A 0.6um BiCMOS Processor with Dynamic Execution

Robert P. Colwell, Randy L. Steck

A next generation, Intel Architecture compatible microprocessor with dynamic execution has been implemented with a 0.6um 4 layer metal BiCMOS process [1]. Performance is achieved through the use of a large, full-speed cache accessed through a dedicated bus interface feeding a generalized dynamic execution microengine. A primary 64-bit processor bus includes additional pipelining features to provide high throughput to this CPU and cache. These and other techniques result in a projected performance of greater than 200 Ispec92. Testability features built into the design allow complete access to all structures without the overhead of a full LSSD implementation. Included in this paper are a microarchitecture block diagram, implementation details and a die photo.

This processor implements dynamic execution using an out-of-order, speculative execution engine, with register renaming of integer [2], floating point and flags variables, multiprocessing bus support, and carefully controlled memory access reordering. The flow of Intel Architecture instructions is predicted and these instructions are decoded into micro-operations (uops), or series of uops, and these uops are register-renamed, placed into an out-of-order speculative pool of pending operations, executed in dataflow order (when operands are ready), and retired to permanent machine state in source program order. This is accomplished with one general mechanism to handle unexpected asynchronous events such as mispredicted branches, instruction faults and traps, and external interrupts. Dynamic execution, or the combination of branch prediction, speculation and micro-dataflow, is the key to the high performance.

The basic operation of the microarchitecture (Figure 1) is as follows:

1. The 512 entry Branch Target Buffer (BTB) helps the Instruction Fetch Unit (IFU) choose an instruction cache line for the next instruction fetch. ICache line fetches are pipelined with a new instruction line fetch commencing on every CPU clock cycle.
2. Three parallel decoders (ID) convert multiple Intel Architecture instructions into multiple sets of micro-ops (uops) each clock.
3. The sources and destinations of these uops are renamed by the Register Alias Table (RAT), which eliminates register re-use artifacts, and are forwarded to the Reservation Station (RS) and to the ReOrder Buffer (ROB).
4. The renamed uops are queued in the RS where they wait for their source data - this can come from several places, including immediates, data bypassed from just-executed uops, data present in a ROB entry, and data residing in architectural registers (such as EAX).
5. The queued uops are dynamically executed according to their true data dependencies and execution unit availability (IEU, FEU, AGU). The order in which any given uops execute in time has no particular relationship to the order implied by the source program.
6. Memory operations are dispatched from the RS to the Address Generation Unit (AGU) and to the Memory Ordering Buffer (MOB). The MOB ensures that the proper memory access ordering rules are observed.
7. Once a uop has executed, and its destination data has been produced, that result data is forwarded to subsequent uops that need it, and the uop becomes a candidate for “retirement”.
8. Retirement hardware in the ROB uses uop timestamps to reimpose the original program order on the uops as their results are committed to permanent architectural machine state in the Retirement Register File (RRF). This retirement process must observe not only the original program order, it must correctly handle interrupts and faults, and flush all or part of its state on detection of a mispredicted branch. When a uop is retired, the ROB writes that uop’s result into the appropriate RRF entry and notifies the RAT of that retirement so that subsequent register renaming can be activated.

The component includes separate data and instruction L1 caches (each of which is 8KB), and a unified L2 cache. The L1 Data Cache is dual-ported, non-blocking, supporting one load and one store per cycle. The L2 cache interface runs at the full CPU clock speed, and can transfer 64 bits per cycle. The external bus is also 64-bits and can sustain a data transfer every bus-cycle. This external bus operates at 1/2, 1/3, or 1/4 of the CPU clock speed.

Clock distribution was carefully designed to minimize skew across the entire die by generating a master clock with a PLL which has been delay synchronized to output signals. Global clocks distributed to 80 different units across the die are then buffered for the specific load in each clock sub-branch. Tuning of these drivers, and careful routing of the global clocks, results in a worst-case global clock skew of 250 pS.

BiCMOS circuits were used extensively thoughout the design, providing lower delay for higher loads, and increasing overall performance by approximately 15%. Figure 2 shows the relative delay to a comparable CMOS inverter and Figure 3 shows the most common implementation of a BiCMOS gate.

A Vdd of 2.9v was selected as an optimal point for CMOS and BiCMOS gate performance while reducing overall power.

Delayed precharge domino logic, shown in Figure 4, was also used for speed-critical paths, particularly in the instruction decode logic, and resulted in lower power than standard domino logic during transitions with fewer race conditions on outputs.

Test structures were defined allowing full access to all processor logic while requiring only 4% of the full die area and incurring no speed penalty. This is accomplished through the use of test registers, mode bits to provide specific logic access, and the use of serial Scanout (scan chains). Scanout in particular provides observability of virtually all important signals within the design with no speed impact, and it is used extensively in test and debug. On-chip BIST is implemented to complement the Scanout observability. The processor is a fully static design accommodating IDDQ testing.

Features for lower power operation, (such as StopClock and standby mechanisms) and features intended to improve system management and RAS (Reliability, Availability, Serviceability) are included. An extensive Machine Check Architecture has been incorporated, with facilities to detect errors in hardware and allow those errors to be handled in software.

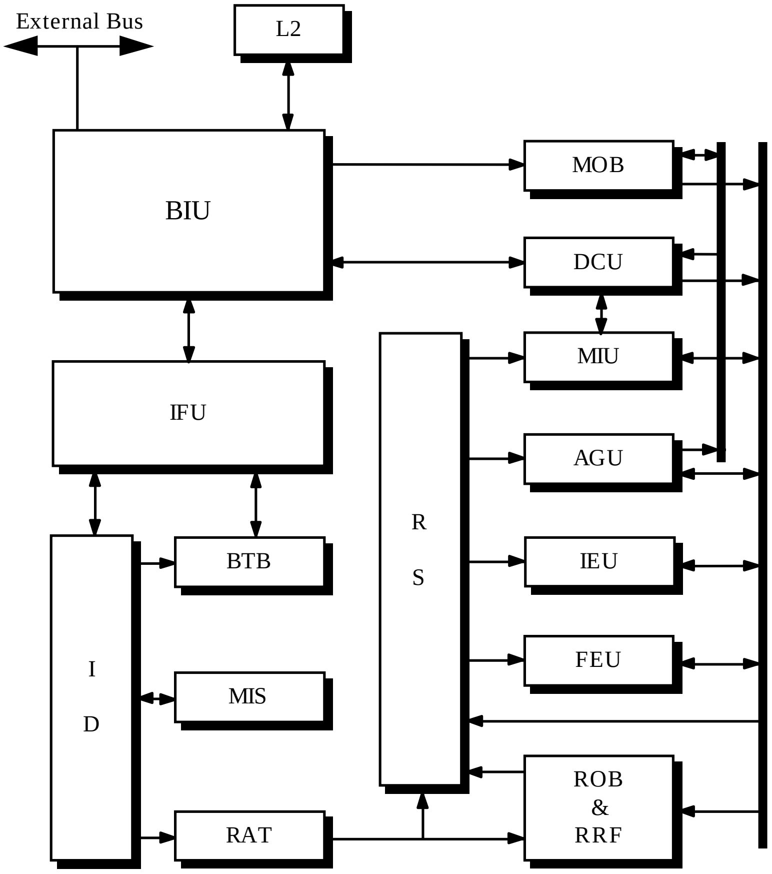
Acknowledgements

We are fortunate to represent the work of many talented, dedicated professionals and it is mainly their efforts that are presented here.

References

[1] Schutz, J., “A 3.3V 0.6um BiCMOS SuperScalar Microprocessor”, ISSCC Proceedings, pp. 202-203, 1994

[2] Hennessy, J. et al, Computer Architecture A Quantitative Approach, Morgan Kaufman Publishers Inc., 1990



- BIU: Bus Interface Unit
- IFU: Instruction Fetch Unit (includes ICache)
- BTB: Branch Target Buffer
- ID: Instruction Decoder
- MIS: Microinstruction Sequencer
- RAT: Register Alias Table
- ROB: ReOrder Buffer
- RRF: Retirement Register File
- RS: Reservation Station
- IEU: Integer Execution Unit
- FEU: Floating point Execution Unit
- AGU: Address Generation Unit
- MIU: Memory Interface Unit
- DCU: Data Cache Unit (includes DCache)
- MOB: Memory ReOrder Buffer
- L2: Level 2 Cache

Figure 1 - Basic CPU Block Diagram

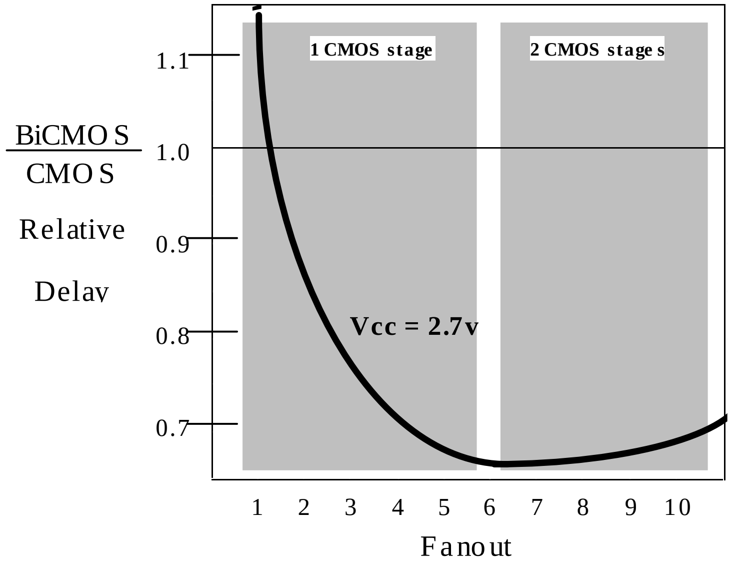


Figure 2 - BiCMOS vs CMOS relative inverter delay

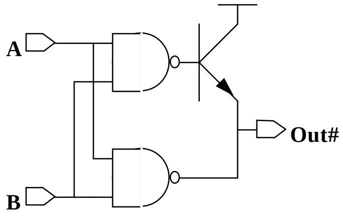


Figure 3 - BiCMOS Gate

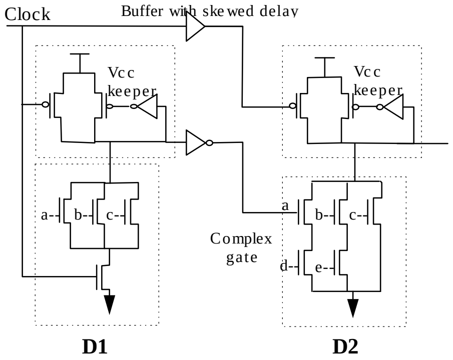


Figure 4 - Delayed Precharge Domino