Design and Integration Methods for a Multi-threaded Dual Core 65nm Xeon® Processor

Raj Varada, Mysore Sriram, Kris Chou, James Guzzo
Intel Corporation, 2200 Mission College Blvd, Santa Clara, CA
{raj.r.varada, mysore.sriram, kris.chou, james.t.guzzo}@intel.com

ABSTRACT

The success of building a complex multi-billion transistor processor is very dependent on robust and silicon proven design and integration methods. The complexity of 65nm process and striving for best in class performance with aggressive time to market schedule put a heavy emphasis on innovative design and integration methods to enable working silicon. In this paper, we describe the design and integration methods successfully used in a multi-threaded dual core 65nm Xeon® Processor.

Index Terms—Design Methods; Integration; processor; Xeon®.

1. INTRODUCTION

The process scaling efficiency as predicted by Moore's law enables more than one processor to be built into the same die in the 65nm process node. Such a dual core processor, when coupled with multi-threading technology [1] makes it appear as four logical processors capable of running four threads simultaneously on the same processor die. The design and integration of such a processor with transistor counts exceeding one billion is a significant challenge given the time to market goals and complexity of pioneering a 65nm processor. The successful delivery of the design hinges on exacting design standards and methods. In this paper, we describe the design and integration methods used to build a multi-threaded dual core Xeon® Processor [2] in the 65nm process technology.

The rest of the paper is organized as follows. Section II gives an overview of the processor and motivates the use of the specific design and integration methods needed for building a complex processor. Section III describes the key features of the block design methods used and Section IV describes the integration methods. Section V concludes the paper.

2. PROCESSOR OVERVIEW

The Xeon® MP processor described in this paper consists of two 64-bit cores and a 16MB unified level-3 (L3) cache. Each core supports two threads, has a unified 1MB level-2 (L2) cache and was largely leveraged from common core architecture. This processor required limited core optimizations to create an efficient, high performance simple direct interface (SDI) connecting the two processor cores, unified L3 cache and processor Front Side Bus (FSB) through a caching bridge controller (CBC).

ICCAD'06, November 5-9, 2006, San Jose, CA
Copyright 2006 ACM 1-59593-389-1/06/0011...$5.00

This architecture reduces the cache and external bus access latencies. Figure 1 shows the SDI interface at a conceptual level. The caching CBC handles the core arbitration, L3 cache accesses, and external bus requests. This was an entirely new design for implementing a low latency, high speed on die communication fabric containing server RAS features for robust server operation.

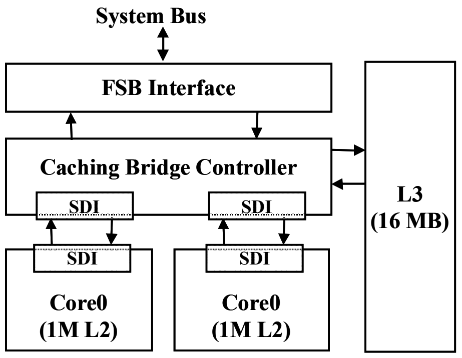


Figure 1: Processor Block Diagram

The CBC logic was partitioned into physical units and physical implementation of the CBC logic and the chip level integration were performed using the techniques described in this paper. The main constituents of the CBC physical hierarchy was control and data exchange blocks. Apart from these blocks, there were other pieces of logic for clocking, DFx and domain crossing; description of the implementation of these blocks is beyond the scope of this paper.

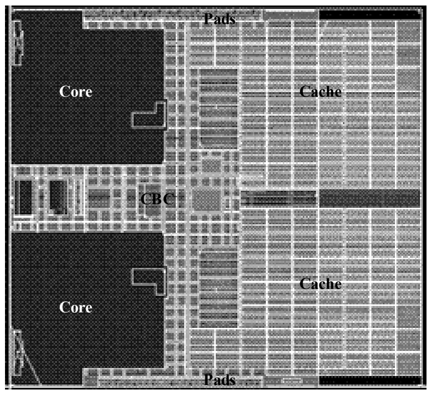


Figure 2: Processor Die Plot

The cache portion of the chip was designed using techniques similar to those in [2] and [3] and was treated as a black box for integration; as such, a description of the cache techniques is beyond the scope of this paper. Figure 2 shows a die plot of the processor at the top level. The processor die is 435mm2 die and has 1.33B transistors. It operates at more than 3.0GHz from a 1.25V core supply. The worst-case power dissipation is 150W while the power dissipation on a typical server workload is 100W. The processor is implemented in a 65nm process technology with eight copper interconnect layers and low-k carbon-doped oxide inter-level dielectric.

3. DESIGN METHODS

3.1. Cell Based Design

All the logic units in the CBC were physically implemented using cell based design (CBD) methodology. In this methodology, the entire design is built from cells in a library. The cells follow a common architecture including the cell footprint and port definition. Most of cells in the library are characterized for timing and other electrical characteristics, but it is not a pre-requisite. Cell based design methodology is advantageous for time critical projects, but requires timing-area-power trade-offs.

Most of the CBC logic was implemented with synthesis, and place and route techniques. The specific methodology used in each of these areas is outside the scope of this paper. Cell sizing was done with a combination of external industry tools and in-house tools. Significant effort was spent up front to correlate the results from the sign-off verification tools to the optimization engines in order to ensure monotonic front-to-back design convergence (for electrical and geometrical rules, timing, noise and reliability).

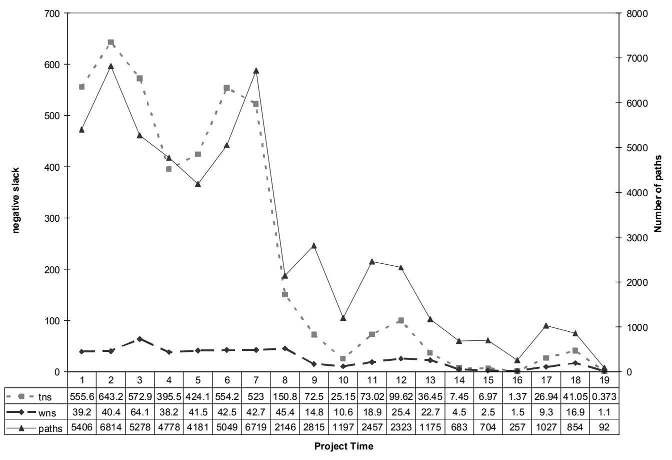
The logic partitioning and block size determination went hand –in-hand with the routing wire width determination. As explained in Section 1V, wider metal with discrete widths were used to allow maximum flexibility in block size. The logic, in addition to being partitioned according to connectivity, was also partitioned so that the layout for the logic partition would fit within a specified maximum physical block size. The block sizes in CBC in this processor were larger than the sizes seen historically in processor designs, which enabled global logic optimization and reduced the effort to maintain design collaterals for each partition. The number of discrete blocks reduced as well as minimization of Full Chip (FC) interfaces for FC floorplan and timing.

The end result of this exercise was the breaking down of the CBC logic into five main control blocks, one data exchange block and one register file block. The control logic handles the entire input/output transaction logic along with the bus and snoop queues and associated transaction queuing and arbitration logic for the requests from different sources. One of the three main control blocks (CONTROL1) is much larger and more complex than the others. The complexity is compounded by the heavy interconnectivity inside the block leading to huge fan-in/fan-out logic cones. With multiple clock domains, there were numerous timing paths with the launching and receiving flops in different clock domains (cross-clock paths).

3.2. Pipeline Convergence

One of the critical tasks for the processor design was the convergence of the CBC logic pipeline, as the RTL was designed from scratch. The pipeline convergence was driven by placement based timing feedback and for this, the post-placement timing results had to be monotonic with respect to the post-layout timing. There were two enablers for this process. First, the correlation effort described above allowed for monotonic convergence within the block. Second, very detailed timing budgets at the interface pin level were stabilized very early on in the design cycle and decoupled the block level timing convergence from the processor top level timing convergence.

A specific methodology was followed for the path analysis. Paths with logic depth greater than a process dependent predetermined number and those which had huge negative slack were the first candidates for logic changes (logic optimization, logic re-partitioning or pipelining). High fan out and fan-in logic cones were analyzed in detail and heavily loaded signals were replicated (in some cases, many times). At all times, the floorplan and timing budgets were kept in-sync with these logic changes and congestion and route-ability were evaluated. Pipeline convergence efforts in the CONTROL1 block based on early CBD feedback resulted in significant acceleration of the overall design convergence. Figure 3 below shows the convergence trend over time of pipeline convergence effort for the CONTROL1 block. It does not show the state of the design at tapeout. The figure shows the decreasing trend in total negative slack (TNS), worst negative slack (WNS) and the number of paths (PATHS) in CONTROL1 over this period.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
tns	555.6	643.2	572.9	395.5	424.1	554.2	523	150.8	72.5	25.15	73.02	99.62	36.45	7.45	6.97	1.37	26.94	41.05	0.373
wns	39.2	40.4	64.1	38.2	41.5	42.5	42.7	45.4	14.8	10.6	18.9	25.4	22.7	4.5	2.5	1.5	9.3	16.9	1.1
paths	5406	6814	5278	4778	4181	5049	6719	2146	2815	1197	2457	2323	1175	683	704	257	1027	854	92

Figure 3: Pipeline Convergence Trend for CONTROL1

3.3. Power Reduction

The CBC was designed to strict power constraints. CBC blocks were placed into one of two categories based on their average activity factor (AF) and performance sensitivity or timing criticality. Timing critical blocks with high average AF were initially designed using primarily nominal transistor based cells. Once a desired level of timing convergence was met, cells along non-timing critical paths were selectively swapped to cells containing low leakage transistors (LL-cells) without impacting any of the design convergence metrics. For lower activity factor blocks the design was built using LL-cells and a few Nominal cells were then used to converge the last few timing paths. It was observed that the cell count and total capacitance was higher with the use of LL-cells only, thus the use of nominal cells for high AF blocks resulted in lower dynamic power. This two pronged strategy enabled fast design convergence, as well as activity appropriate power reduction strategy to be applied.

4. Integration Methods

The key goal for our full-chip integration methodology was to decouple the full-chip layout and block layout, so that work could proceed on both in parallel, to meet the aggressive schedule for the project. This section describes some of the techniques that were used to achieve this decoupling.

4.1. Full Chip Signal Pitches and Routing

The die size for this chip was largely set by the fixed components – the cores and cache – and did not depend as critically on the integration methodology. Additionally, the performance trade-off in the CBC enabled the creation of a full-chip integration methodology that was optimized for rapidly converging the timing and layout of full-chip and CBC blocks, virtually independent of each other.

The first parameters to be tuned were the signal routing pitches. By using a pre-defined set of discrete wire widths and spacing values that were much wider than the process minimum width and spacing, and by making all full-chip wires half-shielded to reduce Miller coupling, the repeating distances for full-chip routing were made as long as possible. Also, the repeating distances on different layers were adjusted such that they had a simple relationship to each other. For example, metal8 repeating distance was twice the metal6 repeating distance, which in turn was twice the metal4 repeating distance. This simplified the placement of repeater stations, as described in the next section.

Simultaneously, the CBC implementation team diligently adjusted the aspect ratios of blocks so that no block was wider than the repeating distance on metal8, the highest horizontal routing layer. This ensured that full-chip routes crossing over a block on metal8 would not need to dive down into the block to get repeated. After studying global routing requirements, minimal wiring channels were allocated in the floorplan to accommodate less critical global wires on lower level metal layers. This approach significantly simplified the process of multiple instantiation of custom blocks such as tag arrays, since no top level routes or repeaters were pushed into these blocks. This also eliminated the need to re-characterize the electrical characteristics of the highly leveraged core design providing maximum reuse and fastest overall time to market.

The net result was that full-chip wires on metal8 crossed over blocks without requiring to be repeated in the blocks, and full-chip wires on lower metals were routed in the whitespace surrounding the blocks and did not need to cross over the blocks. From the block perspective, this resulted in a relatively stable physical environment in which the block design happened, with the block gaining complete ownership of routing resources up to metal7, and having some well-defined metal sharing with full-chip on metal8. Furthermore, the block netlist did not need to be augmented with full-chip signals and repeaters that would change in each integration cycle. This stable environment contributed greatly to the rapid convergence of block-level layout and timing. While this approach involved a small penalty in die area, it was far outweighed by enabling significant design re-use and accelerated time to market (TTM).

4.2. Full Chip Repeater Methodology

For maximum productivity on this project, a *virtual repeater* methodology was used. Virtual repeaters are symbolic annotations placed on wires, with information about the repeater cell that they represent. The extraction and timing tools recognize these annotations and can calculate delays and slopes as if real repeater cells were inserted at the indicated locations, without requiring the overhead of netlist and wire fracturing. The repeaters were maintained in virtual form until quite late in the project – about a quarter before tapeout. Once the design was stable, real repeaters were inserted into the netlist based on the virtual repeater locations. Keeping the repeaters virtual allowed the integration team to respond quickly to late design changes, while maintaining acceptable accuracy in the timing model.

The virtual repeater insertion tool minimized the total delay while meeting slope targets at every receiver and repeater input. As mentioned in the previous section, all full-chip repeaters were inserted in the whitespace between full-chip blocks. The floorplan was populated with a grid of *repeater stations,* which are narrow horizontal or vertical blocks placed in the whitespace in the floorplan that mark the location of legal repeater placement sites. These were spaced according to the repeating distance of the lowest full-chip routing layer in that direction. For example, in the horizontal direction, repeater stations were placed approximately 600u apart. Metal4 wires would need to be repeated every time they crossed a station, while Metal6 wires would need to be repeated in every other station and Metal8 wires could skip over 3 stations before requiring a repeater. The virtual repeater insertion tool was constrained to insert repeaters only in the keep-in regions defined by the unpopulated repeater station sites. Figure 4 below shows the grid of repeater stations in the CBC area.

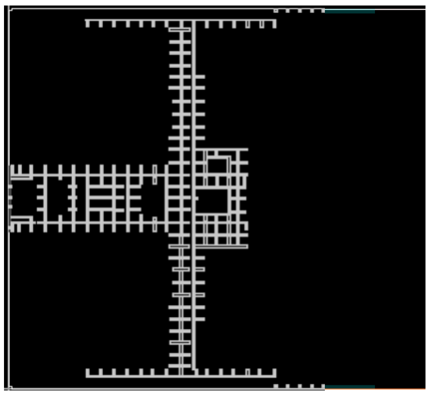


Figure 4: Repeater Stations in CBC Area

To further simplify data flows and decouple repeater insertion from timing tools, the repeater insertion engine did its own parasitic estimation, and used default "tail" data for block internal parasitics, as opposed to stitching in RC from a block extraction run. The default tails could be overridden by a control file, as explained in the next section.

Converting the virtual repeaters into real repeaters was done by a tool that traced the routing for each net, breaking the wires as they crossed over repeater stations with virtual repeater annotations over them. The broken segments of the nets were assigned unique names. Pins were created on the repeater station boundaries, and the repeater cell was inserted into the station netlist. The repeater stations were then assembled by a simple custom place-and-route flow.

4.3. Full-Chip Convergence Techniques

Since time to market was an important parameter on this project, it was essential to leverage as much automation as possible for full-chip routing and repeating. However, one typical drawback of this is the slower convergence caused by variation in the results produced by automation tools. A spec-based mechanism in the auto-router and a fine-grained control mechanism for the repeater insertion tool helped to keep the variations under control and to generate predictable results.

The auto-router allowed routing specs to be specified with different levels of granularity. In the early stages of the project, the specs were simple layer-pair directives. However, as the design stabilized and more paths met their timing, additional constraints were provided in the form of topology directives to the router, effectively "soft-freezing" the routes. When the design was near-converged, almost every FC net had topology specs attached, to prevent unnecessary variations.

Since virtual repeaters were converted to physical repeaters just one quarter before tapeout, it became necessary to ensure that the virtual repeater insertion results were predictable, and that any manual tuning of the virtual repeater locations could be reproduced the next time we ran the tool. To this end, a control file mechanism was devised which allowed the user to specify how far the first or last repeater got inserted from a specific block pin, which subset of repeater stations to use to repeat a specific net, or overrides for the default slope target values for some set of nets, etc. By maintaining this control file, the virtual repeater insertion tool was able to generate highly reproducible results even though it was run from scratch in each integration cycle.

Full-chip timing feedback was incorporated into the floorplan by one of several manual approaches. Scenic routes, layer changes, topology changes, etc. were fixed either by manually routing the net and preserving it as a hard pre-route, or by editing the spec for the net so that the auto router could generate the preferred topology. Tweaks to the repeater solution were captured as changes to the control file.

In order to further decouple and speed up the convergence of full-chip and block-level timing paths, the virtual repeater insertion tool was directed to insert the first repeater on a signal emanating from a CBD block as close to the driving pin as possible. Similarly, the last repeater on a signal entering a CBD block was constrained to be within a specific radius of the input pin. This ensured that the CBD blocks saw very stable external loads on driver pins and slopes on receiver pins, enabling faster convergence.

The total number of CBC signals that required routing at full-chip level was about 7000. Nearly 99% of these nets were routed within 20% of their optimal wire length, which indicates that the restrictions imposed on routing over the blocks did not hurt route optimality significantly. 231 repeater stations were placed in the floorplan, and over 30,000 repeaters were inserted on CBC signals at the full-chip level.

5. CONCLUSION

Block design and integration methods are key elements of bringing billion+ transistor server processor to market. We have presented a number of novel strategies and methods used in the design and integration of a Xeon server processor that have enabled delivery of industry leading performance on an accelerated development timeline. Aggressive usage of cell based design methods enabled early pipeline convergence using placement based timing. Power targets were achieved by block appropriate power reduction methods, including upfront design with high threshold voltage cells. Use of multiple wire pitches and half shielding allowed maximal repeating distances per metal layer thereby enabling bigger block sizes, minimizing repeater count, and eliminating the need for embedded FC repeating inside functional blocks. This correct by construction approach to FC metallization significantly increased electrical robustness of the overall design (minimal effort for FC noise, EM, self heat convergence), enabled maximum hard IP reuse of the core, and significantly decoupled the design convergence of functional blocks from convergence at the full chip level. Virtual repeating methodology achieved near final timing quality early in the design cycle with minimal perturbation in the design process.

Through a set of comprehensive and coordinated methodology decisions made early in the design cycle we were able to strategically decouple several aspects of IC design. RTL development, full chip assembly, timing, cell based design, core, and cache physical design were largely completed in parallel. Decoupling of FC assembly, timing, and robustness from functional block convergence was achieved by reducing the FC interfaces through bigger functional block sizes, pre-defined metal sharing, eliminating embedded repeater stations, and spec-based timing budgeting. By making calculated trade offs in area, schedule, performance and complexity we were able to deliver a robust design with market leadership performance, in break though development times. Many of the methodology innovations described in this paper are now a standard part of the processor design process for bringing the highest performance, most reliable server processors to market.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the work of the talented and dedicated Intel team that implemented this processor.

7. REFERENCES

[1] Hyper-Threading Technology, Intel Technology Journal, Vol. 6 Issue 1, Feb 2002

[2] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, "A Dual-Core Multi-Threaded Xeon® Processor with 16MB L3 Cache", International Solid State Circuits Conference, pp 102-104, January 2006.

[3] J. Chang, J. Shoemaker, M. Haque, M. Huang, K. Truong, M. Karim, S. Chiu, G. Leong, K. Desai, R. Goe, S. Kulkarni, A. Rao, D. Hannoun, S. Rusu, "A 0.13/spl mu/m triple-Vt 9MB third level on-die cache for the Itanium/spl reg/ 2 processor", International Solid State Circuits Conference, January 2004.