Partitioning of Large HDL ASIC Designs into Multiple FPFA Devices for Prototyping and Verification

H. Selvaraj, P. Sapiecha[1], N. Dhavlikar,
Department of Electrical and Computer Engineering,
University of Nevada, Las Vegas,
Las Vegas, NV USA 89154
[1] Department of Electronics and Information Technology,
Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract

The ASIC designs are growing larger everyday. It is very hard to simulate these designs because the simulation time has risen tremendously. An alternate solution is to partition the large design into modules and perform incremental simulation. Hardware Embedded Simulation (HES) is a technology that facilitates incremental design verification of large ASICs. On the other hand, since the introduction of FPGAs, they have been playing an important role in ASIC design cycle. But due to very large size of today's ASIC designs (millions of gates) compared to FPGAs, it is not possible to fit an entire ASIC design into a single FPGA device. This problem can be solved by partitioning the given design into multiple small size designs (modules) and fitting those modules into multiple FPGAs. This paper takes a large RTL design of an ASIC into consideration, analyzes the size of each module in terms of number of CLBs, I/Os, flip-flops, latches and applies the algorithm to partition it automatically into minimum number of FPGAs.

Introduction to Hardware Embedded Simulation (HES)

Typical ASIC (Application Specific Integrated Circuit) design cycle stretches for a minimum of 3 to 4 years. Design verification (that includes functional and timing verification) of devices takes the major part of this cycle. The increase in design size causes dramatic increase in the simulation run time. Hardware Embedded Simulation is a technology that facilitates incremental design verification of ASICs using FPGA (Field Programmable Gate Array) devices.

Emulation [3], often referred to as prototyping, has emerged as a major ASIC verification technology where speed of verification is a primary consideration. Weeks and days of simulation time are reduced to hours and minutes. The emulation methodology uses commercially available Field Programmable Gate Arrays (FPGAs) or custom processors to duplicate ASIC design. Emulation is the only verification approach that can attain the speed required for verification of an ASIC design in the real operating environment. Testing in the real environment (in-circuit) rather than in a simulated environment significantly increases the probability that the device will perform as required.

The HES (Hardware Embedded Simulation) environment consists of software simulator and HES boards and is used for speedy design development and verification. This environment assures correct communication between modules located in silicon and in software simulator. HES is flexible and easy to use for ASIC prototyping. Any part of the verified design can be downloaded seamlessly into hardware that works on event-by-event basis with an RTL simulator. In order to use HES for ASIC design, we need to convert RTL design into FPGA code. This can be accomplished using, for example the Synplicity's Certify [2] program. The fact that HES can accommodate hardware models speeds the design verification. HES can also be used as functional simulation accelerator, providing typically 50 to 1000 times speed improvement over the best RTL simulators. Another important factor that makes HES more attractive is that it is very effective in IP core analysis and certification. We only need to decide which modules should be placed into the HES hardware and the rest will be done by the Wizard. Only synthesizable designs can be put into the HES hardware. However, if we are working on RTL designs, code should be synthesizable and ready for HES applications. If we have a mixture of synthesizable and non-synthesizable codes, the design will be split between HES (for synthesizable sections) and software simulator (for non-synthesizable) modules. The simulation will be somewhat slower but still we will reap the major benefits of HES technology such as instant design modifications, speedy checking of selected design paths, etc. The first HES boards have been built with Xilinx FPGA Virtex devices. However, HES boards with Altera's CPLD Apex parts will appear shortly. Since the HES hardware implementation technology (FPGAs and CPLDs) is hidden from the user through the software layer, there should be no additional learning curve when switching from one HES technology to another.

Partitioning is the process of distributing the logic and I/Os among multiple FPGAs. The development of an RTL solution required major innovation. Early in the development cycle, it became apparent that the level of RTL granularity was not compatible with solving the I/O distribution problem and this restriction severely limited the efficiency of logic distribution. The task is one of grouping the most highly connected entities internal to the FPGA and the least connected entities among FPGAs. The smallest entity at the RTL level is a module. A module is a collection of processes and instantiations of other modules. Partitioning at the module level provides a limited number of solutions making it difficult to provide a balanced logic distribution; and in many I/O intensive cases, making it extremely difficult to find an I/O solution. Moving down to the next lower level, the gate level provides too much granularity and the database size becomes unmanageable. The time required to solve the I/O distribution becomes prohibitive. This is why other partitioning tools encountering I/O intensive designs go directly to a multiplexing scheme allowing a single pin to handle multiple signals. Multiplexing should be considered as the last resort to solving I/O distribution. Although it is an effective way to solve the problem, there is a severe speed penalty. If the emulation prototype speed goes below the minimum threshold due to multiplexing, it is impossible to emulate in the external environment causing the emulation project to fail.

Since the ASIC design has to be partitioned and put into multiple FPGAs, it is necessary to extract the information about overall size of the design. The size of the given design can be specified in terms of the following parameters for each module (if the design is modeled using Verilog as the hardware description language) or the entity (if the design is modeled using VHDL): 1) Total number of I/Os (Input Output devices), 2) Total number of CLBs (logic blocks), 3) Number of flip-flops, 4) Number of latches and 5) Number of clocks in the design.

All the above information about individual modules can be extracted only after synthesizing each module. Synthesis of a given design means converting the given HDL design into lower (RTL) level representation. Note that Hardware Description Languages like VHDL or Verilog are only meant for documenting the given design in a systematic higher-level language format. It is the job of the synthesis tool to express the design at the gate level representation. Many synthesis tools are available in the market. In our experiment, we used FPGA Express, which is a complete FPGA logic-synthesis and optimization tool developed by Synopsys company. With the help of this tool we can create optimized FPGA netlists from VHDL or Verilog HDL code.

Once all the parameters about the individual modules are extracted, partitioning algorithm needs to be applied to partition the given design so that it can be implemented using multiple FPGAs. The partitioning decision should be taken automatically. Once the algorithm is applied, again using the FPGA Express tool the individual FPGAs are synthesized and top-level netlist is generated to connect all the modules in the given design. The algorithm takes into consideration only the top level since the synthesis is done taking into consideration the sub-hierarchy.

2. Partitioning Algorithm:

The given ASIC design is described using VHDL or Verilog [4] as the Hardware Description Language. The given ASIC design consists of a number of modules and sub-modules that are described using VHDL or Verilog. The design modules are arranged in a hierarchical way. The proposed algorithm has been implemented in experimental software called "PARTITION".

It is necessary to synthesize all the top-level modules individually using a hierarchical approach. PARTITION invokes the FPGA Express that will look for a correct DPM interface. The following information needs to be supplied in the command line while executing PARTITION: the project and design name, the list of all HDL files (.v for Verilog files and .vhd for VHDL files.) for the top level module and all the sub-modules connected to the same module and arranged in a hierarchical way and also specify the target vendor, the family and the corresponding device. PARTIRION invokes Synposys defined FPGA Express functions to perform the following tasks:

1. Set up the design and analyze the source files. (Analyzing the files involves checking the syntax for VHDL or Verilog HDL language and to report errors/warnings.)
2. Accept the target device and synthesize, optimize the HDL source code and capture netlists to produce an optimized EDIF or XNF netlist for placement and routing.
3. (optional) Generate a Verilog or VHDL netlist for functional simulation.

Once the synthesis stage is over, the five parameter mentioned in section 1 are derived for individual modules from project report generated by the synthesis tool. The aim of the partitioning algorithm is to find the optimum number of devices and divide the entire logic evenly among them. The algorithm can be described as follows:

The total number of I/Os, CLBs, flip-flops and latches decide the number of FPGA devices needed to fit the entire ASIC design. The given ASIC design consists of various modules and sub-modules arranged in a hierarchical manner. The top level connects all the individual modules together to form the entire ASIC design. The algorithm assumes that various modules and sub-modules in the hierarchy are described using VHDL or Verilog as the hardware description language. Since most of the designs are I/O bound, the task of I/Os distribution is the toughest one. It is desired that the highly connected blocks go into the same device so that those connections will be routed inside the device in order to reduce the I/O count. Once the design is synthesized and the netlist is formed, all the information about the connections among different blocks can be extracted and partitioning decision can be taken based on the information. But it is practically very difficult to read the entire netlist for big ASIC design as it is going to occupy a large memory space and the process is very much time-consuming. So the algorithm eliminates the task of reading the netlist. First all the top level modules are individually taken into consideration and the size of each module is analyzed in terms of total number of I/Os, CLBs, flip-flops and latches. If it exceeds the constraints of the given FPGA device, then it is essential to break down that particular top level module to the next hierarchical lower level so that all the sub-modules in the sub-hierarchy are individually synthesized and all of them satisfy the given device constraints. If they do not, then it is necessary to traverse down further to the next sub-hierarchical level. For a particular module the optimum number of devices (N) is calculated based on the number of I/Os, CLBs, flip-flops and latches. The actual formula is given in the mathematical format in the next section. Then all the sub-modules (M) are randomly distributed into N number of devices. It is necessary to satisfy the device constraints in terms of total number of I/Os, total number of CLBs as well as number of flip-flops and latches. Initially, the first device is selected and the constraints are checked. If they exceeded the constraints specified by the manufacturer, then the module that is using the largest number of resources (I/Os, CLBs, flip-flops and latches) is selected. Then the so called the swapping procedure starts. This module will be swapped with a particular module in the successive devices such that the device constraints are satisfied. The swapping process is iterated for all the modules in the individual devices till the constraints for all the devices are satisfied. Then the next top-level module is taken into consideration and steps are repeated till all the top level modules in the entire ASIC design are covered.

Mathematically the algorithm can be described as follows:

1. Let M be the total number of top-level modules present in a given ASIC design. Let N be the number of FPGA devices required to fit the entire ASIC design.
   Let TIO be the total number of I/Os in the entire design and TACTIO be the actual number of useful I/Os in a single FPGA device.
   Similarly TCLB is the total number of CLBs in the entire design and TACTCLB is the actual number of useful CLBs in a single FPGA device. and TFLIP is the total number of flip-flops in the design and TACTFLIP is the actual number of flip-flops in a single FPGA device. Similarly TLAT is the total number of latches in the entire design and TACTLAT is the actual number of useful CLBs in a single FPGA device. Then the optimum value of N is calculated as:
   N = MAX (TIO/ TACTIO, TCLB/ TACTCLB, TFLIP/TACTFLIP, TLAT/TACTLAT)
2. Then distribute M modules randomly into N devices. Each device will incorporate maximum M/N modules.
3. Swapping process:

```
for (I=1, I<=N-1, I++)
{   for (J= I+1, J<=N, J++)
    {   while ( device constraints exceed the manufacturer's limits)
        {   select the module using maximum resources (I/Os + CLBs + Flip-flops +
            latches)
            Find the module in the successive device which can satisfy the constraints.
            Swap (module(I) , module(J) )
              select the next largest module in that device.
        }
    }
}
```

Repeat the above steps till all the top-level modules in the design are covered.

3. Practical example for implementing the algorithm

This example takes into consideration the ASIC design of INTEL 8085 microprocessor [5]. Basically it focuses on the data-path section of this particular 8-bit microprocessor. The data-path section includes following units: Arithmetic Logic Unit, shifter unit, accumulator unit, program counter, flag byte register, memory address register, temporary register, instruction register unit. All the above blocks are described in Verilog as a hardware description language.

The top-level module (data-path) connects all these blocks together to form the data path section of the microprocessor. Once the coding part is done, first the top-level module is synthesized using FPGA Express. The target device is Xilinx- XC3000: 3120A. Since it is not possible to fit this entire module into a single device, the partitioning needs be done. It is necessary to synthesize all the sub-modules and extract values of the required parameters. From the FPGA Express project report generated after synthesizing the top-level data-path module we collect the following information:

Number of I/Os: 75, Number of flip-flops: 69, Number of latches: 0, Number of CLBs: 203,

Specification for Xilinx device: XC3000: 3142ATQ144:

Useful Number of I/Os: 96, Total Number of useful CLBs: 144, Number of flip-flops: 480, Number of gates: 2000 to 3000

Since the resources required by the data-path module exceed the specifications of the device, the design has to be partitioned. Also, it is necessary to synthesize all the sub-modules to extract all information needed to take the partitioning decision.

Results generated after synthesizing all the individual sub-modules.

	Arithmetic Logic Unit	Shifter Unit	Accumulator unit	Program counter.
CLBs:	139	17	8	38
Flip-flops:	0	0	8	16
	Flag byte reg	Memory address reg	Temporary reg	Instruction reg
CLBs:	2	1	0	0
Flip-flops:	5	24	8	8

4. Summary, conclusion and recommendations

HES is a totally new concept. This work shows that HES technology is an innovative solution to increase the simulation speed by 10 to 100 times compared to the speed of software simulator. It has been shown experimentally that the partitioning algorithm gives reasonably good results. The software implementing the algorithm, PARTITION is capable of handling large ASIC designs and can fit them into multiple FPGA devices. Moreover, it takes the partitioning decisions automatically. The software uses FPGA Express synthesis tool and its DPM interface. FPGA Express functions are invoked using the DPM interface and based on the report generated by the tool, the partitioning decisions are taken to evenly distribute the entire logic into multiple FPGA devices

5. References

1. SYNOPSYS FPGA Express Online Help: www.synopsys.com
2. Synplicity Online Help: www.synplicity.com
3. HES Online Help: www.aldec.com, www.speedgateinc.com
4. Zainalabedin Navabi (1999). Verilog Digital System Design, Mc-Graw Hill
5. W. Kleitz (1998). Microprocessor and Micro-controller Fundamentals: 8085 & 8051, Prentice Hall.