Fast$_{14}$ Technology: Design Technology for the Automation of Multi-Gigahertz Digital Logic

Steve Horne (presenter), Don Glowka, Scott McMahon, Paul Nixon, Mike Seningen, Gopal Vijayan

The authors are with Intrinsity, Inc. in Austin, Texas, USA (www.intrinsity.com)

Introduction

Fast$_{14}$® Technology is a new design technology developed by Intrinsity, Inc., a US-based semiconductor and design technology company located in Austin, Texas. Fast$_{14}$ Technology automates the implementation of multi-GHz digital logic circuits in standard CMOS fabrication processes.

Fast$_{14}$ Technology derives its name from the atomic number of silicon (which translates to "Fast Silicon Technology"). Fast$_{14}$ Technology is comprised of five critical design elements:

- Multiphase Overlapped Clocking
- 1-of-N Dynamic Logic (NDL® family)
- Expert Routing Technology™
- Unified Design Database
- Design Methodology and Electronic Design Automation (EDA) Tools Suite

Fast$_{14}$ Technology enables a significant improvement in chip design productivity for high-speed digital logic. Fast$_{14}$ Technology also provides significant power and silicon area efficiency benefits over other design methodologies in the high-performance logic design space.

These benefits enable embedded processors and special-purpose digital logic products to achieve the levels of performance previously achieved only through custom design flows while maintaining efficient levels of power. These high performance levels are achieved at greatly reduced development costs and with lower risk of circuit problems as compared to the custom design flows used for desktop processors.

FastMATH/FastMIPS Processors

Initially proven in August of 2001 with test chip results, Fast$_{14}$ Technology is now commercially deployed through Intrinsity's first products, the FastMATH® and FastMIPS® processors. Both products operate at 2 GHz at 1.0 V in the TSMC 130-nm LV CMOS process. The FastMATH processor, which contains approximately 70 million transistors, was selected by In-Stat/MDR as the 2002 Extreme Performance Product of the Year.

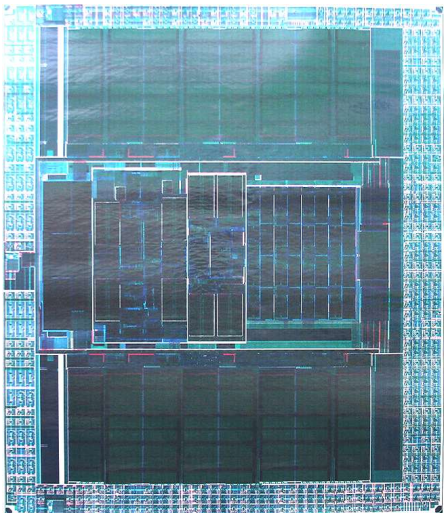

Figure 1: FastMATH Processor

These processors were the first processors developed for the embedded market to break the 1-GHz barrier and remain the highest frequency embedded processors in existence today.

Desktop processors from Intel and then AMD were the first to attain multi-GHz speeds. However, the FastMATH processor remains the first and only processor to achieve 2-GHz operating frequencies in a 130-nm process at a nominal VDD of only 1.0 V.

This low operating voltage and the advantages of Fast$_{14}$ Technology permit the FastMATH processor to operate at a typical power of only 15 W at 2 GHz. At lower operating voltages and frequencies (.85 V and 1 GHz), typical operating power levels of less than 6 W are achieved. Multi-GHz desktop processors typically consume over 50 W.

The FastMATH processor was designed by 45 chip designers in 16 months. First silicon was functional and booted a Linux kernel within 2 months of receiving the first samples. First silicon also met the 2 GHz speed goal at the target voltage and with predicted manufacturing yields. Other high-performance processors have required over 150 engineers and take 36 to 48 months of development time.

Background: Static vs. Dynamic Logic

Historically, two primary digital logic techniques have been available to CMOS chip designers: static logic and dynamic logic.

Dynamic logic was a popular design style for digital logic in the 1970s and early 1980s, before CMOS processes became economical. Prior to CMOS processes, a designer had only NMOS or PMOS manufacturing choices. In these older processes, a static-logic gate consumed significant DC power, which drove the need to design with a dynamic-logic gate. However, dynamic logic was more error prone and hazardous to use than static logic because of various timing sensitivities.

With the advent of CMOS process technology, the DC power of a static gate became very small, removing one of the main advantages of dynamic logic at that time.

Starting in the 1980s, Electronic Design Automation (EDA) tools were created that enabled automated, correct-by-construction layout for static logic. The tenfold productivity gain enabled mass deployment of new classes of digital circuits. Over the last two decades, the development and use of EDA tools for static-logic design has resulted in high design productivity and the ability to create electrically correct circuits with a very low risk of failure. For this reason, static logic has become the workhorse of the semiconductor industry.

One significant disadvantage of static logic is its relatively slow gate speed. Dynamic circuit speeds are two to three times faster than static circuit speeds. Only engineers who designed circuits in the 1970s remember the speed 'lost' when dynamic circuits were abandoned in favor of the more productive static logic. Of course, any engineer who has experience designing dynamic logic is also aware of the difficulty and high risk involved in designing electrically correct dynamic circuits.

Dynamic logic differs from static logic in that a clock is used in every gate (or almost every gate) and regulates the data flow through the gate. During one state of the clock, the gate is in a precharge state and the outputs are driven to an inactive state (their state conveys no logical state information). During the opposite phase of the clock, the input signals are evaluated and the outputs are driven to the appropriate state. With each clock cycle, the dynamic gate transitions from the precharge state to the evaluate state and then back to the precharge state. As in the case with static logic, after a series of dynamic gates, the final results are latched in preparation for the next set of logic gates. The following figure shows the components of a typical dynamic gate.

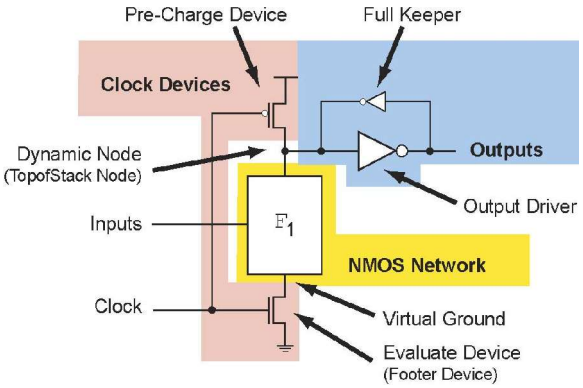

Figure 2: Dynamic-Gate Structure

The speed of dynamic gates comes in large part from the use of NMOS transistors to perform the logic function of the gate. In a CMOS gate, the logic function is duplicated in NMOS and CMOS transistor networks. PMOS transistors have one half to one third the mobility of NMOS transistors, and need to be designed with larger channel widths than the NMOS transistors to achieve optimum speed. The elimination of the complementary PMOS logic network from the dynamic gate reduces its size and also removes a large parasitic load from the gate. Additionally, because the preferred logic evaluation direction is through the NMOS network to ground, the gate can be sized to speed up the evaluation operation versus the precharge operation.

With this significant speed advantage, it would seem that dynamic logic and not static logic would have become the dominant design technology. However, dynamic logic has two fundamental technical challenges that limit its usability in logic devices. These two technical challenges are the main reasons why dynamic logic has been abandoned in favor of static logic. Each technical issue results in circuits that take longer to design, are difficult to design correctly the first time, and are challenging to manufacture over a standard process specification.

The first technical issue with traditional dynamic logic involves timing problems. The nature of the timing problems varies with the specific style of dynamic logic. Once a dynamic gate evaluates, it cannot evaluate its inputs again until after the precharge phase. This causes a large number of timing problems. Other problems are caused by the need to latch or register the result of a logic operation at the end of a cycle. There is a risk that the dynamic gate precharge state will race through the latch before it closes.

Some dynamic-logic styles require careful and error-prone tuning of clock delays to prevent these timing problems. Signal delay variability due to load and RC

delay variability has also been a recurring problem with traditional dynamic-logic styles.

The second technical issue with dynamic logic is noise sensitivity. As previously mentioned, dynamic logic cannot reverse a logic evaluation except by going through a precharge phase. Signal and power or ground noise can cause an inadvertent discharge of the dynamic gate, resulting in a logic malfunction.

For example, consider a dynamic gate that is not expected to discharge its dynamic node through its NMOS network because one of the inputs is low. If this input glitches above ground for a short time, the dynamic gate may falsely evaluate. The input noise glitch might be caused by capacitive coupling from a nearby, unrelated signal route. It may also be caused simply by a slight difference in ground potentials between the driving and receiving gate.

Another well-known noise risk involves charge sharing inside the NMOS network of the dynamic gate. In this case, the inputs to the gate may be noise-free, but the circuit design of the gate can result in a false evaluation due to charge sharing between the top-of-stack dynamic node and one or more of the internal NMOS network nodes.

Because of these two classes of problems, timing issues and noise issues, commercially available EDA tools have not been developed to automate the design of dynamic logic. Until now, only high-end, large processor design companies have been able to afford the labor-intensive design effort required to design dynamic logic on any significant scale. Even these processor teams typically use dynamic logic only in the most critical speed paths, as it is too costly in terms of productivity and design risk to use it everywhere.

For the first time, Fast$_{14}$ Technology has solved the problem of automating multi-GHz logic design. Additionally, it does so in a way that provides significant power savings over conventional dynamic logic and significant performance improvements over static logic.

Multiphase Overlapped Clocks

One of the key elements of Fast$_{14}$ Technology is the use of multiphase overlapped clocks with a nominal 50 percent duty cycle. The patented scheme requires at least three uniformly overlapping clock phases. For example, the FastMATH processor employs a four-phase clocking scheme. In the four-phase scheme, each phase clock is delayed 90 degrees from the preceding clock. Schemes have been studied with three to six phases. Multiphase overlapped clocks provide a number of benefits, as described below.

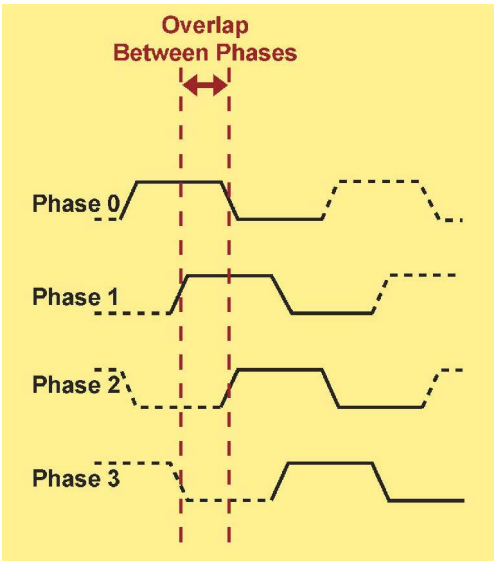

Figure 3: Multiphase Overlapping Clocks

One of the most important benefits of this clocking scheme is that synchronizing latches or registers are not required at cycle boundaries. At high frequencies, latches and registers in static and traditional dynamic-logic styles represent a large percentage of time in the clock cycle that cannot be used for performing useful work. The Fast$_{14}$ Technology design palette does include latches and register elements, but these are used primarily for efficient data storage and transmission purposes. Critical paths such as arithmetic units, control logic, decode logic and SRAM interfaces can cross cycle boundaries as needed without incurring the penalty of a register delay.

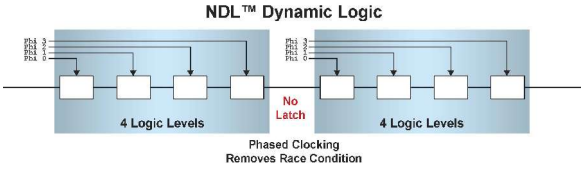

Figure 4: No Need for Registers

Another, more subtle, advantage of removing registers and latches from logic paths is that the designer has greater flexibility in arranging pipelines and logic feedback paths. This is because a pipeline stage and logic feedback path may begin and end at any of the phase clocks, not just at a cycle boundary.

The second critical benefit derived from this clocking scheme is a tolerance for moderate amounts of clock

uncertainty. The primary types of clock uncertainty in digital logic are clock-period jitter, duty-cycle error, and clock skew. Because the successive clock phases are overlapped, moderate amounts of uncertainty in the location of the clock edges does not affect the operating speed of the logic. Additionally, hold time problems and race conditions are virtually eliminated. This fact enabled the automation of logic design using a rich mixture of dynamic and static-logic elements by the Fast$_{14}$ Technology EDA tools.

The nature of the clock distribution system allows this clocking scheme to be applied to large chip areas, if desired. The multiphase clocking domain also interfaces easily to slower speed clock domains.

1-OF-N DYNAMIC LOGIC (NDL) FAMILY

The second critical element of Fast$_{14}$ Technology is the novel 1-of-N dynamic logic (NDL) family.

Traditional static logic encodes data values on signals in binary format. For example, the four data values 0, 1, 2 and 3 are encoded in binary on two signal nets. Traditional dynamic logic does not include the inversion function (gates are non-inverting), and so both the true and complement values of a bit of information must be generated and maintained. This logic style is known as dual-rail dynamic logic, and can be thought of as a 1-of-2 logic encoding. In dual rail dynamic logic, two signal nets are needed to represent one bit of information. It is apparent that with twice the number of nets needed to represent a value, this is a more costly method for representing data than the binary encoding method. Another disadvantage to the dual-rail dynamic-logic scheme is that one of the two wires switches every cycle, whether the data value is changing or not. This is not as power efficient as static logic, where the wire switches only when the data value is changing. Therefore, while dual rail dynamic logic provides significant speed advantages over static logic, it has the disadvantages of higher power consumption and higher wiring density.

The NDL logic family, on the other hand, represents data values in a 1-of-N format, where the radix N can be between one and eight. Radix values greater than eight are possible as well, but this sometimes introduces excessive wiring complexity. The radix value of four is preferred for many logic operations. However, other radix values are very valuable and allow for efficient, dense logic design.

Data Value	Static Logic 2 wires 0,1, or 2 wires switch		Dual–rail Dynamic 4 wires 2 wires switch				NDL Logic 4 wires 1 wire switches			
	A1	A0	A1	$\overline{A1}$	A0	$\overline{A0}$	A3	A2	A1	A0
null	–	–	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1
1	0	1	0	1	1	0	0	0	1	0
2	1	0	1	0	0	1	0	1	0	0
3	1	1	1	0	1	0	1	0	0	0

Figure 5: NDL Data Representation

As can be seen in Figure 5, radix-4 NDL logic has half the switch factor of dual-rail dynamic logic. Fast$_{14}$ Technology provides additional features beyond the scope of this paper that reduce this switch factor even further. Due in part to the NDL logic family, Fast$_{14}$ Technology provides better power efficiency than static-logic styles when comparing designs that operate above 500 MHz.

NDL gates (and traditional static elements such as latches, registers, and static combinatorial logic) are generated at compile-time. Unlike standard-cell based design flows, the designer is not limited to a fixed library of gate functions. The designer codes the desired function of the logic, but does not need to code or design the details of the gate. From this source code, Fast$_{14}$ Technology EDA tools create the detailed transistor-level netlist of the gates and optimize the size of all the transistors for the desired speed goal. This process is described in further detail later.

As in traditional dynamic logic, NDL gates perform their logic functions using NMOS transistors. The capacitive loading and area impact of the complementary PMOS transistor network found in static logic is not present, resulting in significant speed gains over static logic.

The 1-of-N encoding of NDL logic provides a further advantage over classical dual-rail dynamic logic. As noted before, representing data in a high radix 1-of-N format reduces the switching factor of the signals. This signal encoding, when applied to the inputs of an NDL gate, also result in a relatively high fraction of the NMOS transistors in the logic network remaining in the off state. Consequently, when the NMOS network evaluates, less electrical charge needs to be discharged to ground. This improves the speed of the gate, making it more power efficient, and reducing its area. This also allows designers to create more complex functions in a single gate than is possible in conventional design styles, reducing the number of stages needed to implement a logic function.

Fast$_{14}$ Technology also provides for the integration of scan functionality directly into NDL gates as well as static gates. This is done in such a way as to have no impact on the performance of the logic family. The designer can choose no scan, full scan, or partial scan as appropriate for the block or project.

The NDL logic family can operate at arbitrarily low frequencies and has a wide voltage operating range (of course, frequency is reduced at very low operating voltages).

NDL Gate Example

As an example, a 1-of-4 adder gate is shown in Figure 6: NDL Adder Gate. This gate generates a 1-of-3 result from two 1-of-4 operands (from two 2-bit operands). The encoding of each operand is the same as in Figure 5. The 1-of-3 output of the adder gate is propagate (PROP), generate (GEN), and halt (HALT). The propagate and generate terms are not dissimilar from normal Boolean adders. The halt term provides the complementary function to propagate and generate.

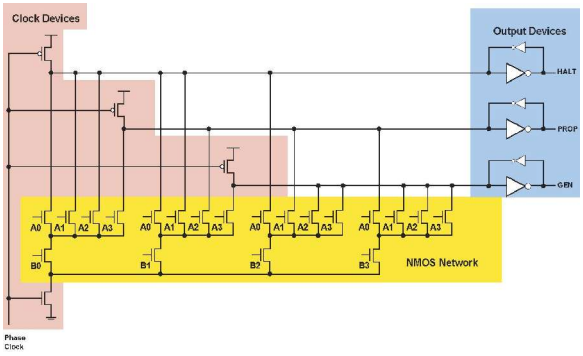

Figure 6: NDL Adder Gate

The PROP output is asserted if the input values propagate a carry-in condition to a carry-out condition (for instance, if the sum of A and B is 3).

```
PROP = A0&B3|A1&B2|A3&B0|A2&B1
```

The GEN output is asserted if the input values generate a carryout condition (for example, if the sum of A and B is greater than 3).

```
GEN = B3&(A1|A2|A3)|B2&(A2|A3)|B1&A3.
```

HALT is the logical complement of PROP | GEN (that is, the terms that sum to less than 3).

```
HALT = B0&(A0|A1|A2)|B1&(A0|A1)|B2&A0
```

A full 32-bit adder can be designed using only three levels of logic using these and similar NDL gates.

As can be seen from this example, some logic functions are naturally represented in high-radix signal encodings. In addition to being efficient logically, NDL gate designers also find this encoding flexibility to be very natural. High-radix representation allows designers to translate their concepts into logic in an intuitive and efficient manner.

As another example of this encoding power, a 6-state state machine can be represented by a single 1-of-6 NDL gate. The high logic density possible in the NMOS network means that the combinatorial logic affecting the next state can also be put into this single gate (though it is sometimes convenient to spread this logic over two or three NDL gates - still well within one clock cycle).

Expert Routing Technology

The third critical element of Fast$_{14}$ Technology is an advanced expert routing technology. For the past four or five process generations, the wiring of an integrated circuit has played a more and more dominant role in the circuit's performance, size and electrical robustness. This trend is expected to continue for the foreseeable future. Fast$_{14}$ Technology addresses this problem with a sophisticated and cohesive set of routing features that include the following components:

- EM/IR-Aware power and ground routing
- Speed/noise-aware signal and clock routing
- Automatic, on-demand signal shielding
- Bundle routing and wire Twizzling™ method
- Redundant via insertion

This routing technology not only enables complex logic to operate at multi-GHz frequencies, but also provides product reliability, deterministic and rapid timing closure, and high degrees of electrical robustness in an automated design flow.

The routing process begins at the cell level, where signal and power/ground connections inside the gate are sized to handle the specific operating currents of the gate. This includes metallization width and contact/via counts. Depending on the specific technology and the design goals, either EM (electromigration) or IR (voltage) drop limits may set the power/ground routing requirements.

After gates are constructed and placed in a block of logic, the interconnect of the block (power, ground and signals) is built in a methodical and context-sensitive manner. The timing, noise characteristics,

and power consumption information associated with each gate and net is readily accessible from a unified design database which is described later. This information is used to precisely size the signal and power routes among the gates in the block. In this way, predictable timing closure, noise immunity, and reliability are achieved without being over designed.

Bundle routing is an efficient and robust way to route NDL signals. The N wires of a radix-N NDL signal are often routed together as a bundle for most of their route length. This provides several advantages. First, the 1-of-N signal encoding requires that, at most, only one of the N wires of this signal will assert at a given time. The result is that an actively switching wire has quiet neighbors and a quiet (victim) wire has at most one switching neighbor (aggressor). At the edge of a bundle, shields may be inserted to protect the edge wires from unrelated signals. While at first this may seem to be a costly use of interconnect resources, in practice the shields (which are tied to power or ground) become part of the power/ground grid. A second advantage to bundle routing is that the routing of the design is more regular. All wires of a bundle have the same source and destinations, and routing them together reduces blockages for unrelated routes.

The wire twizzling technique is a patented method for routing NDL signals. This technique further reduces coupling noise beyond the benefits provided by bundle routing. The wire twizzling method changes the order of the wires in a bundle to reduce by another factor of two the exposure of a victim wire to an aggressor wire. Usually, this signal re-ordering is opportunistic, occurring without cost when a signal changes metallization layers during a direction change. The routing tools keep track of the twizzle state of the route, attempting to keep half the route length in one state and half the length in the alternate state. Wire twizzling is shown schematically in Figure 7.

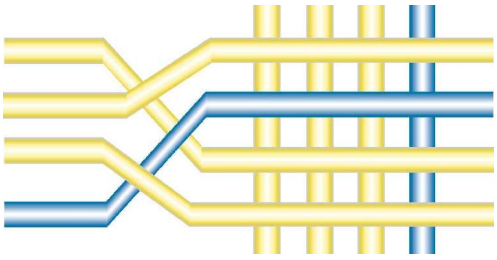

Figure 7: Wire Twizzling (Conceptual Diagram)

Note that in many radix cases (for example, radix-2 and radix-3), a ground or power shield needs to be routed in the bundle to participate in the twizzling process. In practice, however, wire spacing in combination with bundle routing is usually sufficient to solve noise problems. The routing tools automatically choose the lowest cost method to route each signal or bundle, employing wire spacing, wire width, shielding, bundle routing and wire twizzling as appropriate for the specific signal and the noise sensitivity of the attached gates.

The final step in the routing process is the insertion of redundant vias wherever space allows. Note that multiple vias are used in the initial signal or power/ground route when required for signal speed, electromigration or voltage drop reasons. The redundant vias are not required for these purposes, but rather to improve the yield and manufacturability of the chip.

Fast14 Technology Unified Design Database

Fast14 Technology implementations are managed by a powerful, unified design database. This state-of-the-art database was developed by Intrinsity using industry-standard data models. It offers logical (netlist) and physical (layout) capabilities very similar to those found in popular design environments such as those offered by Cadence, Mentor or Synopsys. Intrinsity's database is optimized for storing geometrical data and provides an efficient platform for tools like routers and layout generators.

In addition to the normal netlist and layout data, the database-programming interface provides sophisticated methods for managing hierarchy and occurrence properties. These features allow the database to store timing and wiring constraints and help solve the problem of managing such constraints across the design hierarchy.

This design database can output netlists in many industry-standard formats such as Verilog, CDL, and SPICE. The database can also output standard GDSII layout data.

Third-party logic and layout tools are used to create custom circuits such as SRAM arrays and I/O buffers. Conventional third-party synthesis and place-and-route EDA tools are used for low-speed logic blocks. All of the resultant data from the three design styles (Fast14 Technology, custom, synthesis) is translated into and managed by the Fast14 Technology unified design database.

Fast14 Technology EDA Tools

The Fast14 Technology EDA tool suite is designed to work with the unified design database and automates the design of high-performance circuits. The EDA tool suite provides a design environment with the

following features:

- Hybrid cycle/event-driven logic simulation using inexpensive computer hardware
- Electrical and physical constraint management across levels of design hierarchy
- Module-level and chip-level floorplanning and integration
- Automatic netlist generation and transistor sizing
- Rapid, deterministic timing closure to multi-GHz performance targets
- Automated cell layout
- Block assembly and routing

These tools, including the logic simulator, operate on inexpensive Linux computer hardware.

Interfaces to various commercially available EDA tools are provided for the purpose of simulation and electrical/physical analysis and also to provide methods to integrate custom or synthesized blocks into the chip design.

Figure 8 illustrates how the Fast$_{14}$ Technology EDA tools work together and interface to commercially available EDA tools.

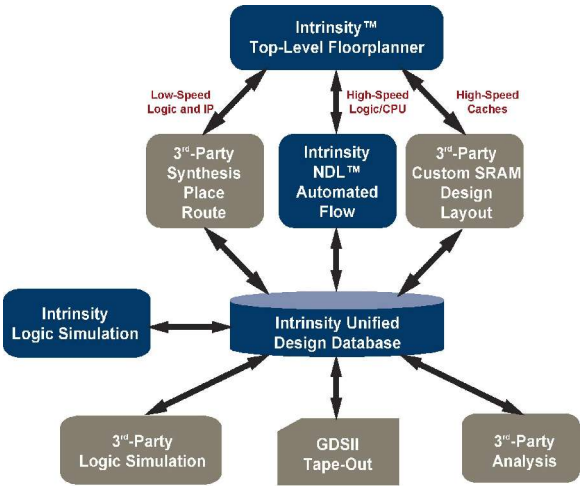

Figure 8: Fast$_{14}$ Technology EDA Tool Overview

The Intrinsity Top-Level Floorplanner tool integrates blocks of NDL gates, blocks of full-custom logic (such as SRAM arrays) and blocks of conventional synthesized static logic. The Floorplanner tool allows the designers to manage timing and physical constraints among the various blocks. It is also the environment in which the top-level route is performed.

The NDL Automated Flow generates multi-GHz digital logic and layout from the designer's source code. This flow includes tools that generate the transistor netlist, optimize gate placement, automatically size all of the transistors to meet the desired speed goal, and generate the layout for the block. This flow is similar in function to the conventional synthesis and place-and-route flows, except that it can generate robust logic circuits operating at multi-GHz rates.

As part of the NDL Automated Flow, NDL gate sizing occurs in the context of the block environment, analyzing all of the gates and the wires connecting the gates. Taking into account estimated wire (RC) delays, NDL gates are assigned appropriate delay goals. Complex NDL gates are allowed to "borrow" time from more simple gates in the upstream or downstream cones of logic. This results in significant power savings in critical paths. This sizing flow converges to the desired frequency target after one or two passes. Each sizing pass takes less than an hour for small blocks of logic, and up to several hours for large blocks of logic. Because of this rapid and deterministic timing closure, designers are able to perform numerous logic optimization iterations on a daily basis. This level of optimization can be achieved at lower frequencies with conventional synthesis tools, but is not achieved in custom design flows because of the labor-intensive nature of those flows.

As described earlier, the NDL logic family enables complex logic functions to operate efficiently at multi-gigahertz speeds. However, the function of data storage and long-distance data transmission is typically best performed by binary-encoded static-logic elements such as registers, latches and inverters. The NDL Automated Flow allows NDL logic gates and static-logic gates to be intermixed in a block. NDL logic gates and static elements are allowed to interface to each other according to a prescribed set of rules. These rules, which are checked automatically by the Fast$_{14}$ Technology EDA tools, ensure that the resulting design is free of timing problems.

The automation of dynamic-logic design in the Fast$_{14}$ Technology EDA flow is unprecedented in the industry and allows Intrinsity to generate circuits that are extremely fast but also highly manufacturable and robust. The possibility of human error is essentially eliminated because of this automation. Additionally, since blocks can be generated quickly from source code to polygons, designers can evaluate many solutions to a design problem before deciding which solution provides the best results.

The Fast$_{14}$ Technology design flow relies on industry-standard analysis tools to check the quality of the design. The design database outputs GDSII layout for

parasitic extraction and analysis. Timing, electromigration, IR drop analysis, DRC, and LVS checks are performed by industry-standard third-party tools (though significant levels of EM, IR and noise analysis are also performed by Fast$_{14}$ Technology EDA tools during the create phase). This ensures the quality and correctness of the design to a state of the art level.

Fast$_{14}$ Technology Value

Fast$_{14}$ Technology represents a significant breakthrough for the automation of high-speed digital logic design. The value of a design technology can be measured by operating speed, power efficiency, productivity, manufacturability, and risk of design error.

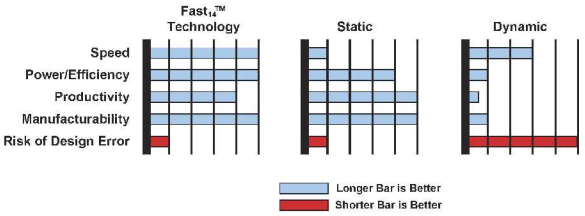

Figure 9: Design Style Comparison

Fast$_{14}$ Technology delivers a substantial speed improvement over conventional static-logic design styles. When compared to dual-rail dynamic logic, the removal of registers, the increased tolerance for clock uncertainty, and the capacitance isolation provided by the NDL logic family provide a significant speed improvement.

When considering power efficiency, it is important to consider the delivered performance. For example, at low levels of performance (below about 500 MHz in a 130 nm technology), pure static logic can be a more power efficient technology than Fast$_{14}$ Technology. Above approximately 500 MHz, the power efficiency of Fast$_{14}$ Technology exceeds that of a static-logic design style. This power efficiency advantage increases further at higher levels of performance. This is evidenced by the fact that there are no commercial logic products of significant complexity implemented in a full static-logic design style above approximately 500 MHz (130 nm, 1.0 V nominal voltage).

NDL gates enjoy a substantial power efficiency advantage over classical dual-rail dynamic logic at any level of performance. The 1-of-N logic encoding provides power savings by reduced interconnect switching power and also because of the improved speed/power ratio inside logic gates. Another source of power savings is the cycle time gained by the removal of registers from critical paths and by the increased tolerance for clock uncertainty (this essentially gives more time to the gates doing the actual work).

Although Fast$_{14}$ Technology derives its speed from the use of the NDL logic family, it allows the integration of static gates in the design. Static elements such as registers and normal inverters are used for power and area-efficient data storage and transmission. NDL gates are used for combinatorial logic and computation elements. The unique Fast$_{14}$ Technology clocking scheme and the supporting EDA tools allow these styles to be mixed in a seamless manner without the design risk that accompanies the interface between static and dynamic logic in traditional designs.

At high performance levels (above about 500 MHz), Fast$_{14}$ Technology has a productivity advantage over conventional synthesis flows. This is because synthesis flows require a great deal of time to achieve timing closure at high clock rates. The effort required to achieve timing closure offsets the advantage that synthesis flows have in the level of abstraction of the source code. At higher levels of performance, conventional static design is not capable of matching Fast$_{14}$ Technology performance with any level of effort. However, at lower frequencies, the higher level of abstraction in the source code gives conventional synthesis flows a design productivity advantage over Fast$_{14}$ Technology.

Relative to high-performance custom design (static or dynamic), Fast$_{14}$ Technology has a dramatic design productivity advantage.

The manufacturability of a Fast$_{14}$ Technology design is significantly improved over a traditional dynamic-logic design and over custom static designs. This is due to the elimination of human error in the circuit design and layout of these critical circuits. The robustness of circuits from the NDL logic family, the expert routing technology, and the powerful Fast$_{14}$ Technology EDA tools deliver products that have similar (or in some case, better) manufacturability than products developed with established static-design methodologies.

The automation capability of the Fast$_{14}$ Technology EDA tools enables a designer to pick a performance goal and quickly get feedback on power. The power of the final product depends on speed, voltage, and function. Fast$_{14}$ Technology enables such a leap in speed that often fewer functions are required to achieve a given performance goal. With the ability to easily generate multi-GHz circuits, it is also simple to retarget a design point to 1 GHz at a very low voltage

to achieve a more power-efficient chip.

Future Developments of Fast14 Technology

Fast14 Technology is a breakthrough technology that has reached commercial production status. The technology will continue development by Intrinsity to improve speed, power, area, and productivity.

Power is an area that will receive continued invention. While products from Intrinsity today deliver breakthrough performance in systems requiring 3-W to 15-W power budgets, future products will continue the migration to markets that require battery power. Fast14 Technology is capable of delivering significant product advantages where performance, power efficiency, or cost efficiency are key market requirements.

Opportunities exist to take the technology to broader markets. Logic synthesis tools that support dynamic logic are key to enabling engineers familiar with static logic to enjoy the two-to-three-times speed improvement seen with Fast14 Technology.