HES-7 ASIC Prototyping

Rev. 1.9

September 14, 2012

Co-authored by:

Slawek Grabowski and Zibi Zalewski, Aldec, Inc.

Kirk Saban, Xilinx, Inc.

Abstract

This paper highlights possibilities of ASIC verification using FPGA-based prototyping, considering the latest Virtex®-7 devices and Aldec HES-7 dual Virtex-7 2000T FPGA ASIC prototyping board. In addition, the most common partitioning issues and resolutions are described.

Table of Contents

New opportunities for ASIC prototyping

In the current verification world no one doubts that ASIC prototyping is beneficial to the ASIC delivery process, and the most common way to do that is through the usage of constantly-improving FPGA technology. This does not mean it is an easy and straightforward process, though, especially with the most common sizes of ASIC (under prototype) being 10-20 million gates. Considering currently available FPGAs like Xilinx® Virtex-6 LX760 devices it is necessary to use prototyping boards that contain six or more devices. However, using multiple FPGAs introduces challenges, as the ASIC design needs to be partitioned and shared. That introduces new problems, such as managing timing constraints, inter-FPGA connectivity congestion and clocking problems, even proper FPGA resources estimation becomes difficult when trying to determine how many FPGAs are required on the prototyping platform.

Fortunately there is a new player in the game – Xilinx introduced its mammoth Virtex-7 2000T device which can accommodate up to 12M ASIC, not including DSP and memory resources. Prototyping boards with one or two Virtex-7 2000T FPGAs can therefore accommodate most ASIC projects, certainly in terms of shear capacity.

Partitioning

When design partitioning, there are two choices: manual partitioning done by the engineer (on the RTL sources) and semi- or fully-automatic partitioning performed by EDA tools (on both the RTL source and the design netlist). See figure 1.

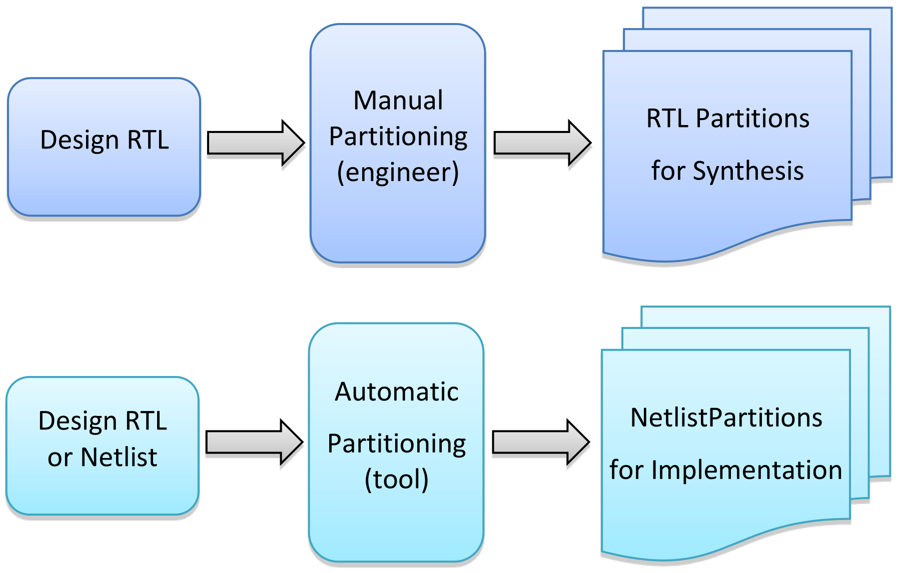

Design RTL
Manual Partitioning (engineer)
RTL Partitions for Synthesis
Design RTL or Netlist
Automatic Partitioning (tool)
NetlistPartitions for Implementation

Figure 1: Common partitioning options used in ASIC prototyping.

In cases where an ASIC design needs to be partitioned and shared between six or more FPGAs, manual partitioning requires tremendous effort and frequently results in poor speed (of the design under test) and functionality problems. In such cases the only solution is to use the EDA partitioning tools available on the market. Of course, even for those tools the problem increases when the design size grows - which means the number of FPGAs also increases, further complicating the partitioning process.

Clearly, the fewer partitions (i.e. FPGAs) the easier the implementation of the ASIC prototyping will be. The Virtex-7 2000T FPGA, with its 12M ASIC capacity, should accommodate most designs. A single Virtex-7 FPGA will be ideal for small- to medium-size designs; and all that's required for the prototyping is the Virtex-7 FPGA board and Xilinx Vivado™ Design Suite. For bigger designs, two Virtex-7 devices should suffice and partitioning is greatly simplified, as there is only a single partition with which to contend.

Assuming multiple FPGA devices are employed for ASIC prototyping the engineer must:

1. Balance FPGA resources while creating design partitions
2. Place and route the partitions on the FPGA board
3. Meet timing requirements
4. Be able to debug of the prototyped design

These problems grow in severity with the number of FPGAs. We will discuss in the next sections how to divide the problems into smaller ones that are easier to solve. However, the complexity to create an optimal prototype using a multi-FPGA platform is very high. Most users settle with a suboptimal solution for the above problems because finding a better solution requires too much effort. It is essential to use the largest FPGA devices and avoid the complexity that rapidly grows when using multiple small FPGAs for prototyping.

Balancing FPGA resources while creating design partitions

Balancing FPGA resources between multiple partitions is the most basic problem that must be solved during partitioning. In order to find the optimal solution an engineer should:

- Observe different FPGA resources and keep them balanced in multiple partitions:
    a. Look Up Tables (LUTs)
    b. Registers,
    c. Block RAMs
    d. DSP blocks
    e. Clock buffers
- Map large memories to on-board RAM, usually to DDR2/DDR3 RAM.
- Perform global logic optimization to remove redundant or unused logic

Resource balancing is also a part of these partitions during placing and routing. Frequently, while trying to find the ideal balance between partitions we run into routing problems. Optimally balanced partitions usually do not have optimal interfaces and require a high number of PCB tracks. Going further, we will run also into timing issues if routing is not optimal. In conclusion, the partitioning, partition placement and routing, and timing problems cannot be really solved in separation. Focusing on one of these problems may lead to trouble with the others.

It is essential to map big memories to on-board DDR2/DDR3 devices. Offloading FPGAs with some memories always helps with partitioning. However, it does not help too much with placing and routing. Each FPGA platform has limited DDR2/DDR3 resources and usually the memories are connected only to selected FPGAs on the board. Mapping a memory to on-board RAM also requires the placement of its associated logic close to the memory. This leads to additional considerations for placing and routing.

Most of the partitioning tools on the market try to find an optimal solution by following design structure. Partitions are created from instances of different modules. However, separating some modules may lead to poor logic optimization and speed that is too low to run the prototype efficiently. Some redundant or unused logic may not be removed and complicate partitioning and resources balancing. This reinforces the message that the larger the device the better.

Placing and routing partitions on FPGA board

Prototyping engineers usually pay lots of attention to FPGA resource balancing during partitioning and run into trouble with partitions placing and routing. It is much better not to balance FPGA resources perfectly so as avoid problems with partitions placing and routing. In order to achieve an optimal solution for place and route engineers should:

- Observe and try not to exceed FPGA pin count in partitions interface
- If the partition interface exceeds the FPGA pin count, try to keep the lowest multiplexing ratio
- Find shared wires and avoid routing them multiple times
- Pay attention to constraints applied by DDR2/DDR3 memory locations
- Avoid routing signals without source or without load
- Do not route constant signals between partitions

It is very difficult not to exceed an FPGA's pin count while mapping partitions, especially for big ASIC designs. In most cases pin multiplexing is required. Multiplexers are used to share the same connection in time between multiple signals. Transferring multiple signals over the same connection usually requires design clock frequencies to be reduced, typically in accordance with the multiplexing ratio. It is essential to keep the multiplexing ratio as small as possible and avoid performance problems caused by reduced clock frequencies.

A partition is usually created from multiple design modules grouped together. It is difficult to find a single module instance that fits perfectly to given partition resources. Some designs are implemented to be partition-friendly and engineers intentionally create hierarchy that corresponds with FPGA platform architecture. However, design hierarchy usually is created according to design functionality and does not fit well to the architecture of any prototyping board. In such cases, multiple modules must be grouped to create the partition. Frequently, there is a group of the same signals connected to several modules. It is essential to find such signals and avoid routing them multiple times between partitions; which may increase the multiplexing ratio and reduce performance.

As mentioned in the previous section, partitions that contain modules mapped to on-board DDR2/DDR3 RAM must be placed in the FPGAs that have access to the required memory. This is an additional constraint for partition place and route because such partitions can be placed only in specific FPGA chips and cannot be moved to other chips where memory is not available.

Separating design modules and juggling with them to create partitions may again lead to optimization problems. During routing the most important are constant signals or source/load less signals. They should not be routed between partitions and consume critical routing resources.

Meeting required timing

The prototype must meet timing and performance requirements. Most of the timing issues that must be solved in FPGA prototype are concerned with building the clock tree for the partitioned design and managing the timing of FPGA interconnections. In order to achieve a high performance solution engineers should:

- Ensure low clock skew between multiple FPGAs
- Control the number of clocks and use dedicated resources for clocking
- Avoid gated clocks
- Group logic with the same clock
- Isolate clock generators and clock buffers to enable replication
- Avoid combinatorial paths between FPGAs
- Provide timing constraints for the FPGAs' interconnections

All clocks must be distributed using dedicated resources on the board. Clock lines are the critical resource on each FPGA board. Especially when the number of global clock lines is limited. All internal clocks should be implemented inside the FPGA devices. Gated clocks are not recommended in FPGAs and should be converted to clock enables.

Routing internal clocks between FPGA devices is usually not possible because of the lack of sufficient clock lines and clock buffers. This is why logic clocked by an internal clock should be grouped and placed inside a single FPGA device. If the logic is too large and must be partitioned between multiple devices, it is better to replicate the clock generators in each FPGA device to avoid routing the clocks over PCB. The best solution is to distribute global clocks only to each FPGA device using dedicated clock lines and replicate local clock generators in each FPGA to avoid routing any additional clocks between FPGAs.

Similarly it is not recommended to route any clock domain crossings between FPGAs. All signals crossing clock domains should be implemented inside FPGA device.

Signals routed between FPGA devices should be registered. Partitioning big combinatorial logic between FPGAs complicates constraining and may result in poor performance. Each path routed between FPGAs should be properly constrained in each FPGA to make sure that it will work at its required speed.

Debugging of prototyped design

The debugging of high-speed ASIC prototypes is the most challenging task. Usually, hardware engineers create debug probes by connecting some essential signals to test points or external connectors available on a given board. In such instances, an external logic analyzer is used to capture the states of all the test points. Built-in checkers are also very popular. Analyzing waveforms captured by a logic analyzer is tedious and time-consuming. Frequently, design engineers add probe points or assertions to the code that can be synthesized to hardware. The embedded logic analyzers insert simple test status information and do not require analysis of complicated waveforms. The FPGA and EDA vendors also provide these embedded logic analyzers, such as ChipScope™ analyzer from Xilinx or Identify from Synopsys that can be inserted into the FPGA devices. These embedded logic analyzers are easier to use but usually have limited memory capacity and cannot capture longer waveforms.

Engineers should be prepared for bug fixes and ECO changes too. Bug fixes or ECOs may complicate prototyping if the design hierarchy is changed or the interfaces of some modules are significantly increased to implement the bug fix. This may lead to FPGA resource balance problems or partitions place and route problems. Bug fixes, if possible, should be done locally without changes in design hierarchy and interfaces. In such cases only one FPGA chip needs to be re-implemented. Discovering and fixing bugs on multi-FPGA prototypes is a daunting task, but help is on the way.

In the following section we present a design example where the use of the Virtex-7 2000T device shows how much easier and better ASIC prototyping has become.

Xilinx design example

Xilinx leveraged ASIC prototyping and emulation methodology internally with the creation of the Zynq™-7000 All Programmable SoC emulation platform. This platform was used to emulate the first Zynq-7000 device prior to silicon tape-out. Utilizing a combination of four Virtex-5 LX330T and two Virtex-6 LX240T FPGAs, the Zynq-7000 All Programmable SoC emulation platform enabled Xilinx and its ecosystem partners and customers to get a jumpstart on IP and SW development. Xilinx has since ported the Zynq-7000 All Programmable SoC emulation platform to a single Virtex-7 2000T device. The entire design occupies less than half of the Virtex-7 2000T devices and enables a 5X increase in performance, a 6X reduction in devices required, and an 80% reduction in total system power.

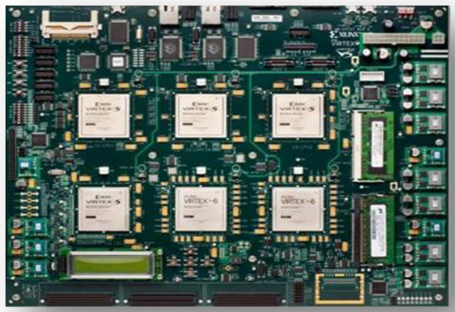
The single chip Virtex-7 2000T FPGA implementation also resulted in dramatic improvements in productivity. The design no longer needs to be partitioned across six FPGAs; the entire RTL for the full design can be implemented in the Virtex-7 2000T FPGA. This eliminates the need for IO multiplexing and dramatically reduces time to market for end users. The implementation of the Zynq-7000 All Programmable SoC on the Virtex-7 2000T FPGA utilized the new Vivado Design Suite from Xilinx. With the new analytical place and route engine in the Vivado Design Suite, no manual placement constraints were required to achieve 50MHz performance across the device. See figure 2 below:

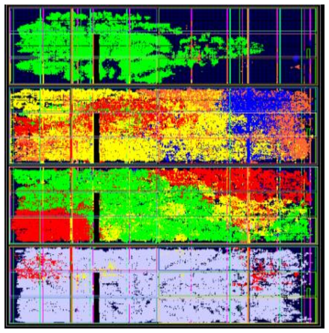
Zynq-7000 All Programmable SoC

› 6 FPGAs (4x V5LX330, 2x V6LX240T)
› Weeks of laborious partitioning effort
› Emulates Cortex-A9MP @ 10MHz

All 6 FPGAs combined into one Virtex-7 FPGA

› No manual design partitioning
› ARM processor <u>increased performance by 5x</u> from 10 MHz to 50 MHz
› Utilization: 48% LUTS, 34% BRAM, 52% IOs





Stacked silicon interconnect

Over the past five years Xilinx has worked with a number of partners in packaging and related technologies to provide a mature and reliable manufacturing flow for 3D FPGAs. Microbumps have been used for years in the image array space and are very reliable. Nearly every smart phone or cell phone includes the technology to integrate the camera. These microbumps are used to link each FPGA slice (power, ground, IOs) to the interposer. Through-Silicon-Vias or TSV are used to transmit power, ground, and IO signals from each FPGA slice though tens of thousands of routes in the interposer.

The 65nm passive silicon interposer contains no active components, which increases the reliability of both the interposer and the TSVs. Due to the fact that the interposer is also manufactured from silicon, it provides thermal conductivity and helps to mitigate package stress.

The side-by-side layout of the FPGA die allows for efficient heat dissipation and also reduces warping. The layout maintains a conventional FPGA topology that minimizes the impact to software place and route algorithms. All these technologies individually have been proven to be very reliable and their application in stacked silicon interconnect (SSI) is equally as reliable The resulting Virtex-7 2000T device, the first FPGA using SSI technology has been shipping since October 2011.

Breaking through

ASIC prototyping with FPGAs enables fast and accurate SoC system modeling and verification as well as accelerated software and firmware development. Xilinx takes FPGA-based prototyping one step further with the introduction of the Virtex-7 2000T device which allows end users to break through multi-chip partitioning barriers with the capacity and performance needed to prototype complex ASICs in a single FPGA.

Conclusion

Implementing a high-speed prototype using multiple FPGAs is very challenging and requires an engineer to find optimal solutions to several technical problems. Anyone who had to deal with multi-FPGA prototyping can confirm that. To help ease this difficulty, it is essential to use a prototyping board with the largest possible FPGA devices. The number of FPGAs determines the complexity of the prototype implementation and debug. Previously, using fewer FPGAs was impossible due to FPGA capacity, at the time, but with the latest Virtex-7 2000T device most ASIC designs can be accommodated with one or two devices; which will make the design bring-up less painful and much quicker. To accommodate, that Aldec provides the HES-7 product with the largest Virtex-7 FPGA devices. The next section provides more information about the main features of HES-7.

Aldec HES-7 ASIC prototyping board – key features

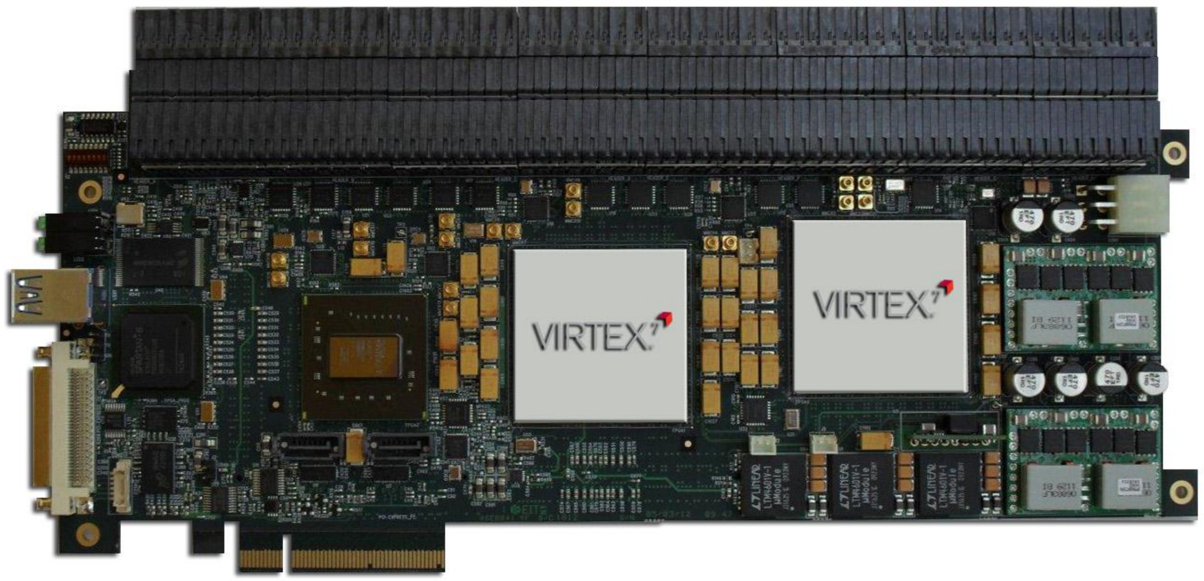


Figure 2: HES-7 ASIC Prototyping board.

Configuration	Capacity (Up To)	FPGAs	FPGA I/O Connectivity	
			720 I/O/360 LVDS	**44x GTX (6.6 Gbps)**
HES-7 XV690	4m ASIC Gates	1 XC7VX690T	240/120 FPGA1	16 FPGA1
HES-7 XV1380	8m ASIC Gates	2 XC7VX690T	240/120 FPGA1 480/240 FPGA2	16 GTX FPGA1 28 GTX FPGA2
HES-7 XV2000	12m ASIC Gates	1 XC7V2000T	240/120 FPGA1	16 GTX FPGA1
HES-7 XV4000	24m ASIC Gates	2 XC7V2000T	240/120 FPGA1 480/120 FPGA2	16 GTX FPGA1 28 GTX FPGA2
HES-7 Backplane	96m ASIC Gates	8 XC7V2000T	240/120 FPGA1 (3/5/7) 480/240 FPGA2 (4,6,8)	16 GTX FPGA1 (3/5/7) 28 GTX FPGA2 (4/6/8)

I/O Connectivity (Cont.):

- 322 high-speed inter-FPGA connections
  - FPGA0 ↔ FPGA1: 96/48 (SE/Diff)
  - FPGA1 ↔ FPGA2: 226/113 (SE/Diff)
- High-speed 25 Gbps connectors for expansion
  - Backplane and daughterboard interface
- 5 Global clock inputs connected to all FPGAs
- 89 special purpose global lines to all FPGAs
  - For programming, expanded debug, etc.

Customizable Clocking:

- 5 customizable, low skew, global clocks from backplane
    - User programmable PLLs (2KHz to 1.4GHz)
    - MMCX connectors for external clock inputs (0 - 450MHz)
- 60 differential local clock inputs from backplane
    - 20 multi-region and single region differential pairs to FPGA1
    - 40 multi-region and single region differential pairs to FPGA2

Memory:

- Up to 16GB of DDR3 memory supported with SO-DIMM sockets
- UP to 64GB of DDR3 with 4 HES-7 boards connected through backplane
- 8GB Flash memory connected to FPGA0
- Micro SD card socket available for stand-alone configuration

Protocol Support:

- PCI-Express finger connector for inside PC operation
- PCI-Express cable connector for outside PC operation
- USB 2.0/3.0 connector
- JTAG connector
- 2 SATA connectors (Device/Host)

Diagnostics:

- Self -diagnostic button and status LEDs for hardware diagnostic

Daughterboard Support:

- Leverages non-proprietary 25 Gbps backplane connectors
- Technical specification on the interface available for daughterboard development

About Aldec, Inc.