20.1 Design of an 8-wide Superscalar RISC Microprocessor with Simultaneous Multithreading

Ronald P. Preston, Roy W. Badeau, Daniel W. Bailey, Shane L. Bell, Larry L. Biro, William J. Bowhill, Daniel E. Dever, Stephen Felix, Richard Gammack, Valeria Germini, Michael K. Gowan, Paul Gronowski, Daniel B. Jackson, Shekhar Mehta, Shannon V. Morton, Jeffrey D. Pickholtz, Matthew H. Reilly, Michael J. Smith

Compaq Computer Corporation, Shrewsbury, MA

A proposed 4-way multithreaded 8-way superscalar microprocessor implements the Alpha instruction set in a 0.125µm partially depleted SOI technology (pdSOI). Improvements over the previous design [1] include doubling instruction issue width to 8, increasing the L2 Cache from 1.75MB to 3MB, and migrating from a bulk 0.175µm CMOS process. The combination of architectural enhancements and advanced process technology produce-single-threaded performance simulated at over 2.5x that of the previous design [2]. 4-way simultaneous multithreading (SMT) further increases utilization of the 8 integer and 4 floating point execution units by allowing concurrent execution of up to 4 independent program threads. Figure 20.1.1 is a block diagram for this processor. Die size is estimated at 420mm^2 as indicated in Figure 20.1.2.

To support 8-wide superscalar issue, the instruction fetch logic must assemble a group of 8 valid instructions each cycle. This block is created by fetching 16 instructions from two separate instruction cache blocks, each 8 instructions wide, and then collapsing over instructions not on the predicted path. The cache indices for the fetches are determined by reading a line prediction array and need not be sequential if the predictor indicates that a flow change occurs in the first fetch block. The 64kB instruction cache is organized into 8 independent banks to support the two fetches per cycle. The two fetched instruction blocks are passed into a collapsing buffer network organized with instructions interleaved on a bit-by-bit basis. In the collapser a series of one-high dynamic pull-down multiplexers removes instructions not on the predicted path (Figure 20.1.3). The collapser first left-justifies each block so the first valid instruction occupies position 0. The 2nd block is then right shifted to align with the end of the 1st block. Finally the blocks are merged with the aligned 2nd block overwriting invalid instructions from the 1st block. The start and length pointers for the collapser are one-hot controls from a multilevel branch predictor that tracks a unique prediction for each instruction in the window and resolves two predicted taken branches (one per 8 instruction block), allowing collapsed fetch blocks to span the end of a basic block.

The collapsed instruction blocks pass through a register-renaming stage, are aligned to a set of functional units, and enter a 128-entry out-of-order issue queue that selects up to 8 issue-ready instructions for dispatch. While issue occurs out-of-order, performance is improved by selecting the oldest data-ready instructions. The previous CPU design uses a FIFO structure to maintain the original program order, shifting unissued instructions into vacant FIFO positions [3]. In this CPU, an age mask vector is used to avoid critical paths from the FIFO employed on the predecessor (Figure 20.1.4). As a new instruction enters the queue, it is assigned an empty position in one half of the 128-entry array. It is also given a 64b age vector with all bits but its own entry number set to 1. Subsequent instructions entering the queue clear the bit in this mask corresponding to their entry position. When the instruction becomes data-ready, it bids for issue on its assigned pipeline by asserting a bit in the BID vector. A bit-wise logical AND of the age mask and the entire 64b BID vector is performed, if the result is zero, then the instruction is the earliest ready instruction in program order in this half of the issue queue, has won the bid, and is issued to the appropriate functional unit.

SMT is implemented by adding circuitry to maintain the 4 independent program counters and select a single thread each cycle for fetch from the instruction cache. As instructions are fetched, they are tagged with a thread-ID that follows the instructions through the instruction, execution, and memory units. Significant performance gains of up to 2x in instruction throughput are possible as the out-of-order issue queue is free to select issue-ready instructions from any of the 4 simultaneous threads [4]. In contrast to coarse grain multithreading, instructions from more than one thread may be active in the integer and floating-point pipelines simultaneously [5]. The added logical complexity for SMT given the existing out-of-order issue policy and register renaming is modest, accounting for only about 6% additional die area. The most significant change required by SMT is the inclusion of 32 integer and 32 floating-point registers for each of the 4 threads. The architectural state of the machine therefore requires 256 registers. With additional registers required to support register renaming by covering up to 256 in-flight instructions, a total of 512 registers are implemented.

The register file logically requires 16 read ports and 8 write ports to support 8-wide issue. Since a 24-port cell is difficult to implement, a 2 bank design of 1024 total registers with 8-port cells is implemented (Figure 20.1.5). The cells share ports for read and write that can occur in back-to-back phases. Integrated precharge and write amps support the back-to-back read/write operations as shown in Figure 20.1.6. Due to the large register file and the transit delays to route the operands to all of the execution units, the register file stage of the pipeline is extended to 3 cycles. Smaller register caches added to the integer and floating-point execution units reduce the performance penalty associated with this additional latency. The register caches store copies of the last 8 cycles of generated results, act as local bypass result multiplexers, and align result write-back to the main register file avoiding contention for the 8 write ports due to instruction latency.

Using pdSOI devices introduces a number of issues such as SOI pass-gate effect [6]. A number of techniques are implemented to avoid the pass-gate effect including replacing susceptible dynamic circuits with static counterparts, and selected use of body-contacted devices. One example is shown in Figure 20.1.7 illustrating a segment of the core of the dual-rail 64b integer shifter where the bodies of the differential pass gates are tied together with the combination left floating. In the original design with floating bodies, significant pass-gate effects occur when the inputs transition from high to low. Capacitive coupling from the falling source to the body and then from the body to the drain causes a significant charge loss on the output. With body ties, the common body node does not rise beyond about a diode drop above ground and the complementary input signals help balance the coupling to the shared body node.

This design, the EV8 microprocessor, was cancelled prior to release for manufacturing.

Acknowledgments:
The authors acknowledge members of the Alpha Development Group at Compaq, past and present who contributed to this microprocessor design.

References:
[1] Jain, A. et al., "A 1.2GHz Alpha Microprocessor with 44.8GB/s. Chip Pin Bandwidth", ISSCC Digest of Technical Papers, pp.240-241, Feb. 2001.
[2] Bannon, P. "Compaq's EV7: Performance Features and Real World Measurements", Microprocessor Forum, October 2001.
[3] Fischer, T., Leibholtz, D., "Design Tradeoffs in Stall-Control Circuits for 600MHz Instruction Queues", ISSCC Digest of Technical Papers, pp. 232-233, Feb. 1998.
[4] Tullsen, D. et al., "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor", 23rd Annual International Symposium on Computer Architecture, pp. 191-202, June, 1996.
[5] Storino, S. et al., "A Commerical Multithreaded RISC Processor", ISSCC Digest of Technical Papers, pp. 234-235, Feb 1998.
[6] Bernstein, K., Rohrer, N., SOI Circuit Design Concepts, pp. 84-92, 96-112, Kluwer Academic Publishers 2000.

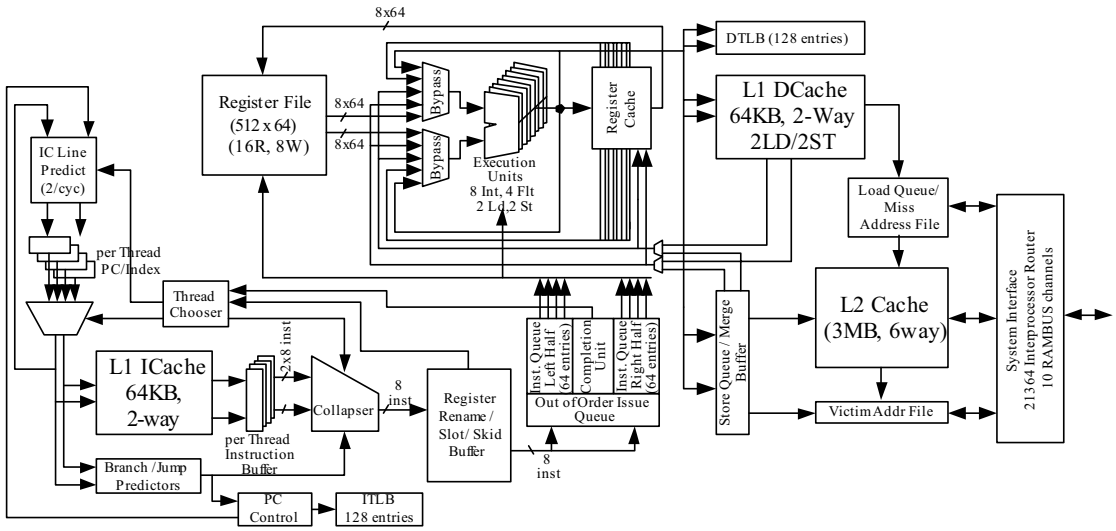

Figure 20.1.1: CPU block diagram.

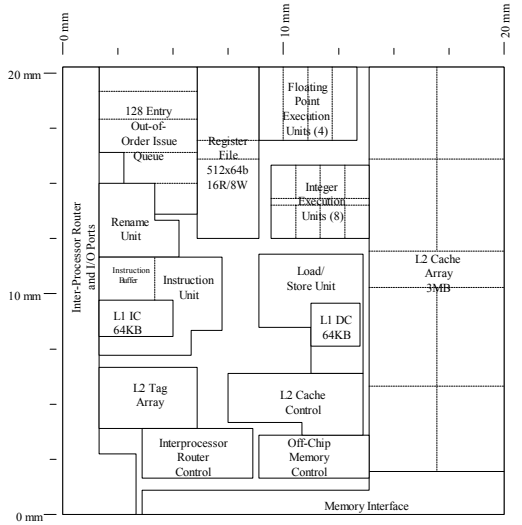

Figure 20.1.2: High-level floorplan.

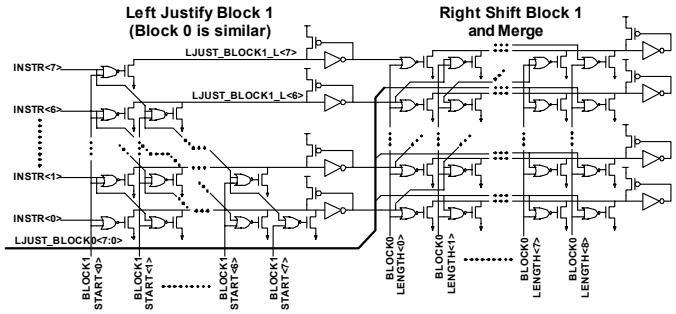

Figure 20.1.3: Collapsing buffer network.

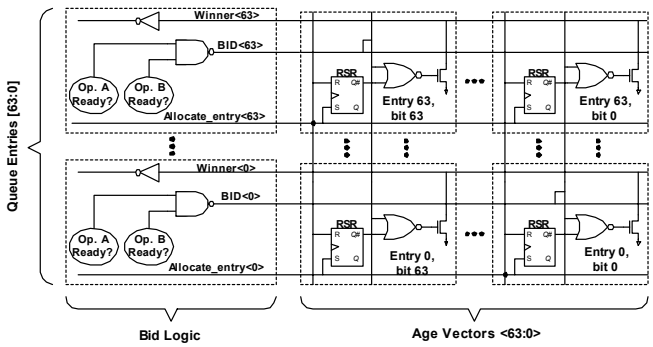

Figure 20.1.4: Instruction queue age mask.

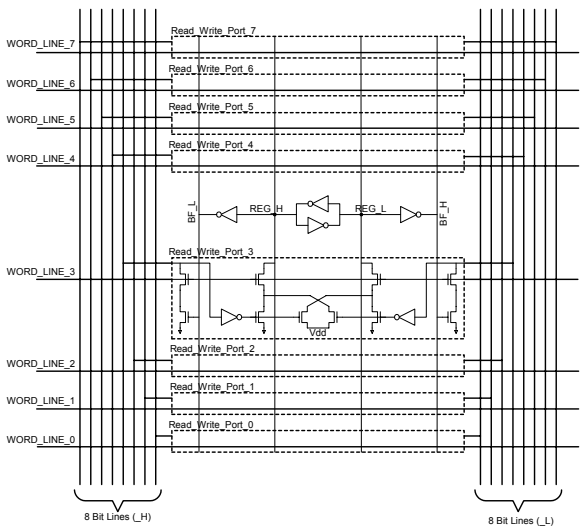

Figure 20.1.5: 8-port register file cell.

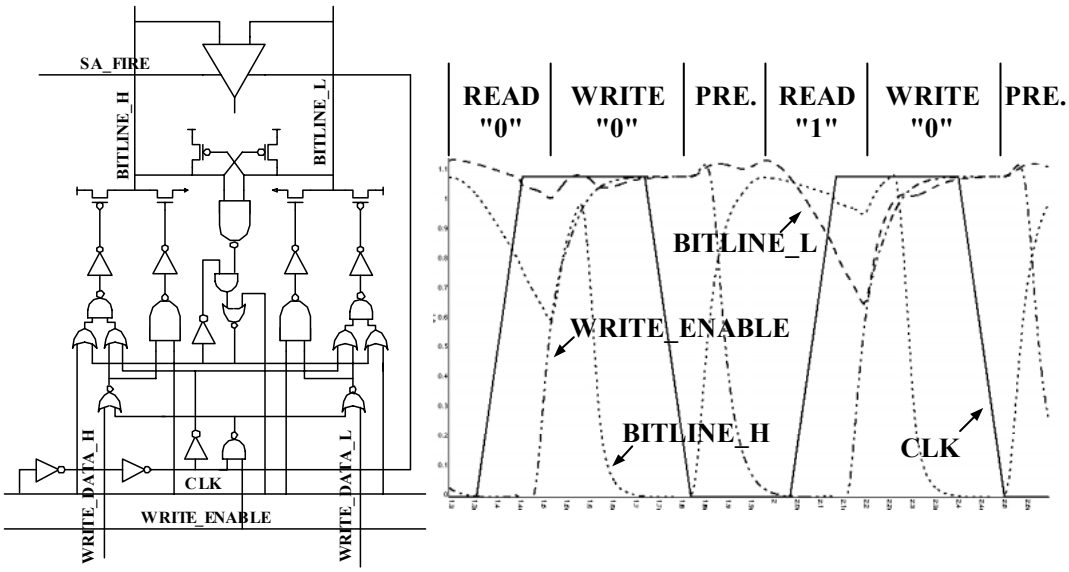

Figure 20.1.6: Register file write amp/precharge.

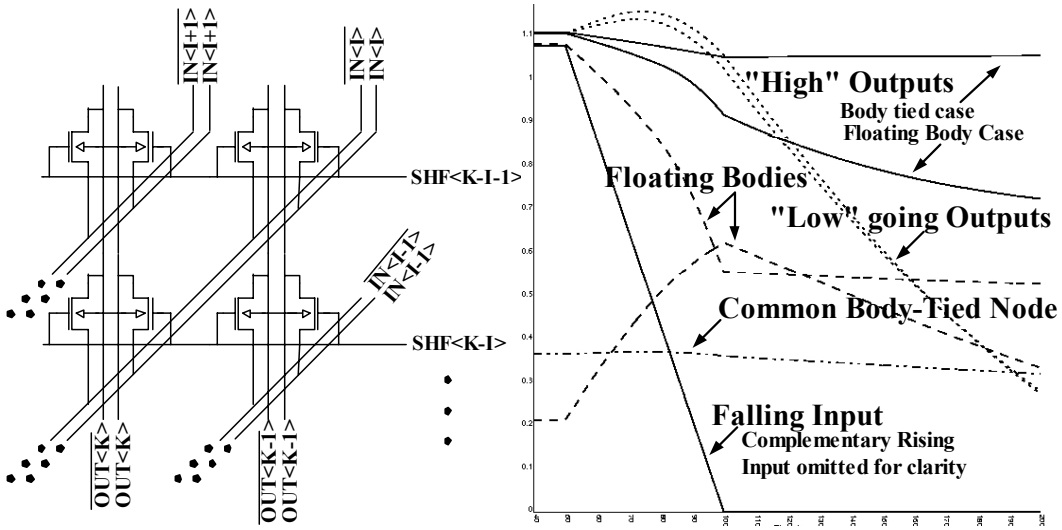

Figure 20.1.7: 64b integer shifter with body-ties.

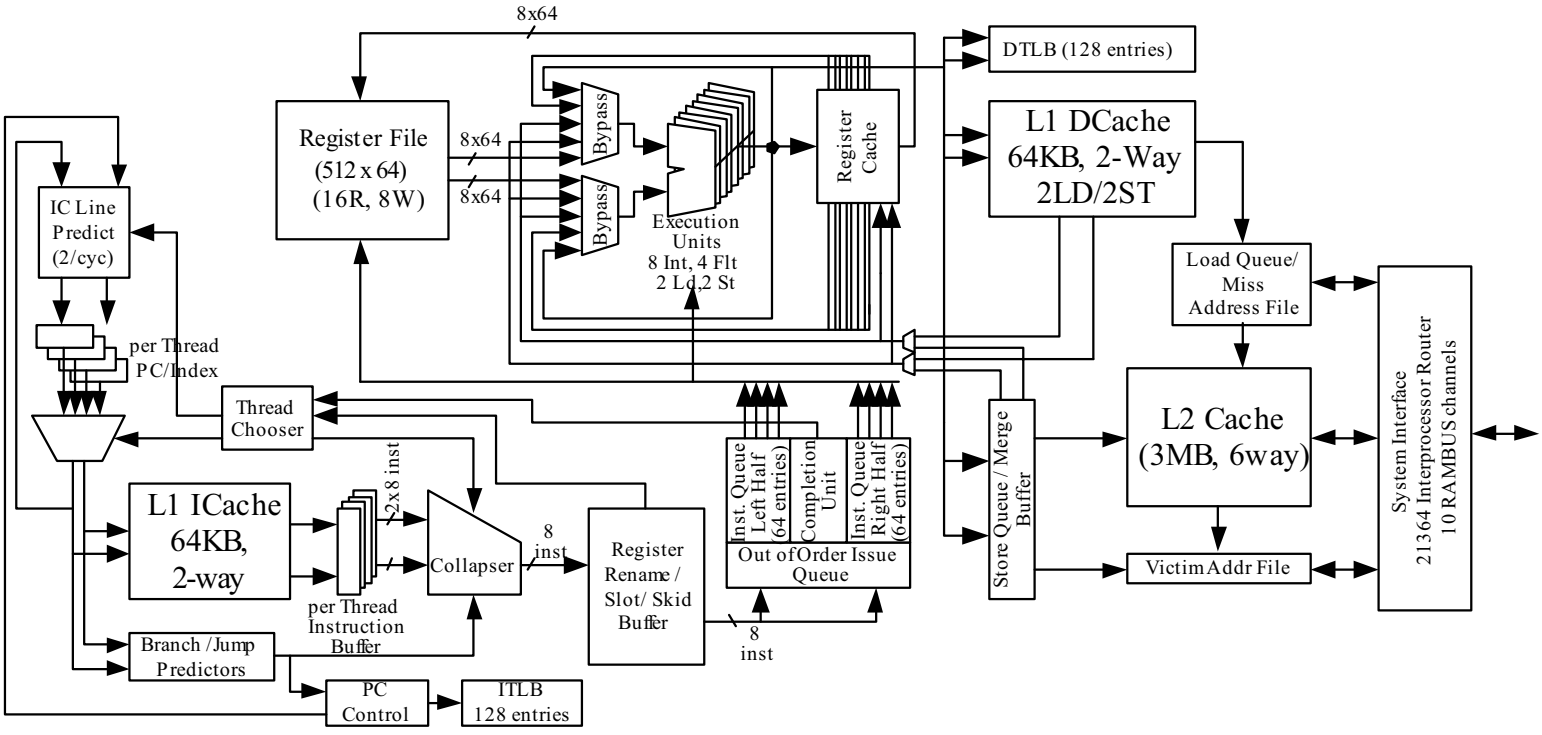

Figure 20.1.1: CPU block diagram.

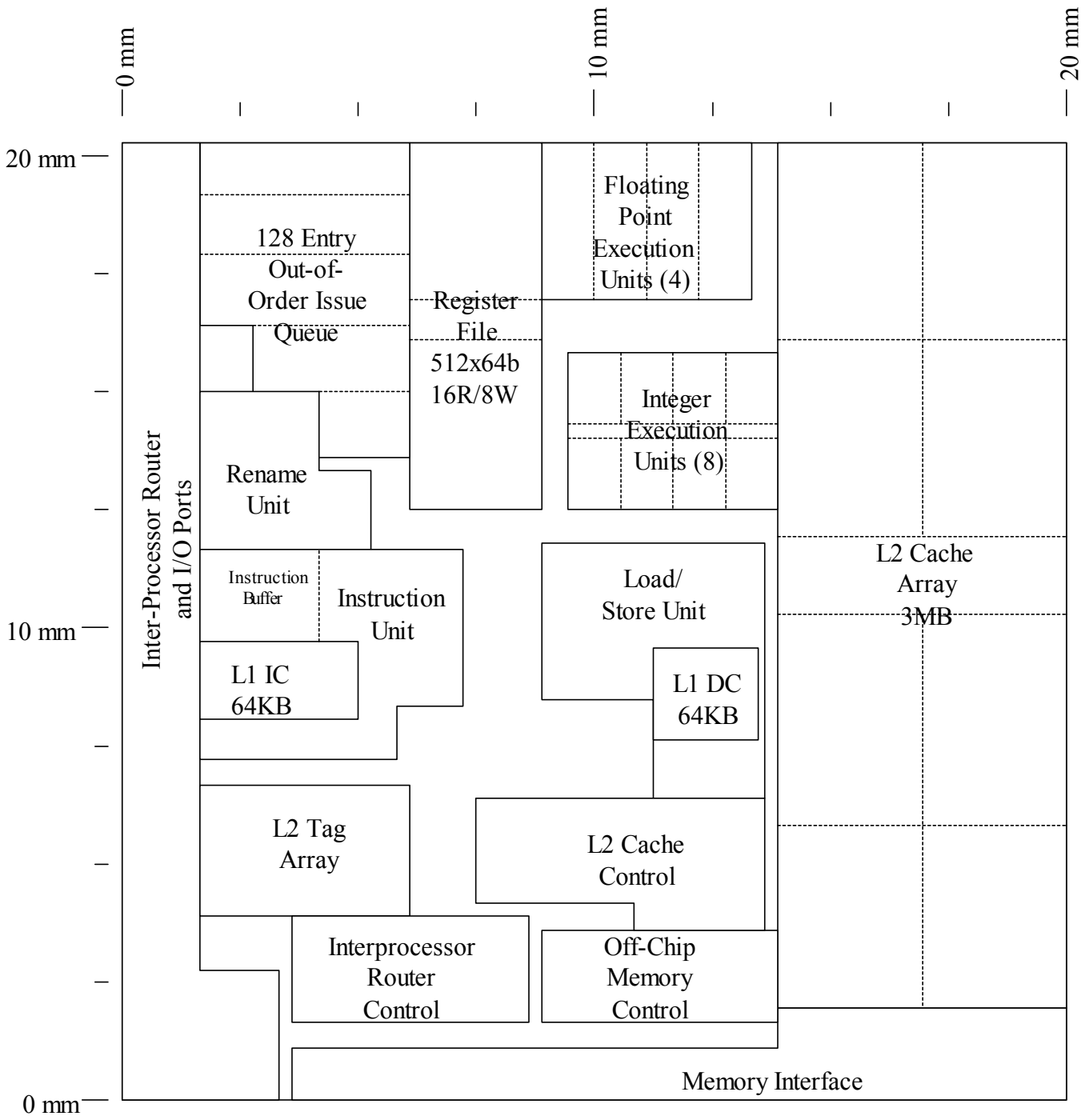

Figure 20.1.2: High-level floorplan.

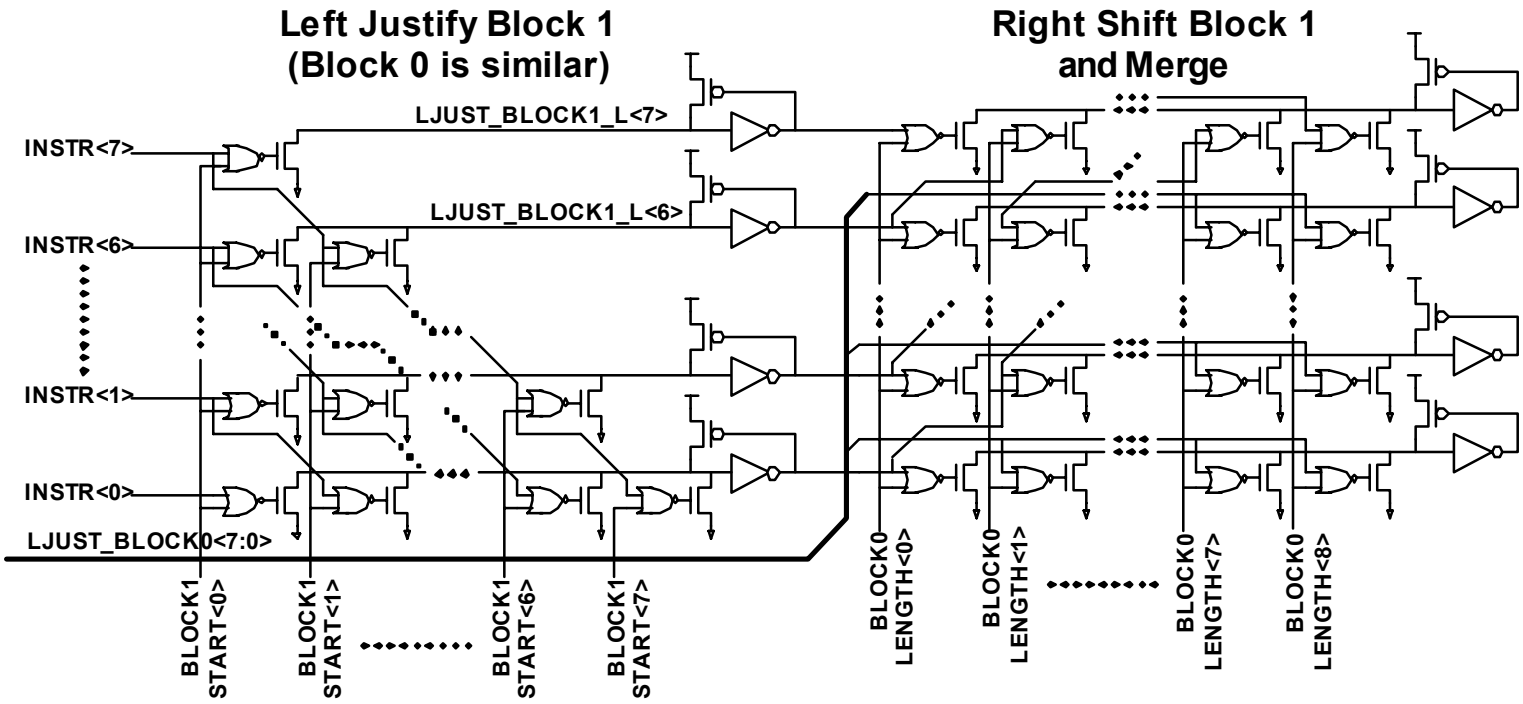

Figure 20.1.3: Collapsing buffer network.

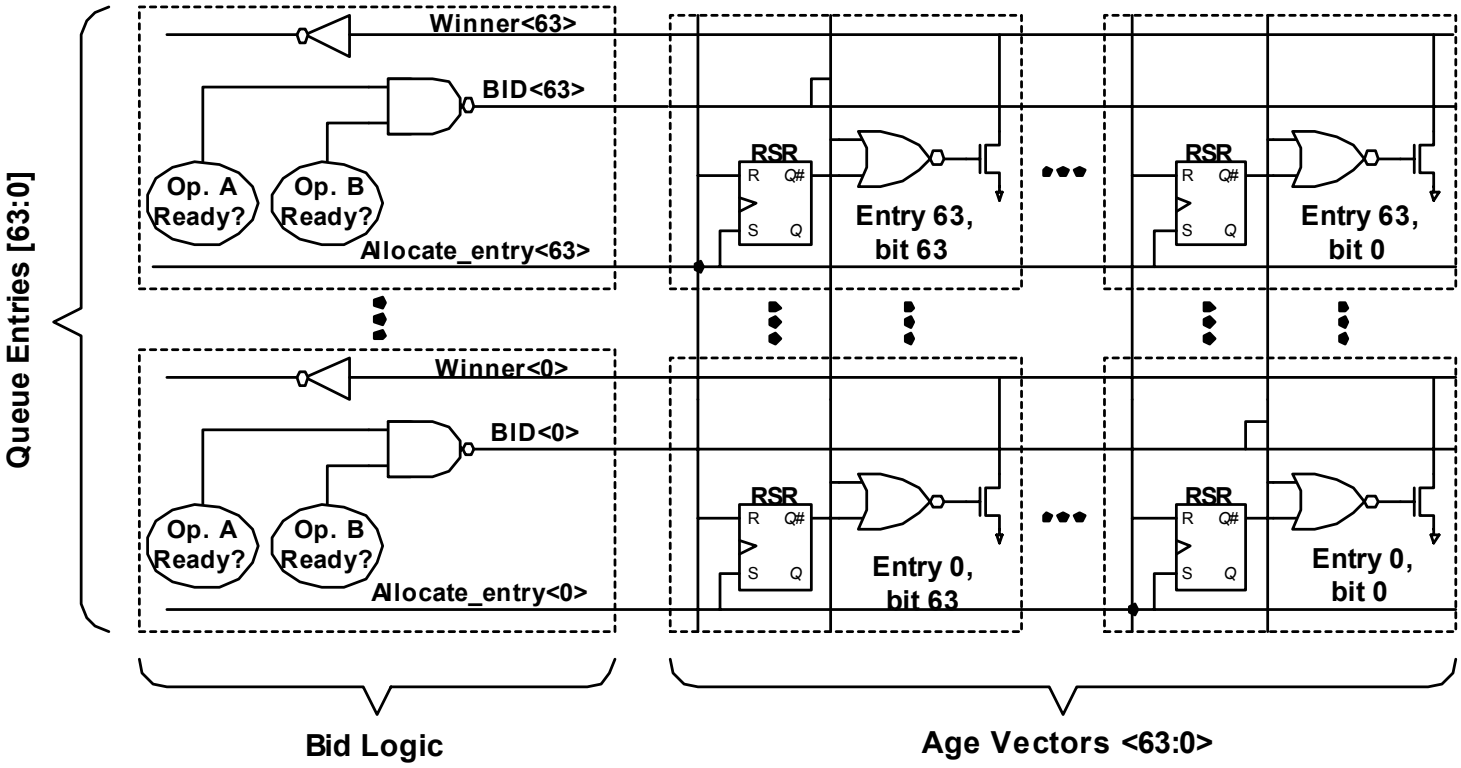

Figure 20.1.4: Instruction queue age mask.

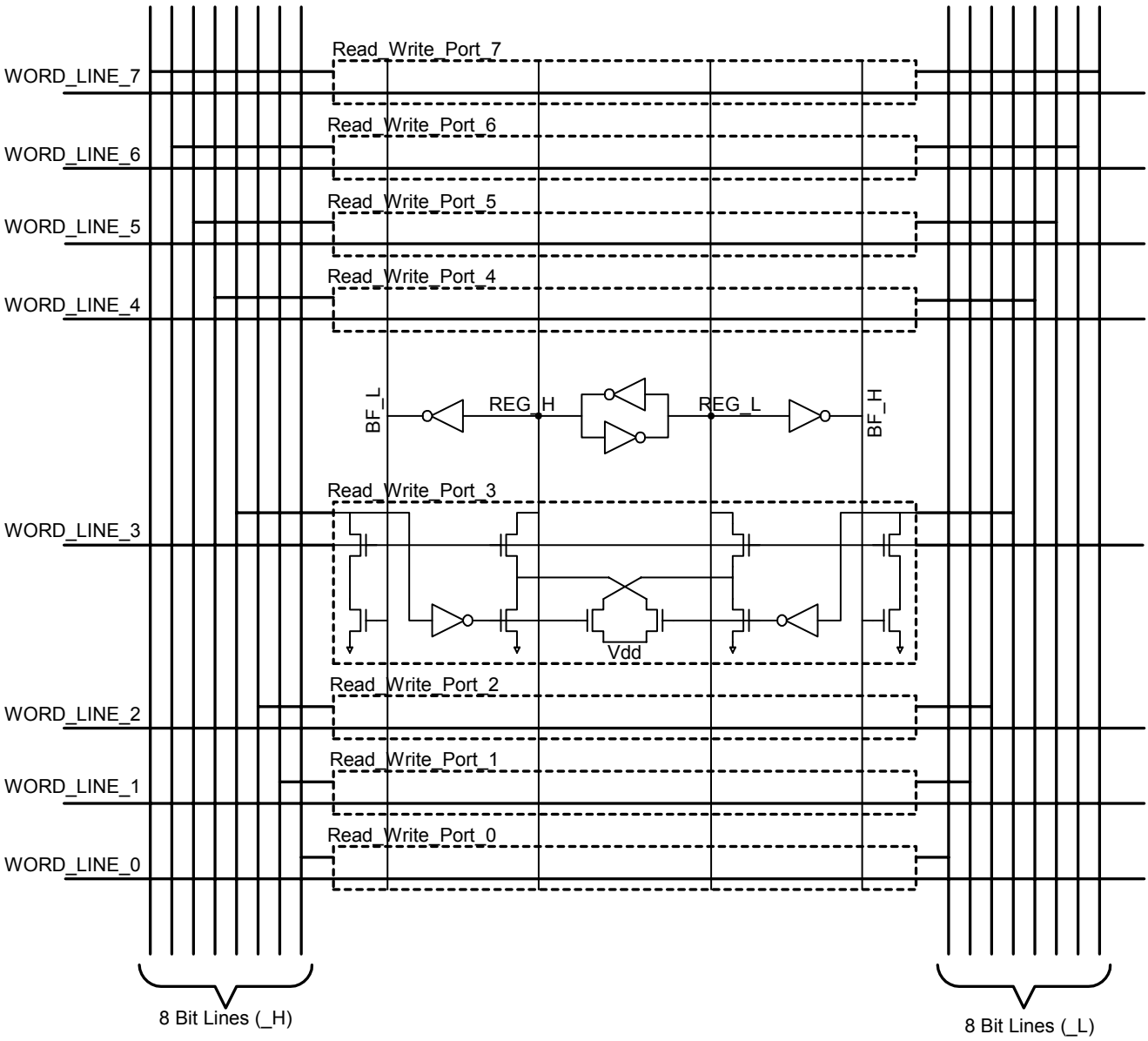

Figure 20.1.5: 8-port register file cell.

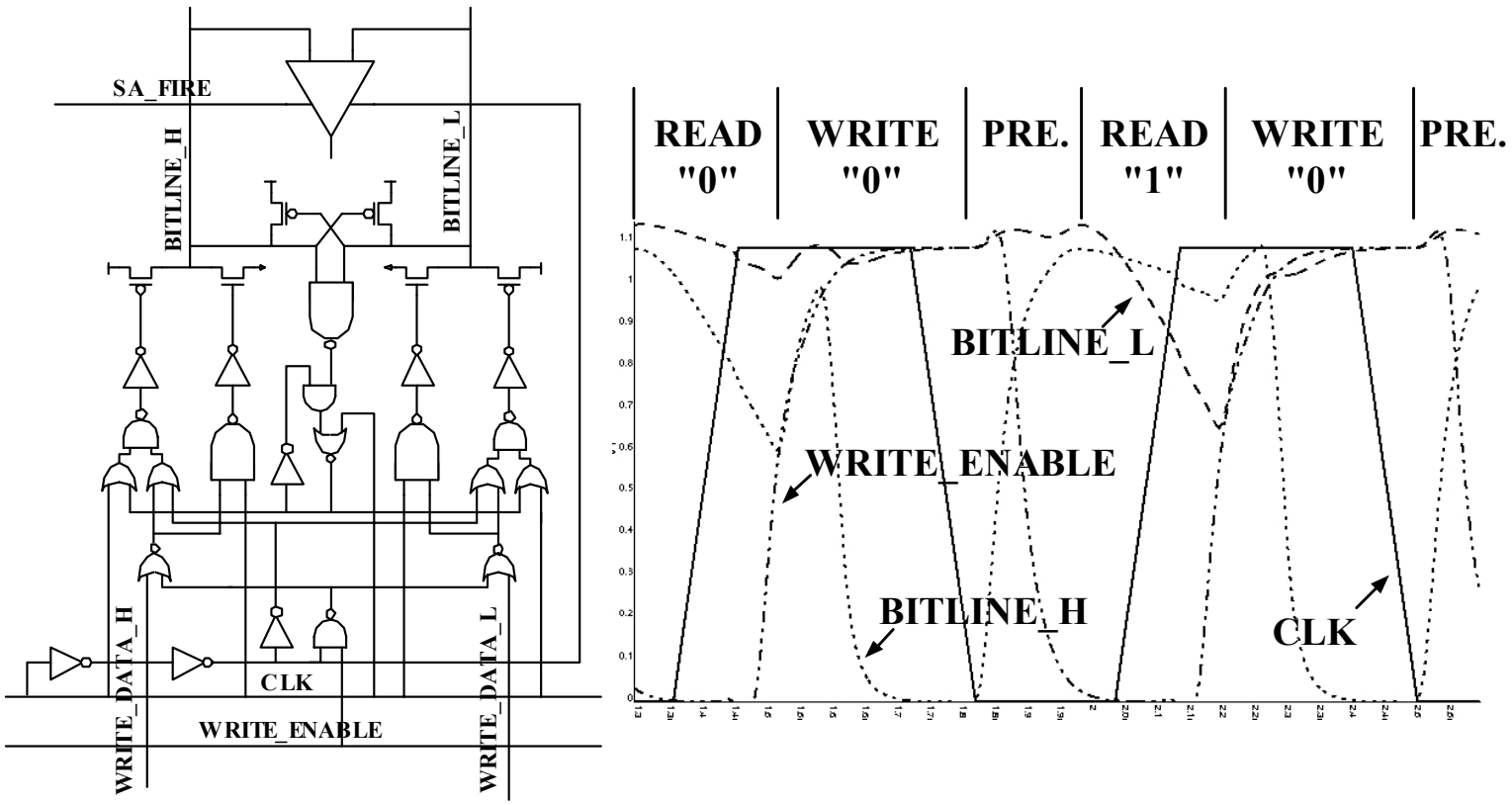

Figure 20.1.6: Register file write amp/precharge.

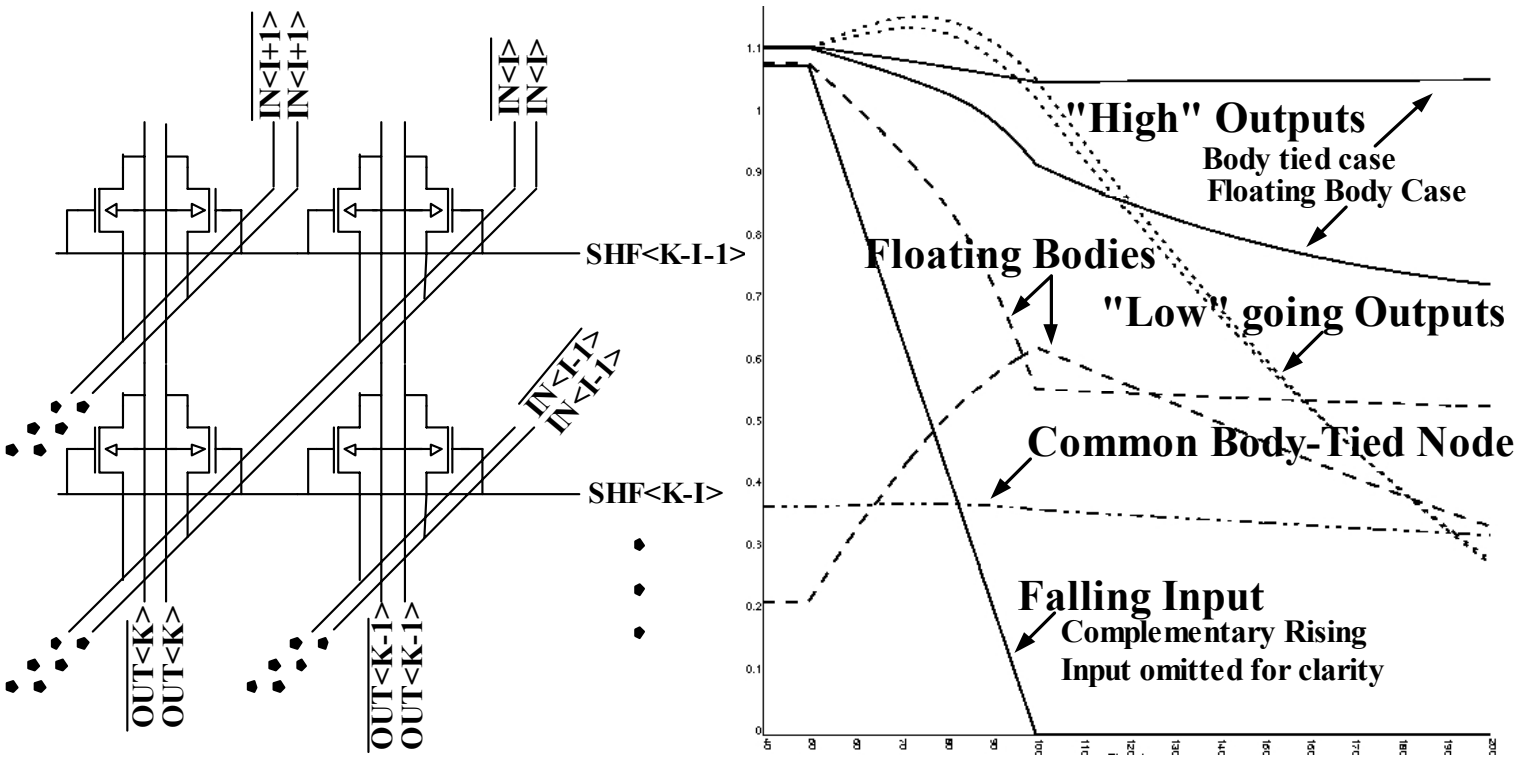

Figure 20.1.7: 64b integer shifter with body-ties.