Power and CAD considerations for the 1.75Mbyte, 1.2Ghz L2 cache on the Alpha 21364 CPU

Joel Grodstein, Rachid Rayess*, Tad Truex, Linda Shattuck, Sue Lowell*, Dan Bailey*, David Bertucci, Gabriel Bischoff*, Daniel Dever, Mike Gowan*, Roy Lane, Brian Lilly, Krishna Nagalla*, Rahul Shah*, Emily Shriver*, Shi-Huang Yin, Shannon Morton

Compaq Computer
334 South Street
Shrewsbury, MA 01545

*Intel Corporation
334 South Street
Shrewsbury, MA 01545

ABSTRACT

A 1.75 MByte L2 cache has been designed and fabricated as part of the Alpha 21364 microprocessor[1] (Figure 1), in a .18μ bulk CMOS process. The cache was designed to run at 1.2 GHz, and pass-1 samples confirm this. While Alpha CPUs are known primarily for high speed, the combination of package constraints and a tight schedule forced careful attention to the integrated whole of power expenditure and the interaction of CAD with design. The cache consumes only 7% of total die power.

Categories and Subject Descriptors

B.3.1 [**Memory Structures**]: Semiconductor Memories – *SRAM*
B.2.2 [**Memory Structures**]: Design Styles – *cache memory*
B.7.1 [**Integrated Circuits**]: Types and Design Styles – *Microprocessors*.

General Terms

Design, Performance, Verification.

Keywords

Cache memory, CPU, low-power, timing verification, logic verification.

1. INTRODUCTION

Alpha CPUs have long been designed at the bleeding edge of performance. Their position on the power-delay curve could be described by "as much performance as possible without melting the system." Nevertheless, the relentless physics of power affects even this type of design, especially in the caches. The Alpha 21364 CPU is built from a 21264 out-of-order CPU core[2]. We then added a 1.75Mbyte, level-2, combined I-D-cache and a system interface (memory controller and router) around the core (Figure 1). To keep to a tight schedule, the 21264 core was modified as little as possible.

The 21364's power budget can be broken down as follows:

- L2-cache clocking power: 3.9 watts
- L2-cache RAM-array banks: 0.8 watts
- L2-cache data distribution between the banks and the CPU core: 2.9 watts
- L2-cache tags: 3.5 watts
- Other (core, system interface, I/O): 138 watts
- Total chip power: 150 watts.

It is immediately obvious that engineering the L2-cache power consumption offers little opportunity for glory – reducing its power to zero would only reduce chip power by 7%. However, the opportunity for a spectacular failure is large. Simply put, the L2 cache covers 37% of the Alpha 21364 chip area and contains 85% of the total devices; without careful attention to power, it may overwhelm the chip's power budget. By contrast, the StrongARM 110, a low-power RISC CPU, spent up to 43% of total chip power in the caches[3].

Chip power is rarely reduced without a corresponding tradeoff in chip speed, signal integrity, or schedule. Our goal in this paper is to show some of the design tradeoffs made in this cache, and highlight some of its innovative features. We further hope to show how, by carefully integrating CAD considerations into the overall design, the design tradeoffs were made without undue schedule impact or project risk.

2. CACHE OVERVIEW

The tags contain 28 banks of 128 rows by 248 columns. It contains a total of 5.4M RAM devices and 350K non-RAM devices.

The data arrays are far larger, and the rest of this paper will focus on them. They contain 108M transistors. This includes 6.9M non-RAM transistors – as many non-RAM transistors as the *entire* 21264 CPU[2]. The cache data arrays consist of 224 banks, organized into 28 sections of 4 bank pairs each (Figure 2). Each section contains ¼ of one of the 7 ways. That is, each way spans four sections, one in each corner. Tags are kept separately, in a central location.

Each cache access requires one tag lookup; it then returns 128 data bits in four fully-pipelined cycles. The tag lookup is *decoupled* from the data access; probes in the multiprocessor system require frequent tag lookups with no concomitant data access.

Each bank of the cache contains 128 rows and 512 columns. An additional two spare rows and 8 spare columns provide redundancy for improved yield. Banks are layed out in *mirrored pairs* – pairs of banks mirrored about their X axis, and sharing a common set of tristate super-bitline drivers in the center. We use dynamic sense amps similar to [5,p.303].

We use M1 for local routing, M2 for horizontal bit lines, and M3 for vertical word lines alternated with shields. Since the L2 cache is at the periphery of the chip, the only global routing above the cache is its own global bit lines, address, and control. This enables us to use free upper metal layers to enhance electrical robustness.

M4 and M5 are both horizontal(!). M4 is strictly V_{dd}/V_{ss}, while M5 contains the main clock spines and global signals. Running these two metals in the same direction allows critical M5 lines to be shielded on both sides *and* below, creating a virtual coaxial cable. It also isolates the critical lower-level bit and word lines from global M5 routes. M6 is used for more V_{dd}/V_{ss} and a small number of vertical clock bus-bars, and M7 for bump routing. The overall dense upper-level V_{ss}/V_{dd} metallization gives us an extremely low-R and low-L path from V_{ss}/V_{dd} bumps down to the lower metals.

3. CACHE POWER

The very first tradeoff we made was to give away performance in an area where it would not be missed. This L2 cache was being added to the existing 21264 CPU core. The core had originally been designed to interface with an off-chip L2 cache made of fast commercial SRAM. Bringing both the L2-cache tags and data on chip allowed us to significantly reduce their latency. However, beyond a certain point, the core was not ready to accept data any sooner. Since the core could not take advantage of further L2 latency reduction, we reduced power instead.

We were able to meet this minimum latency while still serializing tag and data access. That is, we first do a tag access to determine which (if any) of the seven cache ways holds the current address. Only then do we activate the relevant 1/7 of the data arrays. This enabled a drastic power reduction vs. looking up data in each way and then muxing to pick the relevant one.

When combined with using 3 more address bits to restrict ourselves to enabling 1/8 of the banks, this means that only 4 of the 228 banks are ever accessed simultaneously. Aggressive conditional clocking of inactive word and bit lines thus greatly reduces the total power in the banks. With this done, only 11% of the total cache power is spent in the RAM arrays and surrounding circuits. Thus, techniques such as subbanking or bit-line segmentation[4] would not be highly effective in our situation

However, 38% of the cache power is spent on the long wires distributing data between the cache banks and the CPU core. Given the size of the cache, this is not surprising. It could be reduced by low-voltage signaling techniques (e.g.,[5,pp.371-5]). Unfortunately, sending data over these wires represents the most critical timing path in the cache, and reducing the power spent here would thus directly impact our cycle time. Instead, we focused on clocking power.

3.1 Clocking.

Previous Alpha chips have used a dense global metal grid driven by an HTree or RC tree[1,2,3]. The main section of the 21364 is in fact clocked this way[6]. However, for power reasons, the cache has its own clock distribution network. Each of the East and West halves of the L2 has a separate distribution network; they are synchronized to the main clock by DLLs.

A dense metal grid as used on most Alpha designs provides low skew across wide loading variations. However, this has also represented in clock power of up to 65% of the total chip power[3]. Clearly, this would have resulted in an unacceptable amount of cache power.

Fortunately, due to the replication of the basic cache bank, our clock usage is extremely regular, rendering the dense grid less necessary. We thus investigated the use of a buffered HTree. Since the cache area is large, significant spreads of temperature, voltage, and intra-die variation can occur. Thus, a pure HTree would not provide tight skew; different buffers in the HTree could have different delays. We also investigated the use of an RC tree[5,p.270]. Since this relies on matching diffusion capacitance against metal capacitance, it cannot give low skew across process corners without a dense clock grid, and would have worse skew than an HTree

We chose two identical global HTrees, one for each of the East and West halves of the cache (with one endpoint for each of the 14 sections) driving an extremely sparse grid. The grid consists of a single horizontal M5 clock spine in each of the 28 sections, running the entire width of the section. These are tied together with a total of 4 vertical bus-bars (two each in the East and West). Each of the bus-bars runs nearly the entire height of the chip.

RLC simulation shows that insertion of the bus-bars reduces total clock skew by 30% vs. a pure HTree. The sparse grid requires minimal metal usage. A denser grid would have reduced the effect of process/V_{dd}/temperature variations on clock skew, at the cost of extra clocking power.

Based on their size, these bus-bars constitute 39% of the total clock-wire capacitance. This may seem like a large amount of clocking power. However, the total metal of the clock grid (including the bus-bars) is only 17% of the total gate loading attached to it. Thus, the bus-bars are only 7% of our total clock capacitance.

3.2 Clock Shielding

As mentioned, our M5 clock spines are shielded on both sides as well as below. This was done chiefly for signal-integrity reasons – with this configuration, and assuming that 75% of the metal layer above is populated with V_{dd}/V_{ss}, only 6% of the wire capacitance is susceptible to crosstalk. Including the gate loading implies that less than 1% of the total clock load is from coupling. Since coupling cap is data-dependent loading, the shielding reduces skew. By eliminating the possibility of crosstalk aggressors switching in the opposite direction as the clock, it also reduces clocking power.

The shielding also decreases the clock-spine inductance. At high frequencies, shields running parallel to a wire are much better current returns than shields perpendicular to a wire. Worst-case RLC ringing is only 2% of V_{dd}, even though the clock grid is strongly driven for a fast edge rate.

3.3 Synchronous Design

The use of self-timed design is common in caches[e.g.,3]. For example, the strobe which fires the sense amplifiers may be derived from the discharge of a dummy bit line, or may occur a fixed delay interval after (e.g.,) firing of a word line. For obvious reasons, the use of a self-timed design reduces clocking power. However, it also introduces significant design risk –firing sense amplifiers before enough bit-line differential has developed will result in non-functional operation. For this reason, critical delays are often made programmable.

Instead, we used a fully-synchronous design. Bit-line differential is generated during phase 1, and then dynamic sense amps [5,p.303] are strobed at the start of phase 2. This design style is nearly risk free. Lowering the clock frequency will increase the bit-line differential as needed. For this reason, a synchronous design was deemed mandatory.

However, the penalty in clock power was severe. Our most critical timing path in the cache starts from firing the sense amplifiers, and continues through repair logic and driving the very long wires back to the CPU core. Thus, in our synchronous design, we had to fire the sense amplifiers as early as possible in the clock cycle. Since the sense-amp clock is a gated clock, the minimum delay is through a single gate.

A standard means of generating a conditional clock with a single gate delay is CCLK = NOR2 (CLK,EN_L). CCLK will then pulse high in phase 2 if ENA_L is active. However, due to their large P devices, highly-loaded NOR gates are both power and area inefficient. Instead (Figure 3), we conditioned the sense-amp strobe using CCLK=NAND2(CLK,ENA_H). CCLK is thus always high in phase 2. In those cycles where the strobe is due to fire in phase 2, it goes low in phase 1, thus ensuring a rising edge will occur. For a constant output loading, the use of the NAND gate instead of the NOR reduces both clock loading and gate area by 20%.

We further reduced area by downsizing the data P device. Since small circuits consume less power, this also reduces overall power. With ENA_H set up before the start of phase 1, this P device is used only as a keeper, and can be downsized as much as leakage concerns allow.

Finally, we added a small internal precharge device inside the NAND to equalize the loading the NAND places on the main clock line, independent of the value of ENA_H. This improves skew, at a negligible power cost.

Though the decision to use a synchronous design improved our schedule and reduced risk, it had a significant impact on clocking power. This single NAND gate represented 52% of our clock gate load. In perspective, however, this represents less than 2% of the chip power.

3.4 Conditional Clocking

The prime candidate to decrease clocking power is usually conditionalizing the clock[5 p.268,7]. In any cache, clocklike signals such as word lines are only clocked when necessary. We also investigated shutting off the main clock grid. This would be difficult:

- The chip goes into high-memory-bandwidth servers for databases and scientific computing. This class of machine can use the L2 cache almost constantly.
- The clock for the L2 cache is driven by a delay-locked loop[6]; the L2-cache clock must tick constantly to drive the DLL's phase detector.

It is thus difficult to conceive of turning off the entire global clock grid. Another option would be to break the grid into multiple smaller grids – one unconditional driving the DLL and a small amount of clock-control logic in each bank, and a separate smaller grid for each of the seven ways. We could calculate our power savings as:

- Based on cache layout, the unconditional part of the grid would comprise ¼ of the total grid area.
- Naively, only one of the other seven grids is ever powered on at the same time. In fact, due to overlap in the pipes, we typically have two ways with simultaneous transactions in their pipe at any time.
- Power savings=1-[¼*1 + ¾*2/7]=54%.

However, the L2-cache clock power represents only 3% of the total chip power. Thus, conditional clocking would save us only 1.5% of the total chip power.

On the other hand, conditionally clocking the main power grid would increase the clock skew [5,p.265](as conditional clocking almost always does). Furthermore, it causes severe disruptions to the verification flow, as detailed next. On the whole, then, it did not seem worth the risk.

4. CAD OVERVIEW

Understanding up front the limitations of our CAD tools in dealing with a design of this size helped us avoid design choices that would have hurt our schedule. In other cases, being able to enhance our tools allowed us to reduce our design margin.

4.1 Logic Verification

Verifying the correct logical operation of a cache this size presents obvious difficulties. Simulation in a switch-level simulator such as Cosmos[8] is not possible in any computer available to us, due to virtual-memory limitations. We could, of course, remove all of the RAM cells and then simulate it with Cosmos. However, even this is quite slow – and with 8.4M non-RAM devices, exhaustive non-symbolic simulation is unthinkable[9].

Symbolic simulation (e.g., Symbolic Trajectory Evaluation[10]) would have been possible. STE has been successfully used to verify several memory arrays [11]. However, it is most useful when a functional block, though internally complex, can have its external behavior described by its response to a small number of symbolic stimuli. This is not the case for us. First, the internal structure of the cache is quite regular and not complex. Moreover, the cache is driven by a complex BIST engine. Our real problem is not verification of the cache by itself, but rather the combination of cache and BIST engine – and the BIST engine's behavior is not easily described by a small number of assertions. Finally, STE engines are not readily available, and the pool of verification engineers skilled at using them is correspondingly small; this would have hurt our schedule.

Instead, we verified the RTL with focused tests. We then did formal equivalence checking of RTL vs. schematics. This placed severe constraints on the RTL. First, it must be written at a low enough level that a formal-equivalency-checker tool can compare it to the schematics. On the other hand, it must be written at an abstract enough level to run very quickly in the RTL simulator – the BIST sequence is over 6 million cycles long.

Our RTL simulator uses levelized compiled code and phase partitioning. Rather than using bit-parallel simulation to simultaneously simulate different patterns, it packs 64-bit-wide busses into a single machine word for better single-simulation performance. Since the RAM datapath is largely wide busses, this would seem a natural fit. Unfortunately, the combination of swizzling the address and datapaths, 4-1 column muxes, and row/column repair made this very difficult without significant bit packing and unpacking.

Our resolution was to pack the RTL bits in whatever manner was necessary to make the RTL fast. This was quite different than the actual bit order in the schematics. However, the discrepancy was only visible in the RTL↔schematic mapping file used by the equivalency checker. Since the equivalency checker is run only rarely, and compares each bit of a bus separately in any case, this was not a problem.

With this done, the full-chip model ran only 15% slower with our detailed model than with a previous abstract cache model. We mentioned earlier that conditionalizing the main clock grid would have prevented this. The reason is that the cache needs clocks in both phases. Assume that the main clock grid is a conditional phase-1 clock. Inverting it to make a phase-2 clock would create a signal that is always high in phase 2, and conditional in phase 1. ANDing this with a condition to make a phase-2 conditional clock would create a signal that could be high or low in both phases. Though correct architectural use of these clocks ensures correct functionality, the use of "clocks" which may be high or low in either phase confuses CAD tools.

In particular, the phase partitioner used in the RTL simulator model fails to understand the clocking. The majority of the nodes actually switch only once per cycle; without proper phase-timing information, the code generator must schedule them for evaluation every phase. This nearly double the number of gate evaluations vs. more traditional clocking.

4.2 Race Verification

Our use of a very sparse clock grid, while saving power, causes more skew than on the rest of the die. If this skew caused a race, the chip would be non-functional at any speed. To minimize this risk, each of the 28 HTree endpoints was treated as its own clock domain. Essentially, we assumed arbitrary worst-case skew between endpoints, and designed around it with deskewing circuits (e.g., Figure 4). This guarantees functionality at any clock skew (though clock skew will still hurt the cycle time). Similarly, all paths between the cache and a clock grid on a different DLL were deskewed. Our race-checking tool[12] was extended to check that proper deskewing was done. In addition, it allowed us to break down the design and analyze it hierarchically. Long wires requiring RLC simulation were identified with heuristics similar to [13], and then manually verified. Understanding the clocking is fundamental to the race checker's operation. Thus, again, conditionalizing our main clock grid would have made this tool's operation quite difficult.

5. HINDSIGHT

Hindsight is always 20-20; we could have done many things better. The design of the data arrays, commonly thought of as the "entire" project, went smoothly. However, we greatly underestimated the layout complexity of the wiring between the data arrays and the core. The global timing of these wires proved similarly complex – they encompass every clock domain on the chip.

Similarly, we underestimated the contribution of this wiring to the global power budget. The write-data paths to the data arrays were not speed-critical. With more schedule time, we could greatly reduce the power they used; either by inserting "dummy" latches to mask unneeded transitions or by using low-swing signaling.

Hooking up our global power grid to the rest of the chip proved difficult. The power grids of the L2 cache and of the rest of the chip were both done in a very regular fashion – but independently of each other. This simple oversight required a time-consuming final stitching operation.

With hindsight, we have justified significant percentages of the L2-cache power as being insignificant to the overall chip power. Early in the design, these numbers were less clear. Instead, we made the decisions somewhat by gut feel and later reversed the ones which were wrong. For example, we originally did condition the main clock grid; but then reversed that decision when the CAD implications and the total chip power became clearer.

6. CONCLUSIONS

While "traditional" Alpha designs have been done with power as an afterthought, this is no longer possible. Though it covers 37% of the die area and 85% of the transistors, the Alpha-21364 L2 cache represents only 7% of the chip power. This was achieved by careful attention to clocking and to schedule. The requisite constraints were met by treating speed, power, and CAD algorithms as one integrated whole. Design decisions were made only in light of verification capabilities. Even at the high-power, low-volume end of the CPU spectrum represented by Alpha CPUs, we expect this trend to become more pronounced in the future.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge contributions from David Asher, Jeffrey Chu, Steven Felix, Patrick Fitzgerald, Chris Houghton, and Jen-Tsung Lin.

8. REFERENCES

[1] A.Jain et.al, "A 1.2 GHz Alpha microprocessor with 44.8 GB/s chip pin bandwidth," ISSCC Proceedings, Feb.2001,pp.240-.

[2] B.Benschneider et.al., "A 1GHz Alpha Microprocessor," ISSCC Proc., pp.86-7, Feb. 2000

[3] J.Montanaro et.al., "A 160Mhz,32b,.5W CMOS RISC Microprocessor," ISSCC 1996

[4] K.Ghose&M.Kamble, "Reducing Power in Superscalar Processor Caches Using Subbanking, Multiple Line Buffers, and Bit-Line Segmentation," ISPLED'99, pp.70-5.

[5] Chandrakasan et.al.,"Design of High-Performance Microprocessor Circuits," IEEE press, 2001.

[6] T.Xanthopoulos et.al, "The Design and Analysis of the Clock Distribution Network for a 1.2 Ghz Alpha Microprocessor," ISSCC Proc., pp.402-3, Feb.2001

[7] E.Friedman,“Clock Distribution Networks in VLSI Circuits & Systems,” IEEE Press, 1995.

[8] R.Bryant, “Boolean Analysis of MOS Circuits, “IEEE T. CAD of ICs, July 1987

[9] R.Bryant,“Formal Verification of Memory Circuits By Switch-Level Simulation,” IEEE Trans. On CAD of ICs, Jan 1991

[10] M. Pandy and R.Bryant, “Exploiting symmetry when verifying transistor-level circuits by symbolic trajectory evaluation,” IEEE Transactions on CAD of ICs, Vol. 18, No. 7 (July 1999).

[11] M.Pandy et.al., “Formal Verification of PowerPC Arrays using STE,” DAC’96, pp.649-54.

[12] J.Grodstein et.al., “Race Detection for Two-Phase Systems,” ICCAD-90.

[13] Y.Ismail, E.Friedman, and J.Neves, "Figures of Merit to Characterize the Importance of On-Chip Inductance," IEEE T. on VLSI Systems, Dec 1999, pp. 442-4.

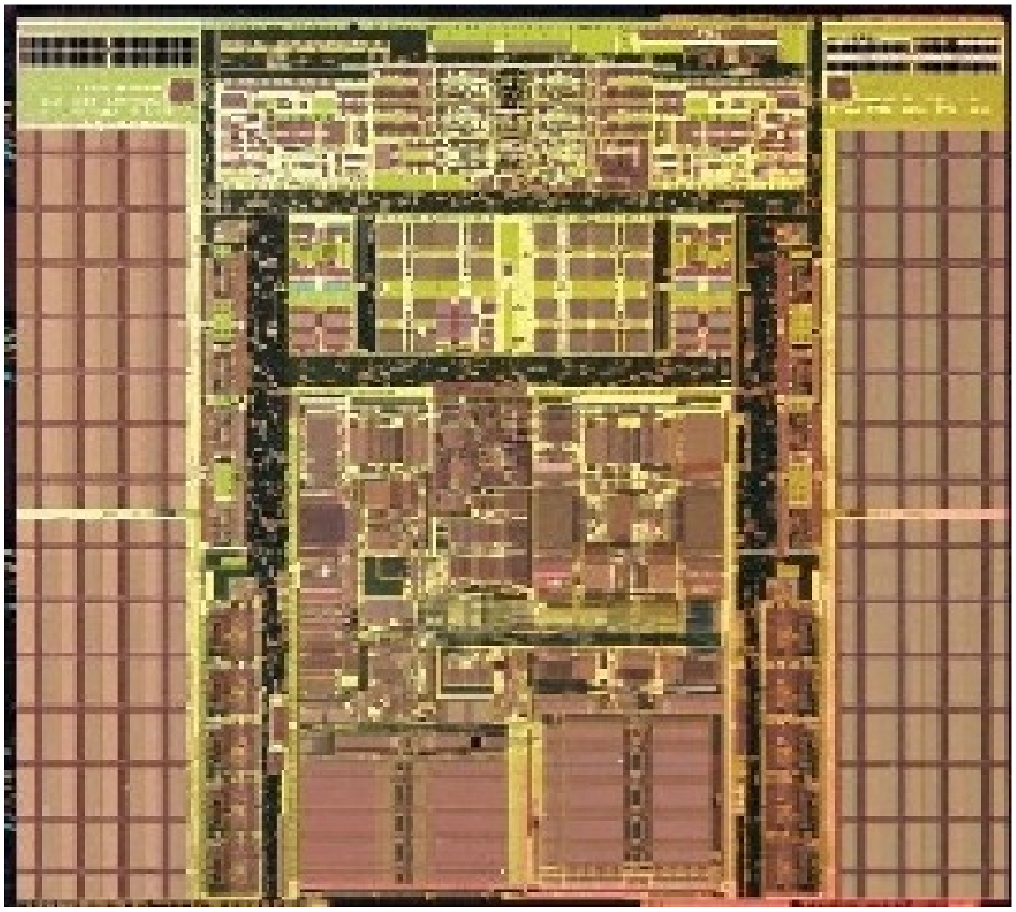

Figure 1

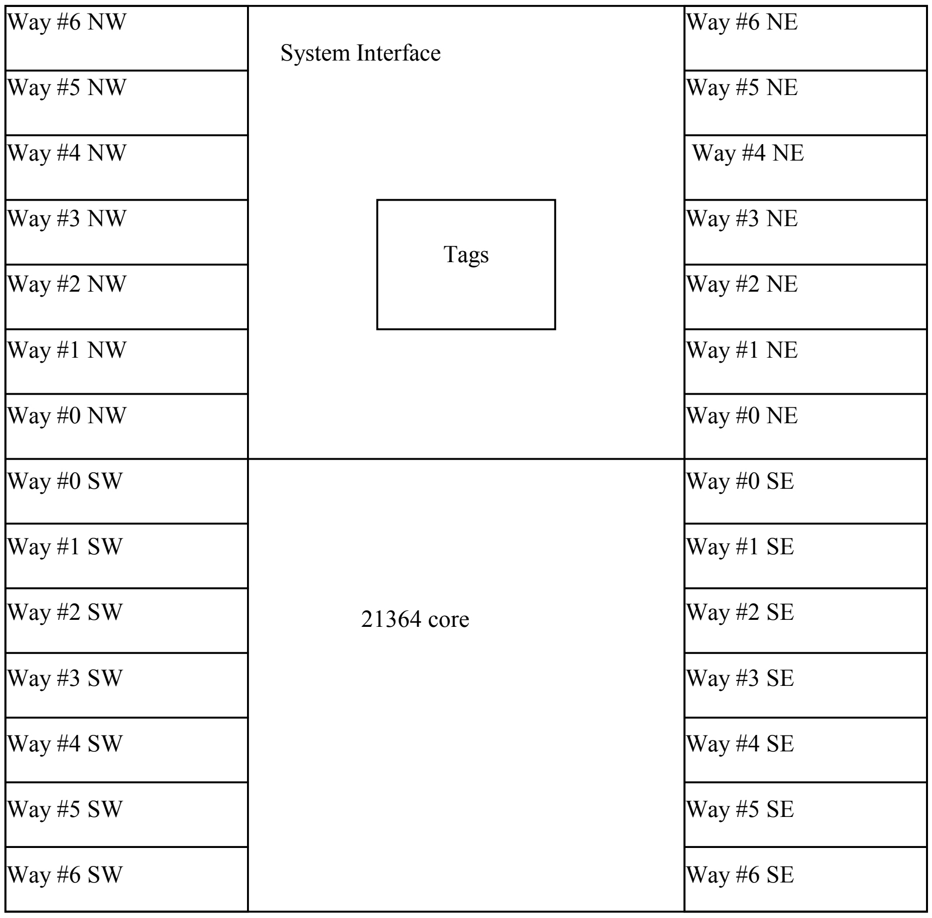

Figure 2

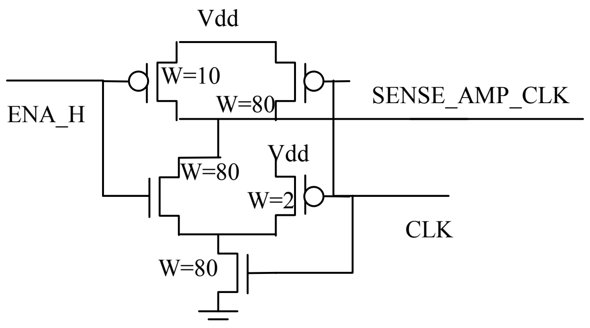

Figure 3

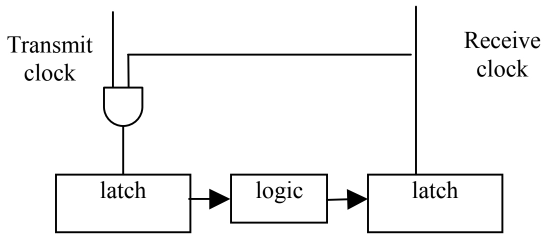

Figure 4