on a board using multiple FPGAs, and then later be translated into a single ASIC.

The designer can plan ahead to ensure a smooth conversion from FPGA to ASIC by targeting an FPGA-to-ASIC conversion methodology flow at the beginning of a design cycle.

3.2 Parallel Design Flow

The more advanced parallel design flow shown in Figure 4 is for designers with previous conversion experience that desire to compress their FPGA prototype and ASIC development schedules. Designers work with an ASIC vendor to develop a flow for co-designing or designing the FPGA and ASIC in parallel such that once the FPGA prototypes are approved, the ASIC design can be immediately released into fabrication.

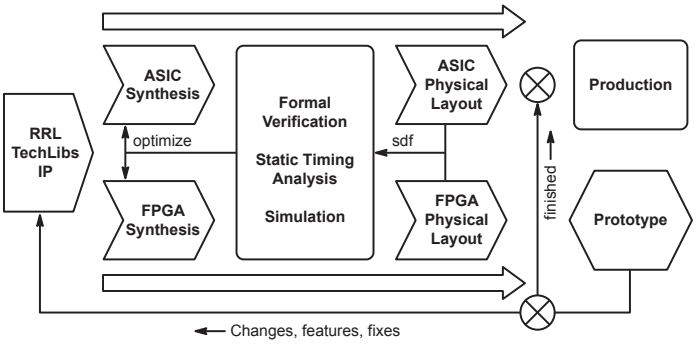


Figure 4. Parallel Design Flow

This approach requires a common compatible FPGA and ASIC tool set, requires both implementations to be developed in parallel and requires rigorous maintenance of changes to the RTL code and automatic recompiles. However, the total design cycle time savings to the final production ASIC can be significant.

3.3 Design Documentation

Good documentation reduces TTM by eliminating ambiguity and reducing the need to communicate with the designer. A solid design specification is important for system design and will help the FPGA-to-ASIC conversion flow. Documentation should be updated as the design progresses and is valuable both for the designer and anyone who may need to support the product in the future.

Documentation checklist:

- Naming conventions
- Design tricks
- Use of special FPGA features
- Operating conditions
- Chip-level/system timing budgets
- Asynchronous timing
- Timing margins
- Verification suites
- Synthesis scripts
- I/O characteristics
- Spare pins for testability
- IP blocks

Any unusual features of the FPGA that are used and any special design tricks should be documented. Chip- level timing, which documents set-up and hold requirements and clock-to-output performance, is very useful. Asynchronous portions of the design should be very carefully documented. You may not have thought about it this way, but simulation test benches provide a form of documentation by showing how outputs are affected by inputs. Synthesis scripts will be important if the design is to be re-synthesized. Be very careful to document the desired I/O characteristics of each pin, for example: LVTTL/LVCMOS, 4 mA/12 mA, slew rate, etc. Also make sure to document any unused pins which may be used for test modes or future enhancements.

Any third party IP blocks used in the design need to be identified in the documentation. Most IP used for the FPGA design will either need to be re-licensed for the ASIC implementation or converted to a compatible solution. Therefore, a brief definition of the IP block plus enough information to fully identify the block needs to be recorded. This should include the IP vendor's name, vendor part number, IP revision number, and any data sheets provided with the core.

3.4 System Timing Budgets and I/O Timing Constraints

Understanding how the device interacts with the rest of the system is one of the most common issues that delays the conversion process. System timing and especially system timing margin need to be clearly understood and well documented.

If the ASIC is going to be a drop-in replacement for the FPGA, the elements in the system timing budget must be maintained even though the ASIC I/O timing may be slightly different. Without knowing the ASIC timing, leave

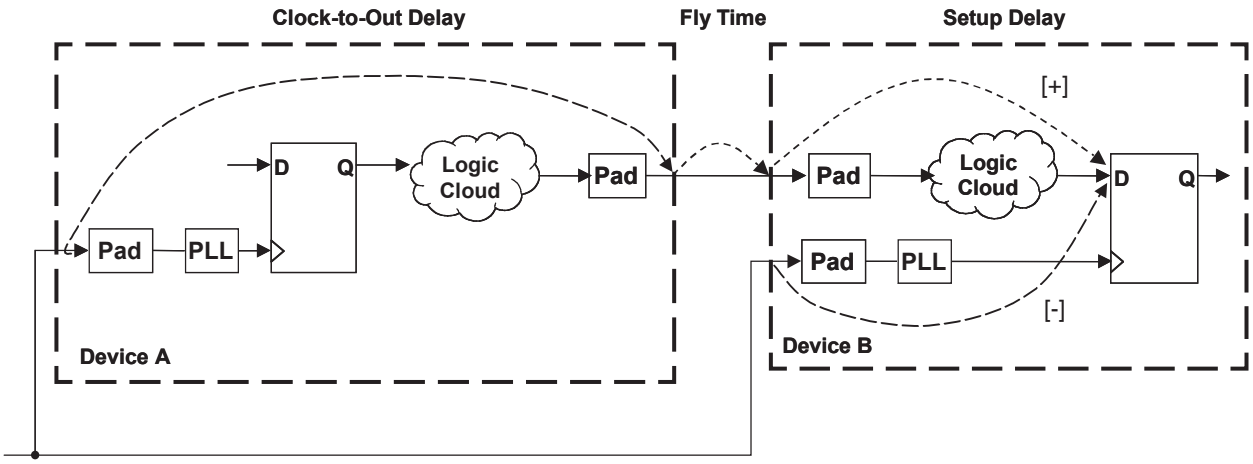


Figure 5. System Timing Diagram

some margin for the system timing budget since the ASIC I/O does not necessarily behave in the same manner as the FPGA I/O. Figure 5 illustrates the typical system timing involving two devices in a synchronous design. Figure 6 computes the system timing margin. The total of all the delays must be less than the clock period. The difference between the total delay and the clock period is the margin.

Parameter	System Timing Budget
T_clk-to-out	2.5 ns
T_fly	2.0 ns
T_setup	2.0 ns
T_jitter	0.1 ns
T_noise	0.2 ns
Total time	6.8 ns
Clock period	8.0 ns
Margin	1.2 ns

Figure 6. System Timing Budget

Extra margin is useful to know about when converting FPGAs to ASICs. For example, extra margin may make it possible to utilize an even less expensive ASIC technology. Many systems designers include 20 to 30 percent of the clock period as margin.

The definitions of setup time and clock-to-out time are illustrated in Figure 7. For FPGA-to-ASIC conversion, document any programmable delays used in the FPGA I/O cells as this information does not appear in the netlist.

Fly time is the time budgeted for signals to propagate across the circuit board and is normally a function of the permitted trace length. Jitter from PLLs and DLLs must be accounted for and may be different between the FPGA and ASIC. Noise comes from multiple sources such as simultaneous switching outputs (SSO), coupling and crosstalk.

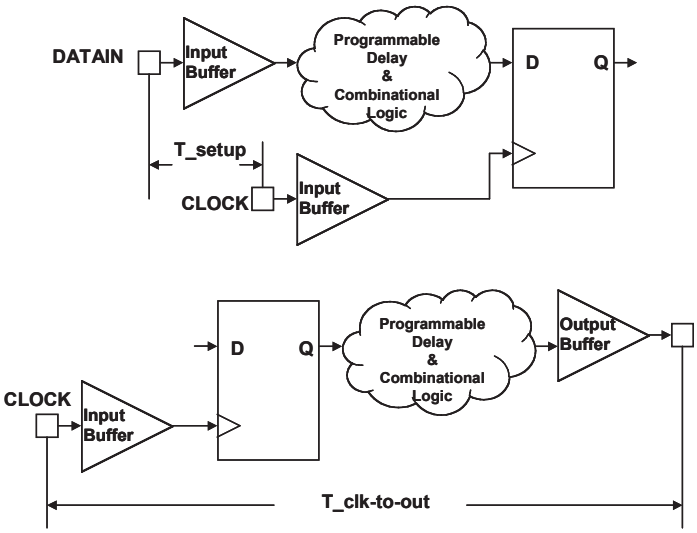


Figure 7. Definitions of Setup and Clock-to-Out Time

3.5 Internal Timing Constraints

Information needed to reach timing closure within the device is another area that must be well documented. The following characteristics need to be defined and documented for successful timing closure:

- Identify all clock domains (external or internally generated)
- Specify clock period and duty cycle for each domain
- Expected clock latency (worst case)
- Expected clock skew (worst case)
- Cross-clock domain definition
- Critical signals
- Multi-cycle paths
- Zero-cycle paths
- False paths
- Other design exceptions

Identifying and specifying all clock domains within the device, both system generated and device generated, is critical for successful timing closure. System clocks are generated externally and are brought into the chip through an input pad. Timing budget information is required for all system clocks. Internal or derived clocks are clocks generated internal to the device from one of the external system clocks or from another derived clock. An internal PLL, DLL or custom logic block may be used to create internal clocks. Gated clocks should be avoided whenever possible.

The clock period, duty cycle, latency, and skew must be defined for each clock domain within the device. Worst case clock latency is the delay from the clock origination point through the longest path to the register clock pin. The expected clock latency for all external system clocks in the design must be defined and documented. Clock latency for all internally generated clocks will be automatically computed during static timing analysis. Clock skew is the difference in arrival time of the clock signal to different registers in the clock domain or between different clock domains. Worst case clock skew must be defined and documented for all clock domains within the design.

The design method used to cross clock domains within the design must be reviewed at the beginning of the conversion cycle to ensure the integrity of the data is maintained across worst case corners. Section 7.3 Multiple Clock Domains, discusses reliable methods for passing data between clock domains.

All critical signals within the device need to be identified in the design documentation. Critical signals should include all high-speed, jitter sensitive networks, as well as all other time critical signals. All critical signals receive special attention from front-end design engineers when implementing static timing analysis (STA) and by back-end design engineers when performing layout of the device.

All design exceptions must be defined and documented. Design exceptions include multi-cycle paths, zero- cycle paths and false paths. All design exceptions are clearly explained in Section 5.3 Static Timing Analysis.

3.6 Design Organization and Hierarchy

It is good design practice to use hierarchy to manage design complexity. Hierarchy is also useful for managing critical timing path optimizations and computer run time.

The design problem is greatly simplified if all module outputs are registered; that is, they go through flip-flops. This effectively groups all combinational logic paths into a module and makes area and timing optimization more efficient. Each module should use a single common clock and a single common set/reset signal.

It is a good idea to place all chip-level I/O functions at the top of the hierarchy, making the core logic a sub- module.

3.7 RTL and Netlist Handoff Considerations

Register transfer level (RTL) is a high-level of abstraction used to define digital hardware structures. The two high-level languages used for RTL definition are Verilog and VHDL. The synthesis tool takes the RTL and provides a gate level netlist ready for place and route.

RTL handoff is a resynthesis methodology complete with supporting tools. RTL handoff allows the silicon vendor to accept RTL designs from their customers. RTL handoff has many advantages over netlist handoff:

- Allows more flexibility in addressing timing related issues
- Easier integration of soft IP
- Easier implementation of designs with high gate counts

The biggest disadvantage of using the RTL handoff methodology is that it may require more interaction from the customer at the beginning of the conversion process. If the principles set forth in this guide are followed then this disadvantage can be mitigated.

The deliverables that the customer is responsible for providing include the golden RTL code, FPGA synthesis scripts, timing constraints, and the identification of any embedded soft or hard IP.

Even with all of the advantages of the RTL handoff flow, the netlist handoff flow is still required for many applications. For example, if the application is an older design that is going through a cost reduction flow, the golden RTL may not be available, and in some cases the netlist may not be in sync with the RTL. Another example is when the RTL is restricted due to security reasons. Whatever the reason, if RTL is not available then the design can be converted using a netlist handoff flow.

Like the RTL handoff flow, the netlist handoff flow is a set of tools and methodologies that enable the silicon vendor to accept netlists for conversion to an ASIC. The deliverables the customer is responsible for providing include the golden Verilog or VHDL netlist, timing constraints and the identification of any embedded soft or hard IP.

4.0 How to Avoid Getting Locked into IP

IP portability is an area where the designer needs to be careful. IP selected for the application needs to be supported for both the FPGA and the ASIC devices. There are numerous cores offered by FPGA vendors that are proprietary to the vendor's technology. However, most ASIC vendors who specialize in FPGA-to-ASIC conversions offer IP solutions that are functionally equivalent to the more popular proprietary FPGA cores. Usually the best solution is to select a third party IP vendor who licenses a RTL version of the core, then use the IP core for both the FPGA and ASIC.

4.1 Proprietary IP

Even if synthesizable IP is selected for all of the application's functions there is still the legal issue to consider. FPGA vendors produce their own IP that they will not license for conversion to ASIC technology. Proprietary FPGA IP, even though it may be free, may block any true cost reduction operation, making the free IP very expensive indeed. At the very least, the use of proprietary IP will increase the complexity and cost of the ASIC conversion. This will equate to substantially higher NRE and a longer development span. Therefore, proprietary IP should be avoided whenever possible.

4.2 Engage with the Silicon Vendor Early in the Design Cycle

Using third party IP for both the FPGA and ASIC implementations requires some due diligence early in the design cycle but will payoff in the long run during cost reduction. The earlier in the design cycle the silicon provider is engaged, the better. Silicon vendors can help identify all of the hurdles associated with IP and can recommend third party solutions that can be used for both FPGA and ASIC implementations. Early engagement will also help to ensure there is time to deal with any IP modifications that might be needed for your specific application.

4.3 Use Soft IP Cores Whenever Possible

IP cores can be characterized as either soft, firm, or hard. This refers to the degree to which the core has been targeted toward a particular fabrication process.

Soft cores are in synthesizable HDL, and are more flexible than firm or hard cores. They have the disadvantage of not being as predictable in terms of performance (i.e. timing, area, power). They are also harder to protect because RTL source code is more portable and readable than either a netlist or physical layout data.

Firm cores are soft cores that have been pre-placed and routed as a block, having been optimized for performance and area using the target technology cell library. Firm cores offer a compromise between soft and hard. Firm cores are more flexible and portable than hard cores, yet their area and performance are known. They are easier to protect than soft cores.

Hard cores have been optimized for power, size, or performance and mapped to a specific technology. Examples include embedded cores such as block memory, timing generators and high-speed interfaces. Since hard cores are process specific, they are much more predictable, but consequently less flexible and portable due to process dependencies. Hard cores are difficult to reverse engineer and provide the best IP protection.

For most applications using a soft IP core over a hard embedded core is optimal. Only applications that require bleeding edge performance, are very area/power sensitive or include some analog functionality should target a hard macro. Using soft IP for your application will break down all technical barriers in using that IP in both FPGA and ASIC technologies.

4.4 Embedded IP

Any IP block that is part of the base architecture of a device is referred to as embedded IP. Advanced FPGA products come with numerous hard embedded IP cores. These cores have the pros and cons noted earlier in this section and include:

- Memories
- Timing generators (DLL/DCMs and PLLs)
- High-speed I/Os
- Processors
- SerDes transceivers

Embedded cores can present a hurdle for FPGA-to-ASIC conversion. However, most conversion products have equivalent IP, either in the form of hard or soft blocks. The advantage to using soft IP blocks in the ASIC is they only consume resources if your design uses them. In other words you are not paying for unused IP.

If the application calls for a processor then a third party soft processor core should be utilized versus the embedded core in the FPGA. A proprietary soft processor core should be avoided as well.

High-speed SerDes transceivers require the use of a hard embedded core due to the performance requirements. For programmable silicon vendors, embedded SerDes transceivers are still immature. Once the embedded technology matures, the cost reduction potential will increase significantly. Until then, for stability of design and long-term reliability, the best solution for current system designs is to utilize an off-chip ASSP solution.

One of the hidden problems with FPGAs containing embedded IP is that engineers may not initially plan on using the IP, but during the development process they may find it useful for solving a problem. This makes the conversion process much more complex.

4.5 Back Porting ASIC IP into FPGA Designs

In addition to recommending third party solutions, silicon vendors often maintain their own library of IP that can be deployed in the FPGA device early in the design cycle. This allows the designer to work with the same IP in the FPGA and ASIC resulting in a seamless ASIC conversion.

While soft IP is generally capable of being implemented in an FPGA, there are a few design issues to be aware of when implementing ASIC IP in an FPGA design. ASIC IP may not be as fast when implemented in an FPGA as compared to highly tuned FPGA IP. This means you may need to prototype the function at reduced speed in the FPGA, and then increase the speed to the desired level in the ASIC implementation. When implemented in an FPGA the ASIC IP may have a larger gate count than a similar function designed specifically for FPGAs. This may require a larger FPGA than originally planned, but as the FPGA is only used for prototyping and limited production the overall impact to the project cost is low.

4.6 IP Quality

One challenge in selecting and using IP in an application is determining which core offers the highest quality and, thus, the lowest risk. Poor quality IP is a leading cause of ASIC silicon re-spins. For most applications there is IP available from a third party IP vendor that is recommended by both the FPGA vendor and the ASIC silicon vendor. This usually means both the FPGA and ASIC vendor have certified that particular core for use in their technology, reducing the risk. In the absence of commonality, the best course of action is to utilize the IP recommended by the ASIC vendor.

Using the following IP selection guidelines can help to minimize risk:

- Evaluate the maturity of the IP. How often has it been used in real world applications? How many bug fixes were there in the past 12 months? What types of applications have utilized the IP?
- Evaluate the IP vendor. How big is the company? How long has it been in the IP business? Does the company have a reputation for quality IP? How many staff members are focused on IP design and verification? Does the company develop its own IP or act purely as a marketing firm for other IP developers?
- Evaluate the level of verification. Is the verification environment regressive, self-checking, portable, and well documented? What level of code coverage was achieved? What level of functional coverage was achieved by the verification environment? How was timing verified? Was the IP core verified by any industry standard or independent source? If so, is the report available? Is the vendor willing to improve or modify the level of verification if requested?
- Evaluate the IP vendor's quality of service. Ask for and verify references of other companies who have implemented the IP core. How fast does the IP vendor respond to issues with the core? Is the vendor willing to make modifications to the core if needed by the application?

Silicon vendors, both FPGA and ASIC, not only work to answer all the questions noted above but also work to independently verify IP that they recommend for use in their silicon. That extra verification effort helps to minimize risk and is the added value silicon vendors provide when acting as the middleman in the licensing of IP. Lower licensing fees are another benefit of obtaining the IP through the silicon vendor. These vendors generally license IP in bulk for multiple applications, thus getting a lower price from the IP vendor than possible for a single application.

5.0 FPGA-to-ASIC Conversion Verification

One of the primary strengths of the FPGA-to-ASIC conversion methodology is the reduced risk associated with FPGA prototyping. The design can be functionally proven in simulation, in the lab, as well as in beta systems under real world conditions prior to being converted to an ASIC, avoiding the majority of the functional risk associated with the pure ASIC flow.

5.1 Verification Strategies

This guide briefly discusses the verification methodologies used in the conversion process and provides some tips on how to make the verification complete and successful.

The designer's job is to make sure the product meets its marketing specifications. This can be verified by building FPGA prototypes and by developing test benches and simulating the design. Formal test benches work well because they allow the designer to establish regression tests to make sure the whole design works correctly after fixes or feature enhancements are incorporated.

The conversion engineer wants to make sure the design functions correctly after conversion from FPGA to ASIC. This can be done by using the designer's regression test bench to verify the results are the same across the conversion. Formal verification (logic equivalency checking) and static timing analysis (STA) can be used to "prove" the design is ported correctly from functional and timing perspectives.

The manufacturing engineer wants to make sure that manufactured parts with silicon defects are rejected. This can be done by inserting design for test (DFT) structures in the converted ASIC design to make it easily testable on automatic test equipment (ATE). DFT insertion is typically performed during the conversion process and should not alter the results of simulation regression tests or formal verification and STA. The DFT process also creates special high-fault coverage ATE vectors.

5.2 Formal Verification

Functional simulations are an essential part of verifying the design meets the product specification. During conversion, formal verification proves the converted logic has the same functionality as the original design source. The timing is not verified - just the functionality. Logic equivalence checks (LECs) are preferred to simulation because they are an exhaustive analysis of all logical possibilities. Unfortunately simulation is only as thorough as the test bench. Hand written test benches often overlook important functions. LEC checks are performed between the original RTL and synthesized netlist or between the original netlist and converted netlist.

Formal verification tools have limitations. For this reason it is good design practice to isolate memories, tri-state logic and IP blocks into separate modules. Be aware that flip-flops are key correspondence points used by LEC tools. In some cases, FPGA optimization tools will replicate flip-flops to isolate loads and this can cause compare problems. One solution is to code the RTL with the replicated flip-flops, as they will add little expense to the ASIC and are already present in the FPGA.

5.3 Static Timing Analysis

Static timing analysis is the primary tool used to verify timing closure on the ASIC during the conversion process. STA measures the register-to-register delay across the complete design and flags any errors or warnings based on a set of conditions defined by the library elements and in a script that is written based on the timing information provided in the design documentation. By default, STA assumes single-cycle timing for all paths in the design. Single-cycle timing means the data is expected to arrive at its destination within one clock cycle, as shown in Figures 8 and 9.

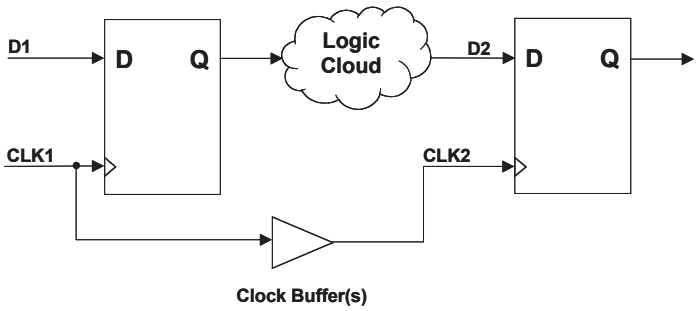


Figure 8. Single-Cycle Path

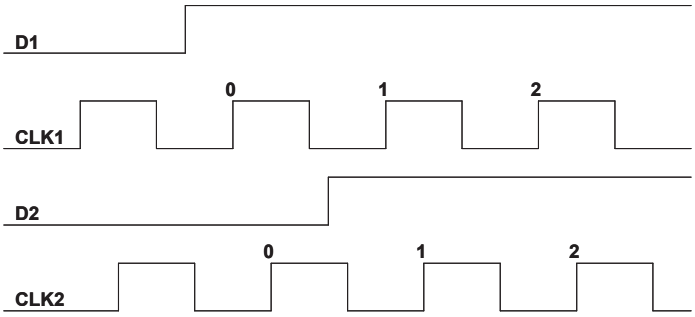


Figure 9. Single-Cycle Timing

For some designs there will be situations where single-cycle timing will not be achievable and a design exception will exist. Design exceptions include multi-cycle paths, zero-cycle paths, and false paths.

In multi-cycle timing paths the data is expected to take more than one clock cycle to arrive at its destination. A multi-cycle path is illustrated in Figure 10. Assuming the clock period is 2 ns and the delay through the logic cloud is 3.2 ns, the data will arrive at the destination register at clock edge 2 of CLK2 as shown in Figure 11. All multi-cycle paths in the design must be well understood and documented to include the instance name of the register where the path begins, the instance name of the register where the path ends and the number of clock cycles required for the data to propagate through the path.

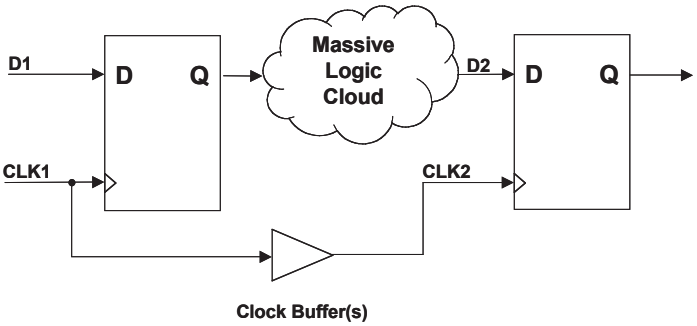


Figure 10. Multi-Cycle Path

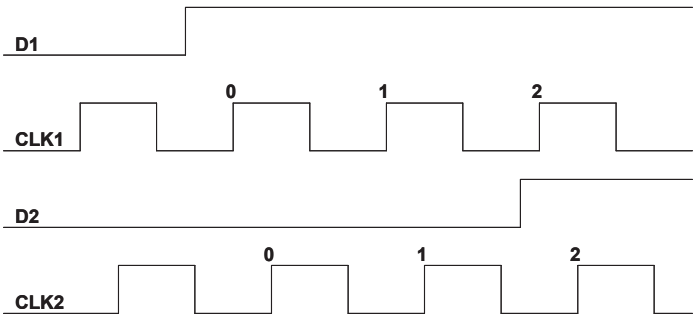


Figure 11. Multi-Cycle Timing

In zero-cycle timing paths the data is expected to arrive at the destination register in the same clock cycle that it is launched. A zero-cycle path is illustrated in Figure 12. Assuming the clock period is 2 ns and the delay through the clock buffer is larger than the clock to “q” delay through the launching register, the data will arrive at the destination register at clock edge 0 of CLK2 as shown in Figure 13. This is the same clock edge used to launch the data. All zero-cycle paths in the design must be well understood and documented to include the instance name of the register where the path begins and the instance name of the register where the path ends.

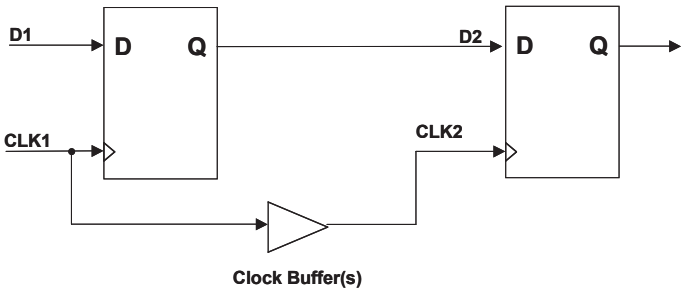


Figure 12. Zero-Cycle Path

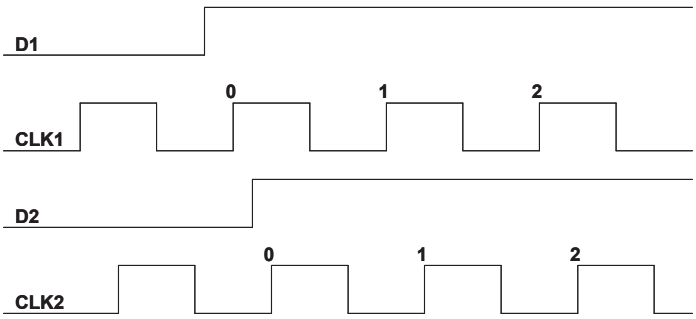


Figure 13. Zero-Cycle Timing

False paths are logic paths that exist but cannot be or are not intended to be analyzed. An example of a false path that cannot be analyzed is an asynchronous interface between unrelated clock domains. All false paths in the design must be well understood and documented to include the instance name of the register where the path begins, the instance name of the register where the path ends, and the reason for the denoted path to be false.

During the exhaustive analysis done by STA, the following timing checks are performed:

- Data setup: The setup time for each logic state at each sequential device’s data pin is checked. The setup time is defined in relation to the rising or falling edge of the corresponding sequential device clock signal. Setup time is defined as the amount of

time the data must be stable prior to the assertion of an active clock edge in a sequential device.

- Data hold: The hold time for each logic state at each sequential device's data pin is checked. The hold time is defined in relation to the rising or falling edge of the corresponding sequential device clock signal. Hold time is defined as the amount of time the data must be stable after the assertion of an active clock edge in a sequential device.
- Set/reset recovery: The recovery check is similar to the setup check and represents the minimum time the asynchronous set or reset pin must be stable after being de-asserted and prior to the assertion of an active clock edge in a sequential device.
- Set/reset removal: The removal check is similar to the hold check and represents the minimum time the asynchronous set or reset pin must be stable after being de-asserted and after the assertion of an active clock edge in a sequential device.
- Minimum pulse width: The minimum pulse width is checked for each sequential device's clock pin.

5.4 Test Vectors

Test vectors are generated from the simulation environment and can be used for power analysis during the conversion process and in manufacturing test. Test vectors are simply text files containing columns of ones and zeros that include the stimulus going to and coming from the chip I/O during functional simulations. Test vectors can be captured in either print-on-change or cycle-based format. In print-on-change format the I/O stimulus is captured every time any of the I/O change state. Obviously, print-on-change vector files can become huge for large designs in a relatively short amount of simulation time. In cycle-based format the I/O stimulus is captured every time the designated clock edge is asserted. The disadvantage of capturing cycle-based vectors is that asynchronous transition times will not be captured in the vector set.

Both methodologies for capturing vectors have their strengths and weaknesses. At the beginning of the conversion process the need (if any), amount, and type of test vectors for that application will be determined.

So what makes a test bench portable? If the intention is to use the test bench on ATE, it must only access the top level pins of the chip, as ATE is not capable of "probing" inside the design. Through the conversion process, internal node probing may not work well because of net name changes and logic optimizations. It is generally a good idea for test benches to only access top level pins.

Simulations should use full timing accuracy when capturing output data. To make a useful regression test, the input stimulus and output results should be captured in the output file, along with I/O control signals.

A good test bench will demonstrate the basic functionality of the design by presenting normal input to the device and capturing and checking the output. A good test bench should also test any special tricks in the design, and most importantly, any asynchronous timing.

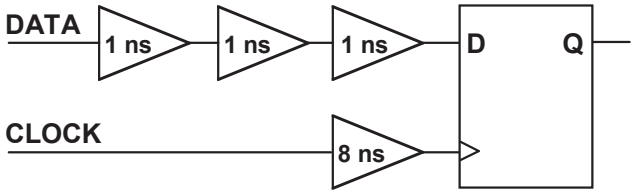


Figure 14. Simulator Delay Circuit

Delay Path	Zero Delay	Unit Delay	Assignable Delay
DATA → D	0	3 U	3 ns
CLOCK → C	0	1 U	8 ns

Figure 15. Results for Different Simulator Delay Models

An area of confusion with logic simulators centers around delay models. Some simulators permit gates to be modeled with unit delay or the actual gate delay. Unit delay simulations run faster in the computer, but are not as accurate.

For simulations to be meaningful, especially with asynchronous circuits, be sure to turn on the assignable or actual gate delay mode. For best results, use post-place and route back-annotated timing.

Figure 14 illustrates a simple circuit with several buffers having different delays. Figure 15 shows very different behavior can be obtained depending on the delay model used by the simulator.

5.5 Power Simulations

Simple gate count estimates can be used to calculate the amount of power dissipated by a design. However, power simulations generate better estimates of power dissipation. Power simulations are dynamic in nature and require a simulation pattern (test vectors) to execute. The closer the simulation pattern mimics real world conditions the better the power estimate. A simulation pattern that mimics real world conditions is referred to as a representative simulation pattern.

Not only is a representative pattern required for an accurate power estimate but the netlist also needs to be close to completion in the conversion process. In order to get the most accurate estimate of the power dissipated by the design, power simulations need to be executed using a representative simulation pattern against the final netlist.

5.6 Design for Test

All ASIC vendors have test coverage requirements that must be met for each ASIC produced. Design for test (DFT) is a methodology and toolset that enables the ASIC vendor to meet those requirements. The DFT techniques utilized by the ASIC vendor include SCAN and BIST insertion, IDDQ testing and at-speed testing.

Adding DFT to a design is easier if some simple rules are followed. Many of these issues cause other problems as well, so if you are using good design practices to begin with the impact should be minimal.

- Avoid latches. Latches don't work well in shift register scan chains and must be converted to flip- flops in test mode. It is much easier to use a flip-flop instead of a latch in the first place.
- Avoid combinational feedback loops. They act like latches.
- Don't use scan flip-flop library elements. Flip-flops in the design will be replaced by scan-flops and if that is already done, then extra muxes will need to be added anyway.
- Use synchronous design. It makes controlling the shift register much easier.
- Use a single external reset. It makes controlling the reset very easy, especially when the scan chain is shifting data.
- Include a test signal to turn off all DC biased circuits. This includes memory sense amps, I/O bias generators, I/O pull-up and pull-down circuits, etc. A special test signal that does this simplifies IDDQ testing.

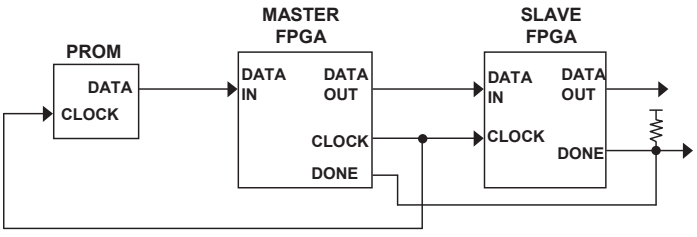


Figure 16. Daisy Chain FPGA Configuration

6.0 Good and Bad Design Practices Specific to FPGA Conversions

This section discusses issues that are very specific to conversion from FPGA to ASIC.

6.1 FPGA Configuration Dependencies and Emulation

During configuration an FPGA loads in data that personalizes it to perform the mission mode function. If the configuration process is not completed the part will not become operational. When power is applied to an ASIC it comes on instantly since it does not need to be programmed. An ASIC can be designed to emulate the FPGA configuration sequence, but implementation of this feature in an ASIC increases gate count and implementation effort and should be avoided if possible.

There are some cases where it is important for the ASIC to emulate the FPGA in terms of following the power up and configuration sequence. In a multi-FPGA chain, devices are normally programmed in a serial or daisy- chained fashion as shown in Figure 16. If one of the FPGAs is converted into an ASIC, then the ASIC might need to emulate an FPGA and it must pass the programming data through to the next FPGA in a correct manner.

Various configuration control signals might not be required when implementing the configuration logic in an ASIC. It is critical to understand the function of all control signals required during the FPGA configuration and the role of those signals, if any, in the design phase of an ASIC. This ensures that the functionality can be implemented correctly when converting an FPGA to an ASIC.

Configuration emulation can be implemented using various strategies. Some designers, concerned about whether the FPGA is correctly programmed, build checking into their system. This usually involves having a microprocessor monitor various FPGA configuration signals, such as the DONE pin. In some cases, they just wait for DONE to assert. This can be emulated by having the DONE signal tied high on power up. In other cases, designers make sure DONE asserts after exactly the right number of clock cycles. This can be emulated by counting the number of bits, and in case of a daisy chain operation, passing the data for the next FPGA in the chain once configuration is completed.

Generally in supervised systems, the effort required to change software is deemed greater than adding configuration emulation logic to the ASIC. However, if

a software change is possible it is a better solution than implementing the configuration logic in the ASIC design. Better yet, design the system and software to sense if the device is an FPGA or an ASIC and do the appropriate checking.

6.2 Resets

FPGAs contain a power on reset (POR) function that initiates configuration each time power is cycled. The configuration process loads all registers with the appropriate initialization values to make it appear as though a reset was applied. POR signals and the initial reset are generally not represented in the RTL code or the FPGA design netlist. They appear in simulation models as global signals, usually permanently de-asserted.

In the ASIC conversion, initial state values are implemented as a special initialization reset. In most cases this is the same as a system wide reset. Ideally an explicit reset signal is implemented in the FPGA. This reset signal should be brought out to a pin.

Use of system wide resets is good design practice because it establishes initial state. This makes logic simulation significantly easier and avoids FSM latch-up (dead) states. Every flip-flop in a design should have a reset, unless it needs to be initialized to '1' in which case a set makes sense. Resets may be either asynchronous or synchronous.

Global asynchronous resets are a good idea, but what happens if the reset is de-asserted right on the clock edge? Slow slew-rate reset signals further exacerbate the problem.

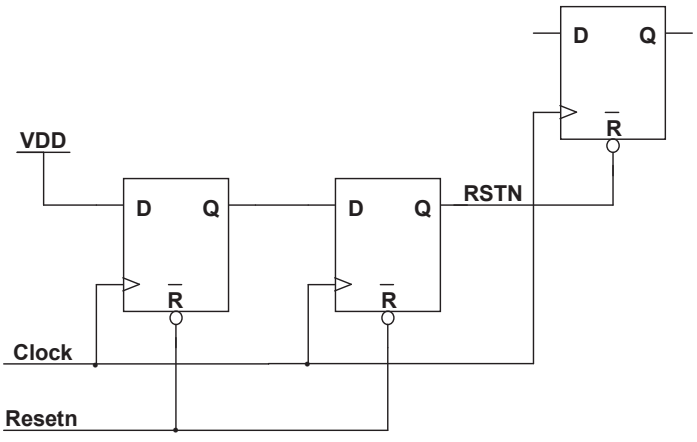


Figure 17. Synchronized Reset Circuit

Figure 17 shows a circuit to address this issue. This circuit causes an immediate asynchronous reset, but also ensures a synchronized recovery occurs. A reset circuit is required for each clock in multiple clock domain circuits and extra care is required to design them to interact properly.

Use of asynchronous set-reset flip-flops is discouraged because of ambiguity when both set and reset are asserted at the same time. While designers can usually predict what will happen in this case, unpredictable behavior will result if both signals are de-asserted at the same time. Designs that depend on specific behavior when both set and reset are asserted are not portable to other technologies, which may have different behavior for this condition. HDL coding priority issues can cause behavioral simulations to mismatch with gate- level structural simulations.

PORs should not be included in a chip design. While the function seems simple, they are an analog circuit with critical electrical characteristics, such as voltage threshold and turn-on time. When there is more than one chip in a system with POR you cannot ensure they will all complete their resets at the same time. Generally it is best to have one master POR generator driving all the chips in the system.

6.3 Memory Initialization

FPGAs provide a means to define the content of RAMs via configuration. Initialization patterns are usually passed as parameters to RAM instances in the FPGA netlist. By default these parameters are all zero. The FPGA configuration circuitry writes data into the memories during configuration, such as all zeros, or any desired memory pattern. Some FPGAs offer memories that are not initializeable and power up to an unknown state.

ASIC RAMs generally power up in an unknown state. Making them power up in a known state requires more logic or a special RAM. Either an on-board ROM must be copied into the RAM at configuration time, or register file RAMs with appropriate sets and resets must be used. Either solution can be expensive. In addition, the memory used in an ASIC cannot be easily re-programmed as is the case when using FPGAs.

Some ASIC vendors offer memory solutions to match the features offered by various FPGA vendors. These memories are completely initializeable and can be used as a one for one replacement for memories found in typical FPGA designs.

6.4 Synchronous vs. Asynchronous Memory

Synchronous memories are safe and easy to use because data, address, and write enable are registered, making the timing relationship between these signals simple.

Asynchronous memories are less portable than synchronous memories and should be avoided. Reliable operation is dependent on the design and timing of peripheral support circuitry. The timing and quality of the write enable pulse is critical and it is difficult to meet timing with simultaneously switching inputs. It is essential the address does not change while the write enable is asserted or the entire memory content may be corrupted. In addition, STA becomes a very complex task.

Most memories have synchronous write and read capabilities, and some have asynchronous read. As memories get bigger and performance demands increase, most FPGA vendors have moved away from supporting large asynchronous read memories.

6.5 I/O Standards and Matching FPGA I/O Characteristics

Before starting the FPGA design, check the ASIC vendor's I/O library against the I/O standards that will be used in FPGA design. Not all I/O standards supported by FPGA are popular, so ASIC vendors often only support the more common I/O standards. Most likely, the ASIC vendor supports the same I/O standards as the FPGA vendor but their electrical characteristics are not exactly the same. Items to check are DC current, edge rates, and propagation delay. There are also options for the same I/O standards. Whether an I/O has pull up, pull down, or bus hold, and whether an I/O is 3 V tolerant or has input hysteresis (Schmitt trigger), is the type of information that needs investigation and should be documented for a smooth conversion.

Most companies specializing in FPGA-to-ASIC conversions can provide I/O characterization reports. These reports provide a comparison between the FPGA I/O used in a design to the I/O provided by the ASIC vendor.

It is a good practice to review this information before selecting the ASIC vendor. Assessing the high speed I/O or critical clock lines may take more effort. The signal integrity team should simulate with I/O models provided by the ASIC vendor to ensure the ASIC I/O performs at an acceptable level compared to the FPGA I/O.

6.6 Operating Conditions

While it is normally not an issue, it is a good idea to double-check the expected operating conditions for the FPGA and ASIC. Specifically check the junction temperature range and limit. An ASIC typically draws five times less power than the FPGA it replaces, so junction temperature is not normally an issue unless there is an upgrade to military temperature requirements. Also check the tolerance of all power supplies for ±5%, ±10%, etc. Finally, check the input over-voltage capability.

6.7 ESD, Latchup, Hot-Socketing

FPGA-to-ASIC conversion users can assume an ASIC will have the same ESD, latchup and hot-socketing capability as an FPGA. Nevertheless, always ask for the quality report to avoid any surprises later on.

Depending on test methods, ESD can be categorized as human body model (HBM), machine model (MM), and charged device model (CDM). Failures found in HBM and MM testing are typically in the diffusion regions of the protection circuits. Failures found in CDM testing are usually gate oxide damage. The industry standard for HBM testing is 2K V. The typical testing voltages for MM and CDM are 200 V and 500 V, respectively.

6.8 Simultaneously Switching Outputs, Input Noise

Simultaneously switching outputs (SSO) create noise on the power and ground supplies. If the supplies bounce enough, the output timing is changed and the input noise margin is reduced. FPGAs and ASICs normally have SSO guidelines. Before locking the FPGA I/O pinout, it is a good idea to check the ASIC guidelines.

Some tips for reducing SSO effects:

- Isolate output and core/input pad supply pins
- Spread out high-drive output pins
- Interleave the I/O pads with more power/ground pins
- Use controlled slew rate drivers
- Stagger output signal transitions (~5 ns skew)
- Use lower inductance package
- Reduce capacitive loading/driver size
- Use differential outputs
- Move high-drive buffers off-chip

Some tips to reduce input noise:

- Isolate input and output pads, as shown in Figure 18
- Use differential inputs
- Keep away from high-drive outputs
- Stay close to power/ground pins
- Use Schmitt trigger input buffers for the most sensitive inputs (clock and reset)
- Skew the output transitions away from the input transitions

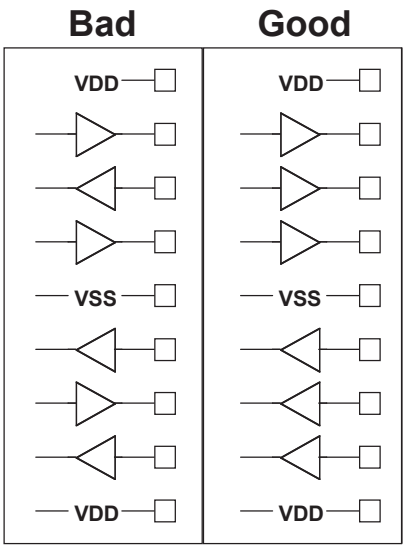


Figure 18. Group the Input Pads and the Output Pads and Isolate them with Pwr/Gnd

6.9 I/O Voltage Banks

The ASIC vendor will match the FPGA's I/O voltage banks. For both FPGA and ASIC design, it is a good practice to assign pinout such that the adjacent I/O banks share the same voltages. Not only does it cut the number of power supplies on the board, but also it improves the ESD protection by having more I/Os tied to the same power supplies. For ASICs with built-in on-die decoupling caps, two adjacent I/O banks sharing the same voltage double the benefit of decoupling.

6.10 On-Chip Terminations Using Digital Controlled Impedance

On-chip termination saves board space and alleviates routing congestion. The latest FPGA and ASIC devices offer this feature by adjusting the I/O impedance in reference to external precision resistors. Figure 19 shows different termination modes: series termination, parallel termination, split parallel termination, and differential termination.

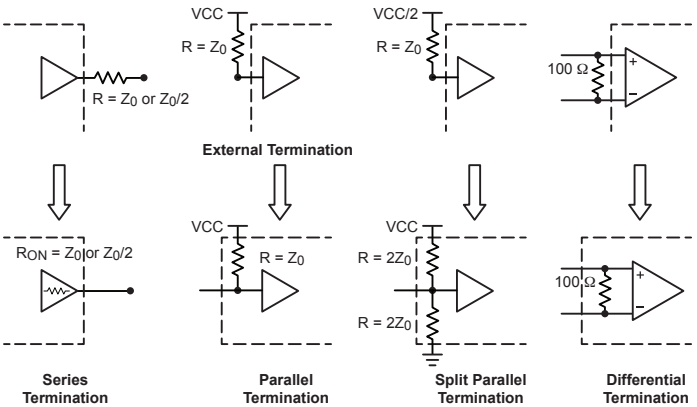


Figure 19. On-Chip Digital Controlled Impedance

Because the transistor impedance is nonlinear, the on-chip termination can only approximately match the target impedance. Therefore, for termination requiring very high precision you should use external termination resistors despite the board space penalty. Judging when to use or not use on-chip termination is best determined by signal integrity analysis.

6.11 Double Data-Rate Registers

The latest FPGA and ASIC devices embed double data rate (DDR) registers in the I/O pad cells. As illustrated in Figure 20, there are typically six registers - two for input DDR, two for output DDR, and two for output enable DDR. These registers are not just for DDR applications but can also be used in any applications that register the inputs or outputs. For those non-DDR cases, use one of the two registers in the pair either for input, output, or output enable. Because these registers reside in the pad cell, they offer the shortest clock-to-out time due to their short connections to the I/O buffers. More importantly, they offer consistent clock-to-out timing among I/Os in the same group.

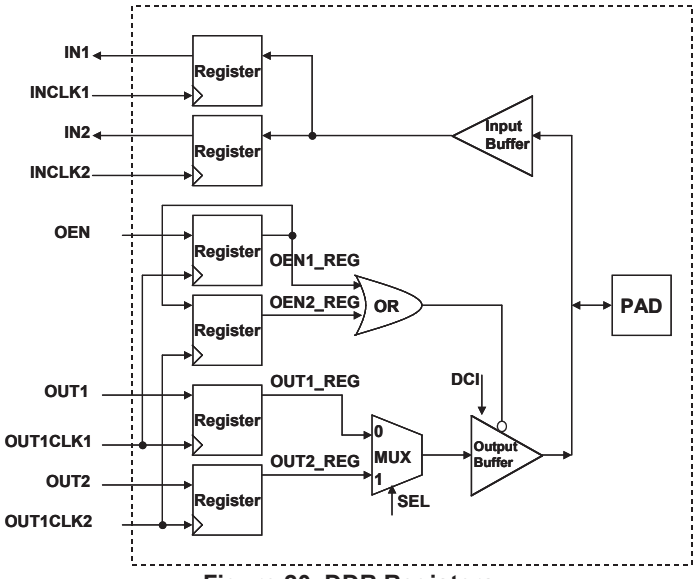


Figure 20. DDR Registers

7.0 General List of Good and Bad Design Practices

This section discusses a variety of good and bad design practices. Alternative approaches to bad practices are also discussed.

7.1 Synchronous Design

Synchronous design is popular because it eliminates many timing issues. However, the definition of synchronous is ambiguous, so here's a simple definition.

Synchronous designs have a single master clock and a single master set/reset driving all sequential elements in the design. Additionally, all input signals are synchronized to the clock in such a fashion that they never violate setup and hold time requirements. In Figure 21 the register represents all the flip-flops, latches, and memory elements of the design. The logic clouds may be complex logic cores or they may be as simple as a wire. The master set/reset signal may be asynchronous.

The benefit of this design style is that maximum clock frequency, input setup and hold time, and clock to out timing are the only timing issues.

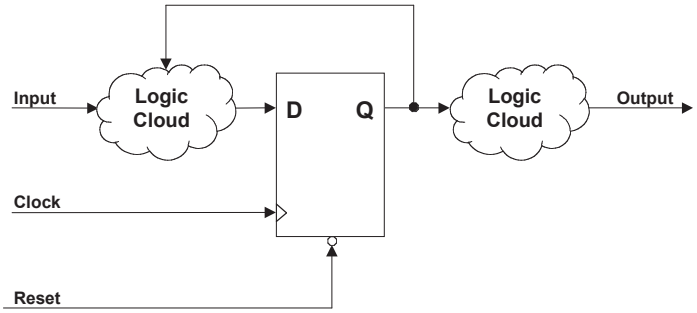


Figure 21. Synchronous Design

7.2 Input Synchronization (Metastability)

The circuit shown in Figure 22 works very well for synchronizing input signals. This circuit offers a high degree of metastability protection and should be used on all asynchronous inputs. Metastability may occur when the data-input changes at the same time as the clock.

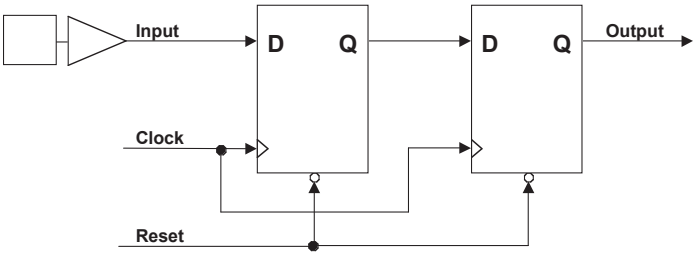


Figure 22. Input Synchronization Schematic

In this case, the flip-flop may capture an intermediate voltage level, often modeled as an "X" in logic simulation. This intermediate voltage level will eventually become a 0 or a 1, but it takes some time for the flip-flop to resolve it. This resolution time is usually several times longer than the clock-to-out time of the flip- flop, but less than the clock period.

By placing two flip-flops in series, designers can be sure the second flip-flop is always capturing stable data even if the first one is metastable for a time after the rising edge of the clock.

If combinational logic is added between the two flip-flops, the time available for stabilization would be reduced accordingly.

Effectively this circuit creates an input data sampling strategy, which avoids metastability problems and safely brings data into a synchronous system.

7.3 Multiple Clock Domains

Developing a design that is fully synchronous to one clock domain is ideal but often not possible. Applications should be designed as synchronously as possible with the fewest number of clock domains feasible.

Special care must be taken whenever there are two or more independent system clock domains and information is exchanged between the domains. Figure 23 shows two clock domains, A and B. It is assumed the data exported from domain A will be asynchronously received by domain B. In the general case, there is no way of knowing the relationship between clock A and B. A may be faster or slower than B. The relationship may even vary over time.

One reliable way to pass data back and forth is to use a handshake protocol. In general there are two protocols. The strobe method requires two edges on the hand shake signal as shown in Figure 24. The toggle method only requires an edge to be passed and is faster.

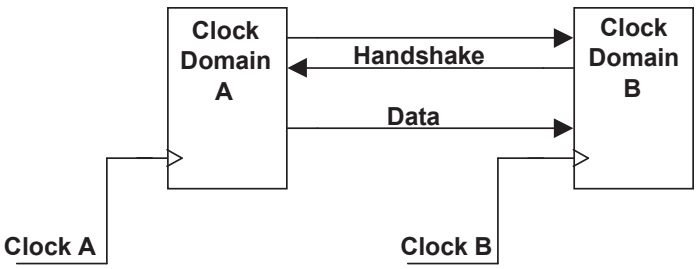


Figure 23. Multiple Clock Domains

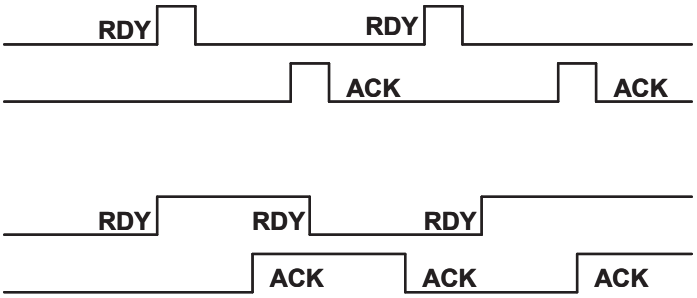


Figure 24. Strobe and Toggle Waveforms

The two protocols are independent of either clock frequency and are guaranteed to work. Note for proper input synchronization it is necessary to double-buffer the receiver flip-flops for the handshake signals.

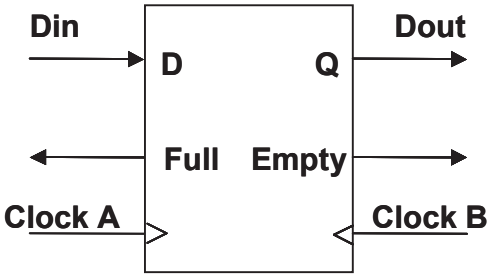


Figure 25. Passing Data Between Clock Domains with a FIFO

Another approach is to use a FIFO as shown in Figure 25. In this case the full and empty flags can be used as handshake signals. Design of asynchronous FIFOs with full arbitration is a difficult task so it is best to use a pre-designed FIFO library element.

7.4 Gated Clocks

Gated clocks are clock signals that include combinational logic in the clock circuit as shown in Figure 26. If not timed exactly right, gated clocks can lead to glitches in the logic or clipping of the clock pulse as shown in Figure 27.

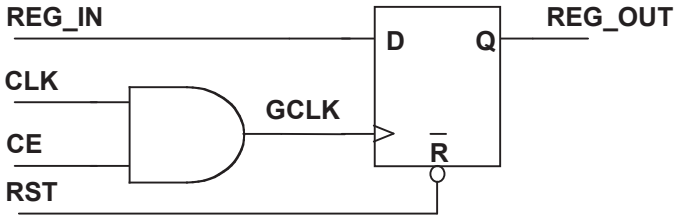


Figure 26. Gated Clock Schematic

Designing gate logic into the data port of a flip-flop versus gating the clock leads to a cleaner, more synchronous design.

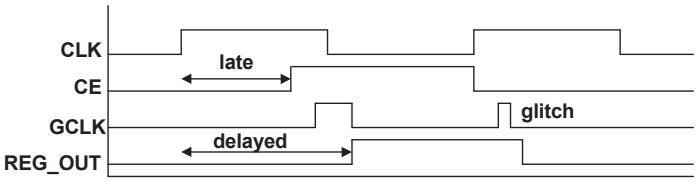


Figure 27. Gated Clock Timing

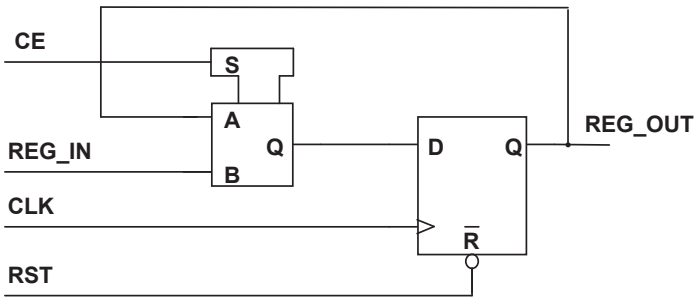


Figure 28. Clock-enabled Flip-flop Schematic

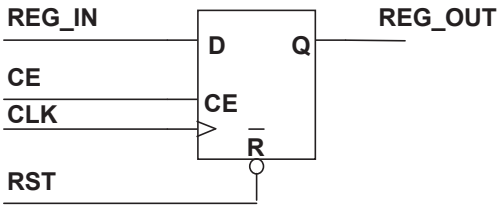


Figure 29. Clock-enabled Flip-flop Symbol

7.5 Finite State Machines

Finite state machines (FSM) are normally designed in a synchronous fashion using binary or one-hot encoding styles. From a portability perspective, the most important FSM design issues are dead (lock-up) states, initialization for testability and synchronizing the FSM inputs to the system clock.

The binary encoded state machine is the most common. One problem with binary encoded FSMs is the possibility of entering a “dead” state. A dead state is a state the machine could enter and not be able to exit from. To avoid dead states in a binary encoded FSM, a default state needs to be defined in the RTL code. That way all undefined states will be able to follow the exit strategy provided by the default state.

For machines with a small number of states, one-hot state machines are a very efficient approach. Essentially, there is one flip-flop for each state. On reset, all flip-flops are reset to “0” except for the initial state flip-flop, which is set to “1”. From then on, only one flip-flop is “hot” at a time. The hot flip-flop represents the state of the machine. Dead states do not exist in one-hot state machines. One-hot state

machine outputs require no decoding and they are very fast. The only potential problem is they do not suppress multiple-ones unless a special recovery circuit is added.

7.6 Latches

Latches should not be used unless absolutely necessary. In most cases a flip-flop will work just as well. When synthesizing designs, be especially careful to avoid accidentally inferring a latch when one is not intended. The problem with latches centers around the transparency issue. In the circuit shown in Figure 30, if Gate A and Gate B were to both go high we might have an oscillator.

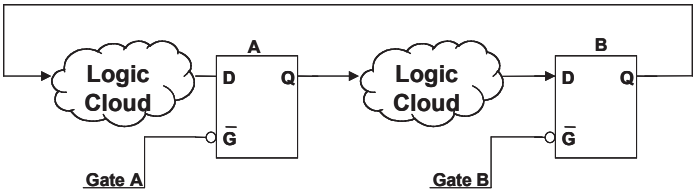


Figure 30. Latch Transparency

Most EDA software tools have difficulty with latches. Static timing analyzers typically make assumptions about latch transparency. If one assumes the latch is transparent, then the tool may find a false timing path through the input data pin. If one assumes the latch is not transparent then the tool may miss a critical path.

Due to the transparency issue, latches are difficult to test. For scan testing they are often replaced by a latch- flip-flop compatible with the scan-test shift-register. Under these conditions a flip-flop would actually be less expensive than a latch.

7.7 Internal Tri-States

The use of internal tri-state buses is discouraged due to testability issues. During an FPGA-to-ASIC conversion, internal tri-states are usually converted to combinational logic.

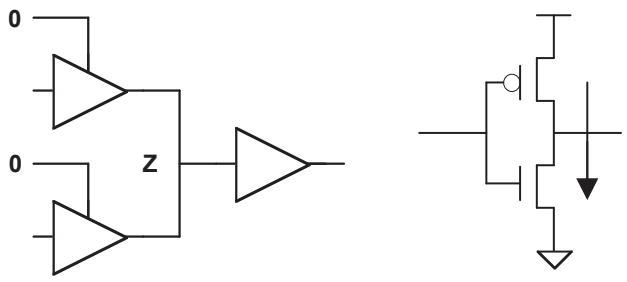


Figure 31. Floating Internal Tri-State Bus

Problems with internal tri-states include the need to ensure that only one driver is turned on. If more than one driver is turned on, bus contention results and high currents can flow. If no drivers are turned on, as shown in Figure 31, the tri-stated signal can float to a metastable state and cause the bus receiver to go into a high current mode. If nothing else, these issues may cause good chips to be rejected by the tester due to high current draw.

When using tri-state buses, be sure to put in a pull-up, pull-down, or bus-latch cell on the tri-state bus to indicate what should happen when the bus is undriven.

7.8 Time Delay

There is really no good way to build reliable and predictable delays in silicon. Designers often attempt to build delays by creating strings of heavily loaded gates, as shown in Figure 32. Their performance is extremely unpredictable and technology-dependent. In addition, they can be very difficult to identify in a design, and even then, the designer's intent may not be very clear.

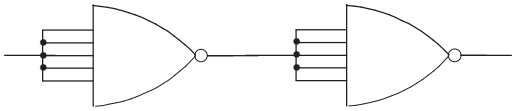


Figure 32. Logic Gates as Delays

Silicon vendors often include delay elements in their cell libraries, as shown in Figure 33. Just because they are delay cells does not mean they are predictable. Their primary purpose is to make it easier to see when and where delays are used by the designer.

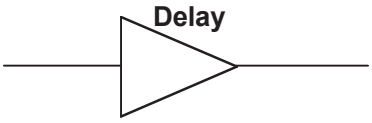


Figure 33. Libray Delay Cell

If delays must be used, a trick for building delay lines with rise/fall symmetry is to use an even number of inverting delay stages each with the same loading conditions, as shown in Figure 34.

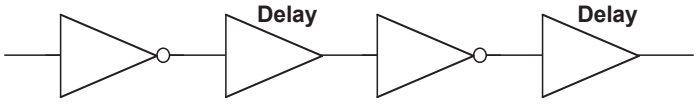


Figure 34. Symmetrical Rise/Fall Delay Circuit

7.9 Pulse Generators

Asynchronous pulse generators, such as the one shown in Figure 35, are difficult to build because of problems in controlling the pulse width. The pulse may be too wide or it may be too narrow and disappear completely as shown in Figure 36. Pulse generators have all the problems associated with delays, such as being unpredictable and technology dependent.

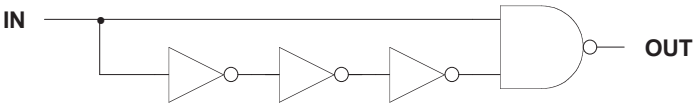


Figure 35. One-Shot Pulse Generator Circuit

Figure 36. Pulse Generator Timing

Figure 37 shows a better way to make a pulse generator using a synchronous circuit. It generates pulses one clock cycle wide, as shown in Figure 38.

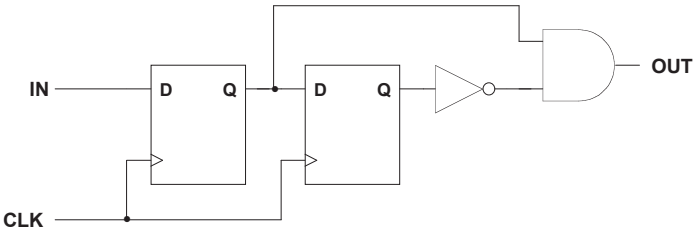


Figure 37. Synchronous Pulse Generator Circuit

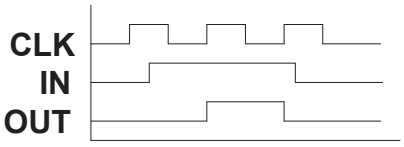


Figure 38. Synchronous Pulse Generator Timing

7.10 Direct Action I/Os

Figure 39 shows a design similar to the glitching gated clock design. This case is one of a direct action signal that comes in through I/O ports. This represents an implicit multiplexer and there is not a good way to make sure the signal is glitch-free. This circuit is technology-dependent and extremely difficult to test. Clock signals should have their own dedicated input pin.

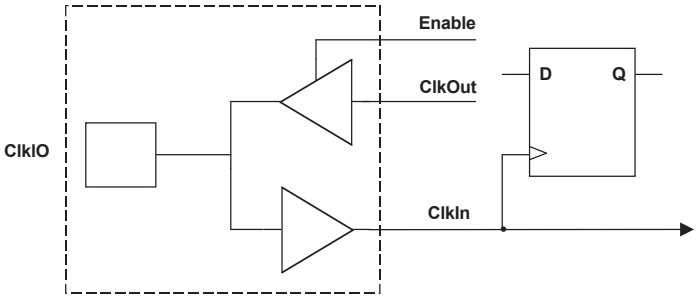


Figure 39. Direct Action I/O Signals

8.0 Conversion Checklist

Below is a short checklist of items to consider when designing an FPGA for conversion:

1. Engage early with ASIC vendor
2. Plan ahead on packaging
3. Select a JTAG approach
4. Make the core supply voltage flexible
5. Consider converting multiple devices into an ASIC
6. Document, document, document
7. Include margin in the system timing budget
8. Don’t use proprietary IP
9. Use soft third party IP
10. Avoid FPGA configuration dependencies
11. Explicitly reset all sequential elements
12. Pay attention to differences in I/O characteristics
13. Check SSO rules
14. Use synchronous design
15. Design cross clock domain interfaces carefully
16. Avoid gated clocks
17. Avoid dead states in state-machines
18. Avoid latches

Sales and Design Assistance from ON Semiconductor

ON Semiconductor Distribution Partners

Allied Electronics	www.alliedelec.com	(800) 433-5700
Arrow Electronics	www.arrow.com	(800) 777-2776
Avnet	www.em.avnet.com	(800) 332-8638
Chip One Stop, Inc.	www.chip1stop.com/maker/on	(81) 45 470 8771
Daiwa Distribution Ltd.	www.daiwahk.com	(852) 2341 3351
Digi-Key	www.digikey.com	(800) 344-4539
EBV Elektronik	www.ebv.com/en/locations.html	(49) 8121 774-0
Fuji Electronics Co., Ltd.	www.fujiele.co.jp	(81) 3 3814 1770
Future & FAI Electronics	www.futureelectronics.com/contact	1-800-FUTURE1 (388-8731)
KH Electronics Inc.	www.khelec.com/kor	(82) 42 471 8521
Mitsui Electronics Inc.	www.btel.co.jp	(81) 3 6403 5900
Mouser Electronics	www.mouser.com	(800) 346-6873
Newark/Farnell	www.farnell.com/onsemi	(800) 4-NEWARK
OS Electronics Co., Ltd.	www.oselec.jp	Japanese: (81) 3 3255 5985 Other Languages: (81) 3 3255 6066
Promate Electronic Co.	www.promate.com.tw	(886) 2 2659 0303
RS Components KK	jp.rs-online.com	(81) 45 335 8550
Segyung Britestone Co.	www.britestone.com	(82) 2 3218 1511
Serial AMSC	www.serialsystem.jp	(81) 3 5302 1569
Serial Microelectronics, HK	www.serialsys.com.hk	(852) 2790 8220
Taewon Inc.	www.taewon.net	(82) 2 6679 9000
World Peace Industries Co.	www.wpi-group.com	(852) 2365 4860
WT Microelectronics Co.	www.wtmec.com	(852) 2950 0820
Yosun Electronics	www.yosun.com.tw	(886) 2 2659 8168

INTERNATIONAL

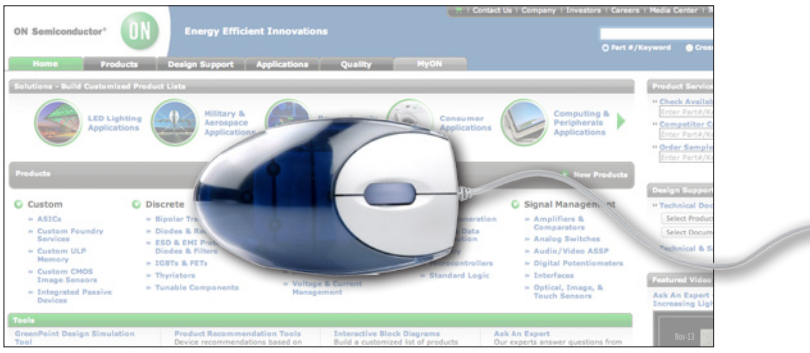
GREATER CHINA	Beijing	86-10-8577-8200
	Hong Kong	852-2689-0088
	Shenzhen	86-755-8209-1128
	Shanghai	86-21-5131-7168
	Taipei, Taiwan	886-2-2377-9911
FRANCE	Paris	33 (0)1 39-26-41-00
GERMANY	Munich	49 (0) 89-93-0808-0
INDIA	Bangalore	91-98-808-86706
ISRAEL	Raanana	972 (0) 9-9609-111
ITALY	Milan	39 02 9239311
JAPAN	Tokyo	81-3-5817-1050
KOREA	Seoul	82-2-2190-3500
MALAYSIA	Penang	60-4-6463877
SINGAPORE	Singapore	65-6484-8603
SLOVAKIA	Piestany	421 33 790 2450
UNITED KINGDOM	Slough	44 (0) 1753 70 1676

AMERICAS REP FIRMS

Alabama	Huntsville	e-Components	(256) 533-2444
Brazil	Countrywide	Ammon & Rizos	(+55) 11-4688-1960
California	Bay Area	Electec	(408) 496-0706
	Southern California	Tech Coast Sales	(949) 305-6869
Canada	Eastern Canada	Astec	(905) 607-1444
Connecticut	Statewide	Paragon Electronic Systems	(603) 645-7630
Florida	Statewide	e-Components	(888) 468-2444
Georgia	Atlanta	e-Components	(888) 468-2444
Illinois	Hoffman Estates	Stan Clothier Company	(847) 781-4010
Indiana	Fishers	Bear VAI	(317) 570-0707
Kansas	Overland Park	Stan Clothier Company	(913) 894-1675
Maine	Statewide	Paragon Electronic Systems	(603) 645-7630
Maryland	Columbia	Third Wave Solutions	(410) 290-5990
Massachusetts	Statewide	Paragon Electronic Systems	(603) 645-7630
Mexico	Countrywide	Ammon & Rizos	(+55) 11-4688-1960
Michigan	St. Joseph	Bear VAI	(440) 526-1991
Minnesota	Eden Prairie	Stan Clothier Company	(952) 944-3456
Missouri	St. Charles	Stan Clothier Company	(636) 916-3777
New Hampshire	Statewide	Paragon Electronic Systems	(603) 645-7630
New Jersey	Statewide	S.J. Metro	(516) 942-3232
New York	Binghamton	TriTech - Full Line Rep	(607) 722-3580
	Jericho	S.J. Metro	(516) 942-3232
	Rochester	TriTech - Full Line Rep	(585) 385-6500
North Carolina	Raleigh	e-Components	(888) 468-2444
Ohio	Brecksville	Bear VAI Technology	(440) 526-1991
Puerto Rico	Countrywide	e-Components	(888) 468-2444
Rhode Island	Statewide	Paragon Electronic Systems	(603) 645-7630
Vermont	Statewide	Paragon Electronic Systems	(603) 645-7630
Wisconsin	Evansville	Stan Clothier Company	(608) 882-0686
	Oconomowoc	Stan Clothier Company	(608) 882-0686

For a comprehensive listing of ON Semiconductor Sales Offices, please visit:

www.onsemi.com/salessupport



PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303-675-2175 or 800-344-3860 Toll Free USA/Canada
Fax: 303-675-2176 or 800-344-3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800-282-9855 Toll Free USA/Canada.
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81-3-5817-1050

ON Semiconductor Website: **www.onsemi.com**
Order Literature: http://www.onsemi.com/orderlit
For additional information, please contact your local Sales Representative