

FPGA Implementation of a Maze Routing Accelerator

John A. Nestor

Department of Electrical and Computer Engineering
Lafayette College
Easton, Pennsylvania 18042 USA
nestorj@lafayette.edu

Abstract. This paper describes the implementation of the L3 maze routing accelerator in an FPGA. L3 supports fast single-layer and multi-layer routing, preferential routing, and rip-up-and-reroute. A 16 X 16 single-layer and 4 X 4 multi-layer router that can handle 2-16 layers have been implemented in a low-end Xilinx XC2S300E FPGA. Larger arrays are currently under construction.

1. Introduction

The classic Lee Algorithm for *maze routing* [1] remains popular in electronic design automation because it is guaranteed to find a shortest-path connection if one exists. The algorithm represents the routing surface as a rectangular grid. During the *expansion* phase, it searches outward from the *source* node of a desired connection in breadth-first fashion while labeling each encountered node to indicate the shortest path back to the source. When the *target* node is found, the *backtrace* phase selects a path by following these labels while marking the path as an obstacle. The *cleanup* phase clears the remaining labels before additional connections are routed.

Although popular, the Lee Algorithm is computationally expensive. For single layer routing, expansion is $O(d^2)$ for a connection of distance d , and cleanup is $O(N^2)$ for an $N \times N$ grid. Multilayer routing is even more costly. This has motivated several proposals for hardware accelerators. *Direct grid* accelerators (e.g., [2]) map each node into an array of simple processing elements (PEs). This reduces the expansion time to $O(d)$ and cleanup time to $O(1)$ but requires N^2 processing elements for a single-layer. *Virtual grid* accelerators (e.g. [3]) map more than one node onto each PE. This requires fewer PEs, but each PE must be significantly more complex to handle the multiple mappings. Other acceleration approaches include raster pipelines [4], specialized processors [5], and accelerators intended for routing FPGAs, which have a specialized routing structure [6].

This paper describes a return to the direct grid approach called L3. L3 improves over previous direct grid approaches by providing (1) efficient support for multiple layers; (2) a significant reduction in PE logic; and (3) support for quick initialization and removal of obstacles and connections during rip up and reroute. A preliminary version of L3 was described in [7]; this paper describes a refined version and its implementation in an FPGA.

2. The L3 Architecture

Figure 1 shows the general organization of L3S, the single-layer version of L3. Each *cell* (PE) is connected locally to neighboring cells in the grid using output XO. XO is true when a cell is labeled during expansion. It is connected to the WI, EI, NI, and SI inputs of the cell's west, east, north, and south neighbors, respectively. An attached control unit (not shown) communicates with the array of cells using a global command bus CMD, and a row and column decoder that allow the selection of either individual cells or rectangular regions of cells. Cells pass status information back to the control unit using a global tristate/wired-OR STATUS bus. Global input PF supports preferential routing and will be discussed later.

Each cell is a finite state machine with 6 states: E (empty), BL (blocked), XE (expanded east), XW (expanded west), XN (expanded north) and XS (expanded south). Table 1 describes the function of each cell as it responds to one of four commands: CLEAR, SET, EXPAND, and TRACE.

To perform maze routing, the control unit first selects all cells and broadcasts a CLEAR command to set all cells to the E state. It then selects the source node and uses the SET command to set the source cell to the XE state. It next applies the EXPAND command while selecting the location of the target cell. During each successive clock cycle, a cell will enter an expanded state if one of its neighbors is in an expanded state and the corresponding preference input (PF1 for east/west, PF0 for north/south) is high. Expansion continues until either the target cell is reached (at which point status bit S0 is pulled low) or expansion has failed (in which case "watchdog" status bit S1 goes high).

If expansion is successful, the control unit starts the backtrace phase by selecting the target cell and broadcasting the TRACE command. In response, the target cell asserts its state code on the STATUS bus and enters the obstacle state (BL). The control unit uses the direction information encoded in the state code to determine the address of the next cell in the path and repeats this process until the source node is reached and the entire path is marked as an obstacle. To route another connection, the CLEAR command is applied with no cells selected to remove expansion labels (but not obstacles) and the process repeats. The entire process takes d clock cycles for expansion, d clock cycles for backtrace, and 1 clock cycle for cleanup.

L3M is the multi-layer version of the L3 architecture. It uses the same array structure as L3S but time-multiplexes cell hardware over multiple layers. Figure 2 shows the organization of a single L3M cell, which uses a shift register to store the states of each layer. The L3M cell processes states from bottom to top on each successive clock cycle. The state sequencer is similar to that of the L3S cell except that it has two additional states XU and XD to support vertical expansion. Expansion information from the layer "above" the current layer is taken from the next shift register stage. Expansion information from the level "below" the current layer is stored in a flip-flop at the end of the preceding cycle. Both calculations are suppressed by the /TOP signal when the top layer is reached. The preferential routing input PFV allows vertical expansion to be suspended; when zero the sequencer recirculates the current layer so that horizontal expansion can continue on successive clock cycles. The L3M array will find a connection in $O(L*d)$ cycles for L layers.

3. Implementation Results

As a proof of concept, the modules of the L3S and L3M accelerators have been coded in Verilog HDL and synthesized using the Xilinx ISE 4.2i and Synopsys FPGA Express tools targeting a Xilinx XC2S300E FPGA [8] on a Memec Development Board [9]. The synthesized L3S cell requires 17 4-input lookup tables (LUTs), 3 D flip-flops (DFFs), and 3 tristate buffers (TBUFs). The L3M cell requires 32 LUTs (including 3 SRL16 shift registers), 3 DFFs, and 3TBUFs.

Table 2 shows the results of synthesizing the complete L3 router including cell array, decoders, control unit, and a serial interface to communicate with a host computer. All designs were tested and work properly, although the 16 X 16 single-layer design failed to meet the timing constraint.

The LUT requirements of the L3S and L3M cells can be used to predict the size of router that can be accommodated by a larger FPGA. For example, the Virtex-II Pro XC2VP125 device [8] contains 111,232 LUTs and could accommodate an 82 X 82 array of single-layer L3S cells or a 58 X 58 array of L3M cells.

5. Conclusion

This paper has described a new architecture for a direct-grid hardware routing accelerator and its implementation using FPGAs. The L3 architecture supports multiple layers, preferential routing, and iterative rip-up-and reroute. A working 16 X 16 single-layer router and 4 X 4 multi-layer have been demonstrated in a low-end FPGA. Future work will include implementation of larger routers, further design refinement to improve clock cycle time, and detailed performance comparisons of the hardware router to software implementations.

References

1. Lee, C. Y. "An Algorithm for Path Connections and its Applications," *IRE Transactions on Electronic Computers* vol. EC-10, no. 2, pp. 346-365, 1961
2. Breuer, M., and Shamsa, K. "A Hardware Router," *Journal of Digital Systems*, vol. IV, no. 4, pp. 393-408, 1981
3. Ventkateswaran, R., and Mazumder, P., "Coprocessor Design for Multilayer Surface-Mounted PCB Routing," *IEEE Trans. VLSI Systems*, vol. 1, no. 1, 1993
4. Rutenbar, R. and Mudge, T., "A Class of Cellular Architectures to Support Physical Design Automation," *IEEE Trans. CAD*, Vol. CAD-4, No. 4. October 1984
5. Won, Y, Sahni, S., and El-Ziq, Y., "A Hardware Accelerator for Maze Routing", *Proceedings Design Automation Conference*, June 1987
6. Huang, R, Wawrzyniek, J., and DeHon, A., "Stochastic, Spatial Routing for Hypergraphs, Trees, and Meshes", *Proc. International Symposium on FPGAs*, February 2003
7. Nestor, J. "A New Look at Hardware Maze Routing", *Proceedings Great Lakes Symposium on VLSI*, March 2002
8. Xilinx, Inc., *Xilinx Databook*, 2003. Available online at <http://www.xilinx.com>
9. Memec, Inc., *Spartan-II Development Board User's Guide*, 2002

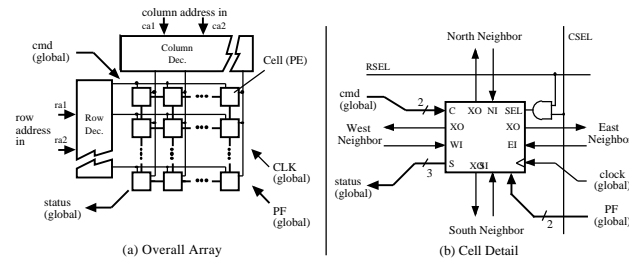


Fig. 1. L3S Organization. The attached control unit is not shown

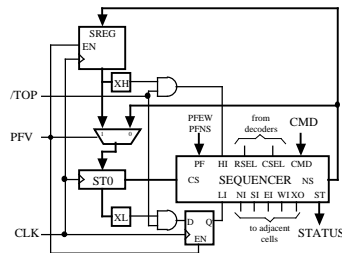


Fig. 2. The L3M Cell design. Layers are processed from bottom to top

Table 1. Cell state sequencing in response to CMD. X* indicates any expanded state

CMD	SEL	EW * PF1	WI * PF1	NI * PF0	SI * PF0	PS	NS	S1	S0
CLEAR	0	-	-	-	-	X*	E	1	1
	1	-	-	-	-	Any	E	1	1
SET	1	-	-	-	-	Any	XE	1	1
TRACE	1	-	-	-	-	Any	BL	D1	D0
EXPAND	-	1	-	-	-	E	XE	0	SEL'
	-	0	1	-	-	E	XW	0	SEL'
	-	0	0	1	-	E	XN	0	SEL'
	-	0	0	0	1	E	XS	0	SEL'
	-	-	-	-	-	X*	PS	1	SEL'
All other conditions							PS	1	1

Table 2. L3S and L3M implementation costs and predicted performance

Array Size	Serial LUTs/FFs	Control LUTs/FFs	Array LUTs/FFs	Total LUTs/FFs	Clock (ns)
4 X 4	173 / 141	167 / 15	260 / 48	600 / 204	38ns
8 X 8	173 / 141	171 / 19	1083 / 192	1427 / 352	40ns
16 X 16	173 / 141	370 / 23	4067 / 768	4610 / 932	51ns
4 X 4 X 4	173 / 141	238 / 23	425 / 50	836 / 214	41ns