A new high density and very low cost reprogrammable FPGA architecture

Sinan Kaptanoglu, Greg Bakker, Arun Kundu, Ivan Corneillet
Actel Corporation
955 East Arques Avenue, Sunnyvale CA 94086
email: {sinan,gwb,kundu,ivan}@actel.com

Ben Ting
BTR Inc.
20410 Town Center Lane, Suite 210, Cupertino CA 95014

1. ABSTRACT

A new reprogrammable FPGA architecture is described which is specifically designed to be of very low cost. It covers a range of 35K to a million usable gates. In addition, it delivers high performance and it is synthesis efficient. This architecture is loosely based on an earlier reprogrammable Actel architecture named ES. By changing the structure of the interconnect and by making other improvements, we achieved an average cost reduction by a factor of three per usable gate. The first member of the family based on this architecture is fabricated on a 2.5V standard 0.25µ CMOS technology with a gate count of up to 130K which also includes 36K bits of two port RAM. The gate count of this part is verified in a fully automatic design flow starting from a high level description followed by synthesis, technology mapping, place and route, and timing extraction.

2. OVERVIEW

Actel introduced its first reprogrammable FPGA two years ago, the ES family. The ES family addressed the low to medium gate count ranges with good performance but only average cost per gate compared with the other commercial reprogrammable FPGAs.

In this paper, we describe Actel's second generation reprogrammable FPGA architecture. This architecture is similar to the earlier ES architecture in terms of its choice of logic blocks, but it has a different interconnect structure. The main motivation for the new architecture has been the desire to improve the *ease of use* with higher gate counts and lower cost per gate. Ease of use is loosely defined as the ease of place and route success, ability to fix pins, and predictable performance. In this regard, we surpassed our goals by making it possible to build FPGAs with *gate counts* of up to a million gates[1] at 0.25µ technology, and up to two million gates at 0.18µ. Furthermore, we achieved a sizeable reduction in the average cost per usable gate by a factor of 3 even excluding the geometry shrink factor. Despite this cost reduction, the new architecture seems to be more robust for ease of use than the ES family. When compared with other state of the art commercial reprogrammable FPGAs, this architecture delivers competitive performance, while delivering the lowest cost per gate.

The next section describes the architecture. Section 4 discusses the CAD software that supports this architecture, where we present preliminary results from synthesis, technology mapping, and place and route.

3. THE FPGA ARCHITECTURE

We call our architecture *semi-hierarchical* for lack of a better term. More precisely, this is an architecture with three levels of (true) hierarchy. However, within each level of hierarchy, the structure is *linear*, more commonly known as a two dimensional mesh. Each mesh is segmented. The size, the properties, and the segmentation of each mesh is different, the details of which appear in the next few sub-sections.

Each of the three levels of hierarchy has its own interconnect. Switching from one to another (either going up or down the hierarchy) is facilitated through *tabs* which are active connections with bi-directional buffers. In addition, it is also possible to short circuit the hierarchy by using *local* extensions of some of the resources of the lower hierarchy. Such hierarchy crossing extensions when judiciously used, significantly increase the performance of the involved nets. Clearly the use of the hierarchy crossing extensions ceases to be beneficial if they are used for interconnections that are not very local.

The top level of the routing hierarchy is made out of an arbitrary rectangular array of *tiles* called B16x16. Each tile is surrounded by routing channels on all four sides.

The B16x16 tile is the middle level of the hierarchy. This hierarchy itself is made out of 16x16 arrays of basic blocks

[1] There is no established standard for gate counting for FPGAs as there is for gate arrays. We quote *usable* gates not raw gates.

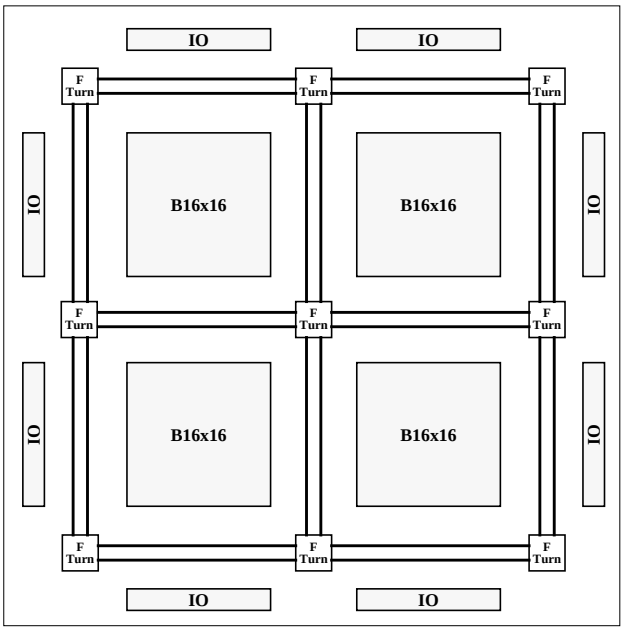
called B1. This array is served by a two dimensional mesh routing structure. The routing channels have logarithmic segmentation. Each B16x16 tile also includes 9K bits of user RAM and additional routing resources dedicated to it.

The lowest level of the hierarchy is the B1 block. This block in a way is the most complicated level of hierarchy possessing several different kinds of interconnect each serving a different purpose. The units inside the B1 block are the combinatorial and sequential logic modules, such as LUTs and flip-flops.

3.1 The Top Level of the Hierarchy

As mentioned earlier, all devices in this FPGA architecture are built as arbitrary rectangular arrays of B16x16 tiles enclosed by IO blocks on the periphery. The IO blocks include digitally implemented Delay-Locked-Loop (DLL) components (see section 3.6.1). Figure 1 shows the floorplan of the 4-tile FPGA that was mentioned in the abstract. Other devices, both smaller and much larger have similar floorplans. Obviously, the smallest device in this architecture contains only a single B16x16 tile surrounded by IOs.

Figure 1 - Floorplan of a 4-tile FPGA

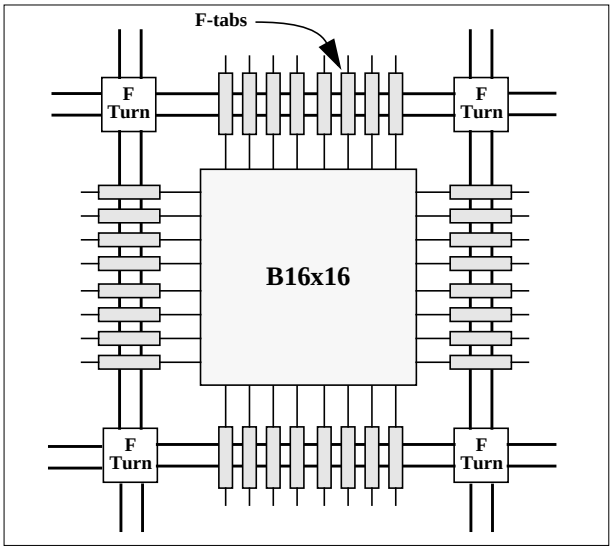


The routing tracks that surround the B16x16 tiles are called *freeways*. The width of the freeway channels is adjusted to different values for different members of the family without disturbing the internal structure of the B16x16 tile. This modification is the only architectural adjustment required for adding new members to this family of FPGAs. This ability to step and repeat B16x16 tiles rapidly is a real advantage in bringing different devices to the marketplace.

The freeway tracks can be extended in any combination of all three directions at each end through programmable switches with bi-directional buffers forming a freeway turn matrix (F-Turn). The IO blocks on the periphery of the FPGA also have identical channels of freeway tracks on the inside edge facing a B16x16 tile. Thus, when an entire FPGA is built with an array of B16x16 tiles and IO blocks, the freeway channels along with the F-Turns form a coarse mesh of their own.

A freeway track will very rarely be utilized all by itself without any extension, since such distances are abundantly covered by the routing resources in the middle hierarchy. The freeways are primarily intended to be used in conjunction with one or more other freeway tracks in *any* direction, together spanning distances of two or more B16x16 tiles.

Figure 2 - F-Tabs



The freeway channels are accessed from the middle hierarchy through tabs with bi-directional buffers. These tabs are called F-Tabs and are shown in Figure 2. F-Tabs perform dual function in that they not only provide on/off ramp access to the freeways, but they also facilitate the local extension of B16x16 routing resources. They can even combine the two roles simultaneously for the same net!

3.2 The Middle Hierarchy

3.2.1 B16x16 and its Interconnect

The structure of the logic array in the B16x16 is based on the repetition and nesting of smaller tiles. The smallest tile that is directly replicated and stepped is the B2x2 block in an 8x8 array. Each B2x2 block contains four B1 blocks in a 2x2 array.

The B16x16 tile also contains two RAM modules. Each RAM module has 512x9 bits with an interface of two completely independent ports for write and read operations. The RAM can be configured in bit mode and it has features to generate or check parity. The read operations may be configured in registered, pipelined or asynchronous modes. The write and read ports interconnect with dedicated routing resources that run vertically through each column of B16x16 tiles in the FPGA.

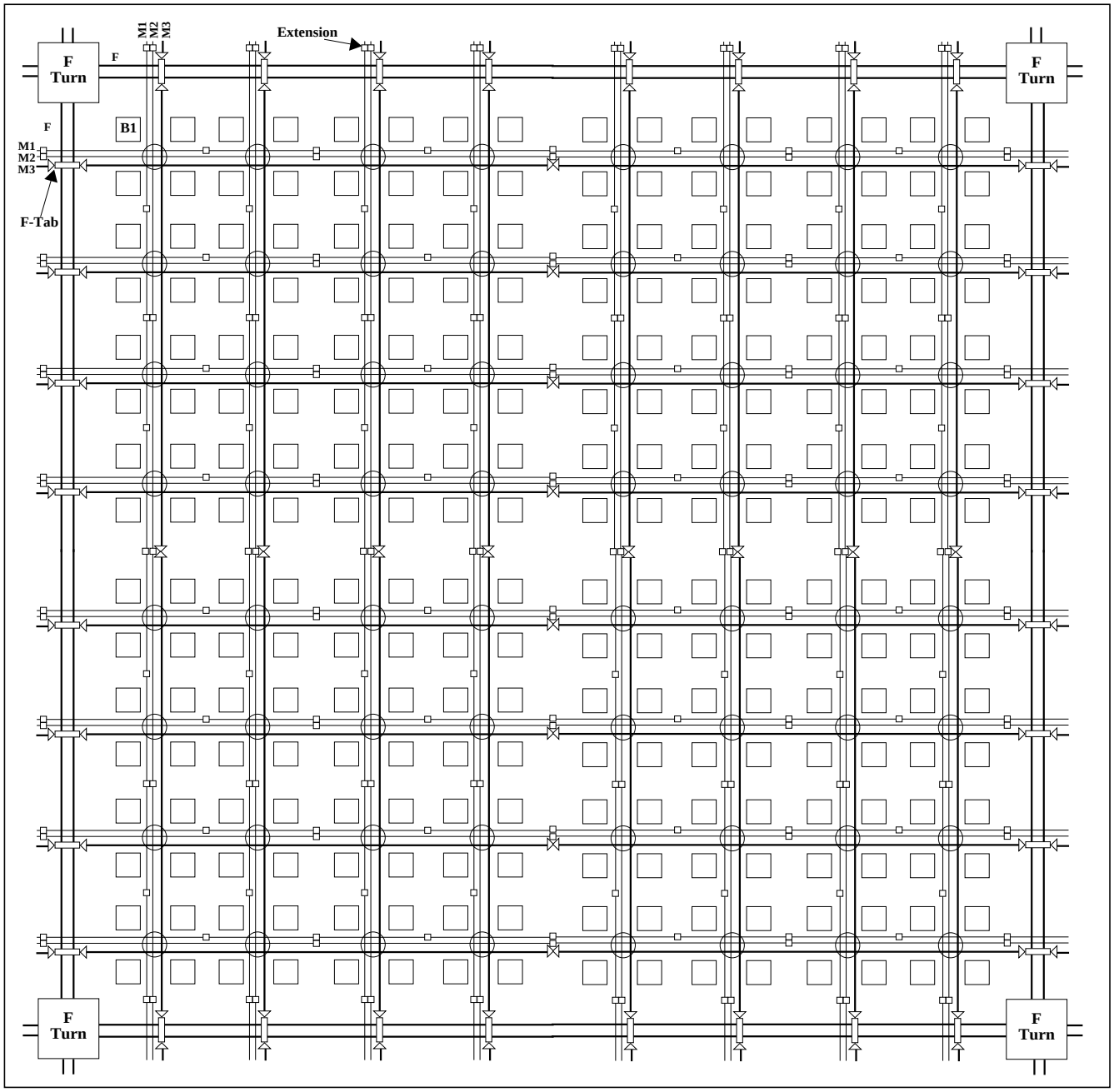
The two dimensional mesh routing in the middle hierarchy is called the *expressway* system and consists of three types of routing tracks — M1, M2 and M3 that span distances of 2, 4 or 8 B1 blocks, respectively. All expressways run both vertically and horizontally through every column or row of B2x2 blocks as shown in Figure 3. Every expressway track may be extended with a programmable switch for an identical distance along the same direction. The M3 tracks have bi-directional buffers at each extension and can be utilized to traverse long distances across the FPGA. The M3 expressways directly connects to the F-Tabs as described in section 3.1.

The vertical and horizontal expressways have depopulated programmable connections to each other in a turn matrix (E-Turn) located at the center of each B2x2 block. This turn matrix is shown in Figure 4.

3.2.2 B2x2 and its Interconnect

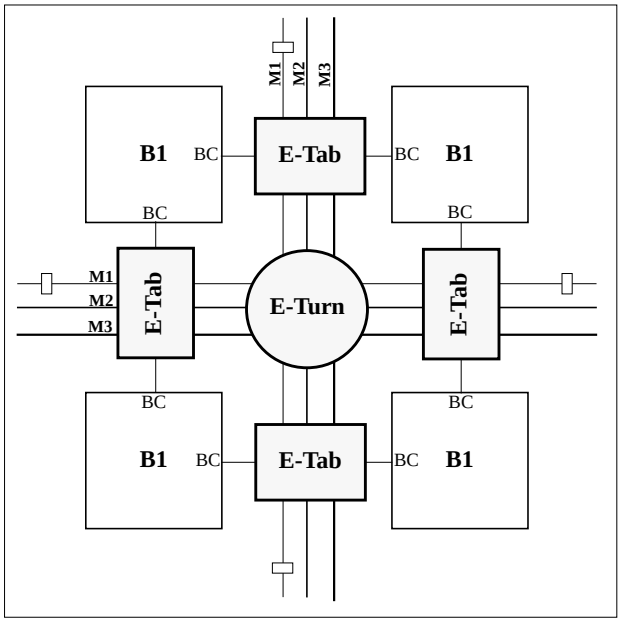
The B2x2 contains four B1 blocks in a 2x2 array and their interconnections with the expressway system. The expressway routing is not accessible to a B1 block on all four sides, only two sides, through two expressway-tabs (E-Tabs), one to the horizontal expressway tracks and the other to the vertical expressway tracks.

Figure 3 - B16x16



An E-Tab has active buffers into the expressway and is shared by two adjacent B1 blocks as shown in Figure 4. The B1 block has two dedicated BlockConnect (BC) channels corresponding to its two E-Tabs. Here, we show the interface of the BCs to the expressways. The BCs directly connect to the E-Tab, which can then connect with any expressway level using its on-ramp capability. Once the required distance has been traversed, an off-ramp E-Tab provides a BC connection back down to the inputs of the driven modules in the destination B1 block. The BC itself is described in section 3.3.3.

Figure 4 - B2x2



The separation of the B1 hierarchy from the higher levels by E-Tabs is designed in such a way as to render the place and route problems inside and outside the B1 blocks to be independent from each other. These two problems, therefore can be solved separately, rather than simultaneously.

3.3 The Lowest Level of the Hierarchy

The lowest level of the hierarchy possesses the most complicated interconnect. It also has somewhat unorthodox logic modules. We first discuss the choice of the logic modules in some length, and then proceed to describe the interconnect.

3.3.1 The Choice of Logic Modules

The sequential modules used in this architecture are quite simple in that they are clock edge triggered D-flip-flops (FF) with dedicated (asynchronous or synchronous) set/reset as well as data-enable signals. The FF can be configured to respond to either edge of the clock signal, and it can also be configured as a D-Latch instead of a D-flip-flop.

The combinatorial modules by contrast, are rather unusual — we use a mixture of LUT2 and LUT3 function generators. Two questions about our choice of the combinatorial modules need to be answered. The first is regarding heterogeneity — why do we not use only LUT2, or only LUT3 logic blocks, and what advantage is to be gained by having a mix of both? The second question is a little less obvious — why not use LUT4s, which are shown to be the most efficient LUT-k blocks for any integer value of k for SRAM based reprogrammable FPGAs by J. S. Rose, R. J. Francis, D. M. Lewis and P. Chow in [1], J. Kouloheris and A. El-Gamal in [2], and S. D. Brown, R. J. Francis and Z. G. Vranesic in [3].

Consider the heterogeneity issue first. This choice is driven by two different observations about the functional usage. The first observation is about the technology mappers — when required to do technology mapping for a *homogeneous* LUT-k type logic block, all commercially available tools produce netlists which contain a mixture of functions ranging from 2 to k inputs. For example, Table 1 presents the results of mapping by such a tool for a homogeneous FPGA architecture made out of LUT3 blocks and FFs.

Table 1 - Technology mapping to LUT3

benchmarks	3-input functions	2-input functions
bm1	83%	17%
bm2	96%	4%
bm3	87%	13%
bm4	98%	2%
bm5	89%	11%
bm6	95%	5%
bm7	66%	34%
bm8	92%	8%
bm9	92%	8%
bm10	75%	25%
bm11	75%	25%
bm12	74%	26%
bm13	86%	14%
bm14	90%	10%
bm15	83%	17%
bm16	91%	9%
bm17	65%	35%
bm18	98%	2%
bm19	98%	2%
bm20	85%	15%
average	**85%**	**15%**

The table gives the percentage of 2 and 3 input functions produced by this mapper on a set of 20 internal Actel benchmarks. The table shows that on average 15% of the blocks in the mapped circuit were two input logic functions even though the mapper was solely targeted for LUT3s. We also observed that this behavior is quite universal and not a peculiarity of one vendor's tools, and it does not depend on the

design size[1]. We therefore take this result as an experimental fact and conclude that a homogeneous FPGA architecture with LUT3 logic blocks gets under-utilized by about 15%.

The second observation is related to the optimal arithmetic and datapath macros that are generated by our ACTGen Macro Builder tool (see section 4.2). Many of these highly optimized macros like up/down counter, adder/subtracter or multiplier tend to use a mixture of $^1/_3$ LUT2s and $^2/_3$ LUT3s. Although the mapping experiment showed an average demand of 15% LUT2s, we settled on the $^1/_3$ fraction for the sake of uniformity. Based on these two observations, we conclude that our heterogeneous architecture with $^1/_3$ LUT2s and $^2/_3$ LUT3s is more optimal than a homogeneous architecture based on LUT3s alone.

Recent mapping algorithms to heterogeneous LUT technology, such as the ones developed by J. Cong and S. Xu in [4] and [5] have shown that mapping algorithms targeted to homogeneous LUTs tend to use more area than those which considered heterogeneous LUTs. The approach in [4] and [5] is directly applicable to heterogeneous LUTs that are composed of smaller LUT blocks, which is not the case in our architecture. [4] and [5] have demonstrated promising reduction in delays as well. The efficiency of mapping should certainly improve when such algorithms emerge that will be able to map to FPGAs with bounded resources and a fixed ratio of heterogeneous LUT blocks.

Next we compare our choice of logic blocks to the more common choice of LUT4s. As we mentioned earlier [1], [2] and [3] systematically analyzed the area efficiency of LUT-k blocks as a function of k, and concluded that the most efficient value of k was near 3.5. The most efficient integer value was found to be k=4, closely followed by k=3 which was about 10% less efficient.

Even though it was systematic, the analysis in [1], [2] and [3] depended on general routing models, and it had certain limitations[2]. For example, the analysis excluded heterogeneous logic blocks and hierarchical routing structures explicitly, both of which directly apply to our architecture. They also excluded *over the cell routing*, which is again very relevant to us. Later, J. He, V. Betz, and J. Rose have studied some heterogeneous architectures in [6] and [7]. They observed that certain combinations of LUT2s and LUT4s (as well as LUT2s and LUT5s) may be more efficient than a pure LUT4 architecture. A. Agarwal and D. Lewis while analyzing LUT based hierarchical architectures in [8], observed that purely hierarchical architectures can be up to 15% more area efficient than linear architectures because they may need fewer programmable switches. None of these later studies however, considered an architecture like ours which is both heterogeneous and semi-hierarchical.

We next make some observations about the issue of over the cell routing. This is not so much a property of the architecture in question as it is a feature of the process technology and the custom layout style one might employ. In the last decade, the CMOS technology has evolved not only towards smaller minimum feature sizes, but also towards more layers of interconnect. Today a typical CMOS process offers at least five layers of metal interconnect, often augmented by additional local interconnect such as a silicide or a salicide, which happen to be particularly useful for the connections from the SRAM memory bits to the routing switches. This should be compared to the state of affairs approximately a decade ago when we had only three layers of metal interconnect. At that time, over the cell routing was not a feasible alternative for SRAM based reprogrammable FPGAs for any reasonable layout style. Today, of course, this is possible, especially if the interconnect has hierarchy[3].

We therefore believe that the conclusions in [1], [2] and [3] should be re-examined based on the latest process technology parameters, as well as using the latest technology mappers. Of course, any strong conclusions are likely to remain unaltered, while others based on small differences may no longer be valid. Indeed this is what we have observed. We have analyzed the mapping results and estimated the routing areas in a 0.25µ technology with five layers of metal. Unlike the general work by [1], [2] and [3], we have not attempted to extend our analysis to arbitrary LUT-k blocks with k larger than 4, but we considered many heterogeneous combinations of LUT4s, LUT3s and LUT2s. Yet again unlike [1], [2] and [3], we did not use general models for routing area estimation. We directly measured the layout area instead. Since our comparison included a small number of competing choices, it was possible to estimate each layout area directly, rather than relying on general models.

At the end of our analysis, we convinced ourselves that our choice of LUT3s and LUT2s in the ratio of 2:1 is as efficient as a homogeneous architecture based purely on LUT4 modules. Parts of our analysis will be briefly discussed in an Appendix at the end of this paper in order not to disrupt the general flow of the architecture description. It turns out however, that the choice we made is not the most efficient one. We have strong indications that a choice containing the right mixture of LUT4s, LUT3s and LUT2s is somewhat more optimal. Such a choice was not adopted in our architecture to maintain backward compatibility to the ES family, but Actel may use such a mixture of LUTs in a future family of reprogrammable FPGAs.

---

[1] Actually there is some dependence on the design size for small designs. However, this slight dependence disappears for designs larger than 1,000 LUT3s. We have observed this behavior with designs ranging all the way up to 20,000 LUT3s.

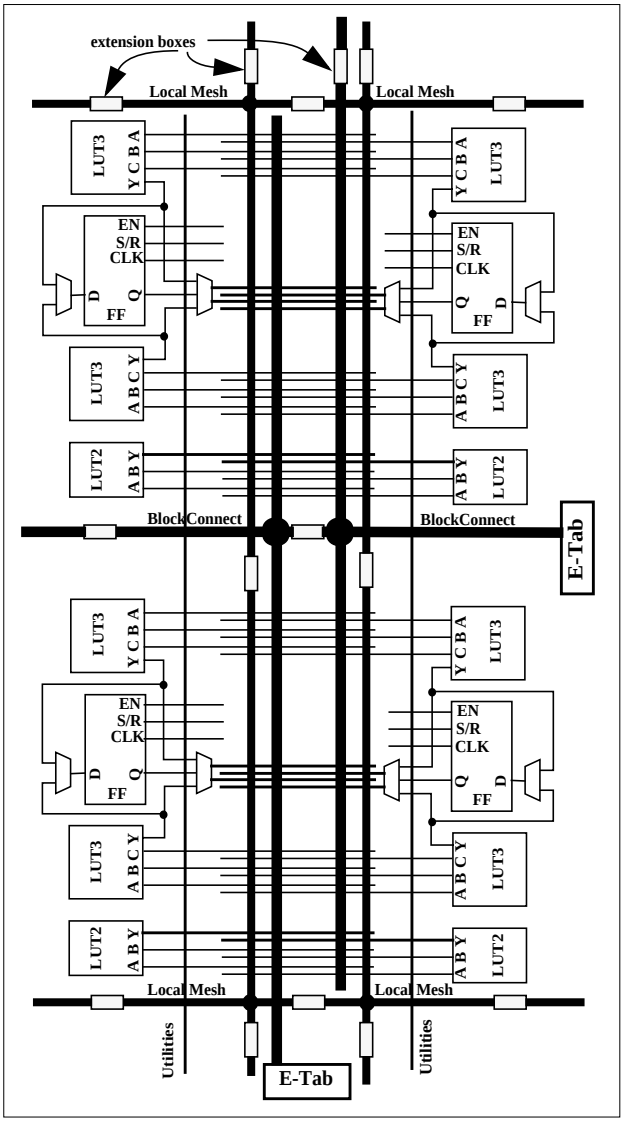
[2] We have not discovered these limitations on our own. They were explicitly stated in [1].

[3] With 5+ layers of general purpose interconnect, this is possible even for FPGA architectures with linear mesh interconnect. But it fits particularly easily and naturally with hierarchical or semi-hierarchical architectures.

3.3.2 The Logic Content of the B1 Block

We associate a FF with a pair of LUT3s. This trio of logic has two outputs, which can be driven either by the pair of LUT3s, or else by the FF and either one of the LUT3s. Each LUT3 has its own unshared inputs, while the FF has no data input other than the one that can be directly driven from the LUT3s. A trio and a LUT2 constitute a *quad* of the basic logic block (B1) which contains 4 such quads as shown in Figure 5. Therefore the ratios of LUT3s, LUT2s and FFs are precisely 2:1:1. The LUT2s do not share any inputs or outputs with any of the others.

Figure 5 - B1 Block



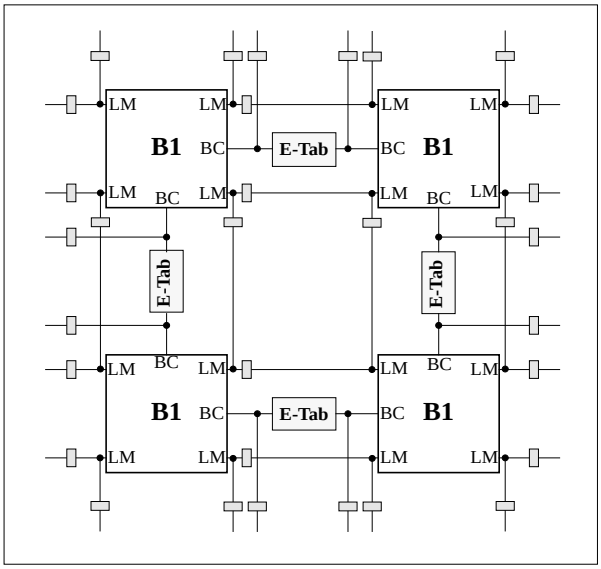
3.3.3 The Interconnect for the B1 Block

The B1 block contains its own dedicated routing of three types — DirectConnect (DC), LocalMesh (LM) and BC. The DC is a high performance direct connection between LUT3s in adjacent quads. The DC forms a vertical connection between adjacent B1 blocks, and provides excellent support for datapath functions such as counters, comparators, adders and multipliers (see section 4.2).

A B1 block has 4 channels of LM and two channels of BC. LM and BC are two-dimensional routing meshes which span two horizontal quads or a B1 block, respectively. The LM provides connection within and between adjacent quads for low fanout connections. The BC performs two functions. The first is to support mesh connections within and between adjacent B1 blocks. The second function is to provide on/off ramp access into the expressway routing as described in section 3.2.2.

Outside the B1, an LM or a BC track may be extended with a programmable switch along the same direction or the perpendicular direction to an adjacent B1 block. Each E-Tab itself provides such an extension facility for a BC in one of the two directions as shown in Figure 6. These hierarchy crossing connections in close proximity allow significantly better performance than a strict hierarchy. They also help avoid congesting the expressways.

Figure 6 - LM and BC extensions



3.4 Delays for General Routing Resources

At the beginning, we mentioned *predictable* delays as an important ease of use goal. The net delays in this architecture, depend on the routing topology (such as whether you cross a hierarchy or not) but they only weakly depend on the net fanout. This kind of predictability is mostly due to the *active routing* approach which provides automatic buffering of nets, thus freeing the designers and mapping tools from doing analysis to buffer them explicitly. Note that the active routing on the F-turn matrices is somewhat similar to the active routing examples one may find in other FPGA architectures. This kind is motivated by reducing the RC interconnect delay. On the other hand, the active routing in between the hierarchies has the additional advantage of providing predictability by *isolation*. Table 2 shows the span and the typical delay on a path including a LUT3 module in

0.25µ technology for the various general routing resources in the architecture.

Table 2 - Routing Resources

Hierarchy	Resource	Span	Delay (ns)
Lowest level	DC	1 quad	0.25
	LM	2 quads	1.10
	LM+LM	1 B1 block	1.30
	BC	1 B1 block	1.25
Middle level	BC+BC	2 B1 blocks	1.60
	M1	2 B1 blocks	2.20
	M1+M1	4 B1 blocks	2.40
	M2	4 B1 blocks	2.30
	M2+M2	8 B1 blocks	2.80
	M3	8 B1 blocks	2.80
	M3+M3	16 B1 blocks	3.50
	F	16 B1 blocks	4.60
Top level	F+F	32 B1 blocks	5.50

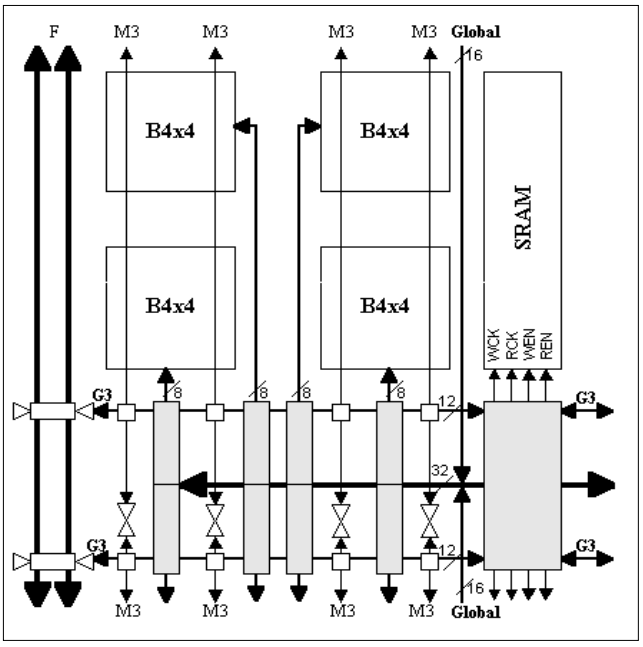
3.5 Global and Other Utility Signals

We classify high fanout nets in FPGA designs into four categories — *global utilities*, *local* clock/set/reset, *control* signals and high fanout *data*. Examples of global utilities are clock, set or reset signals that define the main clock domains in the design reaching many FFs in the FPGA. Local clock/set/reset signals occur relatively few times in a design and usually have low to medium high fanout.

Well known examples of control signals are FF-enable and multiplexor-select, yet there is a more general description. Control functions are always *orthogonal* to the data flow in the design. Such signals have medium to high fanout and may occur several times in a design. In our observation, another important characteristic of any control signal, has been that its source may originate in a different logic component and therefore the control signal driver may not be situated in the same physical hierarchy as its loads. We refer to the remaining high fanout signals that do not qualify for control as data.

We recognize the support of high-speed, high-fanout nets to be crucial in the acceptance of FPGAs with capacity of up to a million gates. We have taken a fresh approach to meet the requirements of all four categories above, keeping the area cost low while maintaining considerable flexibility. First, there are 32 chip-wide utilities in all members of the family that may be sourced from IOs or from logic internal to the FPGA. In addition, each B8x8 block has 12 G3 resources that span the width of a B8x8 block. Every G3 track may be sourced from inside the B8x8 through a vertical M3 expressway track, from an adjacent B8x8 block through a vertical M3 or a horizontal G3 extension, or from a distant B16x16 tile through a vertical freeway track. The two RAM modules in the B16x16 tile may select their clocks or enable signals from the 44 (32 chip-wide + 12 G3) utilities and each B4x4 block can independently choose 8 signals from the same 44, as shown in Figure 7.

Figure 7 - B8x8 Utilities



The 8 utilities in a B4x4 block are pruned down to 4 within a B1 block in two parallel steps. First, each B2x2 block may select a clock and a set/reset signal from the 8 utilities and distribute to all its 4 B1 blocks. Each B1 block in turn may select an enable and a general-purpose signal from the same 8 utilities. The clock, set/reset and enable signals drive all 4 FFs in the B1 block with common controls, effectively on a nibble basis. All 4 utilities in the B1 block can drive most of the LUT3 modules and each of the utilities in turn, may be driven by a LUT2 or a LUT3 module.

The chip-wide resources are targeted for the global utilities. The G3 resources are intended for high fanout control signals. The utility selection at each step into the B4x4, B2x2 or B1 block, is entirely optional from the higher level and may be sourced from logic within the level itself, allowing the possibility of any clock, set/reset, enable, control or data signal in a more localized scope. The provision of so many flexible ranges of utility resources is *unique* to this FPGA architecture, opening up the potential of structured approaches to placement and may considerably ease the routing of complex designs.

3.6 Other Architectural Features

3.6.1 DLL and Clock-doubler

The DLL and Clock-doubler components enhance clock delay control for improved system performance and every member of the family contains 2 or more such components. The DLL technology can track and adjust an internal clock-timing so that it may coincide precisely in time with an external clock from which it is derived. This allows the manipulation of the Clock-to-Out delay (pad-to-pad) from

30ns down to 0ns, in addition to a 100ps/increment user-programmable mode to support a wide range of delays. The Clock doubler can multiply frequencies to 150Mhz.

3.6.2 Power Supplies

This FPGA family can operate in a 2.5V system, a 3.3V system, a 5V system or mixed-voltage systems. Three separate Vcc supply networks are provided on the device — one for the array core, one for the output drive level and one for the input tolerance level. Level transistors are provided to accommodate all possible combinations of voltage levels for the input and output signals. All three supplies can be driven with 2.5V, the output drive level can go up to 3.3V while the input tolerance level can be raised to 5V.

3.6.3 I/O

Programmable options for IO pads include 3.3V PCI drivers, four-level slew rate control from 2.9V/ns down to 1.2V/ns at 35pF loading, and polarity control for output data and output enable. Open-source or open-drain output configurations are possible to support the emerging standards like GTL.

3.6.4 JTAG

This family implements a subset of the IEEE 1149.1 Boundary Scan test (BST) instructions, in addition to a private instruction to allow Actel's *ActionProbe* facility for real-time debugging of user designs (see section 4.4). The device supports in-system programmability and may itself be programmed via the JTAG inputs with yet another instruction.

4. DESIGN FLOW AND SOFTWARE

This architecture is specifically targeted to be used in a completely automatic push button design flow, where one starts from a high level description of the design functionality and timing constraints. Beyond this initial specification, no other user input is needed such as pin assignment, interactive floorplanning or manual hints to the automatic place and route.

In general there are fours steps to design with an FPGA — Design Specification, Implementation, Programming and System Debug. Design specification is supported by schematic capture and high level design. Register-transfer-level (RTL) circuit description in VHDL or Verilog can be readily synthesized and mapped by commercial EDA tools or by our ACTmap. The ACTgen Macro Builder may also be utilized to automatically generate high performance custom datapath macros. Some related architecture considerations are covered in section 4.2 below.

Actel's *Designer* tool performs the implementation step completely automatically. The DirectTime tools allow the user to analyze all paths and specify timing requirements for place and route. The bit-stream for Programming the devices can be programmed into an EPROM in the target system to configure the FPGA. Alternatively, Actel's Silicon Explorer can directly download the bit-stream into the FPGA during prototyping. System debugging capabilities for the final step are discussed in section 4.4 below.

4.1 Synthesis and Technology Mapping

In this section, we present some results of technology mapping for our architecture (based on LUT3 mapping) as well as for the traditional LUT4 logic blocks. The set of 20 internal Actel benchmarks were mapped to the two target technologies with a commercially available synthesis tool. The number of logic blocks and the count of the *routed input pins* in each of the two mapped circuits are shown in Table 3. We exclude those inputs that are driven by the global utilities as they do not consume any place and route resources.

Table 3 - Technology Mapping to LUT3 and LUT4

	LUT3 mapping		LUT4 mapping		routed input pin savings for LUT4s
	logic blocks	routed input pins	logic blocks	routed input pins	
bm1	1908	5435	1454	5116	5.9%
bm2	2258	6731	1829	7218	-7.2%
bm3	1991	5711	1411	5126	10.2%
bm4	2155	6427	1779	6820	-6.1%
bm5	5227	15137	3970	14146	6.5%
bm6	1657	4892	1319	5194	-6.2%
bm7	2746	7311	2017	7359	-0.7%
bm8	2228	6515	1691	6399	1.8%
bm9	2536	7417	1924	7286	1.8%
bm10	2461	6806	1738	6589	3.2%
bm11	3841	10561	2685	10276	2.7%
bm12	5161	14141	3586	13801	2.4%
bm13	3507	10109	2627	9526	5.8%
bm14	5081	14773	3781	13872	6.1%
bm15	1908	5435	1454	5116	5.9%
bm16	2172	6386	1882	5955	6.7%
bm17	3868	10264	2640	10062	2.0%
bm18	6260	18660	5170	20439	-9.5%
bm19	18704	55988	15536	61804	-10.4%
bm20	4705	13455	3467	12420	7.7%
average routed input pin savings for LUT4s					1.4%

It is clear from this table that the LUT3 mapping is quite efficient in terms of the total number of routed input pins which are approximately[1] the same for LUT3 and LUT4 mappings. This was a bit of a surprise since we expected not only fewer blocks used for LUT4s, but also fewer routed input pins. As the logic blocks grow in size and functional capability, one would expect some input pins to disappear from the netlist, having been absorbed as internal nodes inside the bigger logic blocks. Indeed this is exactly what happens when we go from LUT2 to LUT3 mapping. The

[1] 1.4% advantage that the LUT4 mapping enjoys is well within the fluctuations from design to design.

number of logic blocks and the total number of routed input pins both decrease. However, the same did not happen when we went from LUT3 to LUT4 mapping.

4.2 Hard Macro Support

The high speed of DC routing between the quads inside a B1 block in this architecture offers opportunities to create high performance datapath components based on a simple *ripple* style of logic design. This scheme of chaining the critical path of a ripple structure with DC tracks from quad to quad can be extended to the B1 block below to create a ripple macro of arbitrary width. At 0.25ns per bit, the total performance is quite attractive. We have also observed that all the logic inside a quad (including the LUT2) tends to be highly utilized with such 1-bit macros and since a ripple implementation is usually the most compact in logic area, the overall capacity of the FPGA increases with more of these instances in a design. The routing of these macros in a pipelined section with the help of utility resources to carry the control signals, creates a regular structure which may potentially relieve some congestion from surrounding regions.

The ACTgen Macro Builder provides the capability to automatically generate high performance custom datapath macros. It allows the designer to trade-off speed with efficient use of resources to decide on the optimum implementation suitable for the design. Synthesis tools can infer ACTgen macros during optimization.

4.3 Place and Route Results

We have placed and routed the 20 benchmark designs referred above, with the exception of bm19, which is too large to fit on the first device we are currently sampling.

4.4 System Debug

This architecture features Actel's *ActionProbe* circuitry which allows access to any internal node from certain external pins. In other programmable logic devices, designers would have to re-layout their device and add muxing to bring signals out to an external pin. This added to the time to reprogram the device, could introduce other errors, often changed critical timing, and changed loading and fanout on the signals. The number of pins available for looking at internal nodes is usually limited also, so for any one layout only a few nodes are observable. Alternatively, designers could get access to more nodes with some sort of JTAG/ SCAN or static single-stepping methods. A larger number of internal nodes can be accessed, however it is only a static view of the state of the node, making it almost impossible to trouble shoot timing problems.

The Silicon Explorer is an integrated hardware and software solution that, in conjunction with the Designer tools, allow users to examine *all* internal FPGA nodes while the device is operating in the target system -- in *real* time! Its *non-invasive* method does not alter any timing or loading effects and will help shorten the debug step.

5. CONCLUDING REMARKS

In conclusion, we presented some details of our new architecture, which attempts to combine the best features of both the linear mesh type routing structures and the hierarchical ones, while suppressing the less desirable effects of both, in very large and high performance FPGAs.

During the development of this architecture, we discovered, somewhat to our own surprise, that the LUT3s as building blocks have become as efficient as LUT4s, contrary to the results of earlier studies nearly a decade ago. We also observed that there are many heterogeneous combinations of small LUTs that do better than either homogeneous LUT3 or LUT4 logic blocks. Our analysis was not as general as the earlier studies and does not cover all LUT-k blocks, especially the larger k values. This is still a fertile research area in the light of the new technology parameters.

The first member of the family of FPGAs based on this architecture has already fully functional silicon and preliminary CAD software to support it. The software (especially the place and route) is not yet fully optimized for the features of this architecture. Despite that, the preliminary results fully indicate that we meet the capacity and performance targets even at this early stage, and we expect to surpass them as the software matures.

6. ACKNOWLEDGEMENTS

We are indebted to many past and present Actel employees and contractors who helped us with the development of this architecture. Even though we cannot name them all here, we would like to explicitly acknowledge the contributions by Jeff Schlageter, Robert Smith, Peter Pani, Yinan Shen, Jung-Cheun Lien, Jerome Fron, Chuck Hastings, Gajus Worthington, Bill Plants and Warren Miller. We thank Ken O'Neill and Luther Abel for reading the manuscript and making many helpful suggestions.

7. APPENDIX: Estimating Routing Area for LUT-k Blocks

Before we make logic block comparisons, we start with a brief digression. We ask the following general question — given an FPGA (of which the architecture is already chosen) and two designs with (a) N_1 nets and P input pins to route, (b) N_2 nets and P input pins to route, where $P > N_2 > N_1$, which design requires more routing resources, hence a larger routing area? We considered several architectures and many different routing topologies in each one of them and have convinced ourselves that both problems require about the same total area of routing. The first design has fewer nets, but the average fanout per net is larger requiring more complicated topologies and longer average net length. We then concluded that unless N_2 is much larger than N_1 the routing areas needed for these two problems are about the same irrespective of the choice of logic blocks and the underlying interconnect architecture. By this we do not

imply that the routing area is not affected by the choice of logic blocks or the structure of the interconnect. On the contrary, the required routing area strongly depends on these choices. But once these choices are made, the two problems stated above require the same amount of routing area as each other. The routing areas change from one choice to another, but they always remain approximately equal to each other for each choice. This means that the routing area is proportional to P, the number of routed input pins, but it is to a large degree independent from N_1 and N_2.

This observation of ours is somewhat different from that of [1], [2] and [3], who in their routing models use all pins, both inputs and outputs. If the FPGA is large enough so that the number of logic blocks is much larger than the number of IOs, then the number of output pins is approximately N, the same as the number of nets. Their models assumed that the routing area is proportional to (P+N), whereas we convinced ourselves that the output pins do not matter very much. If P >> N, there is little difference between P and (P+N) and the routing models in [1], [2] and [3] approach that of ours. Indeed, for LUT-k type logic blocks where k is large, we have P >> N, and P ≈ P + N. But for small values of k (especially for k < 5), the two estimates could differ significantly. For small k, our estimate of the routing area will be smaller. If our observation is correct, LUT2s will benefit the most, LUT3s the next, and so on.

We can now summarize our results for the area efficiencies for various logic blocks. There are three apparently unrelated effects, each of which makes the smaller LUTs more attractive than they were a decade ago. The first of these three reasons is the observation we made above, namely that only the *routed input pins* matter for estimating the routing area. The second reason was the dawn of the age of *over the cell routing*. The details of this is beyond the scope of this appendix as they are trade secrets, which we are unable to publish[1]. The third reason is that LUT-k mapping is not equally efficient for all k. It appears that *LUT3 mapping is exceptionally efficient*[2] (see section 4.1).

The combined effect of these three observations is still not enough to promote the LUT2s to the top of the list, even though LUT2s probably get the biggest boost. However the combined effect is more than enough to push the LUT3s to a virtual tie with the LUT4s. Furthermore, it also makes several heterogeneous combinations of LUT4s, LUT3s, and LUT2s significantly better than LUT4s.

8. REFERENCES

[1] J. S. Rose, R. J. Francis, D. M. Lewis, and P. Chow, IEEE Journal of Solid State Circuits, Vol. 25, No. 5, Oct. 1990, pp. 1217-1225.

[2] J. Kouloheris and A. El-Gamal, ACM/SIGDA Workshop on FPGAs (FPGA '92), Feb. 1992, pp. 9-14.

[3] S. D. Brown, R. J. Francis, J. S. Rose, and Z. G. Vranesic, Field Programmable Gate Arrays, Kluwer Academic Publishers, 1992, pp. 87-115.

[4] J. Cong, and S. Xu, Design Automation Conference, June 1998, pp. 704-707.

[5] J. Cong, and S. Xu, IEEE/ACM International Conference on Computer Aided Design, Nov. 1998, pp. 40-45.

[6] J. He and J. Rose, Custom Integrated Circuits Conference, May 1993, pp. 7.4.1-7.4.5.

[7] V. Betz and J. Rose, ACM/SIGDA International Symposium on FPGAs (FPGA '95), Feb. 1995, pp. 10-16.

[8] A. A. Agarwal, and D. M. Lewis, ACM/SIGDA Workshop on FPGAs (FPGA '94), February 1994.

[1] Most of the details we cannot publish are related to the layout style to take advantage of the extra layers of interconnect.

[2] We do not have a good explanation as to why the LUT3 mapping is the most efficient. We are not even sure if this will always be true in the future either. But for the present, we have to take it as an observed fact, and take advantage of it.