Superscalar Scorecard: Who's On First?

Two Design Styles Tradeoff Clock Rate vs. Complexity

by Brian Case

Over the past couple of years, a surprising number of superscalar microprocessors have been announced, and nearly as many have actually been shipped. Most superscalar implementations are high-end microprocessors aimed at the workstation market, but the large and growing number of these high-performance implementations implies that superscalar techniques are becoming widely understood and will be used in all classes of microprocessors in the future.

Superscalar techniques were first explored in the late 1960s for the IBM 360/91. For microprocessors, superscalar designs were discussed in earnest in the mid-1980s when single-pipeline, one-instruction-per-cycle RISC implementations were the state-of-the-art. Aggressive implementations with duplicate execution resources-such as two integer ALUs-were proposed, and these were given the most attention because they seemed to offer the biggest performance boost and were clearly different from single-pipeline machines. As it turns out, many designers have taken a more conservative approach: instead of duplicating resources, many superscalar implementations simply dispatch multiple instructions into existing processor resources. As an alternative, superpipelining-as in the R4000-offers performance improvement without the necessity of multiple instruction dispatch.

Due to the ingenuity of processor designers, the current crop of superscalar designs covers a broad spectrum, and some interesting tradeoffs have been made, such as degree of superscalar dispatch vs. clock rate. The following sections cover basic superscalar design issues, the superscalar design space, and each major superscalar processor. Finally, possible future directions are discussed.

Design Issues

The designer of a superscalar processor faces a set of issues not relevant to simpler organizations. With each issue comes a range of implementation choices and corresponding opportunities for tradeoffs. Refer to Table 1 for a comparison of how the current important superscalar processors dealt with each issue.

The most fundamental characteristic is the maximum number of instructions allowed to be issued and executed in one cycle. To issue an instruction is to fetch it from instruction storage (usually cache) and send it to an appropriate execution unit. To execute an instruction is to perform the operation specified by the operation code. The issue and execution rates in simple superscalar machines are usually equal, but queues (or reservation stations) in execution units can buffer issued instructions, resulting in a potential instantaneous execution rate greater than the issue rate. Since the average execution rate cannot exceed the issue rate, the degree of a superscalar design is usually equated to its issue rate; a processor that can issue two instructions per cycle is called a degree-two superscalar design (or sometimes a "two-scalar" processor).

The ability to issue multiple instructions per cycle implies the ability to fetch multiple instructions per cycle from an instruction cache. Fetching an item from any storage-instruction or data-is always easier when the item is aligned on a boundary equal to its size. Superscalar RISCs benefit from the architectural characteristic of fixed-size, 32-bit instructions; as a result, superscalar RISCs have simpler fetch and dispatch logic than comparable superscalar CISCs.

For a superscalar RISC processor fetching two 32-bit instructions in a cycle, it is easiest to require the pair to be aligned on a 64-bit boundary. For software (compilers, assemblers, and linkers) on the other hand, it is easiest if instruction streams can be generated without regard to alignment. These conflicting considerations raise a design issue for superscalar versions of processors that have large installed software bases. Since existing programs were generated by development tools that did not take into account any alignment preferences of superscalar processors, it is important for these implementations to accommodate arbitrary instruction alignment as much as possible.

Building a superscalar implementation is difficult in part because instructions have interdependencies. For example, a conditional branch might depend on the outcome of a compare instruction, the compare instruction might depend on the result of an add, and the add might depend on a load. It is unfortunate but true that such a group of four instructions must be executed serially. Instruction dependencies create a fundamental limit on available parallelism, and thus a limit to the effectiveness of superscalar processor organizations (see µPR 9/19/90, p. 8).

Two superscalar issue techniques are possible. The first issues instructions to queues in execution units. Dependency checking is performed in each execution unit queue to make sure that instructions are executed in the correct order. The second technique performs depen-

Characteristic \ Processor	Intel 960CF	Intel 960MM	HP PA-RISC 7100	IBM RIOS RS/6000	IBM RSC	DEC Alpha 21064	Cypress hyperSPARC	TI SuperSPARC	Motorola 88110	Intel P5
Group Size	4	4	2	5	3	2	2	3	2	2
Maximum Issue Rate	3 (2 sus.)	3 (2 sus.)	2	4	2	2	2	3	2	2
Integer Issue Rate	1+	2	1	2/1	1	1	1	2	2	2
Dependent Int. Ops.?	No	No	No	No	No	No	No	Yes	No	No
FP Issue Rate	No FPU	1	1	2/1	1	1	2/1	1	3	1+
Simul. Int. & FP Disp?	–	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Memory Units	1	1	0	0	0	1	1	1	1	2
Branch Unit	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Speculative Issue	Some	Some	No	No	No	No	No	Some	Yes	No
Speculative Execution	No	No	No	No	No	No	No	No	Yes	No
System Benchmarked	–	–	887	580	220	–	–	SS10/41	–	–
Clock Speed	–	–	96 MHz	62.5 MHz	33 MHz	150 MHz	66 MHz	40 MHz	50 MHz	66 MHz
SPECint92	–	–	78	59	15.9	70 est.	62	52.6	51*	60 est.
SPECfp92	–	–	142	125	22.9	120 est.	64	64.7	74*	60 est.
Price/Speed/Quantity	\$180.90 33 MHz 1K	n/a 40 MHz	n/a	n/a	n/a	\$1559 150 MHz 1K	\$700† 55 MHz 10K	\$400† 33 MHz 10K	\$495 40 MHz Samples	n/a

Table 1. Key attributes of major superscalar processors (*SPEC89; †1993 pricing).

dency checking in the primary instruction issue unit. This technique has the advantage of simpler execution units but the disadvantage of more restricted issue capability.

The second technique is used in most current superscalar processors where the main fetch-and-issue unit performs dependency checking, and most execution units are essentially simple, bare pipelines. Four of the processors covered here go further: hyperSPARC and SuperSPARC have queues in their floating-point units, the RIOS chip set has queues in both its integer and floating-point units, and the 88110 has a sophisticated queue in its load/store unit.

The fetch-and-issue logic in all these processors considers the next group of instructions, where the group is a window into the instruction stream. For most processors, the group size is the same as the maximum number of instructions that can be issued at once, while for others-the 960 series, RIOS, and RSC-the group size is one instruction larger than the maximum issue rate. In any case, the dependency checking logic selects as many instructions as possible from the group and issues them.

Most of the superscalar machines discussed here can only issue instructions in program order. Program-order issue logic that considers a group of, say, three instructions will never issue instruction three unless instructions one and two are also issuable in the same cycle.

It is possible and profitable, however, to design a machine that can issue instructions out of order. The 960 series, 88110, RIOS, and RSC each have some out-of-order issue capability. The 88110 goes further to allow out of-order-execution. It is important to note that out-of-order issue and out-of-order execution are not the same.

Since instruction dependencies can limit the effectiveness of a superscalar design, it is worth asking what can be done to counteract them. To a certain extent, compilers can be written to minimize instruction dependencies, but, again, for processors with large installed software bases, such a compiler can help only new or recompiled applications. This realization led the designers of SuperSPARC to implement a novel cascaded integer ALU structure that can accept and execute two dependent integer arithmetic instructions in the same cycle.

Another hardware technique that combats the anti-parallel effects of dependencies is speculative execution. Speculative execution usually revolves around predicting conditional branches: the branch is predicted and the instructions in the predicted direction are speculatively issued. The 960 series, SuperSPARC, and 88110 implement some degree of speculative issue. It is possible to only speculatively issue or to speculatively issue and execute. True speculative execution requires some sort of recovery mechanism-as with the 88110 register-file history buffer-to "undo" execution effects in the event of an incorrect speculation.

It is possible to devise and implement many hardware techniques to extract the maximum parallelism from an instruction stream, but there is a fundamental tradeoff at work: at some point, the complex logic lengthens the basic machine cycle or pipeline so much that more performance is lost than is gained. In recognition of the "complexity is not free" maxim, some superscalar processors emphasize high clock rate over sophisticated issue capabilities.

Superscalar Processor Organization

The hardware for a simple RISC or RISC-like processor contains a single pipeline, and it is not really organized into units because all instructions flow through the pipeline in virtually the same way. At each pipe stage, an instruction has private access to the logic contained in that stage. In contrast, the hardware for a superscalar processor is functionally partitioned into units that are capable of executing subsets of the processor's instruction set, and each unit is pipelined.

Figure 1 shows a simple block diagram comparison. The simple RISC pipeline allocates processor logic to stages in the single pipeline in a logical sequence. The superscalar pipelines reflect the same functions and sequence, but the functions are grouped into distinct sub-pipelines or units. In Figure 1, the superscalar machine has a fetch unit, two integer units, and a load/store unit. The fetch unit is responsible for executing branches, since branches directly affect the fetch function. The fetch unit also performs all dependency checking before issuing instructions to the simple execution units.

Figure 1 is just an example; many grossly or subtly different organizations are possible. Units may be very specialized, such as separate integer and load/store units, or more general, such as an integer unit that also takes care of loads and stores. The register-read function can be in either the fetch unit or the execution units. Each execution unit can be tightly coupled to the fetch unit pipeline, or execution units can have queues or reservation stations to buffer the flow of instructions.

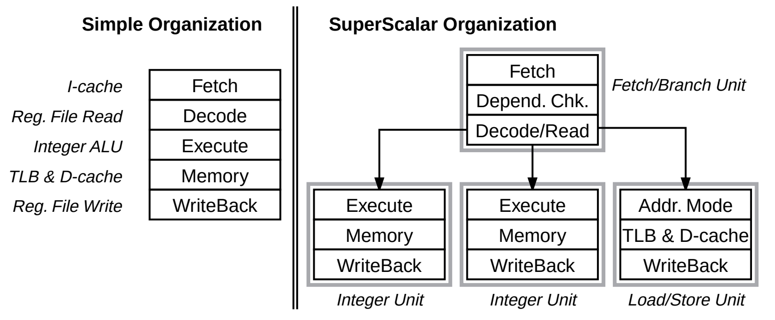

Figure 1. RISC Pipeline vs. Superscalar pipeline organization.

Levels Of Superscalar Aggressiveness

The aggressiveness of a superscalar design depends not only on its degree-the maximum number of instructions it can issue in a cycle-but also on the generality of its multi-issue capabilities.

The maximum number of instructions that can be issued in a cycle depends on the number of independent execution units available and the ability of the issue logic to detect instruction sequences that can use all the available units. It is common to have an issue rate less than the number of independent execution units because parallelism available in the instruction stream is limited and because of the desire to keep issue logic simple and fast.

The simplest superscalar organization is the result of partitioning the hardware found in any generic, pipelined processor. A simple partitioning creates a fetch unit capable of executing branches, an integer unit for integer ALU instructions, a load/store unit to execute memory reference instructions, and an FP unit for floating-point instructions. This organization essentially assigns disjoint subsets of the processor's instruction set to separate execution units. The advantage of this organization is that its implementation can easily build on an existing processor design; little extra logic is required to create autonomous execution units. The largest amount of design effort is spent creating the issue logic.

Note that the simplicity of this partitioning approach is affected by the instruction set. RISC instruction sets tend to have instruction subsets that do not have overlapping semantics; for example, integer ALU instructions do not perform memory references. CISC instruction sets are not as clean. At least one RISC, the IBM POWER architecture, was designed explicitly to facilitate this partitioning approach.

A more complex design duplicates one or more of the execution units. The most natural unit to duplicate is the integer unit because integer computation dominates most applications. While a superscalar organization created by the simple partitioning described above can simultaneously issue instructions only from disjoint instruction subsets, a superscalar processor with duplicated execution units can simultaneously issue two instructions from the same subset.

One problem with duplicated execution units is that dependencies can prohibit simultaneous execution of dependent instructions even when the required units are available. For some types of execution units, it may be possible to cascade duplicate execution resources within a single unit. As done in the SuperSPARC integer unit, this creates a single unit capable of executing two instructions from the same subset regardless of dependency.

Considering these options and the possible subtle variations, it is important to realize that the quality of a superscalar design is determined both by its degree and its issue restrictions. A certain processor may be able to issue four instructions per cycle, but if it has severe issue restrictions, it may actually exhibit lower performance on real applications than a degree-two processor with more general instruction-issue rules.

Partitioned Organizations

Most current superscalar microprocessors reflect the conservative design approach of partitioning traditional resources into independent execution units. Taking this approach allows the designers to reap superscalar performance benefits while leaving time to focus on other goals, such as reaching the highest clock rate possible.

Intel's 960CA was the first superscalar microprocessor announced; the 960CF is an updated version that improves performance with a larger instruction cache. It essentially uses the partitioning approach, but because of the architecture, it achieves some of the benefits of duplicated resources. It has an issue/branch unit, a full integer unit, and a separate load/store unit. Since some address-calculation instructions perform simple integer arithmetic and are executed in the load/store unit, it is possible to dual-issue and execute two integer instructions so long as care is taken to encode the simplest integer operation using the addressing mode equivalent. This capability, however, is not fully symmetric, since many integer operations cannot be coded using address-calculation instructions.

It is debatable whether or not IBM's first superscalar processor-the RIOS chip set used in the RS/6000 workstations-is a true microprocessor, since it uses several chips. The RSC and PowerPC 601 are newer, single-chip versions that unquestionably qualify as microprocessors. The RSC is a scaled-down RIOS implementation, and the 601 is a higher clock-rate version of RSC that implements the architecture and bus changes defined by the IBM/Apple/Motorola consortium. (The 601 will be unveiled at the Microprocessor Forum.)

The POWER and PowerPC architectures were designed to take advantage of the partitioning approach to superscalar implementation. The result is an instruction set that cleanly divides into subsets that use only the resources of a particular unit. The major instruction subsets are branch, condition-code, integer, and floating-point. As a result, RIOS was able to implement a degree-four instruction issue unit while others are limited to two- or three-issue. The integer and floating-point units have queues that allow two instructions to be issued to the same unit in a single cycle even though only one instruction can be executed per cycle.

When conditions allow four instructions to be issued to the separate execution units, five traditional operations can be started in a single cycle. While this is impressive, SPEC benchmark results show only average integer performance, which suggests that opportunities to issue three or four instructions per cycle are rare.

The RSC implements only a degree-two instruction issue unit. It may have been necessary to reduce the issue unit to squeeze the entire processor on a single chip, but the designers probably also realized that a degree-two issue unit would be a better match to the RSC's small 8K combined instruction/data cache.

HP has taken a conservative approach with its PA-RISC 7100 design. This chip is the result of simply integrating the integer and floating-point units on a single chip and permitting an integer and a floating-point instruction to be issued simultaneously. There are no separate branch or memory-reference units, and all non-FP instructions are in the same instruction class for the purposes of instruction issue.

The hyperSPARC from Cypress/Ross is another incremental design that builds on an existing processor design. The chip has five execution units: branch/instruction issue, integer, load/store, floating-point add, and floating-point multiply. The issue unit can dispatch two instructions per cycle, including the case of two floating-point instructions. While this would seem to give hyperSPARC an advantage in floating-point performance, floating-point instructions are first buffered in a queue in the floating-point unit. The queue can receive two instructions from the main issue unit in one cycle, but floating-point instructions are then re-issued to either the add or multiply unit from the queue at a maximum rate of only one per cycle.

DEC's first entry into the merchant microprocessor market-the 21064 “Alpha”-is also a straightforward partitioned design. This degree-two superscalar implementation has four execution units-branch/instruction issue, integer, load/store, and floating-point-and is considered superpipelined as well because integer shift operations require two cycles to execute. Independent shift operations can be issued every cycle, but dependent shifts incur a one-cycle dependency stall.

Among the chips in this group, the 960 and RIOS have the most aggressive raw capabilities. The 7100 is distinguished for its relative lack of independent units (no separate branch or load/store capability). Ultimately, the relative performance of the processors in this group is determined more by clock rate than superscalar capabilities. In this regard, the 21064 and 7100 have the edge.

Duplicated Unit Organizations

The currently available superscalar microprocessors with significant duplicated resources are the 960MM, 88110, and the SuperSPARC, but the industry waits on the edge of its seat in anticipation of the P5 from Intel.

The 960MM is an updated version of the 960CF that adds separate, full-fledged integer and floating-point units. The new integer unit gives the 960MM the ability to issue any two independent integer ALU instructions in the same cycle. Although the floating-point unit has separate add and multiply pipelines, a maximum of one floating-point instruction can be dispatched each cycle.

The 88110 has the most extensive partitioned and duplicated resource organization of any superscalar microprocessor. It offers a branch/instruction-issue unit, a load/store unit, two integer units, a bit-field (shift) unit, two graphics units, and three independent floating-point units (add, multiply, and divide). The degree-two issue unit can simultaneously dispatch two instructions to any two execution units with only minimal exceptions.

In addition to its impressive issue symmetry, the 88110 permits speculative instruction issue and execution after a predicted branch. A register file history buffer keeps track of registers so that they can be restored to their pre-branch values if the branch is later discovered to have been mis-predicted. Speculative execution allows the 88110 to extract the most performance from its two-issue capability.

While the 88110 has exceptionally symmetric issue rules, the three-issue SuperSPARC one-ups the 88110 in maximum issue rate and dependent integer operations. SuperSPARC has a single integer unit with three ALUs arranged as a tree; this cascaded arrangement permits SuperSPARC to simultaneously issue two integer ALU instructions (but not shifts) regardless of dependency (see Figure 2). The 88110 designers expected to rely on compilers to minimize dependencies in pairs of integer operations, which is a legitimate strategy for a processor without a large software base. SuperSPARC, on the other hand must provide significantly improved performance for a large installed base of software that was compiled without regard for minimizing dependencies.

The designers of Intel's P5 faced a problem that the designers of superscalar RISCs did not have to confront: how to deal with complex, non-RISC instructions. Clearly, many of the multi-cycle x86 instructions would not benefit much from multiple issue, but the instructions that perform simple operations in conjunction with memory references are candidates for consideration.

The two-issue P5 creates two integer execution units by duplicating the last three stages of the five-stage 486 pipeline, similar to the superscalar arrangement shown in Figure 1. This arrangement is also similar to the 88110 in that it does not allow simultaneous dispatch of dependent operations. The issue logic will dispatch two instructions into these units only if the instructions are simple and independent. Simple, in this case, includes instructions that perform not only pure register-to-register operations but also register-to-memory and memory-to-register operations.

Even though these two 486-like execution units will require more than one cycle to execute an ALU operation/memory reference combination, the ability to dual issue them is important since they are frequent in x86 programs. The P5 permits two memory references to proceed in parallel because of its dual-access on-chip data cache *(see* ***061201.PDF****)*.

The one significant difference between the P5 and other superscalar processors is its apparent inability to dispatch a floating-point instruction along with instructions from other classes. The P5 can, however, dual dispatch the special case of a floating-point operation together with a floating-point exchange instruction, which helps mitigate the impact of its stack architecture for floating-point operands.

Conclusions & Futures

It seems that the relatively conservative, partitioned approach to superscalar implementation has a clock-rate advantage over the more aggressive duplicated-resource approach: the highest clock-rate designs are the 21064, 7100, and hyperSPARC (assuming Cypress/Ross achieves its 80 MHz goal). The more complex SuperSPARC and 88110 are specified for clock rates no higher than 50 MHz, and SuperSPARC is struggling to achieve even that. At 66 MHz, the P5 will be no clock-rate slouch, but it will fall short of the standards set by DEC and HP.

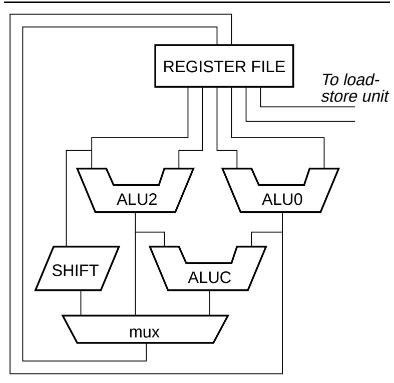

Figure 2. SuperSPARC uses a novel cascaded ALU configuration.

The clock-rate vs. complexity tradeoff is one of the most important issues in superscalar design. While adverse instruction dependencies and alignment can reduce the effectiveness of a superscalar implementation, they will not reduce the performance of a simple pipeline operating at a high frequency. Thus, it is crucially important to make sure that performance gained from superscalar capabilities could not have been gained more easily from clock-rate improvement (except when high clock rates might be inadvisable).

In reality, the clock-rate comparison is only partially fair. It is true that a more complex design, such as used in SuperSPARC and the 88110, leaves less time for performance tuning the hardware, but it is also the case that clock rate was a major focus during the design of the 21064 and 7100 chips. The designers at DEC were willing to make process concessions to achieve a record-setting clock rate that may not be feasible in other fabrication environments. Also, rumor has it that SuperSPARC's clock-rate problems are more the result of a goof in the design of the cache than of an overly complex superscalar design.

Another issue is the semantic richness of the instruction set. The 7100 cannot dispatch two integer operations per cycle, but many composite PA-RISC instructions achieve "superscalar-like" effectiveness. The P5 has a potential advantage in being able to dual-issue integer operations combined with memory references, but the current pipeline organization will require two or more cycles for these instructions.

It is interesting to speculate on how these microprocessors will evolve into more aggressive implementations. The most obvious option for the simpler, partitioned designs is the implementation of duplicate integer units. It is likely that next-generation PA-RISC, PowerPC, and Alpha processors will have two integer units because the performance benefit is large enough to justify the cost. Going beyond two integer units is questionable except in certain application areas or with significant help from compilers because of instruction dependencies. It is probably better to spend effort designing a dependency-tolerant integer unit, as in

SuperSPARC, or to add specialized execution units, as the 88110 has done for graphics operations.

The preliminary P5 disclosures have indicated another future direction: multi-access caches. The two P5 integer units have independent, simultaneous access to the single data cache as long as the accesses are to different cache banks. This allows the P5 to execute two memory operations simultaneously, a capability only approximated by the out-of-order execution of the load/store unit in the 88110.

In the future, the bank-conflict restriction could be lifted by implementing a true dual-ported data cache. Such a cache will be necessary to allow multiple issue and execution of memory references in RISCs. On-chip cache hierarchy design is increasingly important because cache access time is a key clock-rate limiter.

Floating-point is another area that can be improved on most of these processors. The 88110 can dual issue and execute different types of floating-point instructions, but the other superscalar processors cannot (hyperSPARC and RIOS can dual issue to floating-point queue but not from the queue to the floating-point units). While all of these processors would benefit from duplicated floating-point units, the IBM and HP architectures stand to gain less because their composite multiply-and-add operations yield a degree of superscalar-like effectiveness. The MIPS-architecture "TFP" microprocessor under development at Silicon Graphics is rumored to have duplicated floating-point units.

Another future direction that has already been set is the trend toward 64-bit architectures. MIPS and Alpha are available in 64-bit implementations, and PowerPC and SPARC version 9 will result in 64-bit microprocessors in the future. It is interesting to note that neither of the microprocessors with 64-bit ALUs have duplicate integer units, presumably because of the implementation cost.

The P5 is distinguished as the only superscalar CISC microprocessor anywhere near first shipment, but Motorola promises that the 68060 will be the second superscalar CISC. Another contender would have been a superscalar version of the 32000 family, but National chose to abandon the 32000 architecture when it designed its superscalar Swordfish processor (see µPR 2/20/91, p. 1).

The future directions for superscalar microprocessor designers seem clear: those who have high clock rate designs will add more superscalar capabilities and those who have high-degree superscalar implementations will be working for higher clock rates. As always, on-chip cache sizes will increase in order to keep the high-performance processor cores busy. ♦