Cahners
MICROPROCESSOR REPORT
www.MPRonline.com
THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

TECHNOLOGY 2001: ON A CLEAR DAY YOU CAN SEE FOREVER

By Max Baron {2/25/02-01}

Unlike products that can be evaluated by comparing them to their competition and by assessing their usefulness at a given time, technologies offer a glimpse into a probable future and are defined by the likelihood of their becoming reality. At *MPR*, our world consists of microprocessors, digital-signal processors, and application-specific digital engines and the ways they will be created, programmed, and used in the future. Will they transform life as profoundly as the internal combustion engine, the jet plane, the computer, and the Internet? Will they be successful, useful, and accepted? Some technologies seem to have a sure short-term impact. Others, like icebergs in time, may extend their mass forever. Technology nominations and the ultimate Annual Award are *MPR* analysts' votes for the future, glimpses into what might happen and what processors might be feasible. This year's limelight is shining on logic and architecture design technologies.

MICROPROCESSOR REPORT ANALYSTS' CHOICE
Best Technology
Hyper-Threading

Technologies can't appear overnight; their roots are in accumulated scientific knowledge and methods and the hopes of many contributors—universities and other organizations. It is with this humble understanding that we limited our search to technologies that were recently presented by commercial companies or used by them to create test products.

Nominated Technologies

The analyst staff of *MPR* recognizes the following technologies as nominees for the Annual Award for Outstanding Technology in the Field of Digital Processing: Theseus Logic's NULL Convention Logic, Sun Microsystems Laboratories' Asynchronous Technology, Intel's Hyper-Threading technology, and Proceler's Soft Hardware technology.

Asynchronous Logic

The continuing quest for higher performance is increasing the complexity of implementing clocked circuits. Circuit requirements and clock skews contribute an increasing overhead to the higher performance otherwise available from the logic itself. Asynchronous design now seems more important than ever: it may provide significantly higher throughput, lower electromagnetic interference, easier design reuse, and reduced power dissipation. Asynchronous logic design in the United States started as early as 1951–52 with the completion of the ORDVAC, built at the University of Illinois, and the IAS, built by John von Neumann's group at Princeton University. On the other side of the Atlantic, at

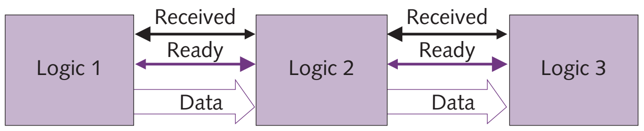
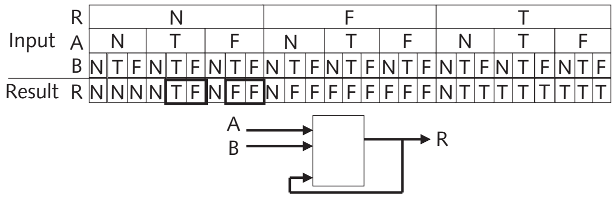


Figure 1. Data and protocol between asynchronous units.



	R	N									F									T								
Input	A	N			T			F			N			T			F			N			T			F		
	B	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F
Result	R	N	N	N	N	T	F	N	F	F	N	F	F	F	F	F	F	F	F	N	T	T	T	T	T	T	T	T

Figure 2. Simplified binary principle of Theseus Logic's NULL Convention Logic (NCL). (Source: Theseus Logic)

Manchester University, the MU5 computer was built between 1969 and 1974.

Conceptually, an asynchronous logic implementation can be viewed as a unit or module that controls its acceptance of inputs and the validity of its outputs on the basis of some form of communication between itself and the units or modules between which it is sandwiched. Lacking clocked registers or latches to store inputs and the results of logic operations, asynchronous implementations need ways to define when input data is complete and valid; when logic operations' outputs become valid; how to latch valid output data until it's ready to be consumed by the next stage(s); and how—upon data acceptance by the next stage(s)—to signal readiness for new inputs for the next logic operation. A simple conceptual block diagram appears in Figure 1.

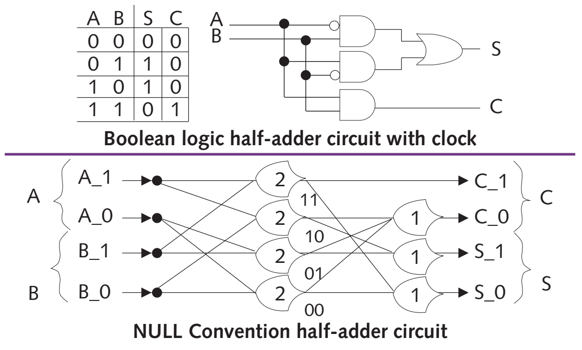
In one implementation, binary data is delivered as one line (wire) per each binary bit, with additional, separate lines used to implement the handshake protocol that indicates the readiness of valid output data and the acknowledgment of its reception by the destination unit. A different approach has data delivered via two lines per binary bit, with the assertion of one line indicating a logical "0," and the assertion of the other line, a logical "1." The remaining states in the two-lines-per-bit design are used for status and handshake protocol; a return to nonasserted status is required between valid data states.

Theseus Logic's NULL Convention Logic

Theseus Logic, founded in 1996, is a semiconductor intellectual property company dedicated to the design of asynchronous ICs using its NULL Convention Logic (NCL) technology. In its asynchronous logic design, Theseus Logic Inc. has adopted an encoding that uses two lines per bit of binary data. The design uses the remaining states to signal lack of data, or "null" data. The result may be fed back to the input gates to latch and protect the unit from further change and also to communicate to the previous unit that it may start working on generating the next set of outputs (separate lines shown for clarity). Assuming a pure binary representation (Figure 2), that on two lines can generate a four-state variable, the truth table shown in Figure 2 demonstrates the result based on inputs A and B and the previous value of R, the result line.

Theseus Logic avoids using four-state logic by incorporating threshold gates whose function is to check that the required (or higher) number of inputs has turned from "null" to valid data. The simple half-adder example shown in Figure 3 is taken from Theseus Logic's white paper, available at *www.theseus.com*. The number inside a threshold gate defines the number of inputs that must be asserted for the gate to assert its output. All lines are indexed as x_1 and x_0 to refer to their being asserted when the binary value of data is "1" or "0." The asynchronous register principle is diagrammed in Figure 4, showing the feedback from the next stage and the output to the previous one. Note that the circuit, shown here in its simple form, is not testing for errors in exclusivity of data.

Theseus Logic announced in March 2001 the results of its translation of Motorola's HC08 processor. Theseus Logic has designed and fabricated the NCL08, an instruction-set-compatible NULL Convention Logic (NCL) implementation of Motorola's S08 processor core, the latest processing element of the 6808 family of microcontrollers. Motorola's STAR08 processor can run at up to 100MHz in 0.25-micron technology. Theseus used a 0.25-micron technology for its chip which was fabricated by TSMC. As with asynchronous designs, the performance of the NCL08 is dependent upon



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A
B
S
C
Boolean logic half-adder circuit with clock
A
A_1
A_0
B_1
B_0
B
2
2
2
2
11
10
01
00
1
1
1
C_1
C_0
S_1
S_0
C
S
NULL Convention half-adder circuit

Figure 3. Half-adder comparison: clocked vs. asynchronous. (Source: Theseus Logic)

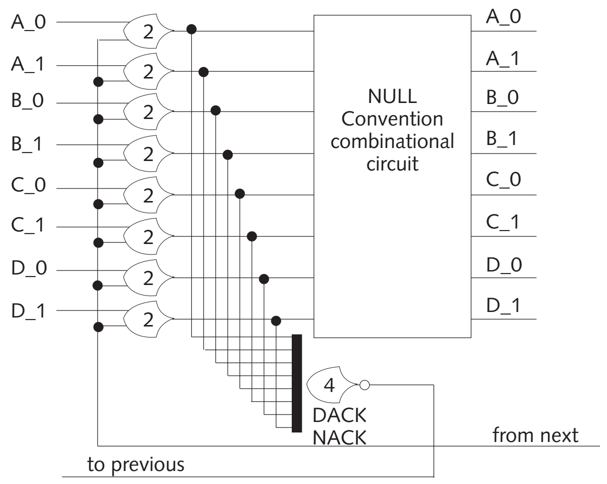


Figure 4. Asynchronous register. (Source: Theseus Logic)

Voltage	Temp	Process	Average Performance f_{equiv} (MHz)	P_{avg} (pJ)	Average Power (μW/MHz)
2.75	-40.0	FF	47.3	1542	207.1
2.75	125.0	SS	25.5	1520	204.2
2.50	75.0	TT	30.5	1248	167.7
2.50	25.0	TT	32.9	1231	165.4
2.25	-40.0	FF	40.4	984.0	132.2
2.25	125.0	SS	21.0	978.1	131.4
2.00	25.0	TT	26.0	754.3	101.3
1.00	25.0	TT	6.8	167.8	22.5

Table 1. NCL08 average performance. (Source: Theseus Logic)

Voltage	Temp	Process	NCL Power Reduction
2.75	-40.0	FF	40.7%
2.75	125.0	SS	38.1%
2.50	75.0	TT	38.8%
2.50	25.0	TT	38.9%
2.25	-40.0	FF	39.9%
2.25	125.0	SS	37.8%

Table 2. Power dissipation comparison: NCL08 vs. Motorola's STAR08 at 27MHz. (Source: Theseus Logic)

process, voltage, temperature, and data. Table 1 shows the results of average performance expressed in equivalent frequency. Table 2 shows the reduction in power dissipation that was obtained by the asynchronous design.

The first results, showing approximately 38% less power requirement, are encouraging, especially for microcontrollers that need to consume as little power as possible.

Sun Microsystems Lab' Asynchronous Technology

At Sun Labs, under the guidance of Ivan Sutherland, the asynchronous group has taken a different architectural and implementation approach to asynchronous design. Guided more by communications between blocks than by logical functions that must be executed in one unit, the group has defined a different protocol and has introduced a design methodology that tailors all logical paths to fit a common overall timing specification.

The Micropipeline

Presented first by Ivan Sutherland in 1989, the micropipeline approach departs from the pipeline approach that is predominant today in clocked circuits that employ logic units interleaved with clocked registers. Instead, it proposes a flexible pipeline stage that is started and terminated not at the beat of a clock but by an event—that event being a transition or handshake between asynchronous units. It is argued that in addition to the speed benefits, the boundary defined by the micropipeline approach makes it easier to integrate reusable cores in system-on-chip (SoC) designs. Sun Labs' asynchronous protocol uses "bundled data," which involves one line per binary bit of data and a separate request-acknowledge transitional protocol (Figure 5).

Using the protocol adopted by Sun Labs, the Sender unit validates data and asserts the Request line. The Receiver accepts the data and signals via the Acknowledge line that the data has been received and that the sender may now start to work on its next set of data. Rising and falling edges are both used interchangeably to signal an event, and the resulting logic value on the signal line is of no importance. On the event lines, only transition events have meaning. The event logic uses Muller C-elements to perform a logical AND of its inputs. The Muller C-element outputs a "0" if all its inputs are "0"; conversely, the output is a "1" if all its inputs become "1." Following a change of this kind, the Muller C-element stores the state and retains it for as long as its inputs do not change or become different from one another. The Muller C-element, also called the "rendezvous" circuit, will produce the next event only if and when all its inputs have each produced events.

Figure 6 is a block diagram of asynchronous logic units connected as a micropipeline. As with synchronous logic, the asynchronous logic function units are sandwiched between latches, the difference being that the latches are controlled by events instead of by clocks. Registers (RG) will be transparent until a Muller C-element (gate with letter "C") detects a Request event and asserts the capture control "C" on the latch. The "Capture done" (Cd) signal is output when the data has been latched. Cd generates an Acknowledge signal that releases the latch of the previous stage by its "P" input. Note also that Cd and a delay generate Request events. This delay has to match the worst-case delay of data flowing through the logic under all operating conditions.

Sun Labs has added to these already-known asynchronous architecture and protocol items, its design technology based on the concept of Logical Effort. The methodology helps obtain uniform gate delay by tuning transistor widths. Aside from the increased simplicity provided for logical design, the

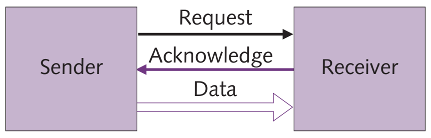


Figure 5. Sun Labs' asynchronous protocol concept. (Source: Ivan E. Sutherland, Turing Award Lecture)

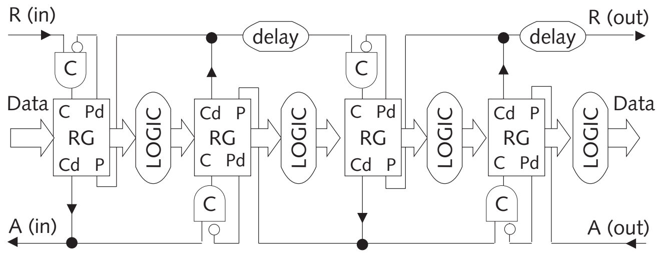


Figure 6. Micropipeline block diagram. (Source: Ivan E. Sutherland, Turing Award Lecture)

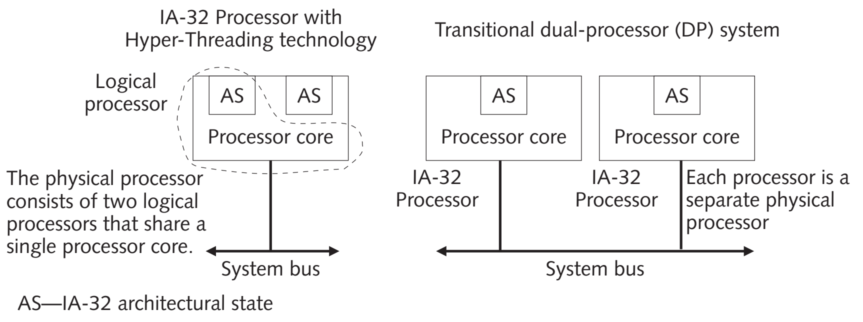

IA-32 Processor with Hyper-Threading technology
Logical processor
AS
AS
Processor core
The physical processor consists of two logical processors that share a single processor core.
System bus
AS—IA-32 architectural state
Transitional dual-processor (DP) system
AS
Processor core
AS
Processor core
IA-32 Processor
IA-32 Processor
Each processor is a separate physical processor
System bus

Figure 7. Hyper-Threading conceptual block diagram. (Source: Intel)

approach may be used to obtain minimal power requirements for any implementation, asynchronous or otherwise.

A Working Chip: FLEETzero

In addition to micropipelines, Sun Labs has developed new asynchronous circuit families, called asP* and GasP, that offer greater flexibility at similar or higher speeds and lower power consumption. The usefulness of both circuit families has been evaluated in recently fabricated chips. Parts of Sun's UltraSPARC III microprocessor use asynchronous FIFOs built from asP* circuits. Sun has also built an experimental chip, FLEETzero, using GasP circuits. FLEETzero was presented in March 2001 at the Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems. Sun built the FLEETzero to test an asynchronous switch fabric that can transport 8-bit data items from eight sources to eight destinations. Throughput results are equivalent to six gate delays per data word. In 0.35-micron technology, it delivered a throughput of more than 1.2Gwords per second at a latency of less than four nanoseconds from source to destination. The design of the FLEETzero chip emphasized data movement rather than logic operations, in line with Sun Labs' strategy for asynchronous design.

Intel's Hyper-Threading Technology

Intel's new approach at improving the performance of modern processor architectures is based on executing more than one thread per processor and is initially targeted at multithreaded server applications. Intel is delivering this technology on its new Xeon processor family for servers in 1Q02, for both dual- and multiprocessor servers. Following Intel's terminology, we will use the term "thread" to refer both to processes and to threads. Hyper-Threading, Intel's name for the new technology, takes advantage of the available cycles of highly pipelined processors that use instruction-level parallelism. Until now, these cycles couldn't be used, owing to limitations in the combination of hardware, workload, and compiler capability. A major effort has been directed toward overcoming the problems encountered in one instruction stream as computers try to increase efficiency of parallelism by executing as far as possible past dense conditional branches; this effort still continues. Increasing memory, cache bandwidth, and on-chip parallelism have brought forth many solutions, some of them stillborn.

Not long ago, a company named TeraGen, now apparently gone, attempted to use available parallelism in running multiple instruction sets on the same chip for embedded applications, all in parallel. One might say, by definition, that if that engine had become a reality, it would then have been able to run multiple programs in parallel. One might also say that recent embedded microprocessor coprocessor and slave processor extensions, such as the ones used for accelerating special data types, are also, on the same chip, executing on different datastreams. In all cases, however, the different types of execution were envisioned to occur on different, dedicated on-chip processors or accelerators, each with its own one-machine state. Intel's approach does not dedicate execution units; it can treat similar execution units equally, as it has introduced on-chip support for one more state machine (Figure 7). Intel claims its Hyper-Threading technology delivers two logical processors that can execute different threads simultaneously, using shared hardware resources, thus providing an average improvement of approximately 40% in CPU resource utilization and in the associated processing throughput. Hyper-Threading technology can provide a performance boost on multithreading and multitasking operations for Intel's future NetBurst microarchitecture processors as well as for other architectures. It is interesting to note, by using the average percentage improvement, the possible contribution of Hyper-Threading to computer architecture state of the art.

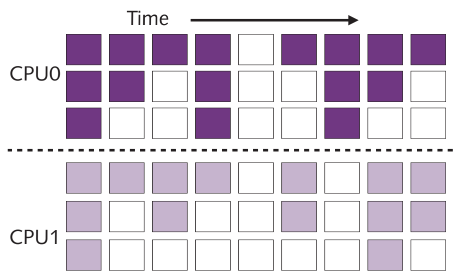


Figure 8. Multithreading using two CPUs. (Source: Intel)

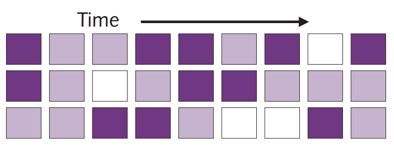


Figure 9. Hyper-Threading. (Source: Intel)

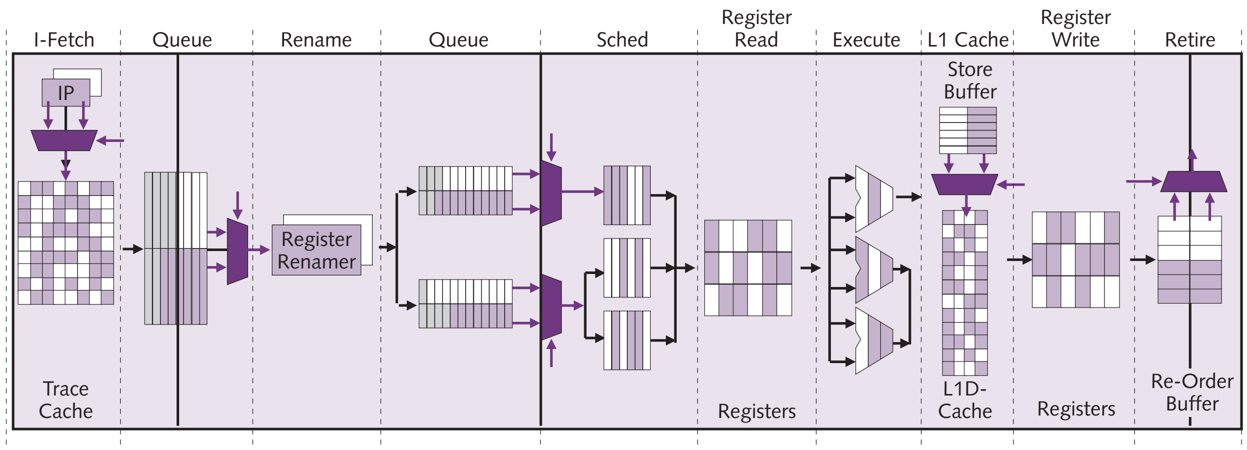


Figure 10. Intel NetBurst architecture with Hyper-Threading capability. (Source: Intel)

A Conceptual Description

Until Intel announced its approach, running multiple threads required multiple CPUs. In Figure 8, two threads are seen running in parallel on two separate CPUs, each with its own efficiency of resource utilization.

In Figure 9, the two threads are shown running on the same CPU, using its resources during cycles that were previously unused because of dependencies.

The innovation in Intel's approach lies in the limited additional circuits required to support two logical CPUs on one machine's resources. Requiring an additional 5% silicon real estate to implement the two logical CPUs is accomplished by duplicating machine-state-maintaining registers, such as the EAX, EBX, and Control registers, the Interrupt Controller, and others. Other resources, such as integer and floating-point registers, are shared by the threads, with register renaming keeping track of the threads in addition to their previous utilization.

Figure 10 shows how two threads are processed on the same CPU and highlights the points at which the implementation can switch from one thread to another.

Status

Intel has published preliminary Hyper-Threading technology performance numbers on a prototype Intel Xeon processor MP platform. The first results are showing an overall improvement of up to 30% compared with the same unit with Hyper-Threading disabled.

Proceler's Soft Hardware

Proceler uses a commercial platform comprising a microprocessor coupled with a commodity FPGA to deliver application-program-customized microarchitectures. Programmers used to dream about being able to define a custom processor by stating what its instruction set should do. Proceler takes the idea one step further.

Proceler's technology processes an ANSI C application program to determine compute-intensive kernels, such as iterative structures and loops. Proceler then uses its compiler and a set of predefined hardware structures to assemble a special-purpose instruction set architecture (ISA) that, downloaded into the FPGA together with the application processor, can speed up execution of the ANSI C program. Pioneered by Proceler, this approach is referred to as "architecture assembly."

Proceler defines its technology as a Dynamically VAriable Instruction SeT Architecture (DVAITA). The hardware-software optimization of an application program (Figure 11) begins with Proceler's compiler identification of compute-intensive kernels and a distribution of processing tasks between the application (host) processor and the special processor about to be created by the DVAITA tools. This distribution will ultimately define the datapaths along which operands and results will flow into and out of the host processor and soft processor engines that the compiler creates.

Performance-intensive sections, having been identified, are translated into an intermediate representation—a program-dependence graph. The performance-demanding kernels are then parallelized at the level of loops—using transformations such as distribution, loop unrolling, and tiling—and at the fine-grained instruction level (Figure 12).

Figure 13 shows a block diagram of the software process. A C source program is profiled, and, following program transformations and partitioning, its execution is distributed

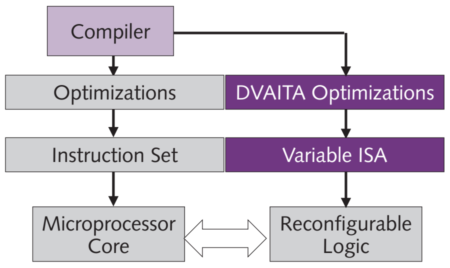


Figure 11. Proceler program partitioning. (Source: Proceler)

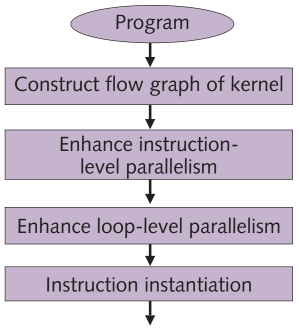

Program
Construct flow graph of kernel
Enhance instruction-level parallelism
Enhance loop-level parallelism
Instruction instantiation

Figure 12. Performance-intensive kernel flow and fine-grain optimizations. (Source: Proceler)

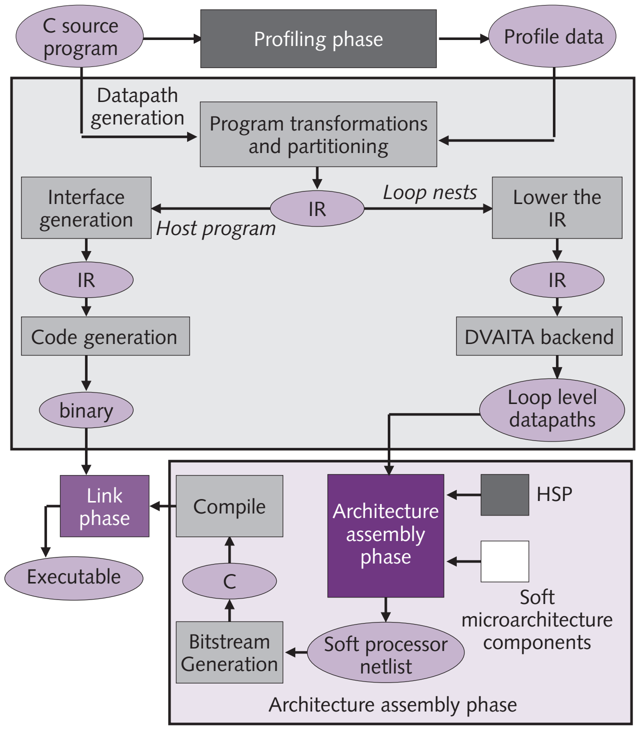


Figure 13. Partitioning, datapath creation, architecture assembly, and final C program generation. (Source: Proceler)

between the host processor and the about-to-be-generated custom processor.

The custom processor part uses subsequent levels of intermediate representation, its hardware support package (HSP), and a library of predesigned hardware building blocks consisting of storage, compute, and interconnect components. These components from the microarchitecture library, including instruction implementations, are then integrated to produce the customized microarchitecture that matches the data and control flow of the application code. The resulting program for the generated custom "soft" processor is compiled, and its binary is linked with the host processor binary to yield the completed executable. The existing version of Proceler's technology yields accelerators that can exchange operands and results with the host CPU via memory—or CPU loads and stores to specific addresses rather than to coprocessor architecture.

Status

Proceler's current offering accelerates ANSI C applications on a commercial platform comprising a microprocessor coupled with a commodity FPGA as the delivery vehicle for the customized microarchitectures. In general, DVAITA technology or variants thereof can have a significant impact on SoC and communications infrastructure implementations. This impact can dynamically reconfigure the implementations' resources to match applications instead of using more-expensive fixed resources. Is Proceler's approach the precursor of the soft host processor? That remains to be seen.

And the Winner Is …

Simple ideas can have an immediate impact on technology and can continue to influence it for many years to come. One can only imagine the performance that will be obtained by implementing extensive portions of computing engines in asynchronous logic. Advances recently announced by Sun Microsystems and by Theseus Logic have carried asynchronous technology into important implementations of processors and circuits that exhibit compelling performance gains and power reduction. Proceler's soft processor technology, now aimed at development tools and still in its infancy, may develop or be reborn into variants that can change our view of configurable processors, extensions of instruction sets, hardware interpreters, and application-specific accelerators. The closest to immediate-throughput results is, however, Intel's Hyper-Threading technology, to be delivered on its new Xeon processor family for servers in 1Q02. It will speed up server applications, and it will improve the performance of desktops running multiple concurrent processes and threaded application programs.

The analyst staff of *Microprocessor Report* is honored to offer the year 2001's Annual Award for **Outstanding Technology in the Field of Digital Processing** to the architecture team that originated **Hyper-Threading technology** at **Intel.**