

An Alternate Wire Database for Xilinx FPGAs

Neil Steiner and Peter Athanas

Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
neil.steiner@vt.edu, athanas@vt.edu

Abstract

This paper presents ADB, an Alternate Wire Database, suitable for routing, tracing, and browsing in Xilinx Virtex, Virtex-E, Virtex-II, and Virtex-II Pro FPGAs. While mainstream design flows and place-and-route tools make good use of available routing resources, they often do so at the cost of comparatively large processing times. An alternative scheme is to modify or generate configuration bitstreams directly, in order to achieve more dynamic designs and to reduce processing times and memory footprints. ADB includes a complete set of compact wire databases for the indicated families, and can generate the necessary bitstream configurations with the help of JBits or an independent bitstream interface. These wire databases can also be used in standalone mode to facilitate routing research in situations where real device data might not normally be available.

1. Introduction

Modern Field-Programmable Gate Arrays (FPGAs) have matured beyond their origins as small glue logic chips, to include very large devices that can support entire System-on-Chip (SOC) designs. And while FPGA development cycles are fast compared to Application-Specific Integrated Circuit (ASIC) cycles, features such as dynamic and/or partial reconfigurability require cycle times on the order of seconds or minutes, rather than hours or days, and this in turn requires supporting tools.

One such tool is JBits [2], a system which operates directly on Xilinx configuration bitstreams to implement or modify designs, bypassing the regular tool chain entirely. Although the use of JBits involves tradeoffs, it does provide low-level access and control of devices, which can be appealing for research.

ADB [4] is a supplemental connectivity database that can interface with JBits to provide wiring information and routing, unrouting, or tracing services. ADB can also operate in standalone mode, providing information and services to client tools, such as browsers, routers, tracers, and placers. ADB may be of interest to JBits users who require exhaustive wiring support, or to researchers who might not normally have access to wiring data for real-world commercial FPGAs. It may also have some value in facilitating JBits support for new FPGA families, and its existence as a wire database distinct from the JBits internal wire database may facilitate mutual verification and validation. In addition, because JBits 3.0 no longer includes JRoute [3], ADB provides the only router currently available to JBits users. The router provided with ADB is presently being used in JHDLBits, an open-source project that connects JHDL [1] and JBits.

ADB has a number of design objectives, inherited and expanded from prior work:

- 100% coverage of device wiring
- Good database runtime performance
- Good database initialization performance
- Compact representation of data on disk
- Compact representation of data in memory
- Compatibility with JBits

The performance and compactness become especially important in the case of embedded systems that need to dynamically reconfigure themselves.

2. Database Design

The primary purpose of ADB is to support routing and tracing services, and it does this with the help of device models read from database files. Table 1 shows the dimensions of the largest supported devices, and suggests the desirability of high compression ratios for both memory and disk representations of the data.

Table 1. Family dimensions

<i>Family</i>	<i>Device</i>	<i>Tiles</i>	<i>Wires</i>
Virtex	XCV1000	7,169	3,020,606
Virtex-E	XCV3200E	19,367	8,019,870
Virtex-II	XC2V8000	15,851	10,463,182
Virtex-II Pro	XC2VP125	22,785	14,108,305

While the connectivity inside FPGAs is substantially more complex than data books would suggest, it is possible to organize wires into common *shapes*, which makes it possible to reduce the memory footprint of much of the data by an order of magnitude.

3. Database Build

The database design determines the internal representation of ADB data, but this internal representation differs significantly from that of the source data. The conversion from source data to internal representation is called the *build process*, and this process is decidedly non-trivial.

The build process is carried out by a collection of Perl scripts that draw upon source data, most of which is Xilinx proprietary information. On a simple level, the build process must reconcile the family and device data and use that data to generate the resulting device and family database files.

In practice this is a difficult problem because of differing conventions and representations used in the various source files, and because of exceptions that show up along the way. The build process takes extensive care in reconciling the data and logging exceptions, and though the exceptions are relatively small in number, they are carefully checked with Xilinx before corrections are applied.

ADB relies on the same data as the Xilinx ISE *bitgen* tool, but unlike *bitgen* which only maps design information to configuration bitstreams, ADB must also be able to map configuration bitstreams back to design information for tracing purposes. This requires inverting thousands of systems of Boolean equations, and because the Boolean AND and inclusive-OR operations have no inverses, ADB relies extensively on the fact that only one connection at a time can exist through a multiplexer.

4. Results and Conclusion

ADB meets its objectives of exhaustiveness, compactness, performance, and compatibility with JBits. It currently supports every available device in the Virtex, Virtex-E, Virtex-II, and Virtex-II Pro families, 44

devices in all, and can be expanded to support future families.

The correctness of ADB rests primarily on the chain of custody of the source data and the manner in which it was processed, since the Xilinx ISE tools do not expose the information necessary for verification. But it is significant that ADB can correctly trace through bitstreams generated by the ISE tools or by JBits or by itself, and that JBits can correctly trace through bitstreams generated by ADB, all of which suggests that the three tools are in agreement as far as they were tested.

Although ADB includes a router, it is primarily designed to be an efficient wire database. Table 2 shows the raw database performance on a 1,400 MHz Pentium III machine, expanding each wire in a device to identify all sink wires to which it can connect.

Table 2. Raw database performance

<i>Device</i>	<i>ms</i>	<i>Wires/s</i>	<i>Sinks/s</i>
XCV1000	1,722	1,754,127	16,028,012
XCV3200E	5,364	1,495,129	18,616,507
XC2V8000	6,774	1,544,609	19,778,778
XC2VP125	11,299	1,248,633	18,194,132

ADB can be used in a variety of situations, but its most significant impact to date is in providing underlying routing and tracing support for the JHDLBits project.

This work benefits from kind support from Xilinx, and was funded in part under DARPA contract DABT63-99-3-0004.

References

- [1] P. Bellows and B. Hutchings. JHDL - An HDL for Reconfigurable Systems. In *Proceedings of the Sixth Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 1998*, pages 175–184, April 1998.
- [2] S. A. Guccione and D. Levi. XBI: A Java-Based Interface to FPGA Hardware. In J. Schewel, editor, *Configurable Computing: Technology and Applications, Proceedings of SPIE*, volume 3526, pages 97–102, Bellingham, Washington, November 1998.
- [3] E. Keller. JRoute: A Run-Time Routing API for FPGA Hardware. In J. Rolim et al., editors, *Parallel and Distributed Processing, Lecture Notes in Computer Science*, volume 1800, pages 874–881, Berlin, May 2000. Springer-Verlag.
- [4] N. Steiner. A Standalone Wire Database for Routing and Tracing in Xilinx Virtex, Virtex-E, and Virtex-II FPGAs. Master’s thesis, Virginia Tech, August 2002.