Issues of Wirelength Cost Models in Routing-Constrained FPGAs

Kenneth Eguro and Scott Hauck
{eguro, hauck}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500

UWEE Technical Report
Number UWEETR-2004-0006
February 26, 2004

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Issues of Wirelength Cost Models in Routing-Constrained FPGAs

Kenneth Eguro and Scott Hauck
{eguro, hauck}@ee.washington.edu

Dept of EE, University of Washington at Seattle
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2004-0006
February 26, 2004

Abstract

Contrary to prevailing consumer conceptions of efficient silicon use, previous research efforts have shown that designing routing-poor FPGAs may yield significant area gains. In this paper we show that conventional wirelength-centric placement tools are unable to deal with the challenges that routing-limited CAD problems present. We believe that this problem is present given today's architectures and will become more important as devices scale.

1 Introduction

Although the FPGA user base has pushed manufacturers to constantly improve LUT utilization, this practice does not insure the best use of the underlying silicon. Unfortunately, a huge amount of interconnection resources are required to provide very high routability across a wide range of circuits for today's large gate-count devices. This problem is further complicated by the natural square arrangement of FPGA tiles since the Rent's exponent is somewhat tied to the perimeter of the array. Either way, routing-poor reconfigurable architectures will become a fact of life in the future. Unfortunately, since conventional placement tools do not consider the difficulties that might be encountered during routing, they will tend to produce very tight placements that are unable to be routed given the available resources. In this paper we demonstrate this phenomenon using VPR and the twenty "Toronto Place and Route Challenge" MCNC benchmark netlists.

2 Problem Formulation

While functional pieces of an FPGA, such as LUTs and monolithic multipliers/memory cores, only occupy approximately 5-10% of the overall die area in current generation FPGAs, the communication pieces necessary to allow near-100% logic utilization occupy the vast majority of the silicon. The work in [2] investigated how LUT utilization was tied to area utilization of the device. Investigating hierarchical FPGA architectures, the author found that requiring 100% LUT utilization ended up creating very large devices and that by maintaining 60-70% LUT utilization one could half that area. While these findings are limited to the specific type of architectures that the author explored, this does introduce some doubt that it is wise to pursue extremely high LUT utilization.

Another scenario that might cause the bulking of routing resources is that very small, highly interconnected sections of a netlist might dictate the overall interconnection architecture. To determine the channel width necessary for a given netlist, the popular place and route suite, VPR [1], places the netlist using simulated annealing, then performs a binary search increasing and decreasing the number of wires per routing channel to determine the minimum allowable global channel width. While VPR chooses to globally decide the minimum channel width for architecture flexibility and design ease, this also means that a small, highly interconnected portion of the circuit dicates the channel width for the entire chip. This is seen in Figure 1a and 1b. However, this does not accurately reflect the average routing requirements of the chip since the majority of the circuit will likely be a more regularly connected datapath. If this highly connected region were to be spread out, as shown in Figure 1C, this would more evenly distribute the routing needs of the chip and the maximum necessary channel width would be greatly reduced. Unfortunately, most existing placement tools are unable to realize this routing distribution since they utilize wirelength-centric cost functions. In this way, they are only able to minimize wirelength without consideration for congestion.

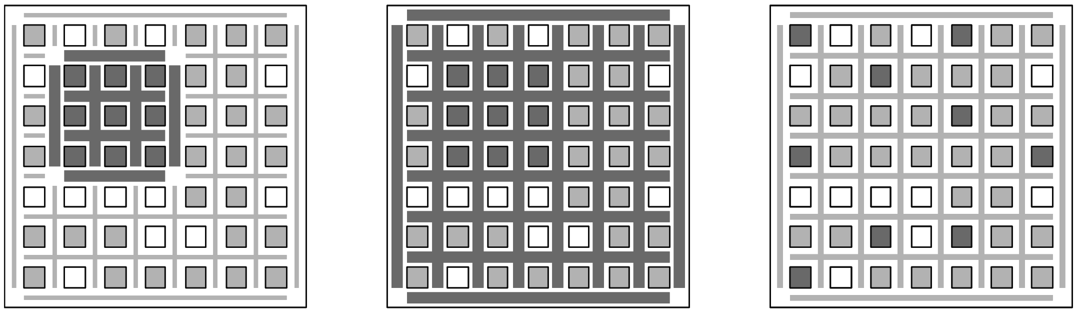

Figure 1 A,B,C – Channel Width Moderation
1A shows only a small portion of the circuit is highly connected and requires a wide channel width. 1B shows that CAD and design issues cause channel widths across the chip to increase to accommodate this small sub-circuit. 1C shows that this highly connected region could be spread out so that the overall routing resource requirements are more evenly distributed throughout the chip.

A more serious consequence of this failure goes beyond architecture exploration and development and into fundamental placement and routing for realistic architectures. If we attempt to recreate the work done in [2] for island-style FPGAs or we follow the natural routing-poor scaling tendencies of 2-D arrays, we realize that today's CAD tools are unable to successfully map to architectures that have limited routing resources. In these cases, they fail to place and route despite the fact that viable mappings exist. For example, consider the case shown in Figure 2A and 2B. Figure 2A is the result of the normal VPR place and route toolflow. The same relative placement and routing should work if we only have an architecture that is quadruple the size, but with half the channel width and double the segment length. Unfortunately, it is unlikely that a wirelegth-centric placement will find this arrangement.

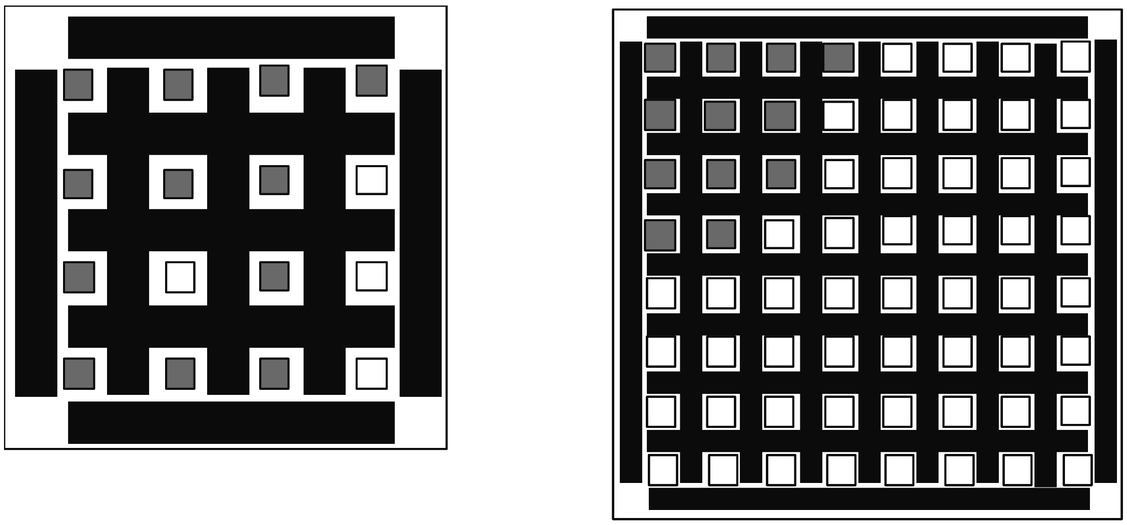

Figure 2 A,B – Logic and Routing Array Growth
The 4x4array shown in 2A has a channel width of 2X. The 8x8 array shown in 2B should be only marginally larger if we give it a channel width of 1X and double the segment length. The placement via conventional CAD tools is far too clustered to route on the modified architecture despite that a viable placement exists.

3 Testing and Results

To demonstrate the shortcomings of conventional placement and routing tools we forced VPR into slightly routing-constrained, but logic-rich architectures. First, we determined the desired routing channel width given a logic-bound architecture. We chose to map these netlists to the Toronto "4x4_sanitized" architecture which features four 4-LUTs per CLB, and both buffered and unbuffered length 4 segments. The default settings of VPR provide a netlist with the smallest square architecture that it will fit on. After placement, it performs a binary search to find the minimum channel width that will allow the netlist to route. These results are shown in Figure 3. After finding the minimum desired channel width, we re-ran VPR, but provided an architecture that had one, two or three fewer tracks than the circuit desired in the logic-bound case. To offset the new lack of routing, we allowed the array to grow up to four times the minimum square array.

Netlist	# CLBs	Array Size (N^2)	Tracks	CW – 1	CW – 2	CW – 3
alu4	390	20	33	fail		
apex2	485	23	43	fail		
apex4	335	19	41	fail		
bigkey	427	27	24	1	fail	
clma	2133	47	51	fail		
des	415	32	24	fail		
diffeq	379	20	29	0	fail	
dsip	343	27	25	fail		
elliptic	906	31	40	2	fail	
ex1010	1201	35	45	2	fail	
ex5p	278	17	43	1	1	fail
frisc	894	30	43	fail		
misex3	361	19	37	fail		
pdc	1187	35	61	fail		
s298	490	23	28	4	fail	
s38417	1609	41	36	fail		
s38584.1	1614	41	35	1	41	
seq	448	22	40	fail		
spla	955	31	56	0	fail	
tseng	266	17	25	fail		

Figure 3 – Logic-bound and Routing-Restricted Architecture Results
The first 3 columns show the results of running the 20 Toronto benchmarks through VPR in the normal logic-bound array and binary-search channel width manner. Note that the netlists in red are I/O bound, not logic bound. The three final columns show the results when given an architecture that has 1, 2 or 3 fewer tracks than desired in the logic-bound case. The array was allowed to grow to four times the original, logic-bound array size before being declared a failure. When successful, we indicate how many additional rows and columns needed to route the netlist.

As Figure 3 shows, 60% of the netlists fail when only given even one less routing track per channel. Furthermore, this falls to 90% when two less routing tracks are given. Finally, all of the netlists fail when given three less routing tracks. These results are particularly surprising when examined more closely. First, the average channel width is 38 tracks. This means that one, two and three less routing track only corresponds to an average of less than 3%, 6% and 8%, respectively, fewer tracks per channel. Given that the netlists were given architectures that approached 400% more overall chip-wide routing resources, we can virtually guarantee that viable mappings existed, but that VPR was unable to find them. Furthermore, we can see that 2 of the netlists routed in exactly the same sized array, that is, the previously found binary search "minimum" was not real lower bound but subject to some variations due to random placement variations. In those cases, the channel width minus two test was really the channel width minus one case. Of course, these random variations might also affect our results in the opposite manner – we declared a failure, but the circuit can place and route given fewer routing tracks if we start with a different placement seed. While more extensive testing could increase the accuracy of our results, we still believe that the basic findings are true – conventional wirelength-centric placement is not sufficient given routing-poor reconfigurable architectures.

[1] Betz, Vaughn and Jonathon Rose. "VPR: A New Packing, Placement and Routing Tool for FPGA Research." *International Workshop on Field Programmable Logic and Applications*, 1997: 213-22.

[2] DeHon, Andre. "Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization)." *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 1999: 69-78.