cadence®

ASIC Prototyping Simplified

To use current solutions for application-specific integrated circuit (ASIC) prototyping using field-programmable gate arrays (FPGAs), you either have to create custom boards or buy off-the-shelf FPGA boards. Off-the-shelf boards don't satisfy the requirements for complex systems on chips (SoCs), and they're expensive and lack scalability. Cadence® Allegro® FPGA System Planner fills the gap, offering a simplified approach to ASIC prototyping.

Contents

Introduction..... 1

Design Flow for
ASIC Prototyping with FPGAs..... 2

Design Flow with
Allegro FPGA System Planner..... 3

Design Requirements..... 5

Designing with
Allegro FPGA System Planner..... 6

Conclusion..... 11

Introduction

With the cost of ASIC design being what it is, you can hardly afford to have any mistakes in your ASIC/ASSP. The need for good verification is hence paramount. Hardware-based verification platforms have been used ever since the advent of ASIC design. More recently, FPGA-based systems have started to address this need. However, off-the-shelf FPGA-based systems are not ideal for a variety of reasons:

1. **Cost**. If you require one prototype board with a limited number of FPGAs, you may be able to justify the cost. However, in a real project, you always require multiple systems for different groups to use. The cost of off-the-shelf FPGA boards becomes prohibitively high

2. **Feasibility**. You would like to prototype your entire SoC, or at least prototype the newer/critical portions of it. Such a prototype is difficult to create using off-the-shelf boards. Mapping your design into off-the-shelf boards can be quite a challenge, especially if your design has wide busses. Mapping a complex design onto off-the-shelf boards can reduce the frequency of operation significantly

3. **Performance**. You would like to prototype your system and test it out at or near the required frequency of operation. With off-the-shelf boards, your frequency is limited

Off-the-shelf boards provide an expensive, partial solution that doesn't enable you to prototype your design and test it fully.

Growing SoC complexity, an increasing amount of embedded software, and the high cost of emulation systems have driven the need for custom prototyping. However, there is significant complexity of designing boards with multiple FPGAs.

Cadence® Allegro® FPGA System Planner was designed to overcome the challenges with multi-FPGA board design. It is designed to be used in combination with commercial tools available for RTL partitioning, FPGA design, and

board design. Allegro FPGA System Planner has been used in several ASIC prototyping designs successfully. It has been found to double or triple the productivity and cut the overall schedule in half. In this application note, we will walk you through a complete FPGA board for ASIC prototyping.

Design Flow for ASIC Prototyping with FPGAs

Figure 1 shows a typical flow for prototyping ASICs using FPGAs.

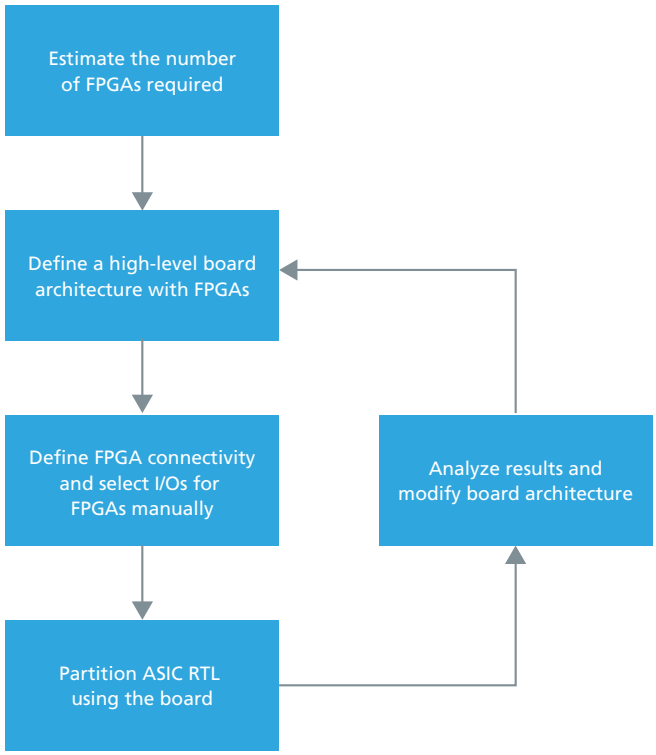

Figure 1: Typical design flow for ASIC protoyping using FPGAs

The process starts by identifying the number of FPGAs required for prototyping. This can be done in several ways:

1. Start with an estimate of the FPGAs. This can be a rough estimate, for example, based on the number of functional blocks that you are integrating to create the SoC. It may be a good idea to separate each functional block into an FPGA or a set of FPGAs
2. Alternately, you can use a synthesis tool to synthesize the ASIC RTL to calculate the equivalent FPGA resources required

Once you identify the FPGAs required, you will need to estimate the number of connections that are required between the different FPGAs. Again, if you use the approach of prototyping each functional block of your SoC in an FPGA, the connections between different functional blocks will be the connections between the FPGAs on the board.

You will then need to create a board Verilog netlist defining all the FPGAs and the connections between the different FPGAs.

Using all of this information, you can start the process of ASIC partitioning into FPGAs. To partition the ASIC/SoC RTL, you will need to use a synthesis tool. Of course, you can partition the RTL manually as well. You may need to add or delete FPGAs and connections between FPGAs to enable satisfactory partitioning.

The key difficulty in the process described above is in creating the multi-FPGA PCB, and iterating between the board architecture and RTL partitioning tool to design a board that works for your ASIC/SoC.

Design Flow with Allegro FPGA System Planner

Allegro FPGA System Planner enables you to simplify this whole process of multi-FPGA board design significantly. Figure 2 shows a flow for ASIC partitioning using FPGA System Planner.

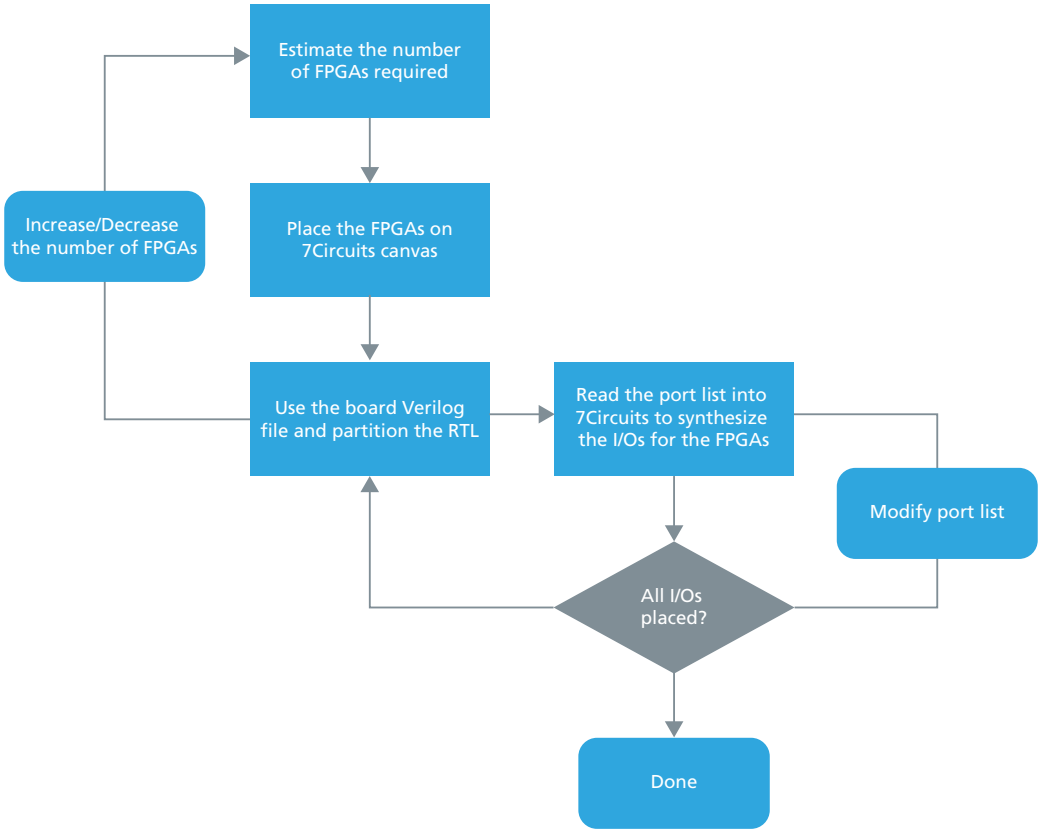

Figure 2: ASIC partitioning flow using FPGA System Planner

Allegro FPGA System Planner automates several steps in the process. Below is the process that you can use for ASIC prototyping with FPGA System Planner in the flow:

1. Estimate the FPGAs required for partitioning your RTL. While estimating the number of FPGAs, keep the following factors in mind:
   - The ASIC gate count that you want to implement in the prototype, translated into equivalent FPGA LUTs. A rough estimate of the FPGA LUTs required for implementing ASIC gates is as follows:
     i. 4 input LUT = 10 ASIC gates
     ii. 6 input LUT = 15 ASIC gates
   - Memory requirement. Separate out the required memory into FPGA internal memory vs. external memory. Also, separate the requirement into external and internal SRAM, SDRAM, and Flash
   - Identify the resources that can be implemented in FPGA hard IP, such as DSP blocks. Note that while functions such as DSPs can be implemented using LUTs, it can become very expensive with the number of LUTs required
   - If you are using processors, such as an ARM® processor, in the design, you may want to consider an off-the-shelf part that can implement this function, and integrate that with the FPGAs
   - Ensure you have sufficient I/Os for interconnecting the FPGAs. The FPGA I/Os or board traces connecting these I/Os represent the wires in the top level of your partitioned RTL

2. Next, place the FPGAs on the Allegro FPGA System Planner canvas, which will allow you to make some architectural tradeoffs. Generate a top-level Verilog® board file with the FPGAs. You can now drive the partitioning tools:
    - Roughly figure out the size of the board you will require
    - Identify a placement that will make sense for the board
3. Now, use an ASIC partitioning tool to ensure that the RTL can be partitioned within the selected number of FPGAs. Some factors to consider here:
    - It's a good idea to keep your FPGA utilization low, say in the range of 50-70%. You may want to keep it lower if you expect your RTL functionality to grow
    - If you plan to use on-chip debug tools, such as Xilinx ChipScope or Altera SignalTap, remember to reduce your SRAM and logic utilization correspondingly
    - Another reason to keep the utilization low is to use your FPGA board for the next generation of your product
4. Next, use the top level of the partitioned RTL to define the connections required among the different FPGAs
5. Allegro FPGA System Planner can now synthesize the design to translate the nets in the top-level RTL and create I/Os for the FPGAs while considering all the aspects for the design:
    - The physical placement of FPGAs on the board
    - The FPGA I/O DRC requirements
    - The logical requirement for the busses between the FGPAs
6. Once you complete the I/O synthesis in Allegro FPGA System Planner, you can generate the Verilog board connectivity file. You may need to iterate with the I/O definitions to ensure that all the I/Os can be placed on the different FPGAs
7. Use this connectivity file once again in the RTL partitioning tool. The RTL partitioning tool can now map the wires in the top level to ports/traces on the board
8. In some cases, it may be required to iterate a bit to get to successful results. Analyze results of the partitioning tools and make the required changes to the FPGA board architecture. You may also want to increase the bus widths to increase the available connectivity
9. Iterate between the steps defined in (2) through (8) to ensure you design a board that can be fully utilized for your prototyping requirement
10. Once you have a board partition that meets the requirements, you can proceed to the next steps in board design. Allegro FPGA System Planner enables you to quickly:
    - Create all the required power supply connections automatically. This is done in a few mouse clicks, and it ensures that your design is going to have correct power supplies connected
    - Generate all the configuration connections required for the FPGAs to be programmed
    - Automatically generate schematics for the most popular board design tools
    - Add the required decoupling capacitors and terminations
    - Take your design into layout. (Allegro FPGA System Planner can read back the layout files from Allegro PCB tools to re-optimize any nets to make the layout easier)

In the next few sections, we will go through a real prototype design example using Allegro FPGA System Planner.

Design Requirements

Consider an SoC designed for a next-generation mobile phone. Typical functional blocks for such an SoC are shown in Figure 3.

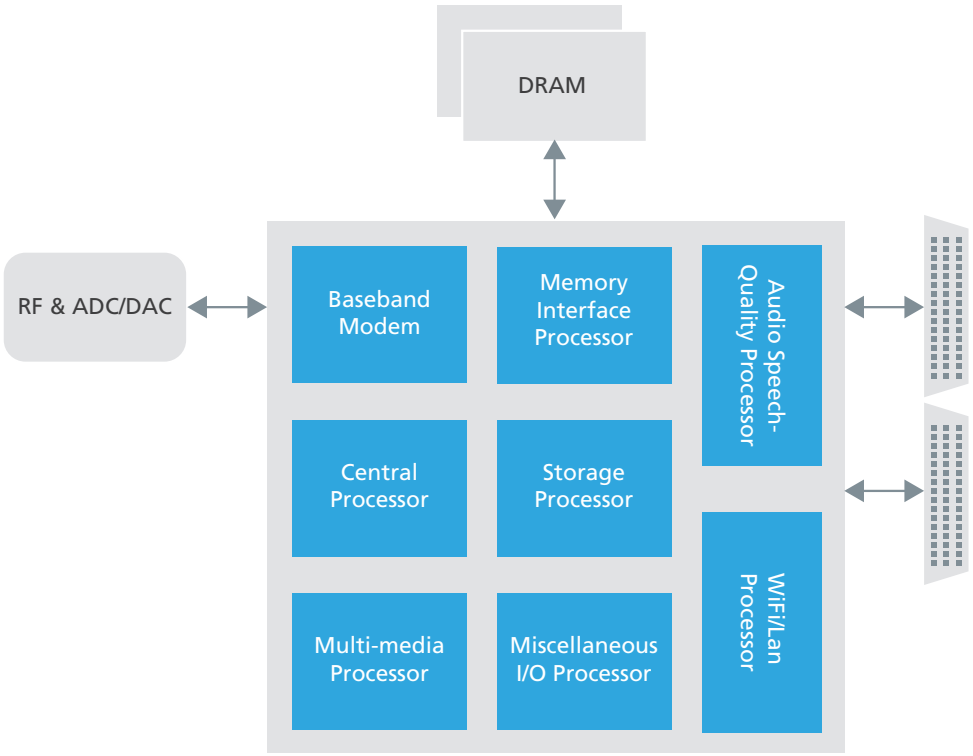

Figure 3: Functional blocks of a mobile phone SoC

For the sake of simplicity, we will illustrate a design with the baseband processor and the central processor. Let us assume that you have determined the following requirements for each of the functional blocks:

1. Control plane busses
   - A 32-bit bus between the central processing unit and all the other blocks
2. Baseband modem
   - Six of the largest available FPGAs
   - Connections among the FPGAs should be the maximum possible, since it is hard to determine how the partitioning is going to work. You would like the board to have the maximum possible connections, so that the partitioning tools can easily do the partitioning
   - An 8-bit data bus and a small control bus between the baseband modem functional block and the audio processor
   - A connector to interface between RF and baseband
3. Central processor
   - Assume four large FPGAs
   - Again, the maximum possible connections among the four FPGAs
   - Control plane bus to all the other functional blocks

Designing with Allegro FPGA System Planner

Now, let's map the high-level design architecture we described in the previous section into FPGAs. The steps are as follows:

1. Open Allegro FPGA System Planner and place your favorite large FPGAs on the canvas
2. Place the FPGAs on the board and create a Verilog file with the FPGA definitions. You would use this file in the partitioning tool
3. After partitioning and identifying the busses required among different FPGAs, define them concisely in Allegro FPGA System Planner
4. Run the Allegro FPGA System Planner I/O synthesis engine to generate connectivity
5. Output the board Verilog file, individual FPGA Verilog files, and constraint files
6. Verify that design requirements are met by using this Verilog file in the partitioning tool
7. Modify the design to meet the partitioning requirements

Allegro FPGA system planner

Allegro FPGA System Planner is a tool designed to simplify multi-FPGA board design. It generates outputs that enable you to iterate with your partitioning tool.

Identifying and placing the FPGAs on the board

Allegro FPGA System Planner has a pre-defined library of all the popular FPGAs that are typically used for prototyping, including Xilinx and Altera FPGAs. A screenshot of Allegro FPGA System Planner with the six FPGAs for baseband and four FPGAs for the CPU is shown in Figure 4. Below is the key design criteria we used:

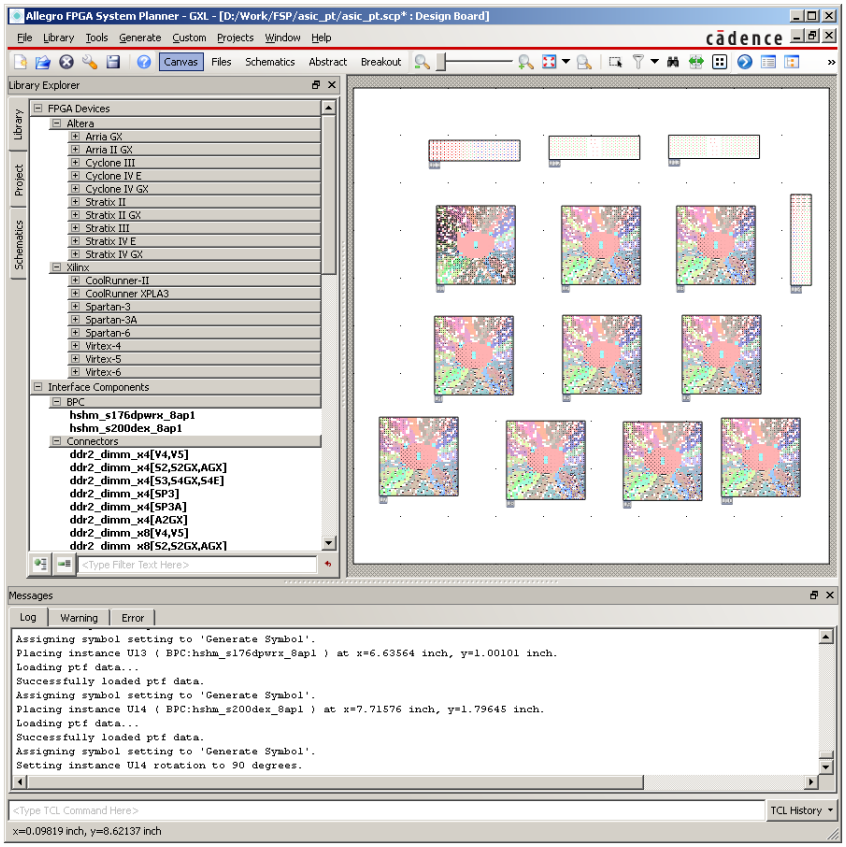

Figure 4: Allegro FPGA System Planner with 10 FPGAs placed on the board

Create a 12"x12" board in Allegro FPGA System Planner

8. Place 10 of the largest Xilinx Virtex-5 FPGAs on the board – 5VLX330FF1760 devices, 6 for the baseband modem and 4 for the central processor
    - Create the modem function with the FPGAs labeled U1 to U6
    - Create the central processor function with FPGAs labeled U7 to U10
9. Add the following connectors to enable input/output to the system
    - A Z-DOK+ connector labeled as U11
    - Two FMC connectors labeled as U12 and U13 (these will enable you to cascade with other FPGA boards or functional modules such as RF/DAC/ADC)
    - A memory part, DDR2 SODIMM connector, XP32

Defining the connections between FPGAs

As previously mentioned, we want to maximize the connections between groups of FPGAs intended for each function. We also want to create a bus between the central processing function and the rest of the FPGAs.

Allegro FPGA System Planner enables you to define the connections at a high level. For example, you only need to define the different types of busses and the different FPGAs with which they communicate. A protocol (a built-in feature of Allegro FPGA System Planner) is used to create busses. A protocol can be used to create point-to-point busses or point-to-multi-point busses.

An example is shown in Figure 5. In this example, we have created a 512-bit bus between the six FPGAs forming the modem function. We also create two additional busses: one between U1, U2, and U3, and the other between the other three FPGAs. These busses are 256 bits wide. We then create a small control bus (32-bit, LVDS) between all the six FPGAs. Let's also define a 32-bit bus between the Z-DOK connector and U1. Note that the generic data busses are defined with the I/O standard LVCMOS12.

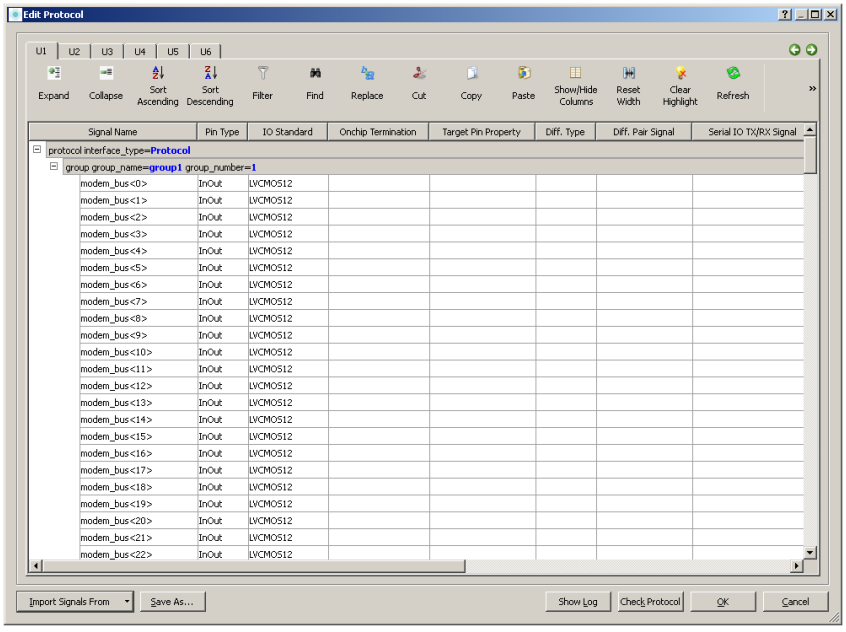

Figure 5: Defining a bus between different FPGAs forming the modem function

A screenshot of the high-level architecture defined so far is shown in Figure 6. This abstract view can be used to define the architecture.

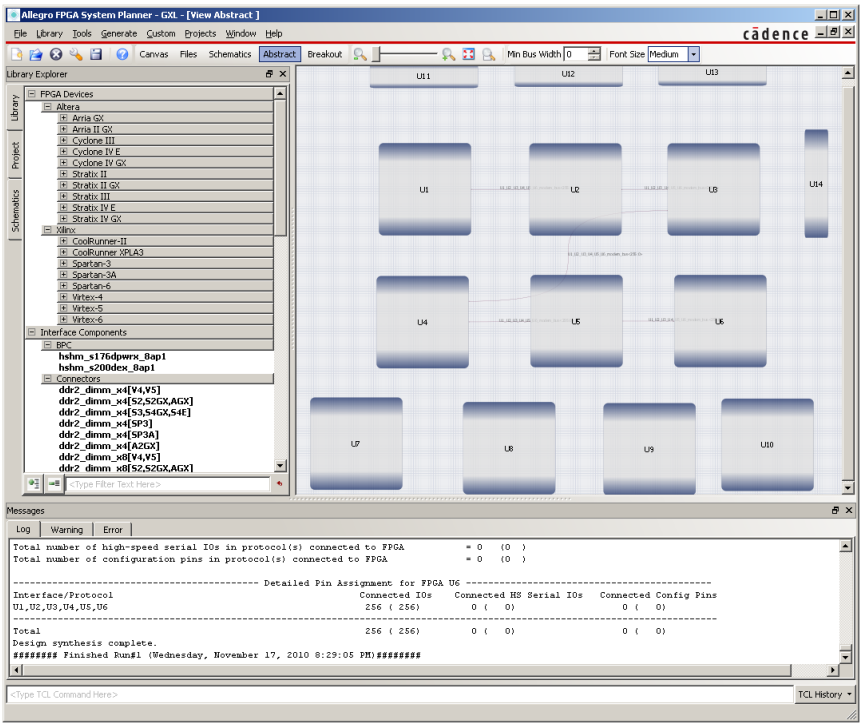

Figure 6: Abstract view of the FPGAs with the different busses defined so far

Running synthesis

Since Allegro FPGA System Planner understands all the FPGA I/O DRC rules, it is able to synthesize the I/O connections based on the high-level definitions. While running synthesis, Allegro FPGA System Planner will consider the logical rule definitions, the physical placement of the FPGAs on the board, and all of the FPGA I/O DRCs. Allegro FPGA System Planner has several synthesis algorithms to help you get the outputs most desirable for your requirements. It is important to consider all of the required factors to pin out the FPGAs correctly:

1. Logical rules. In simple words, following these rules enable the logic to function correctly. An example of a logical rule is as follows: the relationship between clock and data lines in a source-synchronous bus as applied to the FPGA requirement. In a Xilinx Virtex-5 architecture, you will require that the clock line goes to a pin of type "CC" or clock capable pin. The corresponding data pins should go to pins within the same clock region in the Virtex-5 architecture

2. Physical rules. These are rules to enable better board design. Physical rules are derived automatically by Allegro FPGA System Planner based on the layout of the different FPGAs on the Allegro canvas. With this understanding, Allegro FPGA System Planner picks pins that simplify the layout rats-nest

3. FPGA I/O DRCs. These are rules that the FPGA vendors dictate. Allegro FPGA System Planner fully understands these rules for all of the key FPGA architectures

After synthesis, Allegro FPGA System Planner displays the rats-nest of the connections (see Figure 7).

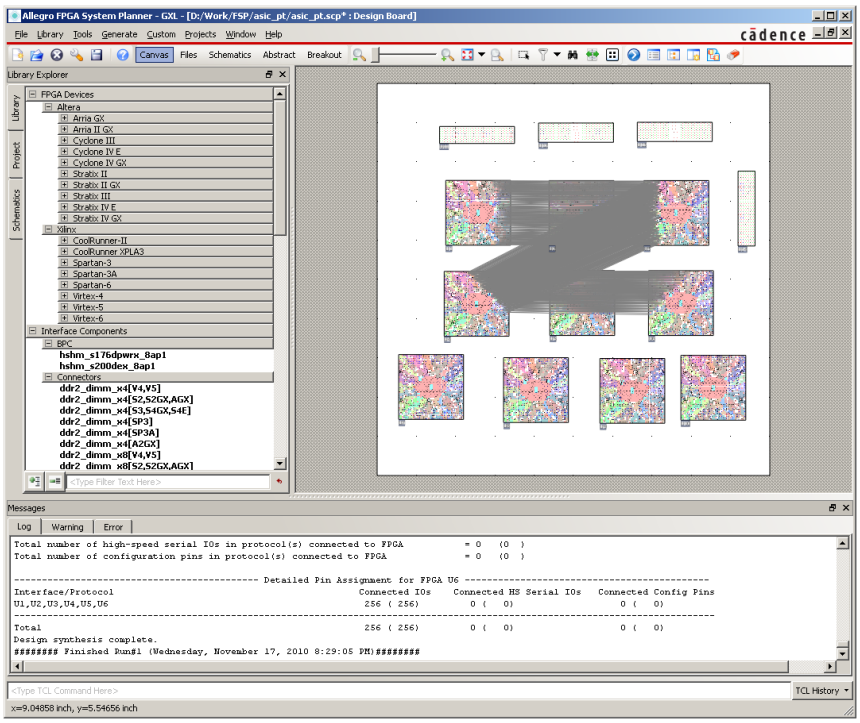

Figure 7: Rats-nest after synthesizing protocols between the modem FPGAs

Allegro FPGA System Planner also generates a detailed report with the I/O utilization. This report can be used to increase or decrease the different bus widths. Reports can be generated for each FPGA or for the entire design (see Figure 8).

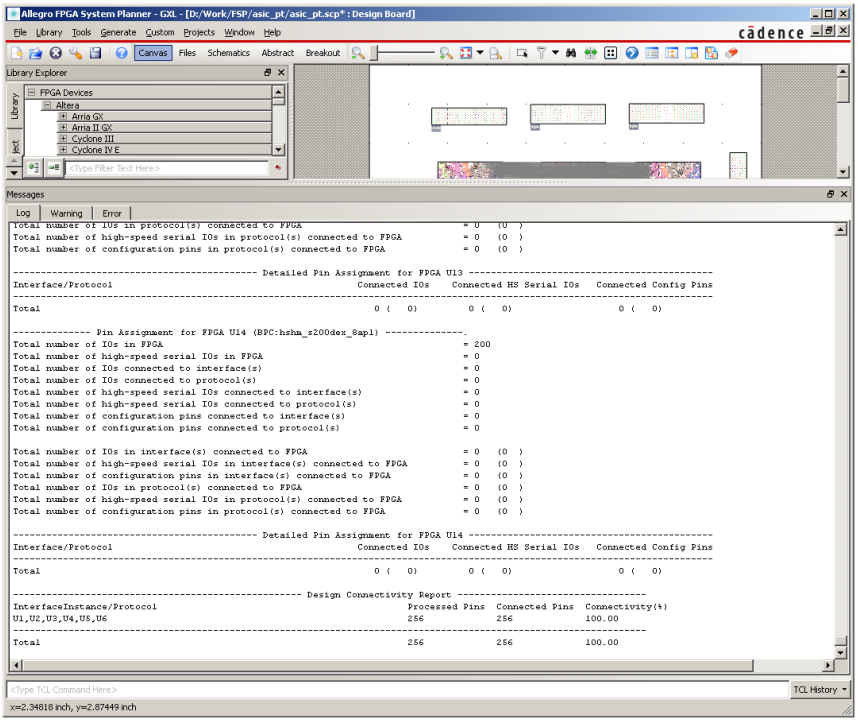

Figure 8: A report of the design so far

Generating outputs for RTL partitioning tools

One key output that the RTL partitioning tools require is the board Verilog file. This file is directly generated by Allegro FPGA System Planner. You can use this file in the partitioning tool to map the wires into FPGA ports (and board traces).

Verifying the partitioning requirements and iterating over the board architecture

You can now use the Verilog board file created in Allegro FPGA System Planner as the input to your FPGA partitioning tool. Use the ASIC/SoC RTL to run through the partitioning tool along with the board Verilog. Based on this, identify if the board that was just designed will be able to satisfy your requirements.

You can easily modify the number of busses between the different FPGAs, the width of the busses, or anything else to satisfy the RTL partitioning requirements. You can iterate between Allegro FPGA System Planner and RTL partitioning tools until you are satisfied with a good balance of board and partitioned RTL.

Once you are satisfied with all aspects of the board architecture, you can also verify the FPGA DRCs with FPGA tools such as Xilinx ISE PinAhead and Altera Quartus. Allegro FPGA System Planner provides a direct interface to verify FPGA pin outs with either Xilinx or Altera tools.

Completing the board design

Once you are satisfied with the board architecture, Allegro FPGA System Planner can be used to complete the rest of your design:

1. Connect the configuration interfaces to configure the FPGAs automatically
2. Connect the FPGA power pins to appropriate voltage rails automatically
3. Generate schematics for popular schematic entry tools automatically
4. Add any required terminations, clock/debug connectors, etc., either in Allegro FPGA System Planner or in your schematic tool

All the mundane and error-prone tasks are completed by Allegro FPGA System Planner to ensure a successful outcome. The next steps are as follows:

1. Perform power mapping in Allegro FPGA System Planner to wire up all the power pins automatically, and bring them up to different rails. You can connect regulators of your choice to the different rails
2. Add a "SystemACE" part and Flash for configuring the FPGAs. Allegro FPGA System Planner connects the devices automatically
3. Define and add the required terminations

Schematics and layout

Now you can generate the schematics for your design. Allegro FPGA System Planner can automatically generate the schematics for different tools, and it can also generate a CSV file format with all the connectivity information, if you want to use a schematic tool that Allegro FPGA System Planner doesn't support.

There are a few things you would need to do in the schematics to take you to board layout. For example, you would need to add all the voltage regulators.

Considerations for successful prototyping

When designing your FPGA board, you should also consider the following:

1. Watch the FPGA LUT utilization and FPGA I/O utilization carefully. Keeping a low utilization will enable you to use the same board for a longer time
2. Ensure there is sufficient RAM left in the FPGAs for debug. Tools such as Xilinx ChipScope utilize a high amount of RAM; you will not be able to use these tools if your FPGA RAM utilization is high
3. Consider connecting each of the FPGAs to a debug connector. This allows you to bring out important signals and observe them using an oscilloscope

4. Consider using high-speed LVDS connections when possible. RTL partitioning tools can automatically use these high-speed busses to multiplex the traces available on the board. Such high-speed busses effectively multiply the available busses on the board
5. Consider a small bus between the FPGAs and use it for complex triggering between the FPGAs. This can prove to be a very helpful debug aid

Conclusion

Allegro FPGA System Planner provides a simplified approach to ASIC prototyping. The current solutions available for ASIC prototyping using FPGAs either require you to spend an inordinate amount of time creating a custom board, or to buy off-the-shelf FPGA boards. The off-the-shelf boards available today cannot satisfy the requirements for complex SoCs because they try to fit one size for all. Secondly, these solutions are prohibitively expensive and so do not scale very well.

Allegro FPGA System Planner provides a simple approach that scales well with the current requirements for ASIC prototyping. It has already been used in several large-scale ASIC prototype design projects successfully.