The Stratix II Logic and Routing Architecture

David Lewis*, Elias Ahmed*, Gregg Baeckler, Vaughn Betz*, Mark Bourgeault*, David Cashman*, David Galloway*, Mike Hutton, Chris Lane, Andy Lee, Paul Leventis*, Sandy Marquardt*, Cameron McClintock, Ketan Padalia*, Bruce Pedersen, Giles Powell, Boris Ratchev, Srinivas Reddy, Jay Schleicher, Kevin Stevens*, Richard Yuan, Richard Cliff, Jonathan Rose**

Altera Corporation, 101 Innovation Drive, San Jose, CA, 95134

(*) Altera Corporation, 151 Bloor St W., Toronto, Ont, Canada M5S 1S4

(**) Department of Electrical and Computer Engineering, University of Toronto,

10 King's College Road, Toronto, Ontario Canada M5S 3G4

dlewis@altera.com

ABSTRACT

This paper describes the Altera Stratix II™ logic and routing architecture. This architecture features a novel adaptive logic module (ALM) that is based on a 6-LUT, but can be partitioned into two smaller LUTs to efficiently implement circuits containing a range of LUT sizes that arises in conventional synthesis flows. This provides a performance increase of 15% in the Stratix II architecture while reducing area by 2%. The ALM also includes a more powerful arithmetic structure that can perform two bits of arithmetic per ALM, and perform a sum of up to three inputs. The routing fabric adds a new set of fast inputs to the routing multiplexers for another 3% improvement in performance, while other improvements in routing efficiency cause another 6% reduction in area. These changes in combination with other circuit and architecture changes in Stratix II contribute 27% of an overall 51% performance improvement (including architecture and process improvement). The architecture changes reduce area by 10% in the same process, and by 50% after including process migration.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]:

General Terms

Design

Keywords

FPGA, logic module, routing

1. INTRODUCTION

This paper describes the Stratix II FPGA logic and routing architecture. The goals for Stratix II were to improve both performance and area compared to Stratix™, independent of the process change. While the process shrink from 0.13um to 90nm could provide approximately 20% performance improvement, this was considerably short of the 50% performance increase that was desired. Although Stratix II includes a number of new features as well as circuit optimizations, the single largest source of architectural performance improvement was the development of a new adaptive logic module (ALM), offering 15% performance compared to a 4-LUT. The inclusion of a set of fast routing connections offered another 3%. The remainder of the performance improvements will not be described in this paper.

Section 2 of the paper gives a brief overview of the Stratix architecture and describes the tools used to evaluate logic and routing architectures. Section 3 describes the development of the adaptive logic module used in Stratix II. It describes various structures for a larger LUT, and the evaluation of them together with the appropriate LAB size. Arithmetic features and carry chain modifications to enhance routability are also described. Section 4 describes modifications of the routing architecture, including fast routing multiplexers and the reduction in routing channel widths used in Stratix-II. Section 5 concludes the paper.

2. ROUTING ARCHITECTURE

This section describes the development of the routing architecture for Stratix. It first briefly describes the FPGA modeling toolkit (FMT) experimental infrastructure used to evaluate various routing architectures, and changes since previous description of it [1]. The new Synthesis modeling toolkit is also described.

2.1 Experimental Infrastructure

The routing architecture was developed using the Altera FPGA Modeling Toolkit (FMT) and Synthesis Modeling Toolkit (SMT.) Based on the academic VPR place and route tool [3] [10], FMT extends the VPR framework to deal with the level of complexity of modeling the details of a state of the art FPGA architecture, and was first used successfully to design the Stratix™ architecture [1].

Since the use of the FMT in developing the Stratix architecture, the SMT has been developed to allow the exploration of logic block architectures. The SMT allows synthesis from HDL into LUT-based architectures, and requires some customization for

FPGA'05, February 20–22, 2005, Monterey, California, USA.

each proposed architecture to efficiently exploit irregular features such as arithmetic structures and control signals for the flip-flops.

The FMT is used to perform place and route experiments on a proposed architecture. The FMT was described in [3] so only a short description is included here. Input to the FMT is an architecture file that describes the various logic and memory resources, and a hierarchical description of the architecture. This includes all timing and physical information needed to perform placement and routing on FPGAs of any specified size, and to determine the overall area and performance of a proposed architecture.

Although similar to the methodology described in [3] [8] [10], the architectural evaluation methodology is extended due to the introduction of the SMT. Given a new logic block candidate, the SMT must be modified to support synthesis into this block. A set of benchmark circuits are then synthesized and used as input to the place and route flow. For any proposed logic block, it is important to determine the best routing architecture for that block. This is done using the well known binary search place and route to determine the minimum channel width. Given the set of circuits, a candidate channel width is then selected that meets the routing demand of the circuit set. While previous work suggested that a 20% increase beyond average minimum channel width was a useful metric for comparing architectural features, a commercial architecture needs to be able to route all of the benchmark set, not just the average, so increased weight is placed on the largest channel width. Because a single outlier result with bad routing can result in a large channel width, our methodology does not strictly use the absolute maximum channel width, but some discretion is allowed. To compare architectures, specific candidates with routing that is sufficient to route on the order of 99% of the benchmark circuits is used, and the area and performance of the proposed architecture computed using the FMT. The placement and routing algorithms use the same basic algorithms as production Altera software, so the FMT can provide a good prediction performance of prototype architectures.

2.2 LAB-Based Architecture

Most Altera devices have used a LAB-based architecture, including the Stratix [1], Flex 6000 [7], Flex 8000, Flex 10K, Apex [6,7], and Mercury [4] architectures. Stratix II continues to use a LAB-based architecture, but with substantial changes to the logic element, as well as some changes to the routing architecture.

A LAB-based routing architecture consists of a highly connected block of logic elements connected to a sparser inter-LAB routing fabric. Stratix II has two levels of hierarchy of routing resources. The lowest level of the architecture is a logic element (LE) which in previous architectures comprises a LUT based logic function and flip-flop, and in Stratix II, includes adaptive logic module and 2 FFs. The first level of routing hierarchy is formed by a collection of logic elements (10 in most previous architectures) which are grouped into a logic array block (LAB). LEs within the same LAB can be interconnected by intra-LAB routing. Figure 1 shows an overview of a LAB, using a conventional 4-LUT as an example. The intra-LAB routing wires consist of LAB lines which route signals external to the LE to the input pins of the LEs, and local lines, which route the outputs of the LEs to inputs of LEs within the same LAB. In Apex and prior architectures each input pin of the LE could connect to any one of the signals in some pool of LAB and local lines. In Stratix and subsequent architectures the connectivity was reduced to 50%, so each input pin connects to half the wires. The LAB also contains a control signal region, typically located in the center of the LAB for reduced delay to the LEs, which conditions and buffers control signals for the FFs and contains other control logic.

The second level of routing hierarchy is formed by a number of rows and columns of routing wires connecting the inputs and outputs of the LABs, shown in Figure 2. The rows and columns will be referred to as H or V wires (horizontal and vertical) for brevity in this paper. Stratix and Stratix II use a three-sided routing architecture, in which each LAB can drive signals or listen to signals on one H channel, and either one of the two V channels on the left or right sides.

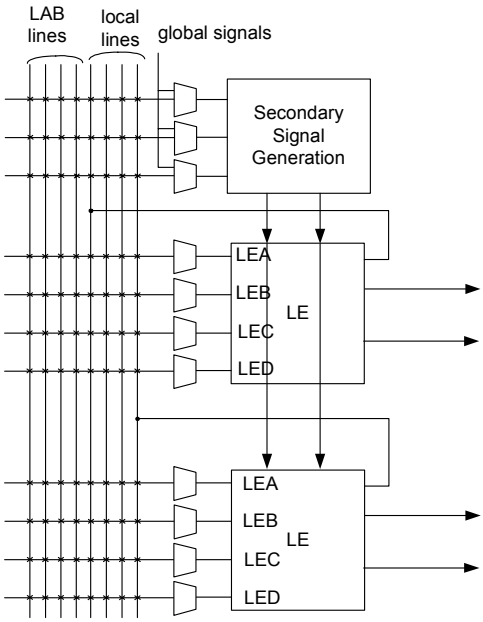

Figure 1: Overview of LAB

3. Adaptive Logic Module Architecture

Based on the desire for improved speed, and largely inspired by previously reported results [2,13], a larger LUT was a key area of interest. The 6-LUT has recently been reported as having better area-delay performance than the 4-LUT, with estimated 14.4% [13, table E.10] performance improvement for a 10-LE LAB, but 17.2% area increase [13, table A.10] for a 10-LE LAB. Other LAB sizes are possible, for different area-delay tradeoffs. For Stratix II, the goal was to achieve as much speed improvement as possible but avoid the area increase if possible. The basic approach that was investigated was based on the ability to construct a larger LUT to reduce the number of levels of logic and increase performance. To avoid the area increase, methods to split the larger LUT efficiently into smaller LUTs were investigated. Two observations help motivate this goal. First, although

synthesis into the conventional 4-LUT produces mostly 4-LUTs as well as a mix of smaller functions, for a 6-LUT the mix is less skewed towards the largest size because not as many large cones of logic exist to be absorbed into a single function. As a result there is a higher proportion of 5-LUTs and smaller functions. Figure 3 shows the distribution of the number of used inputs for LUTs of various sizes, with 4-LUTs using all inputs 57% of the time, but 6-LUTs using all inputs only 36% of the time. There is a 24% reduction in LE count using the 6-LUT, but this is clearly insufficient to achieve an overall area reduction. Thus the 6-LUT will be used to its capacity less often, resulting in a potential waste of its functionality.

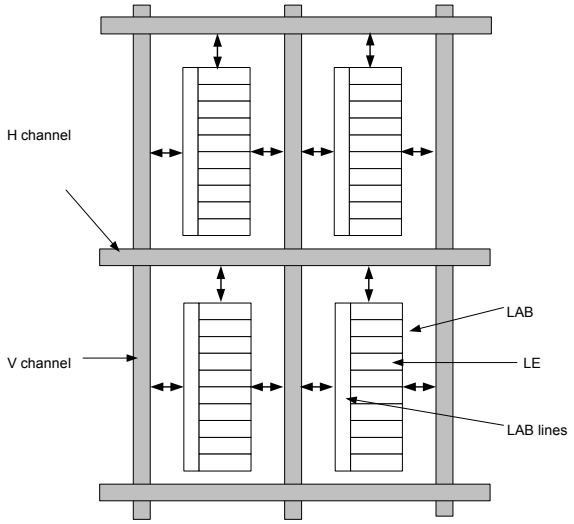

Figure 2. Overview of routing structure

The second observation is that not all paths in a logic circuit have identical depth, so there is the potential for synthesis to use a smaller and less costly LUT for non-critical paths with narrow functions, and to use 6-LUTs only for wider cones of logic to reduce the depth on the critical path. Our investigations were therefore focused on structures that offered the full speed potential of the 6-LUT for performance critical functions, but can also efficiently partition into smaller LUTs for smaller logic functions, or less critical logic that can by synthesized into smaller LUTs.

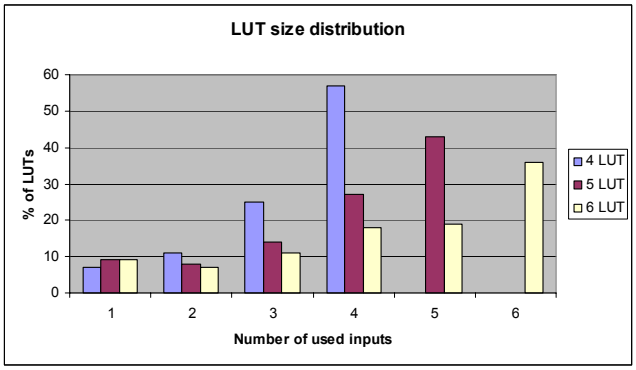

Figure 3: LUT input pin usage distribution

3.1 Logic Block Architectures

There are three approaches that were identified for constructing a set of LUT sizes within a single logic block. All three of these will be described below, although the first was quickly discarded from further consideration and the second and third were the subject of detailed FMT experiments.

The first approach is denoted a composable LUT. In a composable LUT, a set of basic k-LUTs are constructed, together with a tree of 2:1 multiplexers on their outputs to form progressively larger LUTs. A key point is that all of the inputs to the k-LUTs and the subsequent multiplexers have independent inputs from the routing fabric. Figure 4 shows an example composable 6-LUT built out of 4-LUTs. LUT inputs will be labeled a,b,c etc corresponding to their depth in the multiplexer tree when used in widest mode. The independence of the inputs in the composable LUT is also the source of a high cost. A cursory examination of the costs of constructing a composable LUT can be made by considering the final stage of routing alone. In previous Altera architectures, the final stage of routing that connects to the input pin of the LUT consists of a multiplexer with fanin typically ranging from 21:1 to 30:1, requiring from 10 to 13 memory bits to control the selection. A 4-LUT alone consists of a 16:1 multiplexer and 16 memory bits. Thus, each input pin incurs a cost roughly comparable to the cost of a 4-LUT. Although this simplified analysis does not consider the difference in transistor sizes, typically larger in the final stages of the LUT to provide speed, it can be seen that each input pin to the logic block is nearly as expensive as a 4-LUT. Thus the composable 6-LUT, requiring a total of 19 input pins and 4 4-LUTs, is at least as expensive as 4 4-LUTs and considerably more expensive than a conventional 6-LUT with only 6 input pins. This makes the use of 6-LUTs sufficiently expensive that this structure was not considered attractive, and no further consideration of it was undertaken.

This observation drives towards large LUTs with fewer input pins, that can implement a range of LUT sizes while minimizing the amount of replication of the routing. The second structure is denoted a fracturable LUT mask (FLM). A fracturable LUT can be parameterized by the two values k and m. Describing a FLM as a k,m LUT, the value k denotes the size of largest LUT, and the value m denotes the number of extra input pins that are brought into the logic element, for a total of k+m input pins. Thus the 6,2 fracturable LUT implements a 6-LUT with a total of 8 inputs.

In a FLM, the *k*th input, and each of the m extra inputs forms the inputs to a multiplexer tree of depth (m+1). When used to build two smaller LUTs, the logic in the top half of the LUT is used for one logic function, taking the output before the final 2:1 multiplexer that forms the k-LUT output. The *k*th input is not required for functions smaller than size k, so it can be used to drive the select input of a multiplexer that corresponds to the k-1 depth in the bottom half of the LUT. Similarly, the other m inputs are not required for the largest possible k LUT, so they also drive copies of the multiplexer tree using data from the bottom half of the k-LUT. Thus, when the LUT is used to implement a single k-input function, m of the inputs must be duplicated. Figure 5 illustrates how inputs c0 and c1, and inputs d0 and d1 must be duplicated to form the 6 input function z0(a,b,c,d,e,f).

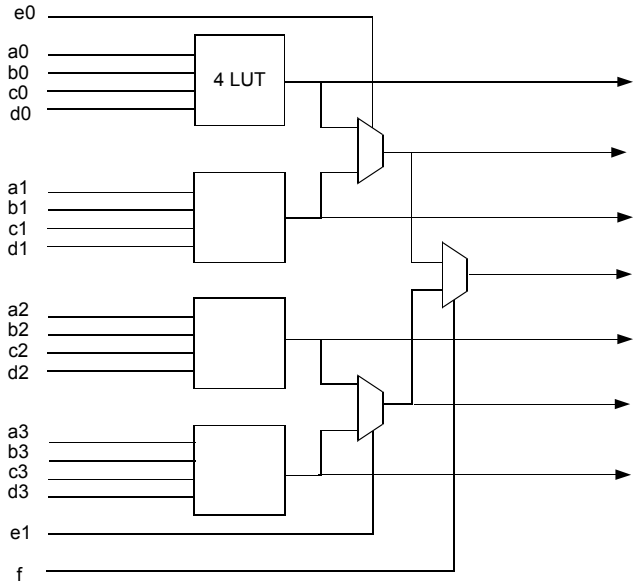

Figure 4: Composable 6-LUT constructed from 4-LUTs

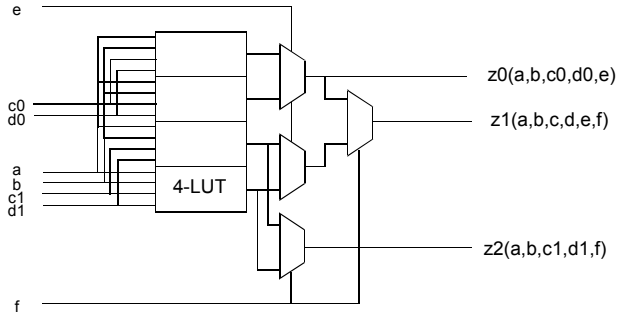

Figure 5: 6,2 Fracturable LUT mask logic module

The FLM can also implement two functions of size up to k-1. In the case that the LUT is used for the smaller functions, each function can use m+1 distinct inputs, but must share (k-1)-(m+1) = k-m-2 inputs. That is, for the 6,2 LUT used as two 5-LUTs, three inputs are distinct for each of the two 5-LUTs, and two inputs must be shared. In Figure 5 it can be seen that the FLM can also implement both z1(a,b,c1,d1,e) and z2(a,b,c2,d2,f), so inputs a and b must be shared when the FLM is used to construct two 5-LUTs. Because the FLM can be partitioned into two 5-LUTs, progressively smaller LUTs will use the 5-LUTs, but do not need to share as many inputs. For example, the example 6,2 LUT can also implement two 4-LUTs with completely independent inputs. Other combinations, such as a 5-LUT and 3-LUT that do not share inputs, or 5-LUT and 4-LUT that share one input are also possible. In summary, a k,m FLM has a total of k+m inputs and can be used to construct either one k-input function, or two functions of size up to k-1 that use no more than k+m distinct inputs. This also enables an ability for synthesis to target either 6-LUTs for higher performance or a mix of 5-LUTs and 6-LUTs for area efficiency.

It is typically expected that two flip-flops would be used with an FLM to support packing up to two logic functions each directly feeding a FF.

The third approach is a shared LUT mask (SLM). Previous architectures targeting data path applications have proposed collections of LUTs that share all memory bits, but have distinct inputs to each of the multiplexer trees [11]. This was intended to allow the efficient implementation of data paths that implement a number of copies of identical logic functions on each of the bits of data, but is too restrictive for general purpose logic. Instead, a variation that a supports the features of the FLM but extends beyond this by allowing LUT mask sharing for the full k input LUT but without providing unique inputs for each of the two functions. A k,m SLM can implement the same functions as a FLM, but can also implement two k-LUTs that have the same logic function, and no more than k+m distinct inputs. While both FLM and SLM allow any pair of logic functions subject to these constraints, the key difference between an FLM and the SLM is the ability of the SLM to implement two identical functions of k inputs in a single logic module, provided that the two functions share (k-m) inputs.

Figure 6 shows an overview of a 6,2 SLM. The SLM operates by introducing an extra level of multiplexing at the penultimate level of the mux tree. This level of muxing can be controlled either by an input pin, or tied to a 1 (top half) or 0 (bottom half). When an input signal is selected, both top and bottom half of the LUT can implement the same k-input function, using no more than k+m distinct inputs. When a 0 or 1 is input to these multiplexers, the top and bottom half of the LUT are divided into independent partitions, and two distinct functions can be implemented as in the FLM.

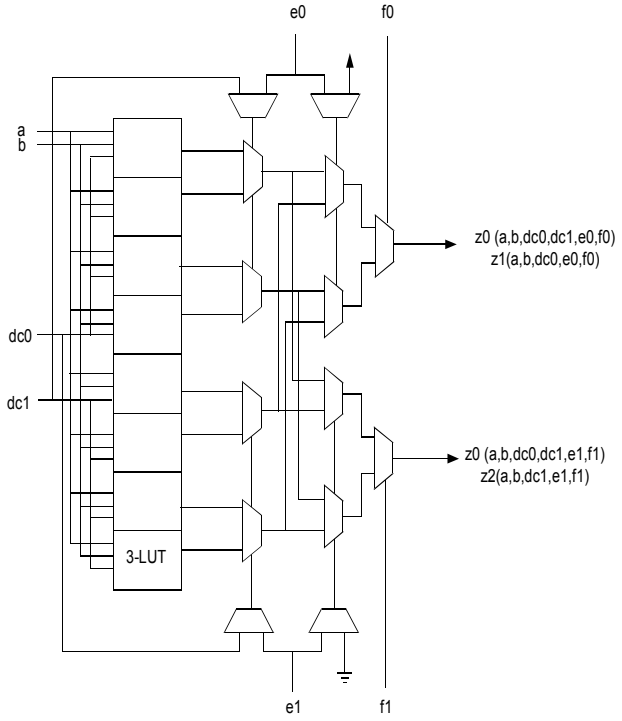

Figure 6: 6,2 Shared LUT mask logic element

The SLM structure was not known when the Stratix II architecture evaluation began, so the first step was to explore the viability of

the FLM. An initial circuit design and area estimate as well as circuit simulations provided the area and delay of the 6,2 FLM. The initial area estimates suggested that the 6,2 FLM and FFs (not including any routing multiplexers) would be approximately 2.4X the area of a 4-LUT, well below the >4X increase required for a composable LUT structure.

Using a Stratix-like routing architecture, minimum channel width binary search place and route was used to determine the required channel width for two candidates. The first was a conventional 10 LE 4-LUT LAB and the second was a 10 LE 6,2 FLM LAB. Both architectures were designed to have routing that provided comparable routability across a range of designs. The 6-LUT FLM was on average 12.4% faster and consumed only 2.1% more area, achieving a net 9.2% reduction in area-delay product. Although the LE by itself is a factor of nearly 2.4X as large as the 4-LUT, the overall LAB area does not increase by this ratio, since the routing makes up most of the overall area, and fewer LEs are required to implement the same logic.

The next step was to investigate the effectiveness of the SLM. This requires placing pairs of 6-LUTs with identical functions and 4 shared inputs in the same LE. Since the FMT is unaware of logic functionality, the SMT was enhanced to support passing the LUT mask into the flow, finding suitable pairs of 6-LUTs, and producing netlist constraints that forced these pairs to be placed in the same LE. Although the area overhead of SLM compared to FLM was estimated as 1.2%, the question was whether it was possible to find enough candidate 6-LUTs with identical functionality that could be shared in the same LE to produce a net area win. In fact, experiments showed that a 5.8% reduction in LE count could be achieved for a net area improvement of 4.7%, although a 1.3% reduction in performance was also found. However, subsequent tuning of the 6-LUT circuit design had improved its performance, so after the 1.3% performance loss, the net SLM performance advantage was 15%. Of the 1.3% performance degradation, about 0.4% is due to the SLM hardware and the remaining 0.9% due to the constraints of the LUT pairs using the SLM feature. This was judged to be a good tradeoff and SLM was adopted for Stratix II, with a net area reduction of 2.6% compared to the original 4-LUT based LAB. Thus the ALM not only achieves a 15% performance improvement, but manages to shrink logic and routing area by 2.6%, for an overall 17.6% improvement in area-delay product.

3.2 Logic Blocks per LAB

After determining that the 6-LUT SLM was the preferred logic block, it was necessary to determine the best LAB size. Although the 10-LE 6-LUT had been demonstrated to be better than the 10-LE 4-LUT, it was not clear if this was the optimal size. It is a separate issue whether the 10-LE 4-LUT was optimal, but the investigation of this did not lead us to change our conclusions on the 6-LUT. Further experiments used the FMT area model and timing models that included all physical wire length scaling for various sizes of LAB to evaluate LAB sizes ranging from 6 to 14 LEs per LAB. LABs with fewer LEs incur more inter-LAB routing, but the routing resources inside the LAB are faster. Conversely, larger LABs need less inter-LAB routing, but the intra-LAB routing is slower. The area model takes into account the appropriate amount of routing, both inter-LAB and intra-LAB to predict overall LAB area. The timing model is also constructed to use correct intra- and inter-LAB delays, assuming that the LAB is laid out as a rectangle with a 2:1 V:H aspect ratio. It can be seen from the results in Table 1 that 10 LEs appears to be area minimum, but the performance of LABs range from 8 to 14 LEs has no consistent trend, due to experimental noise, with any one of these likely to provide comparable results.

Two factors guiding the final LAB size choice were more detailed layout considerations, and a study of performance as a function of the physical aspect ratio on the LAB. The initial experiment on LAB size assumed that the physical aspect ratio of the LAB remained constant at 2:1 height:width across various LAB sizes. Although this is a reasonable approximation for his is not likely true in practice, as the LAB is most efficiently laid out as a column of LEs stacked vertically, with the consequence that larger LABs have more vertically skewed aspect ratios. Figure 7 shows the results of an experiment where the physical aspect ratio of the LAB was varied from 1:1 to 3:1. LABs with more skewed aspect ratios tend to have lower performance due to the difference in delay in the X and Y directions. The faster signal propagation in one direction is not sufficiently compensated for by the slow down in the other direction. Together with detailed layout considerations using the expected amount of routing, this led us to select 8 LEs for the Stratix II LAB.

Table 1: Area and performance results for LABs of various sizes

LAB Size Results Summary				
#ALEs per LAB	# LAB lines	Channel Width	Fmax	LAB Area
6	30	188	-4%	5.50%
8	38	212	2%	1.50%
10	46	236	0%	0%
12	54	260	1%	0.30%
14	62	274	2%	0.50%

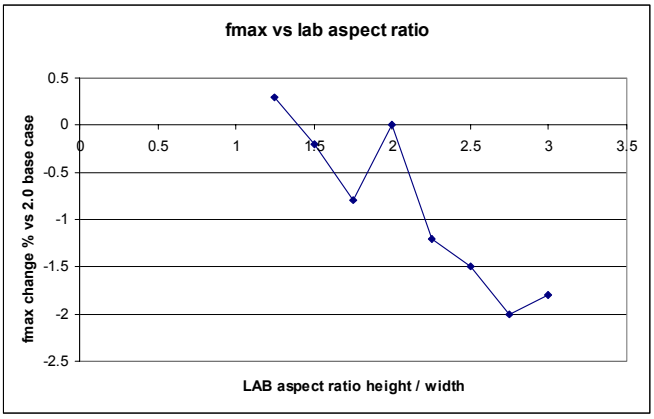

Figure 7: Effect of LAB aspect ratio on overall performance

3.3 Arithmetic Functions in the ALM

Another desire for Stratix II was to improve arithmetic capability. Previous Altera architectures had used a 4-LUT, split into two 3-LUTs to implement a sum and carry function respectively. Since the carry-in counts as an input, the LUT could only implement

additive functions of 2 inputs which effectively limits them to addition or subtraction. Using an entire 6-LUT for one bit of arithmetic was judged too expensive, so our investigations focused on structures that used the ALM for two bits of arithmetic. Implementing the previous approach would have offered two 4-LUTs per bit, supporting arithmetic functions of three inputs and a carry-in. To offer more functionality, a dedicated adder was included in the Stratix II ALM. Two one-bit adders are included in the ALM, with inputs taken from the four respective 4-LUT outputs. Taking the adder inputs from the most obvious set of 4-LUTs at intermediate points in the 6-LUT would provide two sum bits z0(a,b,dc0,e0) + z1(a,b,dc0,e0) and z2(a,b,dc1,e1)+z3(a,b,dc1,e1), which would have wasted the f0 and f1 inputs. To allow a larger set of functions, extra multiplexers controlled by the f0 and f1 respectively were introduced at the last stage of the 4-LUT multiplexer trees, allowing all inputs to be used to compute z0(a,b,dc0,e0)+z1(a,b,dc0,f0) and z2(a,b,dc1,e1)+z3(a,b,dc1,f1). The extra multiplexers are in bold in Figure 8, which also removes other unrelated detail for clarity.

A further enhancement allows the shifting of one of the adder inputs by one bit. This allows each pair of 4-LUTs to be used as a 3:2 compressor, using the adder to form the sum of the outputs. Thus each ALM can perform a 3-input sum, forming a+b+c with two bits per ALM.

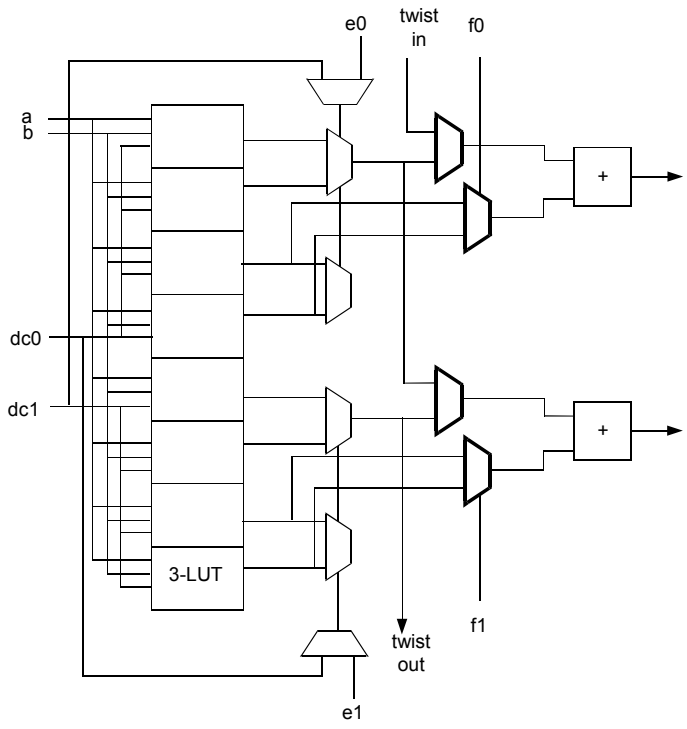

Figure 8: Extra multiplexers and full adder in ALM

The powerful arithmetic can potentially create a high demand for inputs into the LAB, in the event that complex arithmetic functions with many inputs and little sharing are used. To increase the flexibility of packing a mix of arithmetic and random logic functions in the same LAB, multiplexing circuits were added to the control signal region so that the carry chain could be configured to take an "early exit" after half the LEs in a LAB, then to skip to the next LAB. In this way the carry chain can either use all or half the LEs in the LAB, and complex arithmetic functions use only half the LEs and a mix of other related random logic. This tends to make the distribution of the number of LAB inputs more uniform and offers improvements to the overall routing efficiency.

4. MODIFICATIONS TO ROUTING ARCHITECTURE

Stratix II uses a routing architecture that is similar to Stratix. This consists of a direct drive architecture, with each LAB being able to drive two pools of routing multiplexers, and to listen to a horizontal and two vertical channels. Stratix contained wires of length 4, 8, 16 (V only) and 24 (H only). All of these choices were revisited for Stratix II. Although the choice of the length 4 wires was confirmed, it was found that the benefit of the length 8 wires was reduced, and the presence of a single long wire network (either the 8 or 24H/16V) was sufficient. Further, the 24H/16V network offered the highest speed long distance connections a lower area cost than the length 8 network. Thus the 24H/16V network was retained, and the length 8 network was removed. Beyond this, approximately a 20% reduction in routing capacity (normalized to logic density) compared to Stratix was implemented, despite the doubling of target logic density, due to increased ability to tune FMT routing patterns, and improvements in the production router, offering a further 6% improvement in overall area per unit logic.

Although the Stratix II LAB contains about twice as much logic as the Stratix LAB, the total wire count of the channel needed to be increased by only 5%, whereas closer to a 40% increase would be expected based on an approximate $N^{1/2}$ growth of routing demand vs LAB size that is found experimentally. This is due not only to the improved routing efficiency of the ALM, but improvements in the production router as well. The substitution of the more efficient length 4 wires for length 8 also contributes to the net reduction in wire count. It should be noted that [13] does predict a reduction in routing requirements for a 10-LE 6-LUT compared to a 10-LE 4-LUT, but this comparison is not directly appropriate here because the use of a regular 6-LUT has less logic capacity and fewer inputs on average than a SLM 6-LUT and should also be expected to have less routing demand.

Another change to the routing structure was the provision of a small number of fast inputs to the routing multiplexers. The routing multiplexers are constructed as conventional two-level NMOS pass transistor selection stage, followed by two buffer stages. A small number of single stage fanins can also be provided that feed the buffers through only a single NMOS pass transistor, as shown in Figure 9. Each additional fast input provides more potential fast connections, at the cost of not only larger area, but potentially slowing down all connections, including the fast ones, that go through that multiplexer. The slow down arises from the increased fanin of the second stage of the multiplexer, causing increased parasitic loading. FMT experiments were used to sweep various possible numbers of fast inputs, include one, two, and all inputs to the multiplexer being fast, although ignoring the parasitic loading effects in this phase of the investigation. Table 2

shows the results of the area and performance increase, broken out into all circuits, as well as circuits with more than 5000 and 10000 LEs. Although a monotonic increase in performance is expected with more fast inputs, it can be seen that two fast inputs offers slightly less performance than one. This is attributed to experimental noise which can be on the order of +/- 1%, but is also suggesting that the one fast input case benefited positively from noise. Overall it can be inferred that beyond the first fast input there is little incremental benefit to more fast inputs, especially as parasitic loading effects are not included here. After deciding to include a single fast input and performing more detailed circuit design including the slow down due to parasitic effects, detailed area analysis, and a more exhaustive experiment to reduce noise, the performance improvement of a single fast input was confirmed at 3%. Other enhancements to the routing structure have also been implemented, but will not be described in this paper.

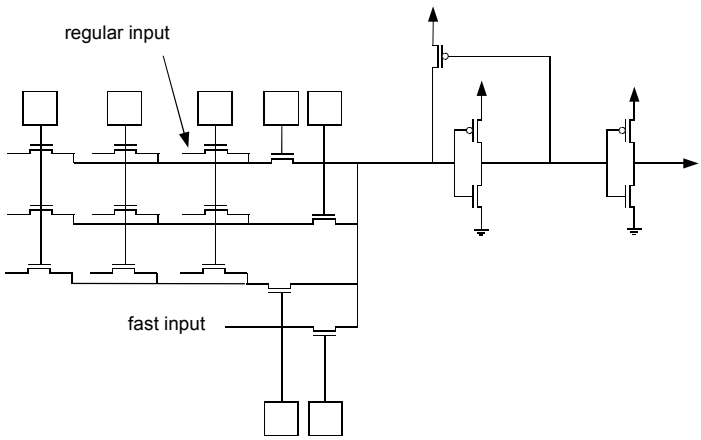

Figure 9: Fast inputs to routing multiplexers

Table 2: Effect of adding fast inputs to routing multiplexers

Fast Inputs per Routing Mux	Fmax All Circuits	Fmax Circuits > 5K LEs	Fmax Circuits >10K LEs
1 Fast	5.7%	5.6%	2.7%
2 Fast	4.0%	3.5%	0.9%
All Fast	7.6%	7.8%	7.7%

5. CONCLUSIONS

This paper has shown how a shared LUT mask LE can effectively improve performance by 15% while reducing overall area by 2%. More powerful arithmetic supports merging logic with arithmetic functions, and supports functions such as adding three independent operands in each LE. Combined with further tuning of the routing patterns, improvements in the production router, fast routing multiplexer inputs, and a process shrink from 0.13um to 90nm, an overall 51% performance increase and 50% area decrease is achieved.

6. REFERENCES

[1] D. Lewis et al, "The Stratix™ Logic and Routing Architecture", Proc FPGA-02, pp 12-20

[2] Elias Ahmed and Jonathan Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", Proc FPGA-00, pp 3-12

[3] V. Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs", Kluwer Academic Publishers, 1999

[4] M. Hutton et al, "Interconnect Enhancements for a High-Speed PLD Architecture", Proc FPGA-00, pp 3-10

[5] R. Cliff et al, "A Next Generation Architecture Optimized for High Density System Level Integration", Proc. CICC-99, pp 175-178

[6] M. Hutton, K. Adibsamii, and A. Leaver, "Timing Driven Placement for Hierarchical Programmable Logic Devices", Proc. FPGA-01, pp3-11

[7] K. Veenstra et al, "Optimizations for a Highly Cost-Efficient Programmable Logic Architecture", Proc FPGA-98, pp 20-24

[8] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density", Proc FPGA-99, pp 59-68

[9] V. Betz and J. Rose, "Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency", IEEE Trans. VLSI, Sept 1998, pp 445-456

[10] V. Betz and J. Rose, "Automatic Generation of FPGA Routing Architectures from High-Level Descriptions", Proc. FPGA-00, pp 175-184

[11] D. Cherepacha and D. Lewis, "A Datapath Oriented Architecture for FPGAs", Proc. FPGA-94

[12] M. Hutton, et al, "Improving FPGA Performance and Area Using an Adaptive Logic Module", in Proc. Int'l Conference on Field Programmable logic and its applications Proc. FPL-04, pp. 135-144, 2004

[13] E. Ahmed, "The Effect of Logic Block Granularity on Deep-Submicron FPGA Performance and Density", MASc Thesis, University of Toronto, 2001