

JPG – A Partial Bitstream Generation Tool to Support Partial Reconfiguration in Virtex FPGAs

Anup Kumar Raghavan
Motorola Australia Software Centre
Adelaide SA Australia
anup.raghavan@motorola.com

Peter Sutton
School of Information Technology and Electrical
Engineering
The University of Queensland
Brisbane QLD Australia
p.sutton@itee.uq.edu.au

Abstract

Reconfigurable computing based on partial reconfiguration of field programmable gate arrays (FPGAs) is yet to move to the mainstream of computing. Hardware devices that support such reconfiguration are now available but no readily available software exists to generate the required partial bitstreams. The JPG tool described in this paper is a Java-based partial bitstream generator designed to fit within the standard Xilinx FPGA design flow. JPG, based on the Xilinx JBits API, is able to generate partial bitstreams for Xilinx Virtex devices based on data extracted from the standard Xilinx CAD tool flow.

switches are usually still more time consuming than CPU context switches. When you include the time taken by software tools to generate new configurations (possibly hours) compared with software compilers (seconds or minutes) this disadvantage is exacerbated.

This paper describes a tool (JPG), designed to fit into the standard Xilinx Foundation 3.1 design flow, which enables the generation of partial bitstreams. Partial bitstreams can be generated by running the standard design tools on individual modules rather than complete designs. This shortens the turn-around time for producing modified designs. JPG is based on the Xilinx JBits API and can generate partial bitstreams for Xilinx Virtex series FPGAs.

The remainder of the paper is organized as follows. Section 2 reviews FPGA CAD tool flows and bitstream generation. Section 3 describes the operation of the JPG partial bitstream generator and how it fits into the standard tool flow. Section 4 discusses the advantages and disadvantages of the approach. Some conclusions are drawn in Section 5.

1 Introduction

Reconfigurable Computing (RC) involves altering the programmed design within a static-RAM field programmable gate array (FPGA) at run-time [1]. RC allows the design of hardware to be changed in response to the demands placed on the system while it is running, or it allows the execution of more complex designs than the number of gates available in the device would traditionally allow. The “ideal” of RC, is to be able to context-switch hardware in the same way that CPU’s context-switch software.

Early FPGAs supported only “full” reconfiguration, i.e., the configuration of the complete device had to be loaded in order to change even a small part of the design. More recent FPGAs such as the Virtex series from Xilinx [3] and the AT6000 series from Atmel Corporation [4] support *partial* reconfiguration, that is, the ability to modify only a portion of the design. Some devices support *dynamic* reconfiguration: the ability to change a portion of the design whilst the remainder of the device continues to operate.

Partial and/or dynamic reconfiguration allow faster context-switches than “full” reconfiguration, however, such

2 FPGA Tool Flows and Bitstreams

In the traditional CAD tool flow for FPGA design, the place and route (P&R) steps usually require the longest execution time [5]. Making changes, therefore, can be an expensive process that involves significant time and effort. For reconfigurable computing applications, however, it is usually desired that changes can occur frequently. Applications mentioned in [1] reuse the same hardware with several different designs that change quite frequently. It is difficult to generate new configurations quickly using standard tool flows.

The traditional EDA CAD flows are generally intended for static designs. These flows produce a complete bitstream file that is downloaded into the device to implement a specific application. For dynamically changing designs, a portion of the design will need to be changed to implement the newer functionality, while the remaining

portion of the design remains unaffected. As conventional CAD tools generate complete bitstreams, they cannot, in general, be used to implement reconfigurable computing designs. Reconfigurable computing therefore needs tools that can produce partial bitstreams.

2.1 FPGA Bitstreams

In most FPGA architectures, the entire device would be configured using one complete stream of configuration data called the *bitstream*. This bitstream contains the information required to activate the resources (e.g. switches and lookup tables) in the FPGAs and implement the designed circuit. If any of the resources in the FPGA need to be changed, the entire configuration data would be regenerated by the available CAD tools and would then have to be re-loaded into the device. The complete bitstream will hold the programming information for both used and un-used resources in the FPGA architecture.

In order to perform efficient partial reconfigurations, one solution is to be able to create *partial bitstreams*. Partial bitstreams are subsets of a complete bitstream. The detail in these bitstreams is less than that in a complete bitstream and should, theoretically, require less effort to generate.

The advantages of producing partial bitstream files are many. These include:

- The overall run time for CAD tools to complete the mapping, placement and routing will be shorter as we are dealing with a smaller area of logic. This can mean shorter design iterations and quicker time to market. Alternatively, it could mean more highly optimized designs in the same design time.
- The time involved in downloading the partial bitstream file and reconfiguring the device will be shorter as the size of the partial bitstream files will be smaller compared to complete bitstream files.

2.2 JBits

Xilinx has developed JBits [6,7], a set of Java APIs [8] which allow programmers (who may or may not be the hardware designers) to create software that manipulates Xilinx Virtex device bitstreams.

A disadvantage of using JBits is that it has a very low level model of the FPGA device. This means that the designer has to understand the device architecture thoroughly and make changes to the design by manipulating the low level resources in the devices e.g. interconnects, switches, connection blocks, switch blocks, and CLBs. Hence, the tool can be useful to work on small sized designs or to make small changes to the design but isn't very effective when trying to make large scale modifications. For large scale designs, JBits is better used

as an API on which to base generic tools, rather than design-specific tools.

2.3 Related Work

The JBitsDiff [9] and PARBIT [10] tools share some similarities with this work. JBitsDiff, like JPG, is built on the Xilinx JBits API. Rather than producing partial bitstreams, however, JBitsDiff extracts information from the bitstream to generate pre-routed and pre-placed JBits cores. A JBits core is a sequence of Java method invocations (using the JBits API) that will manipulate a device bitstream in order to insert the core at some location in the device.

PARBIT is a C program which supports partial bitstream generation for Xilinx Virtex-E devices. The main difference between PARBIT and JPG is that PARBIT uses a separate options file for specifying information about the partial bitstream to be generated, whereas JPG relies on information extracted from design and constraint files within the Xilinx CAD tool process.

3 JPG Tool

The concept of partial reconfiguration performed using the JPG tool is illustrated in Figure 1. To implement designs that use partial reconfiguration, at least one complete base design should be available. The partial reconfiguration designs then provide alternative, possibly upgraded, functionality from the base design.

As seen in Figure 1, the FPGA is partitioned into sub-modules. Each sub-module represents individual logic that can be changed by partial reconfiguration at a later stage.

In the example shown, the device is partitioned into four regions, and one of them is shaded, which indicates that it will be modified at some point in time. There are

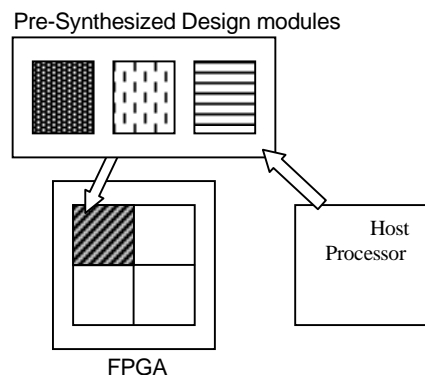


Figure 1. RC environment, the host processor sends design updates to the FPGA

three different implementations for that particular logic module and their partial bitstreams are obtained using the JPG tool. The JPG tool allows the designer to download any particular implementation on the base design thus partially reconfiguring the device.

The complete CAD flow incorporating the JPG tool is shown in Figure 2.

The process to setup a design and create partial bitstreams involves two phases. Phase 1 involves creating the base design, partitioning the base design with sub-modules of individual logic and implementing the sub-modules. Phase 2 involves creating different versions of the sub-modules that will be used to partially reconfigure the sub-module in the base design

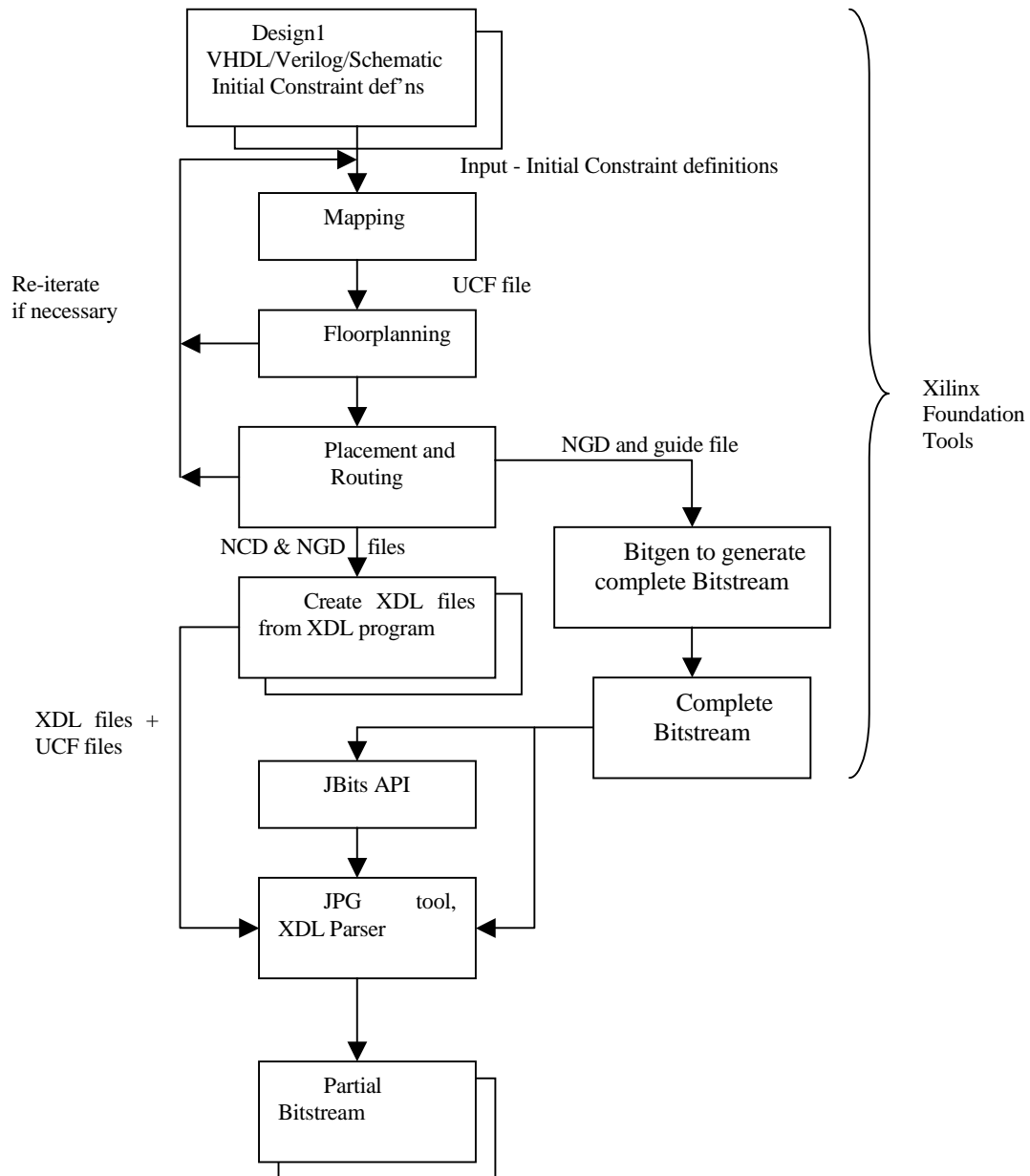


Figure 2. JPG CAD tool flow.

3.1 Phase 1 – Base Design

The design process starts by partitioning the base design into smaller modules that would represent different modules of the design. A HDL or schematic description of the individual modules is created, with one top-level design file. An initial floorplan is developed to constrain the placement of the individual modules onto specific regions of the device. This sets a base location for the modules that will be used later on, when it is partially reconfigured. The files are then synthesized, mapped, placed, and routed using the Foundation tools to generate a complete bitstream file for the base design.

To update the base design with changes in functionality on individual modules, partial reconfiguration can be done on that module alone instead of changing the design completely. To perform partial reconfiguration, partial bitstream files are required. The JPG tool replaces portions of the existing design in the device with the new partial bitstream files thus implementing new/updated designs on the same chip.

3.2 Phase 2 – Partial Designs

Changes on the sub-modules are performed as a new project in the Foundation tool. This means that the sub-module will be the only design file in the project (without other modules as the base design). The steps taken are as follows:

- Guided floorplanning is performed using the constraints from the base design to place the updated sub-module in the same location as that of the base design.
- The Foundation tool is used to complete the mapping, placement and routing for the updated sub-module using the constraint files obtained from the previous step. This step may need to be iterated to obtain the desired results.
- The previous step results in several files. Those relevant to the JPG tool are those with the extensions .ncd and .ucf. The NCD file is a binary database file (Xilinx Foundation tool specific) that has all the connectivity details used in the design. The XDL utility converts the corresponding .ncd file into an .xdl file (ASCII file) that will contain useful information regarding the resources that have been used to implement the design.

At this point, the use of the Xilinx CAD tools is complete, and the flow continues using the JPG tool to produce partial bitstreams.

3.2.1 JPG Tool Usage

The JPG tool (shown in Figure 3) is used to produce partial bitstreams as follows:

- A new project can be created in JPG or an existing project can be opened.
- The complete bitstream file from the base design is used to initialize the environment variables in the JPG tool. The JPG tool needs the base design to generate partial bitstreams.
- The .ucf and .xdl files obtained from the previous steps are passed in as input to the JPG tool. These files are the newer versions of the sub-modules that need to be partially reconfigured. The parser in the tool reads information from these files and makes appropriate JBits calls to initialize the design on the target device. After reading these files, the JPG tool displays graphically the target floorplanned area on the FPGA. This can be used to verify whether the update is happening on the region desired by the designer.
- The tool offers two options. One option is to obtain the partial bitstream of the new design, without downloading the partial bitstream onto the base-design. Option two allows the designer to write the partial bitstream onto the base design, thus partially reconfiguring the device with the updated logic.

When option two is selected to write the partial bitstream onto the base design, the existing bitstream would be overwritten. Care should therefore be taken before modifying the original bitstream. If there is a FPGA board connected to the PC and the XHWIF interface is used to connect the tool to the board, the newly generated partial bitstream is written onto the FPGA thus partially reconfiguring the device.

Details on the XDL parser step are provided in the following section.

3.2.2 XDL parser

The JPG tool extracts programming information obtained from the .xdl and .ucf files and parses them to make valid JBits function calls to program the device. The .xdl file is obtained from a .ncd file using standard Xilinx tools. The .ncd file contains details regarding nets used in the design to complete connections between components and with the I/O pads, and geographical information about the rows and columns occupied by the CLBs (Configurable Logic Blocks) that implement the design as well as detailed information on routing resources used to make connections in the device. Routing information consists of PIPs (programmable interconnection points) details, wiring tracks, and CLB pin details that provide source and sink information for the nets.

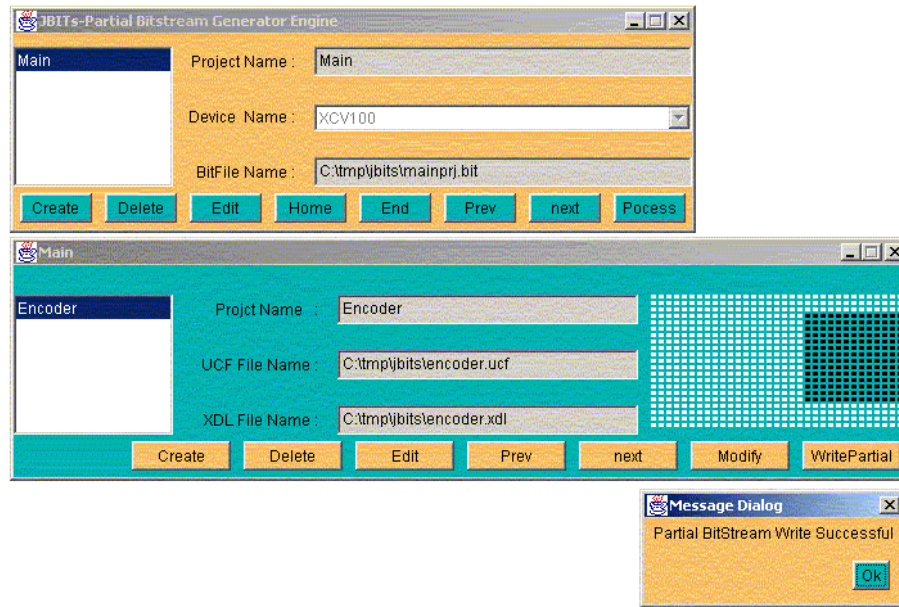


Figure 3: JPG tool user interface, showing the floorplan of the device

A sample .xdl file contains information as shown below:

```
inst "u1/nrz" "SLICE" , placed R3C23
CLB_R3C23.S0 ,
  cfg "CKINV:::1 DYMUX:::1
G:u1/C307:#LUT:D=(A1@A4) CEMUX:::CE
  SRMUX:::SR GYMUX:::G
SYNC_ATTR:::ASYN SRFFMUX:::0 INITY:::LOW
  FFY:u1/nrz_reg:#FF

_PINMAP:24:0,1,2,3,4,5,6,7,8,9,10,11,12
,15,14,13,16,17,18,19,20,21,22,23"
;
```

This configuration information is obtained from an example design. It is this information that is vital to the JPG tool to deliver reliable partial bitstreams. The text above describes an instance of a module named “U1.” “nrz” refers to the name of a signal used in the design. The source node for the signal begins at a Slice S0 in the CLB at location Row 3, Column 23 and traverses to the G input of a CLB on the same module U1, at Column 307. The net traverses through a set of multiplexers, flip-flops and other components as indicated. The pin map indicates the set of pins that have been used to make a connection. These pins can be from the different components, PIPs (Programmable Interconnect Points) and so on.

The JPG parser scans through the complete .xdl file and makes appropriate JBits calls to program the device. JPG assumes that modules to be introduced by partial

reconfiguration have the same interface as those they are replacing.

4 Discussion

4.1 Advantages

The JPG tool establishes a link between the Xilinx Foundation tools and JBits. This allows designers using the conventional tool flow to take advantage of some of the low-level features offered by JBits without having to be familiar with Java or the low-level device architecture.

Projects that require partially reconfigurable solutions are developed using Xilinx’s standard tools which allow the designer to be confident in the tool output, both in terms of correctness and performance (at least more so than using non-commercial design tools). The JPG tool flow has been specifically designed to use the mainstream tools and constraint files where possible, using dedicated software only for the partial bitstream extraction and not for tasks such as placement and routing.

The JPG tool facilitates the creation of partial bitstreams for designs that can be quite large. It enhances the use of JBits to work on larger designs and provides an opportunity to create multiple partial bitstreams that are selected through a GUI interface and downloaded into the device. A potential advantage in having multiple partial bitstreams is that the physical-design time involved in creating partial bitstreams (mapping, placement and routing time) is significantly less than that for the complete bitstream.

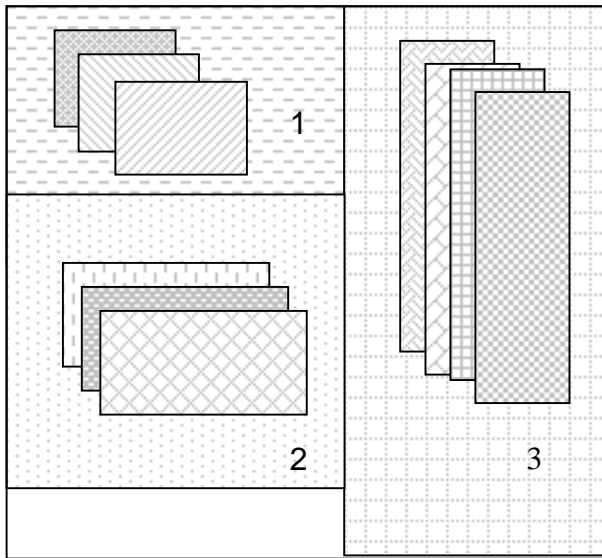


Figure 4: Conceptual model illustrating the partitioning of the FPGA into several regions, with each region having 'x' number of partially reconfigurable designs.

Figure 4 illustrates the concept of a design that has different modules that can be changed dynamically. For each module that needs to be reconfigured, there are several implementations of the same module with modifications from the original design.

The device is partitioned into three regions: region 1 has a module that can be configured in three different ways, region 2 in three different ways, and region 3 in four different ways. The number of combinations of modules is therefore 36 ($3 \times 3 \times 4$). In a conventional CAD flow, which can only produce complete bitstreams, 36 runs of the CAD tool flow would be needed to produce the 36 different bitstreams that would be needed to support all the combinations of modules. (The runs may not be independent – they could take advantage of incremental design support – if present in the tools used.) With the use of partial reconfiguration, a total of 10 ($3+3+4$) partial bitstreams would be needed – each about a third the size of a complete bitstream. In addition, there would be a single complete bitstream to initially configure the device.

JPG is capable of generating these partial bitstreams. The CAD tool flow would be run on each of the 10 modules in their constrained regions, resulting in much shorter place-and-route times, in addition to the reduced configuration times from the smaller bitstreams.

4.2 Disadvantages

The JPG tool is specific to Xilinx Virtex FPGAs, since it is dependant on the JBits API from Xilinx. (JPG could potentially be extended to the XC4000 and XC6200 series of FPGAs from Xilinx, as earlier versions of JBits did

provide support for these devices.) In addition, the CAD flow for JPG usage is slightly more cumbersome than the standard flow as it involves indirect manipulation and a little extra effort on the part of the designer. Tighter integration into the Xilinx tool flow would of course reduce this problem

5 Conclusion

This paper demonstrates a methodology to create partial bitstreams in a manner consistent with the standard Xilinx CAD tool flow. The JPG tool, based on the JBits Java API, generates partial bitstreams using information derived from other tools in the standard flow. The JPG tool brings partial bitstream generation into the conventional design process, making it easier for designers to generate partial bitstreams, whilst maintaining the advantage of commercial CAD tools for most of the design flow. This is another step on the path of moving reconfigurable computing techniques into the mainstream design community.

References

1. Michael Barr, "A Reconfigurable Computing Primer", *Multimedia Systems Design*, Sep. 1998, pp. 44-47.
2. John Villasenor and William H. Mangione-Smith, "Configurable Computing", *Scientific American*, June 1997.
3. Xilinx Inc., "Virtex 2.5 V Field Programmable Gate Arrays", Advance Product Data Sheet, 1998.
4. Atmel Inc., ATMEL AT6000 data sheet, 1996.
5. P. S. Sidhu, A. Mei, and V. K. Prasanna, "String matching on multicontext FPGAs using self-reconfiguration", in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 217-226, Monterey, CA, February 1999.
6. Xilinx Inc., JBits documentation, 1999, Published in JBits 2.0.1 documentation. 8.0 Xilinx Corporation, "JBits- Java Based APIs."
7. S. A. Guccione, D. Levi, and P. Sundararajan, "JBits: A Java-based interface for reconfigurable computing," in *Second Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD)*, (Laurel, MD), September 1999.
8. Sun Microsystems Computer Corporation. The Java Development Kit - JDK 1.3, URL: <http://java.sun.com/>.
9. Philip James-Roxby and Steven A. Guccione. Automated Extraction of Run-Time Parameterisable Cores from Programmable Device Configurations. In *Proceedings of IEEE Workshop on Field Programmable Custom Computing Machines*, pages 153-161, April 2000.
10. Edson L. Horta and John W. Lockwood. PARBIT: A Tool to Transform Bitfiles to Implement Partial Reconfiguration of Field Programmable Gate Arrays (FPGAs). Washington University Department of Computer Science Technical Report WUCS-01-13. July 2001. (Available at <http://www.arl.wustl.edu/arl/projects/fpx/parbit>)