POWER8 Design Methodology Innovations for Improving Productivity and Reducing Power

Matthew M. Ziegler[1], Ruchir Puri[1], Bob Philhower[1], Robert Franch[1], Wing Luk[1], Jens Leenstra[2], Peter Verwegen[2], Niels Fricke[2], George Gristede[1], Eric Fluhr[3], Victor Zyuban[1]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY,
[2]IBM Systems & Technology Group, Boeblingen, Germany,
[3]IBM Systems & Technology Group, Austin, TX

***Abstract* — The design complexity of modern high performance processors calls for innovative design methodologies for achieving time-to-market goals. New design techniques are also needed to curtail power increases that inherently arise from ever increasing performance targets. This paper describes new design approaches employed by the POWER8 processor design team to address complexity and power consumption challenges. Improvements in productivity are attained by leveraging a new and more synthesis-centric design methodology. New optimization strategies for synthesized macros allow power reduction without sacrificing performance. These methodology innovations contributed to the industry leading performance of the POWER8 processor. Overall, POWER8 delivers a 2.5x increase in per-socket performance over its predecessor, POWER7+, while maintaining the same power dissipation.**

***Index Terms* — Design methodology, low power design, synthesis, processors, servers**

I. Introduction

The design complexity of high performance processors continues to increase as the goals of higher throughput and functionality push transistor counts into the multi-billions. As a result, the industry looks to boost productivity by increasingly relying on synthesis for even high performance designs that have conventionally employed more custom methodologies. Time-to-market pressures and the desire for cost savings via smaller design teams further fuel the shift towards synthesis-centric methodologies. But, as the complexity and performance of high-end chips rises, power consumption emerges as a first order design limitation. Power is now a focal point for high-end servers targeting maximum performance.

This paper describes new design techniques and methodologies for overcoming the productivity and power challenges encountered while designing the IBM POWER8™ processor [1]. Leveraging synthesis and automation to a higher degree addressed the complexity and productivity goals. To meet performance levels conventionally attained by custom design, an enhanced synthesis flow called Structured Synthesis, provided the ability to employ custom design techniques within the synthesis flow. Scaling to larger synthesized blocks further improved productivity as well as provided area and power improvements through a flow we call Large Block Synthesis (LBS). We refer to the overall flow that merges custom capabilities and allows scaling macro size as Large Block Structured Synthesis (LBSS) [2].

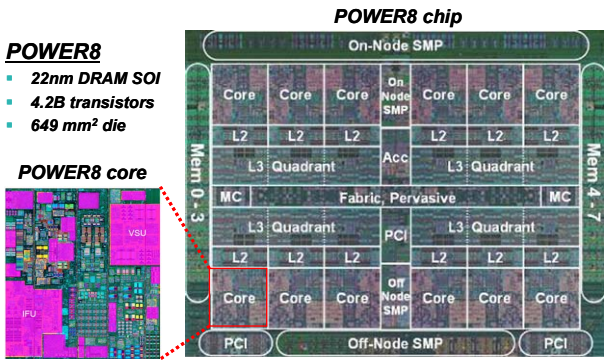

Fig. 1 The POWER8 chip and core flooplans. The larger synthesized macros are shaded pink in the core floorplan.

The shift to a synthesis-centric design approach for a high-end server provides both new challenges and new opportunities. Synthesis streamlines the implementation flow, but reduces control over fine grain tuning. While infusing custom techniques via Structured Synthesis allows designers to retain some fine tuning capabilities, entirely new tuning approaches harnessing modern compute resources are now possible. Leveraging more the powerful synthesis tools as well as more abundant compute resources, we have developed an automated approach for design optimization based on design space exploration called SynTunSys (the Synthesis Tuning System).

The combination of these techniques led to a powerful design methodology that makes use of human design skills for highly critical components while further leveraging automation and compute resources for optimization via design space exploration. Over the course of this paper we will detail these design techniques and methodologies as well as describe how the design process of recent IBM

978-1-4799-3286-3/14/$31.00 ©2014 IEEE

POWER Architecture processors evolved to the POWER8 flow.

II. AN OVERVIEW OF THE POWER8 PROCESSOR

This section provides an overview of the POWER8 processor to frame the complexity of the design challenge. POWER8 is the eighth generation of the IBM Power Architecture targeting industry leading performance [1]. The 649 mm^2 processor die, shown in Fig. 1, is implemented in IBM's 22-nm eDRAM SOI technology [3] and consists of 4.2 billion transistors. It includes twelve architecturally enhanced eight-way multithreaded cores that are supplied by high-throughput private second-level caches, a 96-MB high-bandwidth eDRAM third-level cache, and an on-chip Shared Memory Processor (SMP) fabric. A set of cryptography and memory compression accelerators as well as memory controllers are implemented on-chip. IO links capable of connecting to up to eight memory buffer chips, six high-bandwidth SMP links, and 32 third-generation PCI express lanes provide off-chip communication. Overall off-chip bandwidth can achieve 7.6Tb/s.

All of these enhancements allow POWER8 to deliver a 2.5x increase in per-socket performance while maintaining the power dissipation at the level of its predecessor, POWER7+ [4]. While POWER8 is the cumulative effort of innovations spanning architecture, technology, and design disciplines, this paper focuses on new methodologies and techniques for realizing the POWER8 physical design.

III. THE LARGE BLOCK STRUCTURED SYNTHESIS (LBSS) FLOW

High-end servers have conventionally relied on transistor-level custom design techniques[1] for achieving high performance, e.g., the POWER6 processor which achieves frequencies over 5GHz [5]. At a lower frequency envelope, ASIC design has conventionally achieved design efficiency over high performance processors by more heavily relying on synthesis. While this gap between custom and ASIC design efficiency has been noted in prior publications [6], [7], it has not been until recently that synthesis has made major inroads in the realm of high performance servers running in the 4-5GHz range.

[1] Custom design in our context refers to a methodology based on schematic entry followed by placement and routing. This is opposed to synthesis which automates the design process beginning at the RTL level.

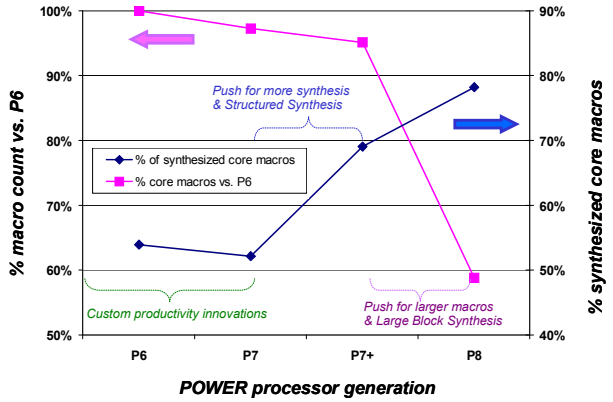

Fig. 2 POWER processors have leveraged synthesis more heavily in recent generations, driven by both Structured Synthesis and Large Block Synthesis methodologies.

One representative metric that helps illustrate the evolution of synthesis in IBM's server design flow is the percentage of synthesized macros in the processor core, as Fig. 2 illustrates. The core is typically the portion of the chip that limits performance from a frequency perspective and where custom design is more conventionally utilized.[2] During the POWER6 and POWER7 time frame, improving custom design productivity was a major focus. These innovations allowed reducing the cost associated in custom design tasks in key areas, such as, automated leaf cell generation, placement, and routing [5], [8]. This focus on custom design productivity was quite appropriate at the time, seeing that custom design represented ~50% of the core macros and the majority of the most challenging macros. Following significant custom design productivity improvements, productivity efforts began focusing more heavily on synthesis, eventually evolving into the POWER8 LBSS methodology.

As Fig. 2 shows, a significant increase in synthesized macros occurred between the POWER7 and POWER7+ chips.[3] POWER7+ provided an ideal situation to push for higher use of synthesis. Underlying improvements in IBM's PDSrtl synthesis tool [9] combined with the higher logic stability of POWER7+, with respect to POWER7, lowered the risk of shifting towards synthesis. The Structured Synthesis methodology was a key catalyst for the transition to higher use of synthesis for core macros. Structured Synthesis provided techniques to span any remaining gap for timing closure. For example, if a macro could not close timing using the standard synthesis flow,

[2] The portion of the chip surrounding the core, which we refer to as the "nest", conventionally uses synthesis more heavily than the core.

[3] Note that Fig. 2 refers to the percent of synthesized macro counts and not percent of synthesized macro area. Also note that this figure does not consider array, I/O, or analog macros.

the designer could intervene and apply custom techniques to only the critical portions of the macro.

Building on the success of POWER7+, the trend of higher synthesis use continued in the POWER8 core. The adoption of Large Block Synthesis (LBS), which improved synthesis turn-around-time, in combination with Structured Synthesis techniques led not only to a greater use of synthesis, but also to larger macros. Entire core units on the chip were synthesized as single large block macros and the number of core macros in POWER8 was reduced by 50% compared to POWER6, as Fig. 2 shows. The following subsections delve into the key aspects of LBSS.

A. Structured Synthesis

Structured Synthesis aims to merge the productivity of a synthesis flow with the fine grained designer intervention of a custom design flow. Conceptually, it strives to allow custom design to be applied to only the portion of the macro that is critical, while allowing the remainder of the macro to be synthesized. Prior to enabling custom techniques within synthesis, the entire macro would have had to be designed custom to allow even a small customized portion. Allocating significant custom design effort late in a design cycle creates schedule risk. Structured Synthesis alleviates this risk by allowing custom techniques to be applied incrementally to only the most critical portions of a design. Thus more challenging macros can be synthesized without the risk of having to revert to complete custom designs late in the schedule, should timing closure difficulty arise.

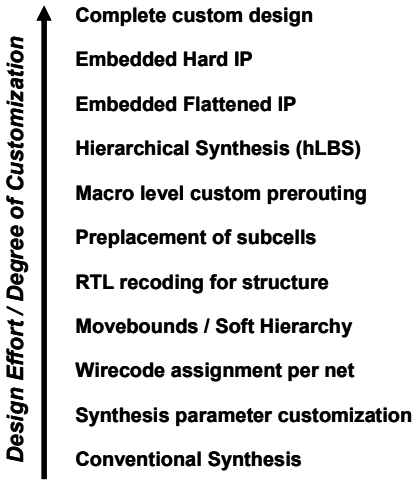

Fig. 3 The spectrum of Structured Synthesis techniques spans from conventional synthesis to custom design.

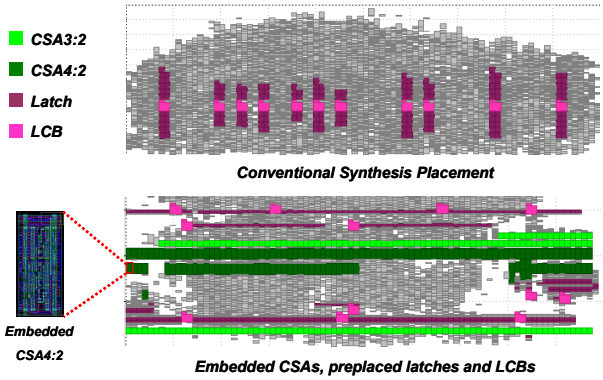

Fig. 4 Preplacement and embedded custom IP blocks are used in this example to close timing, achieve 8% power, and 16% area savings.

Improvements over conventional synthesis

design version	timing	wire length	area
b) designer latch preplacement	28%	30%	5%
c) automated latch placement	16%	27%	2%

Fig. 5 Designer supplied latch placement and automated structured latch placement provide various degrees of control and automation.

Structured Synthesis is based on a portfolio of custom techniques that can be applied within the server synthesis flow, as needed. The general philosophy is to attempt to close macro timing using the standard synthesis flow and then incrementally apply Structured Synthesis techniques, as needed, which attempts to minimize design effort. Once a macro achieves timing closure, additional customization can be applied to further optimize the macro from a power or area perspective, if the return-on-investment (ROI) for additional design effort appears favorable. Fig. 3 shows the spectrum of Structured Synthesis techniques in terms of the first order design effort and degree of customization. Although the design effort for specific application of a custom solution can vary greatly, the general approach is to look to lower-effort techniques first. The choice of using a technique depends on the macro specific problem to be solved as well as the expected ROI in implementing the technique.

Fig. 4 shows a design example showing conventional synthesis, i.e., synthesis without Structured Synthesis techniques, in comparison to a structured implementation. In this example latches and LCBs (local clock blocks) are preplaced. In addition, two types of custom components are embedded preplaced. Whereas the conventional synthesis result does not meet the timing requirements, for this specific example the Structured Synthesis techniques close macro timing as well as provide an 8% power reduction and 16% area reduction.

The Datapath Synthesis placement algorithm was added to the PDSrtl synthesis program to build upon designer supplied preplacement [10], [11]. Using preplaced cells and macro pins as anchors, this new placement algorithm infers structure in the macro and attempts to align datapath portions of the macro to the anchors, while non-datapath portions of the macro revert to convention placement algorithms more appropriate for "random logic". In addition, an automated latch and LCB placement algorithm was added to PDSrtl to emulate the style of preplacements designers supply [12], [13]. Fig. 5 shows an example of a macro containing many latches and the a) conventional synthesis placement, b) designer supplied preplacement, c) the new datapath latch placement algorithm. From this example we see automated latch placement resembles the placements from an experienced custom designer. Although the automated latch placement algorithm does not fully attain the level of improvements as the designer directed preplacements, the design effort of the automated approach was far less and in many cases provides an attractive ROI.

B. Large Block Synthesis

While Structured Synthesis provides a continuum of techniques to attain near-custom quality from synthesis, scaling to larger macro sizes allows another level of synthesis gains. The core benefit of scaling to larger macro sizes is a lower reliance on macro boundary conditions, such as timing assertions and physical constraints, e.g., macro size, aspect ratio, and pin positions. The maintenance and accuracy of these boundary conditions are persistent challenges for physical design. Moving to larger macro sizes removes the need for boundary conditions between the previously small macros within the larger macro and allows the synthesis tool to fully optimize paths across the small macros as native internal paths. On the other hand, larger macros present challenges in terms of design complexity, tool runtime, and design efficiency via portioning work among multiple designers.

To address these challenges, the PDSrtl synthesis tool was optimized for issues that arise in larger macros as well

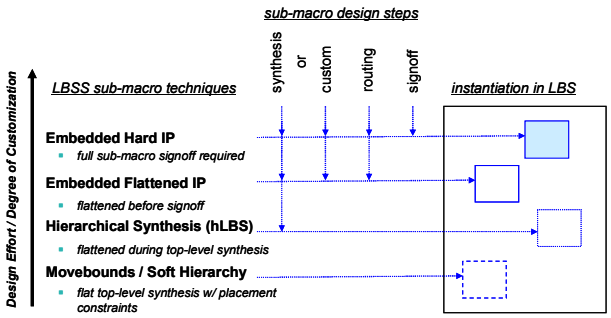

Fig. 6 A spectrum of techniques also exists at the large block level, providing various levels of customization.

to generally improve runtime for large macros. Routing congestion is one issue often facing larger macros. For these macros, the synthesis tool must find solutions for the routing complexities conventionally handled by experienced human unit integrators. These challenges were addressed in POWER8 by enhancing the PDSrtl tool with options for automated congestion aware placement and wire assignment.

In addition, new design techniques for controlling portions of the large block design were developed during POWER8, providing designer guided facilities for improving congestion, timing, and power. These techniques provide a range of customization and design effort tradeoffs, as shown in Fig. 6. The Soft Hierarchy flow allows designers to cluster subsets of logic gates within a large block macro [14], thereby allowing the synthesis tools to increase focus on localized critical portions of the design. This requires low design effort in that it only requires specifying the sub-macro hierarchy to physically group and optionally a bounding box for the sub-macro placement. The Hierarchical LBS flow (hLBS) allows designers to embed pre-synthesized sub-macros in a large macro. A key aspect of hLBS flow is that only the synthesis step[4] is needed for the sub-macro as the sub-macro boundary dissolves during the synthesis process, allowing synthesis to operate on paths crossing the sub-macro boundary. In addition, hLBS allows division of labor among designers and allows each sub-macro to be synthesis with unique synthesis options. Further custom design precision can be attained by designing an embedded flattened IP block using either synthesis or custom design techniques. In this case the sub-macro is designed using standard cells and routed; however the sub-macro is flattened prior to top-level signoff. Finally, embedded hard IP allows full customization, such as using

[4] In this context, synthesis includes logic synthesis and placement, but not routing.

non-standard cell gates, but requires highest degree of design effort, i.e., full signoff of the sub-macro.

Finally, the runtime of tools was improved by adopting a gate-level signoff (GLSO) methodology for macros designated as large blocks. Prior POWER processors mostly employed a transistor-level signoff (TLSO) methodology requiring flat transistor-level extraction. The GLSO tools provide a 3-10x speedup over TLSO for verifying the timing, power, electrical integrity, compliance to electromigration reliability rules, and manufacturing design rules using a gate-level extracted netlist [1]. Prior to POWER8, many of the GLSO tools were not yet adequate for analyzing many of the custom design styles that enabled processor operation in the frequency range of 4-5 GHz. However, an extensive GLSO correlation effort to TLSO and SPICE level simulation led to an accurate and efficient signoff flow for high frequencies designs. Note that the TLSO methodology was still extensively used by the POWER8 team for custom macros, dynamic logic, arrays, register files, clock distribution and analog design components.

IV. LBSS UNIT LEVEL DESIGN EXAMPLES

In this section we profile two POWER8 core units, the IFU (Instruction Fetch Unit) and the VSU (Vector Scalar Unit), which were designed as single LBSS macros. While there are prior reports of large synthesized blocks for processor designs, [15], [16], the LBSS core macros from POWER8 are designed to operate in excess of 4GHz with equivalent performance to custom implementations. POWER8 also contains many other LBSS macros spanning various sizes and design challenges; however the two LBSS units macros employ many design techniques described in previous sections of this paper and are ideal case studies of the LBSS methodology.

A. The Instruction Fetch Unit (IFU)

The IFU, shown in Fig. 7, is responsible for fetching, decoding and dispatching instructions to the sequencing unit, expanding complex instructions into sequences of primitive instructions, predicting and executing the program control flow instructions such as branches and tracking the addresses of all in-flight instructions. The IFU design includes 37 instances of SRAM and register files arrays, 580 thousand standard cells and 640 thousand nets. Area savings was a motivating factor in moving to a single LBSS design for POWER8. Overall, it is estimated that the LBSS implementation of the IFU saved 15% in area versus a conventional server implementation.

Preplacement of the arrays was among the first techniques employed in the IFU to provide an underlying

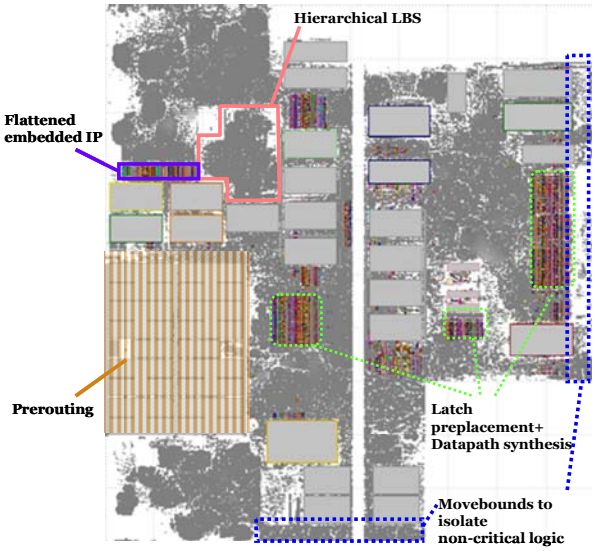

Fig. 7 The IFU (Instruction Fetch Unit) is a unit scale LBSS macro that employed a number of structured techniques for design closure.

structure to the unit. Both movebounds, that confine logic to designer specified regions, and preplacement of latches and LCBs provided additional structure and were effective in guiding the synthesis tools to give better timing and reduce wiring congestion. Approximately 12K out of over 50K total latches were preplaced. The Datapath Synthesis placement algorithm, described in section 3.2, was used to effectively place the regularly structured portions of logic, building on the preplaced latches as anchors. Fig. 7 depicts the dataflow regions as vertical colored lines. The hLBS flow was used to tackle a highly timing critical, physically irregular, control oriented sub-macro within the IFU, allowing high synthesis effort to focus on a specific portion of the unit. Likewise, a critical set of muxes were designed as a custom sub-cell and embedded in the unit, employing the flattened embedded IP process.

In terms of routing, applying layer and width assignments, i.e. wirecodes, to specific nets during synthesis proved effective for timing critical nets. Furthermore, the nets requiring a precise routing topology, such as the IFU cache output bus, were prerouted to ensure a regular topology across the bits of the bus.

Finally, providing organizational framework to the design data to allow multiple designers to work in parallel was crucial. Distributed scripting for preplacements, customized analysis scripts, and versioning programs more conventionally used for software engineering were among techniques developed to manage the complexity of the design task.

B. The Vector Scalar Unit (VSU)

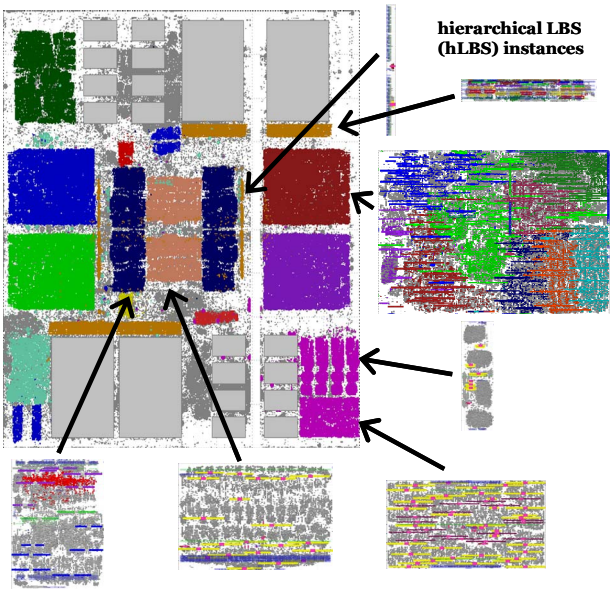

Fig. 8 The VSU (Vector Scalor Unit) leveraged hierarchical LBS (hLBS) to overcome timing and congestion challenges.

The Vector Scalar Unit (VSU) unit implements the logic and storage for executing the POWER PC Book 1 Binary Floating Point (BFU) Instructions, the Vector Multimedia eXtension (VMX) instructions and the Vector Scalar Extension (VSX) instructions. Furthermore, a number of new instructions were introduced for POWER8 to achieve higher performance while running Business Analytics and Big Data applications. Overall, the VSU unit has a total of 697K gates, 723K nets and 20 embedded arrays/register-files for storing architectural and renaming state. The VSU design presented the following challenges:

- Timing critical execution pipeline stages, crucial to overall performance.
- Routing congestion, due to the many wide operand, result, and forward buses.
- High power consumption during the thermal design point (TDP) workload that stresses the execution units.

Preplacing register files and arrays was also the first step in tackling the VSU design. However, without further designer guidance, flat / conventional synthesis led to timing and routing congestion problems. Using movebounds for building the execution units improved congestion, but did not resolve timing and synthesis run time issues. In the end, we found the hLBS method to be quite effective for tackling the VSU timing and tool run time challenges. Fig. 8 shows the seven hLBS sub-macros embedded in the larger VSU design as well as additional movebounds for less timing critical logic.

In particular, the hLBS flow was able to handle the multiple instantiations of sub-macros in the VSU, e.g., the four BFUs (binary floating point units). While the VSU attempts to maintain symmetry across the four BFUs, each copy of the BFU sees different timing at the boundary conditions. A key aspect of the hLBS flow is that the pre-synthesized macros are flattened during top-level synthesis, allowing fine tuning of the pre-synthesized macro interface circuits. This flattening allows each BFU instance to be uniquely optimized during the top-level synthesis run. A second hLBS benefit is that the top-level synthesis runtime is reduced, since the hLBS sub-macros are black-boxed during the initial steps of the top-level synthesis run.

V. POWER8 LBSS MACRO POWER OPTIMIZATION

Power consumption is emerging as a first order design metric even for high performance servers. Therefore, from a server perspective, power reduction may be converted to frequency and performance improvements [17]. POWER8 had the challenge of maintaining the power limit of its predecessor, POWER7+, while providing a substantial performance improvement. Meeting this challenge required is truly a distributed effort across architecture, design, and technology disciplines. Fig. 9 shows the spectrum of power savings techniques employed during POWER8 [1].

In this section we specifically focus on new physical design power reduction techniques for logic macros. Within physical design domain, cumulative power savings is also a distributed effort, i.e., power reduction strives for optimization of all logic macros. However, given schedule requirements and practical limits on optimization effort, power optimization requires careful analysis of the ROI for undertaking a power savings opportunity [18]. Since there are typically more power savings opportunities than can be realized with available design resources, pursuing the highest ROI opportunities is needed to maximize power reduction. During POWER8, the move to a more automated and productive synthesis-based LBSS design methodology provided a number of power savings opportunities, both directly and indirectly.

First, an overall improvement in productivity can reduce the time needed for physical design, allowing more design time for power optimization. Next, the LBSS macros are inherently more flexible in terms of large design changes versus custom designs, allowing exploration of a wider design space. This increase in macro flexibility provided more opportunities for power reduction between the first and second chip releases (RITs) than previous projects,

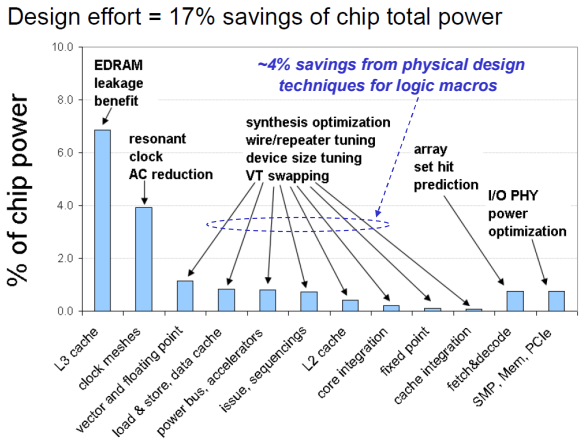

Fig. 9 The synthesis optimization and tuning techniques described in this paper contributed to 4% total chip power savings across various POWER8 units.

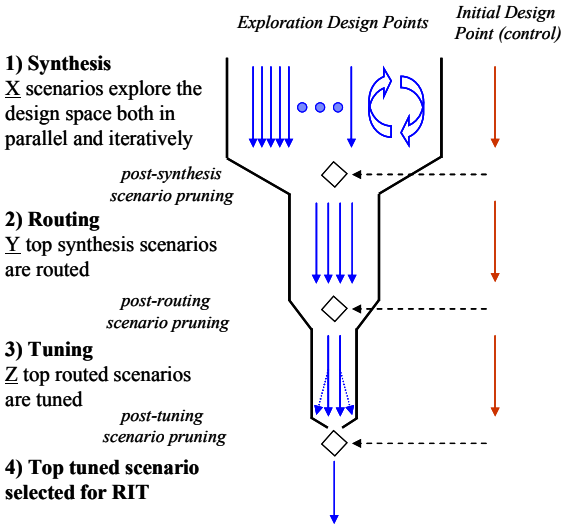

Fig. 10 The macro optimization methodology is based on a tailored flow emphasizing parallel design space exploration.

e.g., POWER7+ [19]. The following sub-section describes new optimization methodologies to take the fullest advantage of the flexibility of LBSS macros.

A. Power Reduction and Macro Optimization by Design Space Exploration

Innovative design methodologies were employed late in the design cycle during the second release to reduce the power consumption of the POWER8 processor. Design optimization by synthesis design space exploration was applied more broadly and systematically than in the prior POWER series chip, POWER7+ [19]. In particular, a new tool called SynTunSys (the Synthesis Tuning System) was developed to automate the design exploration process.

Fig. 10 shows an overview of the optimization flow. This approach is tailored to maximize effort and resources in steps of the design flow that provide the highest power savings ROI. The focusing of effort maximizes the return on the available resources, seeing there is a limited availability of designer effort, compute cluster resources, and disk space for design data. The macro power optimization methodology applies high exploration effort early during the synthesis stage. Multiple synthesis scenarios are submitted in parallel and iteratively refined. The result of each synthesis scenario is a placed netlist with estimated wires parasitics. Based on design statistics, such as timing, power, and routing congestion, the best candidate scenarios are selected to move on to the routing step of the flow. The initial design scenario from the first release is a control comparison at each step of the design flow. Further pruning of candidate scenarios occurs after routing. The remaining post-route scenarios are then tuned using new gate-level power tuning algorithms. This post-route tuning also provides another opportunity for design space exploration.

An illustrative example of the optimization process is shown in Fig. 11, starting with the initial design point from the first release towards the upper right. Multiple synthesis scenarios are generated based on deviations from the initial design point. The high number of options available from the IBM PDSrtl synthesis tool [8] provides a vast design space unique for each macro. Modifying synthesis input settings, such as, logic restructuring, latch configurations, Vt assignments, wide wire assignments, are among others, produces a different design point in the design space. The deviations from the initial design point are submitted in parallel and produce a "blast zone", some of which will improve upon the initial design point. The best solutions from the first parallel exploration, referred to as iteration 0 (*i=0* in Fig. 11), are used for subsequent explorations. Iterations *i=1* and *i=2* then are derivative sets of parallel submissions. When the synthesis design space exploration begins to see diminishing returns or if available compute resources are exhausted, the best scenarios taken through the routing steps of the flow.

The best post-route scenarios are further optimized for power using new gate-level tuning algorithms that target VT and/or gate sizing tuning. This new circuit tuning approach is described in more detail in the following section. This presents another opportunity for aggressive parallel optimization exploration. Often this leads to some scenarios having negative timing results; however, these failing scenarios can be useful for determining which

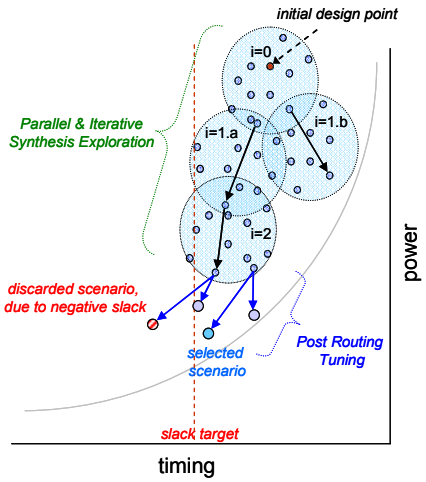

Fig. 11 Parallel and iterative design space exploration followed by post-route tuning was employed for power optimization.

optimization settings provide maximum power savings, without causes timing violations.

The optimization strategy as described above is more a general guideline and is adapted based on properties and expected power savings ROI of a specific macro. Smaller macros having high dynamic power consumption are ideal candidates for aggressive synthesis design space exploration. On the other hand, the compute resource needs of very large macros, like the POWER8 VSU and IFU unit LBS macros, may limit the exploration process to fewer scenarios. These large macros may also focus more heavily on post-route tuning so that power optimizations does not require going through the entire construction flow pipeline.

B. Post-Route Tuning in the Gate Level Flow

POWER8 also leveraged a new post-routing gate-level circuit tuning approach for synthesized macros, whereas previous POWER series chips employed the EinsTuner transistor-level tuning program [20]. Although EinsTuner was still used to tune a few POWER8 custom macros, the productivity of tuning synthesized macros was greatly enhanced from this new flow. The switch to the new gate-level tuning flow enhanced productivity in two ways. First, run time was significantly reduced versus transistor level timing. The new approach allowed tuning unit LBS macros, such as VSU and IFU, with about a day turnaround time. In contrast, EinsTuner could run days on even medium sized macros. Next, the new tuning flow runs within the eFinale [21] platform, which is more tightly integrated into the construction flow. eFinale also provides direct access to the Cadence Finale router for rerouting after gate size tuning as well as design checking. This fully integrated post-route tuning flow allows for parallel design space exploration to determine the optimal power savings scenario per macro.

VI. Conclusion

In this paper we described design methodologies and techniques for tackling the complexity and power challenges associated with the POWER8 microprocessor. The underlying shift to a more synthesis-centric methodology brought both new opportunities and challenges. To bridge gaps between synthesis and custom design methodology, we introduced Structured Synthesis techniques, allowing designers to customize portions of a synthesized macro. The POWER8 design methodology was also enhanced to allow scaling to larger macros. These collective set of techniques culminated in the Large Block Structured Synthesis (LBSS) methodology. Having more synthesized macros in comparison to prior POWER chips provided the opportunity for exploring alternative design points for macro optimization. POWER8 leveraged the lower implementation effort of synthesis and abundant compute resources for design space exploration using a new optimization flow called the Synthesis Tuning System (SynTunSys). SynTunSys and combined with a new gate level post-route power tuning techniques led to a power optimization flow for reducing power late in the design cycle, without sacrificing performance. Overall, these methodology innovations contributed to the industry leading performance of the POWER8 processor, which delivers a 2.5x increase in per-socket performance over its predecessor, POWER7+, while maintaining the same power dissipation.

Acknowledgement

The authors would like to acknowledge and thank the entire POWER8 team for their hard work and dedication over the course of the POWER8 project.

References

[1] E. Fluhr, J. Friedrich, D. Dreps, V. Zyuban, G. Still, C. Gonzalez, A. Hall, D. Hogenmiller, F. Malgioglio, R. Nett, J. Paredes, J. Pille, D. Plass, R. Puri, P. Restle, D. Shan, K. Stawiasz, Z. Deniz, D. Wendel, M. Ziegler, "POWER8™: A 12-Core Server-Class Processor in 22nm SOI with 7.6Tb/s Off-Chip Bandwidth", IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014.

[2] M. Cho, V. Kravets, S. Krishnaswamy, D. Kucar, J. Narasimhan, R. Puri, H. Qian, H. Ren, C. Sze, L. Trevillyan, H. Xiang, M. Ziegler, "Converged large block

and structured synthesis for high performance microprocessor designs," US Patent 8271920, 2012.

[3] S. Narasimha, et al, "22nm High-Performance SOI Technology Featuring Dual-Embedded Stressors, Epi-Plate High-K Deep-Trench Embedded DRAM and Self-Aligned Via 15LM BEOL," IEEE International Electron Devices Meeting (IEDM), 2012

[4] S. Taylor, "POWER7+™: IBM's next generation POWER microprocessor," Hot Chips 24, 2012

[5] B. Stolt, Y. Mittlefehldt, S. Dubey, G. Mittal, M. Lee, J. Friedrich, E. Fluhr, "Design and Implementation of the POWER6 Microprocessor", IEEE Journal of Solid-State Circuits, vol. 43, no. 1, 2008.

[6] W.J. Daily, A. Chang, "The role of custom design in ASIC chips," Design Automation Conference (DAC), 2000.

[7] V. Zyuban, S. W. Asaad, T. W. Fox, A.-M. Haen, D. Littrell, J. H. Moreno, "Design methodology for semi custom processor cores," Great Lakes symposium on VLSI (GLSVLSI), 2004.

[8] J. Friedrich, et al, "Design methodology for the IBM POWER7 microprocessor," IBM Journal of Research and Development, vol.: 55, issue: 3, Page(s): 9:1 - 9:14, 2011.

[9] L. Trevillyan, D. Kung, R. Puri, L. N. Reddy, M. A. Kazda, "An integrated environment for technology closure of deep-submicron IC designs," IEEE Design & Test of Computers, vol. 21:1, pp. 14-22, 2004.

[10] H. Xiang, M. Cho, H. Ren, M. Ziegler, R. Puri, "Network flow based datapath bit slicing," Proceedings of the 2013 ACM International Symposium on Physical Design (ISPD), 2013.

[11] H. Xiang, M. Cho, H. Ren, M. Ziegler, R. Puri, "Network flow based datapath bit slicing," US Patent 8566761, 2013.

[12] M. Cho, H. Xiang, H. Ren, M. M. Ziegler, R Puri, "LatchPlanner: Latch placement algorithm for datapath-oriented high-performance VLSI designs," International Conference on Computer-Aided Design (ICCAD), 2013.

[13] M. Cho, R Puri, H. Ren, H. Xiang, M. M. Ziegler, Structured latch and local-clock-buffer planning, US Patent 8495552, 2013.

[14] M. Cho, A. W. Ng, R. Puri, H. Ren, H. Xiang, M. M. Ziegler, "Soft hierarchy-based physical synthesis for large-scale, high-performance circuits," US Patent 8516412, 2013.

[15] K. Ueno, H. Murakami, N. Yano, R. Okuda, T. Himeno, T. Kamei, Y. Urakawa, "A Design Methodology Realizing an Over GHz Synthesizable Streaming Processing Unit," IEEE Symposium VLSI Circuits, 2007.

[16] R. Varada et al, "Superblock: A Method for Synthesizing Large High Performance Designs without Hierarchy Limits," Design Automation Conference (DAC), 2010.

[17] V. Zyuban, S. Taylor, B. Christensen, A. Hall, J. Gonzalez, J. Friedrich, F. Clougherty, J. Tetzloff, R. Rao, "IBM POWER7+ design for higher frequency at fixed power", IBM Journal of Research and Development, Volume: 57 , Issue: 6, Page(s): 1:1 - 1:18, 2013

[18] M. M. Ziegler, V. V. Zyuban, G. D. Gristede, M. Vratonjic, and J. Friedrich, "The opportunity cost of low power design: a case study in circuit tuning," IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2009.

[19] M. M. Ziegler, G. D. Gristede, V. V. Zyuban, "Power reduction by aggressive synthesis design space exploration," IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2013.

[20] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah. Optimization of Custom MOS Circuits by Transistor Sizing. International Conference on Computer-Aided Design (ICCAD), 1996.

[21] J. Neves, A. Matheny, G. Gandham, E. Hughes, R. Averill III, W. Nop, "eFinale – Integration Platform for High Performance," Design Automation Conference (DAC), 2011.