Ph.D. Progress Report --- Report #2

(March 2001 -- March 2002)

by Andy Gean Ye

This report summarizes my Ph.D. research progress from March 2001 to March 2002. This time period corresponds to part of the third and fourth year of my Ph.D. candidacy. As stated in my first report, the goal of my Ph.D. research is to create an efficient FPGA architecture for datapath circuits. My research methodology is empirical and consists of three phases, two of which have been completed as of March 2002. The first phase consisted of gathering a suite of real benchmark circuits. The second phase consisted of creating a complete synthesis, packing, placement, and routing CAD flow to map these benchmark circuits to my proposed FPGA architectures. During the third and final phase, I will design and perform experiments to study various datapath-oriented FPGA architectures using the benchmarks and the CAD tools.

Currently, I have collected a benchmark suite of 15 datapath circuits from the Pico-Java processor [1]. I also have constructed a complete CAD flow, which is created by augmenting and modifying various commercial and academic tools including Synopsys design compiler [2] and University of Toronto VPR placer and router [3]. Substantial modifications have been made to enable these tools to utilize datapath resources. The details of these modifications are summarized below in chronological order.

March 2001 to July 2001

March 2001 to July 2001 were spent on solving the problem of post-synthesis area inflation. Synopsys can be used to synthesize datapath circuits while preserving datapath regularity. This synthesis method (called structured synthesis), however, often results in much larger circuits than flat synthesis, which does not preserve datapath regularity. We observed an average area inflation of 38% over the 15 benchmark circuits. Using two word-level transformation techniques discovered by us and several traditional structured-synthesis techniques, we were able to improve upon the

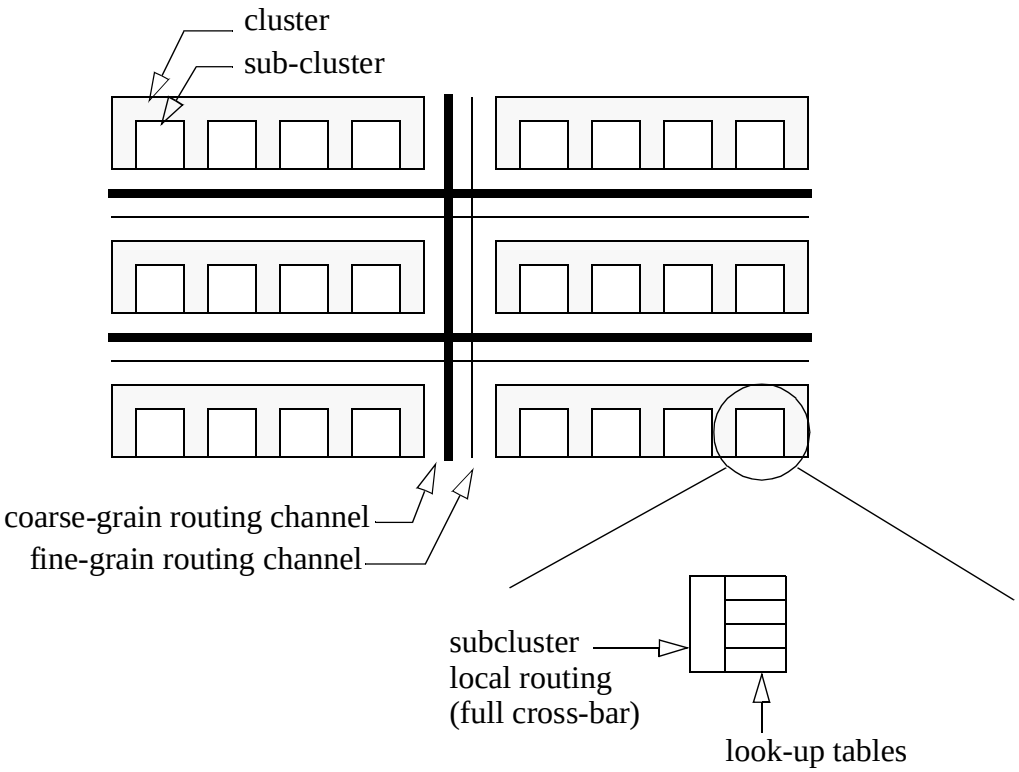

Figure 1: Datapath FPGA Cluster Architecture

structured-synthesis result of Synopsys. Overall we reduced the area inflation to a negligible 3%. These transformations were summarized in detail in the previous progress report [18] and in a paper submitted to the 39th Design Automation Conference [19].

August 2001

We investigated cluster architecture in August 2001. The primary focus of our investigation was on cluster-input specialization. This is motivated by our observation that, in datapath circuits, roughly 40% of two-terminal connections can be grouped into 4-bit wide buses and another 40% of two-terminal connections are from nets with fanout of at least four [18]. We call the first type of signals bus signals and the second type control signals. In our investigation, we designed various specialized cluster inputs which are particularly efficient for routing either bus or control signals. Since we did not have a functional placer and router at the time, the experimental results were incomplete. Nevertheless these preliminary results do give a good indication on the cost and benefit of cluster input specialization.

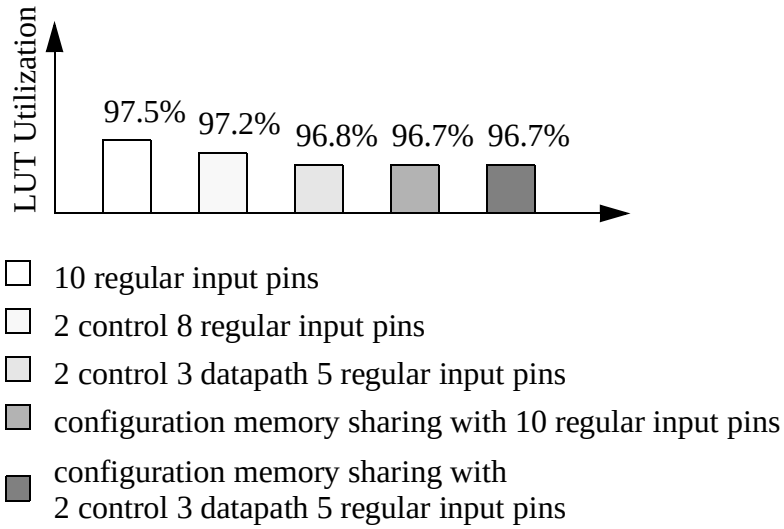

Figure 2: LUT Utilization for various cluster architectures

Our experiments were performed using the improved synthesis tool and our datapath-oriented packing tool written prior to March 2001. All of our experiments assumed clusters each containing 4 subclusters. Each subcluster was assumed to contain four 4-LUTs, 10 input pins, and a fully connected local routing network (full cross-bar). Figure 1 illustrates six clusters and their associated routing resources. More detail on the cluster architecture can be found in the proposal [17] and the first progress report [18].

We proposed two types of specialized cluster inputs. One type is called control-inputs. A control-input pin brings a single signal into the cluster and then distributes the signal to all four subclusters. We observed that clusters with two control-input pins and eight regular input pins per subcluster had an average LUT utilization of 97.2%. As a comparison, clusters with 10 regular input pins per subcluster had an average LUT utilization of 97.5%.

The other type of specialized cluster inputs is called datapath-inputs. Each datapath-input group consists of four input pins, one from each subcluster. These four inputs share a single set of configuration SRAMs. We observed that clusters with two control-input pins, three datapath-input pins, and five regular input pins per subcluster had an average LUT utilization of 96.8%.

The results above demonstrated that the input specialization does not significantly impact on the LUT utilization for datapath circuits. It is very likely that the routing area saving due to these specialization methods can outweigh the slight decrease in utilization. We intend to investigate this area further using a full placement, routing and timing analysis flow.

During August 2001, we also investigated the possibilities of sharing configuration memory across subclusters that belong to a single cluster. When a cluster is used to implement datapath circuits, due to datapath regularity, corresponding LUTs and local routing resources often are configured identically. We can save area by sharing some or all of the configuration memory among these subclusters. Although all of our benchmarks were datapath circuits, they still contained a small amount irregular logic. We assumed two types of clusters in our experiments --- one which shared some or all of the configuration memory, the other which did not share any configuration memory. We packed the regular logic into the first type of clusters. When possible, we also pack the irregular logic into the first type. The remaining irregular logic was packed into the second type of clusters.

We found that clusters, whose subclusters fully shared a single set of configuration memory, had LUT utilization of 96.7%. The combined effect of configuration memory sharing and input specialization was also investigated. We found that clusters with full configuration memory sharing, two specialized control inputs, three specialized datapath inputs, and five regular inputs also had a LUT utilization of 96.7%.

September 2001 to December 2001

September 2001 to December 2001 were spent on writing the paper "Structured Logic Synthesis for Datapath FPGAs" which was submitted to DAC2002. We also modified VPR to accept new datapath-oriented architectural description files. Our packing algorithm was further improved to preserve the regularity of datapath components whose width are less than the architectural datapath width. The VPR placer was determined to be adequate to perform placement for our datapath FPGA architecture.

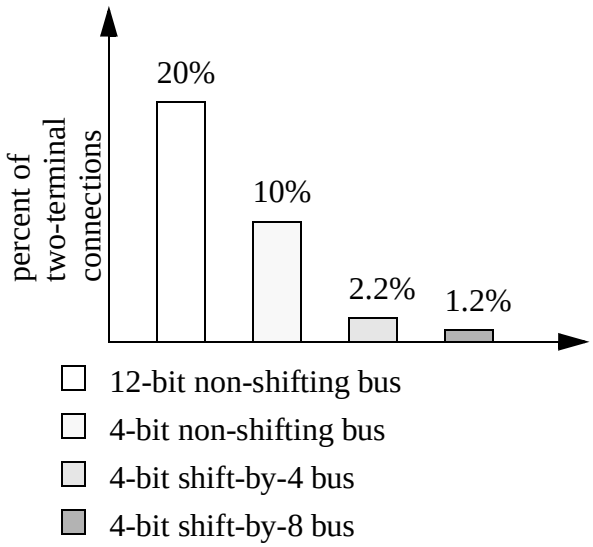

Figure 3: Four most populous bus types for cluster width of 12

During December 2001, we examined the effect of cluster width (the number of subclusters in a cluster), m, on intercluster routing. A series of 9 experiments were performed, where m was set to be 2, 4, 8, 12, 16, 20, 24, 28, or 32 respectively. During each experiment, we synthesized and packed the benchmark circuits into clusters of a given width. We then grouped the intercluster nets into two-terminal buses. Finally these buses were classified according to their width and the amount of shift required to route each bus through the coarse-grain routing resources. The need of shift operations in coarse-grain routing is described in detail by Cherepacha and Lewis in [4]. We observed that m had several important effects on the types and the widths of the two-terminal buses that we were able to recover.

First of all, the number of two-terminal connections that could not be grouped into any buses remained quite constant and consisted of 15% to 20% of the total two-terminal connections. Second, the number of m-bit non-shifting buses decreased significantly from near 50%, when m equal to 2, to a minimum of 14% as m was increased to 20. The figure then increased slightly to around 20% as m was further increased to 32.

Third, for all cluster width, the most populous type of bus were m-bit non-shifting buses. The second most populous type of bus usually was significantly smaller than the most populous type. It usually was also non-shifting. The majority of the remainder bus types usually contained less than 1% of total two-terminal connections each. For example, when m was equal to 12, 12-bit non-

shifting buses consisted of 20% of the total two-terminal connections. The second most populous bus type was 8-bit non-shifting buses, which consisted of 10% of the total two-terminal connections. The third most populous bus type contained slightly more than 2% of total two-terminal connections.

The above observations can guide us to choose the appropriate types of routing resources for various coarse-grain FPGA architectures. It is likely that, regardless of cluster width, we need 15% to 20% of fine-grain routing resources. We likely need at most two types of coarse-grain routing resources. One type always will be m-bit wide and non-shifting. The choice of the other type depends on the cluster width of the architecture and the datapath width of the targeting applications.

January 2002 to March 2002

January 2002 to March 2002 were spent on modifying VPR routing algorithms to route datapath circuits. The modified routers have to deal with both coarse-grain and fine-grain routing resources and nets. Currently, our coarse-grain routing resources consist of m-bit non-shifting coarse-grain tracks, where m is the architectural datapath width. Other types of coarse-grain routing tracks will be considered in the future. Both routability-driven and timing-driven routers were constructed. Experiments were performed using the routability-driven router. Initial results indicated that our datapath-oriented routing architecture can achieve a 20% saving in routing area.

The basic algorithm employed by our routers is a negotiation based maze expansion algorithm. We first identify two types of buses --- pin-buses and net-buses. A pin-bus is an m-bit non-shifting two-terminal bus. We call two-terminal connections in a pin-bus coarse-grain connections. Two-terminal connections that do not belong to any pin-buses are called fine-grain connections. A net consists of a single source and all sinks of the source. We group nets into net-buses, each containing as many pin-buses as possible.

As with VPR, we represent the routing network using the routing resource graph [3]. Extra records are used to keep track of the coarse-grain routing resources in the graph. There are five types of coarse-grain routing resources --- coarse-grain routing tracks, coarse-grain cluster inputs, coarse-grain cluster outputs, coarse-grain input pins, and coarse-grain output pins. A coarse-grain

track is a group of m tracks transporting an m-bit wide signal. Cluster inputs, cluster outputs, input pins, and output pins are grouped into coarse-grain resources based on their connectivity to the coarse-grain routing tracks. In particular, m cluster inputs are defined to be a coarse-grain cluster input, if they brings m-bit wide signals from coarse-grain tracks into their cluster. Similar definitions apply to cluster outputs, input pins, and output pins.

Our algorithm routes one net-bus at a time. First, if a bus has been previously routed, all nets in the bus are removed from the existing routing. Then, each pin-bus is routed through the coarse-grain tracks as a group. At the same time, the first bit of the pin-bus is routed through the fine-grain tracks. The coarse-grain route is accepted only if it costs less than the fine-grain route. Otherwise, all connections in the pin-bus are treated as fine-grain connections and are routed individually. The above process ensures that pin-buses will be routed on the fine-grain tracks when the coarse-grain tracks are congested. Once all the pin-buses are processed, the remaining fine-grain connections are routed individually. All nets that do not belong to any net-buses are also routed individually.

Our maze expansion algorithm uses the same basic cost function as the regular VPR maze expansion algorithm with two exceptions. First, we define the cost of a coarse-grain resource as the maximum cost of its components. Second, we added two penalty costs. One for routing fine-grain connections on the coarse-grain routing tracks; the other for routing coarse-grain connections on the fine-grain routing tracks. Using these two penalties, we encourage the coarse-grain connections to use the coarse-grain tracks and fine-grain connections to use fine-grain tracks.

We also proposed an improvement for the original VPR timing-driven router. When routing a net with multiple sinks, the VPR router sometimes will converge two branches of a routing tree. This convergence will create an artificial over capacity and render an otherwise feasible routing tree infeasible. We found the cause of the problem to be the discrepancy between two VPR cost functions --- the initialization cost and the expansion cost.

The VPR timing-driven router starts routing a new sink with an empty heap. Nodes on the existing routing tree are used to initialize the heap. The cost function used during the initialization is called the initialization cost. Once initialized, the node with the least cost on the heap is repeatedly removed from the heap and its neighbors are put onto the heap based on a different cost function -

-- the expansion cost. A convergence occurs when the expansion cost is smaller than the initialization cost for a given node. To fix this problem, we find the maximum initialization cost during heap initialization. This cost is then added to each expansion cost. This guarantees that all expansion costs will always be greater than all initialization costs, which is a sufficient condition to guarantee greater-than-initialization-cost expansion costs for each node.

Future Work

In the near future, we are planning to run a full range of experiments to determine the properties of a good coarse-grain routing architecture. A full range of issues will be studied including the best granularity for coarse-grain routing resources, the best distribution of coarse-grain and fine-grain routing resources, the best carry chain architectures, etc. We will also revisit the issue of specialized cluster inputs.

Bibliography:

[1] Pico-Java Processor Design Documentation, Sun Microsystems Inc., 1999.

[2] Synopsys Design Compiler Manual, Synopsys Inc., 1999.

[3] Vaughn Betz, Jonathan Rose, Alexander Marquardt, Architecture and CAD for Deep-Submicron FPGAs, Kluwer Academic Publishers, 1999.

[4] Don Cherepacha, David Lewis, "DP-FPGA: An FPGA Architecture Optimized for Datapaths", Proceedings of Ninth International Conference on VLSI Design, Pages 329--343, 1996.

[5] Timothy J. Callahan, Philip Chong, Andre DeHon, John Wawrzynek, "Fast Module Mapping and Placement for Datapaths in FPGAs", Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays, Pages 123--132, 1998.

[6] Miguel R. Corazao, Marwan A. Khalaf, Miodrag Potkonjak, Jan M. Rabaey, "Performance Optimization Using Template Mapping for Datapath-Intensive High-Level Synthesis", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Pages 877--888, August 1996.

[7] Scott Hauck, Thomas W. Fry, Matthew M. Hosler, Jeffrey P. Kao, "The Chimaera Reconfigurable Functional Unit", IEEE Symposium on FPGAs for Custom Computer Machines,

Pages 87--96, 1997.

[8] Andreas Koch, "Structured Design Implementation --- A Strategy for Implementing Regular Datapaths on FPGAs", Proceedings of the 1996 ACM Fourth International Symposium on Field Programmable Gate Arrays, Pages 151--157, 1996.

[9] Andreas Koch, "Module Compaction in FPGA-based Regular Datapaths", Proceedings of the 33rd Design Automation Conference, Pages 471--476, 1996.

[10] Thomas Kutzschebauch, Leon Stok, "Regularity Driven Logic Synthesis", Proceedings of IEEE/ACM International Conference on Computer Aided Design, Pages 439--446, 2000.

[11] Thomas Kutzschebauch, "Efficient Logic Optimization Using Regularity Extraction", Proceedings of 2000 International Conference on Computer Design, Pages 487--493, 2000.

[12] Alan Marshall, Jean Vuillemin, Brad Hutchings, "A Reconfigurable Arithmetic Array for Multimedia Applications", Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on Field Programmable Gate Arrays, Pages 135--143, February 1999.

[13] Ethan Mirsky, Andre DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources", Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, Pages 157--166, April 1996.

[14] A. R. Naseer, M. Balakrishnan, Anshul Kumar, "FAST: FPGA Targeted RTL Structure Synthesis Technique", Proceedings of the Seventh International Conference on VLSI Design, Pages 21--24, 1994.

[15] A. R. Naseer, M. Balakrishnan, Anshul Kumar, "Direct Mapping of RTL Structures onto LUT-Based FPGAs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Pages 624--631, July 1998.

[16] Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, Anant Agarwal, "Baring It All to Software: Raw Machines", IEEE Computers, Pages 86--93, September 1997.

[17] Andy Ye, "Ph.D. Thesis Proposal: Routing Architecture and Place and Route Tools for DP-FPGA", Technical Report, University of Toronto, June 2000.

[18] Andy Ye, "Ph.D. Annual Monitoring Report", Technical Report, University of Toronto, October 2001.

[19] Andy Ye, David Lewis, Jonathan Rose, “Structured Logic Synthesis for Datapath FPGAs”,