Configurability in IP Subystems:

Baseband Examples

Pierre-Xavier Thomas, Grant Martin, David Heine, Dennis Moolenaar, and James Kim
Tensilica, Inc., Santa Clara, California, U.S.A.
{pthomas, gmartin, dlheine, dennis, jameskim}@tensilica.com

***Abstract*—Configurability in IP subsystems has two major motivations. The first is the requirements of the IP subsystem itself; the second the particular customer requirements, as every customer has unique things they want to change in a subsystem. Configurability manifests itself at two levels - the individual components, such as processors (ideally configurable), memories, and hardware blocks for specialized processing; the second one at the subsystem level, where component choices, interconnect and interfaces may all vary considerably. This paper discusses these concepts applied to practical, real, baseband subsystems for wireless communications. Configurability allows both scalability of a reference IP subsystem - e.g. to handle varieties of standards and use cases; and differentiation, so that customers get the optimal IP subsystem for their unique needs. This is illustrated with existing product-ready systems and cores, and future subsystem concepts that will allow even better scalability, performance, and adaptability for the next generation.**

Keywords—IP, IP Substems, configurable processors, configurable subsystems, baseband

I. INTRODUCTION

The definition and design of subsystem IPs is an art. Such IP must address customer needs in a timely way and with appropriate effort for customer integration in their end products. Design teams look to IP, including subsystems, for functions that they cannot afford or do not have the expertise or time to develop, or are not differentiators for their product.

The more that IP is new to the industry or in a new area (e.g. emerging standards), the greater is the scrutiny of the IP provider and their solutions. Customers must minimize risk and maximize the probability of success; so they look at the quality of technology, the IP provider's track record and credibility including previous success, trust, reliability, and even commercial viability. Customers also want to ensure that IP providers can adapt to their unique requirements, methodologies and design flows, and that they will be available during the whole design, and beyond to future generations of product.

In wireless baseband modem solutions, customers want complete solutions with the ability to differentiate, which is key to product success. An IP solution must provide easy integration and easy customer differentiation via:

- Proprietary algorithms and software
- Unique hardware block design
- Special physical design flows for high performance and/or low power

IP providers must offer subsystem configurability and develop reference design solutions so that customers have quick and efficient integration with sufficient differentiation. However, there is a subtle tradeoff: offering infinite configurability will impede the timely offering of the subsystem IP when the market needs it. Making the right tradeoffs is an art that needs good understanding of target customers.

II. CONFIGURABLE PROCESSORS

The foundation for configurable processor-centric subsystems is configurable, extensible processor technology, which has been developed by a number of academic and commercial groups since the 1990s [1, 2]. Tensilica technology [3] dates from the late 1990s and has been applied to a wide variety of Application-Specific Instruction set Processor (ASIP) designs.

Configurable, extensible processors allow designers to control structural parameters and resources in a base RISC architecture. Extensibility allows design teams to add specialized instructions for applications. Automated tool flows create the hardware and software tools required, using specifications for structural configuration, and instruction extensions, defined in an Architectural Description Language.

Configurable structural architecture parameters include:

- Size of register files
- Endianness
- Adding functional units: e.g.MACs and floating point
- Local data and instruction memory interfaces including configurable load-store units and DMA access
- Cache attributes for instruction and data cache
- System memory and bus interfaces
- Debug, tracing, Jtag
- Timers, interrupts and exceptions
- Multi-operation VLIW operation bundling
- Pipeline depth

978-3-9815370-0-0/DATE13/©2013 EDAA

- Port, queue and lookup interfaces into the processor's datapath.

Instruction extensions, defined in Tensilica's TIE language, define specialized register bank width and depth, special processor state, operations of almost arbitrary complexity, their specification and optimized hardware implementation, encoding, scheduling (single or multi-cycle), usage of operands and register ports, and bundling into multi-operation VLIW instructions.

The automated tool flow generates the tooling for compilers, assemblers, instruction set simulators, debuggers and profilers and other software tools, along with HW RTL and scripts for optimized hardware implementation flows targeting current ASIC technologies.

III. DSPs for Baseband

Configurable, extensible processor technology is used to develop IP for particular applications: e.g. data-intensive ones - audio, video and imaging. An area of growing importance is wireless and wired baseband processing. Rapid evolution of standards – 2G, 3G, LTE, LTE-Advanced; and the need to lessen reliance only on fixed RTL blocks for each standard have sparked interest in "Software-Defined Radio" (SDR). Since not all baseband processing can be done economically in software, a better term might be "Software-Intensive Radio".

To increase flexibility and meet bandwidth requirements within a power envelope for baseband, by using more software and processors, there are three approaches:

- General purpose Digital Signal Processors (DSPs) with a bias towards baseband: complex and real vector signal processing. These are complemented by HW RTL blocks for the most data and power-intensive computations, such as FFTs or despreading.
- Computation-specific accelerators: each major portion of the application is mapped to programmable accelerators (and RTL blocks), without a central DSP. A simple microcontroller might be used for scheduling and control, plus interfacing to the MAC layers (Media Access Control) of the protocol stack.
- A hybrid approach, in which general purpose DSPs are mixed with computation-specific accelerators and RTL blocks, to simultaneously optimize silicon area, energy consumption, and flexibility for changing standards.

One example of the use of configurable processor technology in baseband is the combination of a real and complex vector processor, with specialized instructions for FFT for OFDM (Orthogonal Frequency Dependent Multiplexing) [4]. This DSP combined:

- 3-way VLIW
- 4-way Complex MAC (16 MAC) SIMD instructions
- 8-way real SIMD instructions
- Specialized instructions for FIR, Symmetric FIR, FFT, and various kinds of small matrix multiply
- Options for SIMD vector divide, Reciprocal Square Root (RSQRT) and despreading
- Specialized vector register files and processor state to support special instructions such as FFT
- A rich variety of load and store vector instructions with several different addressing modes
- 20-bit (16 bit integer and fractional with 4 guard bits) and 40-bit (32-bit integer and fractional with 8 guard bit) data types mapped into SIMD vector registers.

This 16-MAC baseband DSP concept has been extended to wider SIMD DSPs (32-MAC and higher [5]). One important lesson learned has been to make the DSP architectural base built on top of the configurable processor base itself highly-configurable, both for SIMD width and number of MACs, and the various options, to allow easier reuse of the architecture.

IV. Accelerators for Baseband

The hybrid strategy for baseband described in the previous section optimizes the mapping of baseband algorithms to a heterogeneous set of DSPs, accelerators and RTL blocks. Baseband accelerators include:

- A DSP for soft-bit stream processing in LTE user equipment receive chains with options for Viterbi decoding and QPSK/16QAM and 64QAM soft-bit de-mapping.
- A DSP for "hard-bit" stream processing in LTE user equipment transmit chains, with an option for 32-bit by 32-bit transpose.
- A DSP for turbo decoding, supporting multiple standards including both LTE and HSPA+.
- A specialized accelerator for 3GPP user equipment de-spreading supporting one-dimensional, multi-dimensional, multi-lag and multi-code despreading requirements. These support channel estimation, early/late functions; initial synchronization, path searching; and HS-SCH de-spreading and Hadamard transforms.
- A specialized accelerator for matrix operations, in particular matrix inversion via direct, QR and Cholesky decomposition for 2x2, 4x2, 4x4 and larger matrix sizes (e.g. 8x8, 16x16), as well as matrix-matrix multiplication and matrix-vector multiplication. There is a considerable energy and performance advantage to be gained, especially valuable in user equipment, by offloading these functions to a specialized accelerator.

At this point, designers have a variety of DSPs and accelerators in their toolkit for building baseband modems. However, they need additional components.

V. Configurable Subsystem Components

Any baseband subsystem using heterogeneous processors will need configurable components such as:

- Specialized RTL blocks where a processor cannot offer enough computation or requires too much energy. FFT blocks are a good example of well-known functions that are more area and power efficient in RTL. Other functions where RTL is common include Turbo decoding (usually single-standard, which is why multi-standard Turbo decoding processors may have advantages) and 3G despreading.
- Advanced component and processor interconnects. These may be manually designed with standards such as ARM AMBA AHB and AXI; or use IP generators for bus-based interconnects. Synopsys, for example, provides bus generators for AMBA AHB and AXI interconnect [6]. In recent years, Network-on-Chip (NOC) interconnect 'fabrics', from IP vendors such as Sonics [7] and Arteris [8], have grown popular. The architect needs to consider several interconnect parameters such as bandwidth, latency, power, performance, physical design flow, memory hierarchy, programming models, communications integration and time to market.
- Memory subsystems for advanced processing requirements have grown very complex. Matching the very different speeds of off-chip memory and on-chip processors and RTL blocks requires complex memory systems. Many levels of cache and advanced memory controllers (e.g. DDR controllers) are commonplace. Alternative approaches to cache include local data and instruction memories on cores, with multi-banked shared and private memories, and a variety of Direct Memory Access (DMA) components.
- Other SoC integration blocks support the control of power-on sequence, boot sequence, generation of interrupts internal and external to the subsystem, communication and synchronization between cores inside and outside the subsystem and multicore debug.
- RTL blocks to interface to analogue and RF front end components and those components themselves (for example DigRF interface blocks).

Such components should be configurable for different uses cases, and may follow a similar IP model to that used for processors. At this point, our inventory of processors, accelerators, and hardware components is reasonably complete and our toolkit almost full. One key lack is software, which can range from stacks for PHY processing on baseband hardware – both reference chains and optimized software for processors – through to the complete MAC (Media Access Control) stack running on a control processor that co-ordinates PHY and higher level processing and provides interfaces to application processors and the high-level control of the complete system.

VI. Single-Core Subsystems

The first level of subsystem is a "single-core subsystem", composed of a single DSP or DSP-style accelerator, together with local memory components and a small ancillary DMA block, with other RTL components, to allow the processor to easily communicate with neighboring single-core subsystems, and with the global memory hierarchy.

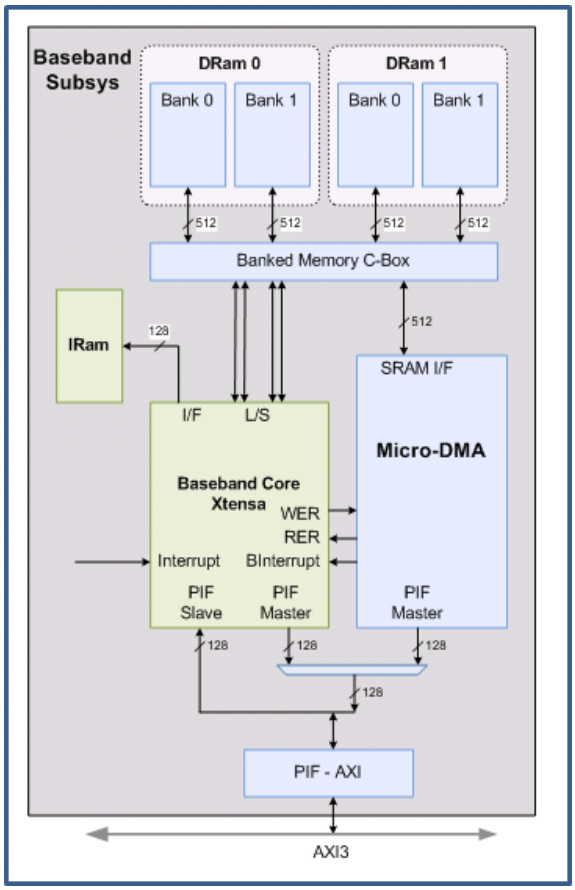

Figure 1. Single-core subsystem

Fig. 1 illustrates a single core subsystem. It contains:

- A baseband processor core
- Configurable micro-DMA
- PIF-AXI interface
- A "connection box" (C-Box) to allow flexible, arbitrated access from the two load/store units on the baseband processor core, to a banked set of data memories. The C-box, together with the compiler, and appropriate placement of data structures in the memory banks, works to minimize stalls and contention.

In this single-core subsystem, the micro-DMA works with the processor core PIF (Processor InterFace) interfaces and the C-box to allow flexible asynchronous DMA transactions to and from system memories, other single-core subsystems, and local data memories accessed via the C- box. Other variations of the configurable micro-DMA can provide direct access to the arbitration local memory network on neighboring single-core subsystems.

A single-core subsystem can become a template for a generator using parameters such as the type of baseband core, local data and instruction memory number, size and width,

system bus choice (PIF, AXI or AHB) and width, and local and global address spaces.

A homogeneous or heterogeneous multi-core subsystem can then be efficiently generated and validated by interconnecting such single-core subsystems with a system bus fabric that can also be adjusted for latency, bandwidth and performance for each channel between the nodes.

VII. Modelling and Verifying DSPs, Accelerators, and Single-Core Subsystems

With configuration and generation of single cores and single-core subsystems come verification issues. There are three distinct levels of verification; the first two are 100% automated and the final one can be automated:

1. Hardware/RTL (Verilog) level
2. System Model (SystemC) level
3. FPGA emulation level

Automated generation technology creates RTL (Verilog) diagnostics and a SystemC wrapper for the core instruction set simulator (ISS) for each core. In addition, beyond each individual DSP and accelerator, the generation process itself is verified through extensive automated verification technology [9]. This includes cosimulation between Verilog and ISS. The SystemC wrapper can be used in system modeling and verification environments as well as third-party ESL tools. It is possible to run examples and real application code on both Verilog and SystemC ISS levels, but for cores meant to be part of a complex multi-core subsystem, this is rather limited – for example, it can cover basic core configuration and sanity, verified in isolation. Finally, individual cores can be placed on an FPGA emulation environment and have appropriate software run, but this is more appropriate for single cores that are complete in themselves (such as for audio applications, running codecs).

It begins to get more interesting with the single-core subsystems. Here the DSP is combined with hardware components such as the micro-DMA, memories, bus interfaces and special registers. At this level, an appropriate Verilog testbench – a "bringup" or "Level One" testbench – is necessary. This can be manually or automatically generated to match the core configuration. It must include sanity connectivity between the components, and basic transactions such as memory accesses and micro-DMA transfers. Software run on the cores will program the micro-DMA and access memories. Bus-functional models for system bus ports can be used in the testbench along with appropriate monitors. It is also possible to extend this kind of testbench from functional to rudimentary performance monitoring, although that is more useful at the higher levels of verification.

VIII. Delivering IP Subsystems

IP subsystems for baseband modems must be delivered to allow easy integration of architecture, hardware, software, and easy verification in the complete system. Deliverables include:

- RTL
- Integration testbench and tests to demonstrate the basic functionality, reset, and boot sequence
- Documentation (user, installation and programming guides)
- Implementation scripts and constraints for synthesis, placement and routing
- Software libraries (general math, communication standard specific, communication and synchronization)
- SystemC model for integration
- Integrated software examples
- Integrated standard communications software

Ease of use and integration is important and must enable the customer to differentiate. Integration of customer hardware functions or third party IP is facilitated by offering the needed interfaces. These are characterized by latency, bandwidth, and topology. Subsystem IP must have the right interfaces to allow connection of such hardware to their internals. Verification of connections at the architecture, logical and functional levels requires a number of views, including high level modeling.

To facilitate integration of customer and third party software stacks, the programming model(s) of the modem need to be exposed and easy to use. This requires providing software communication and synchronization functions.

IX. Complete Baseband Subsystem Composed of Single-Core Subsystems

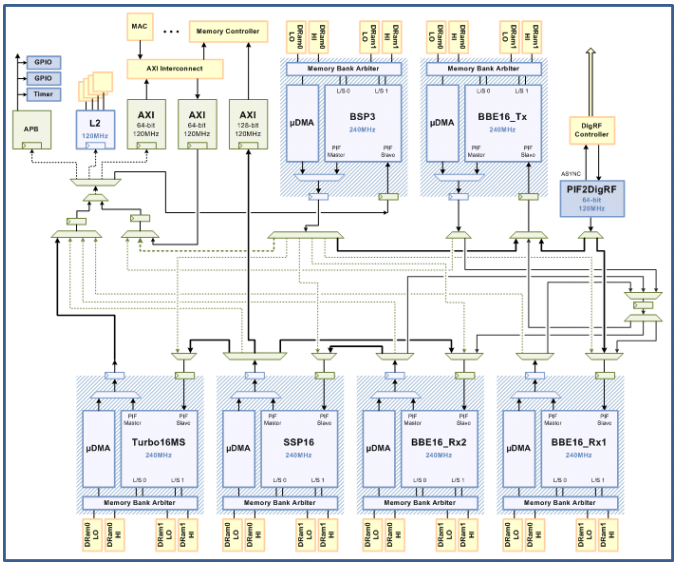

Figure 2. Atlas baseband subsystem composed from single-core-subsystems

We developed a complex baseband subsystem, "Atlas", for user equipment LTE Category 4; this has been used for real products. This subsystem (Fig. 2) consists of six single-core subsystems, and additional components. The single core subsystems include 3 16-MAC baseband DSPs, two in the receive chain and one in the transmit chain. The receive chain

also includes a soft bit DSP and a multi-standard turbo decoder DSP. The transmit chain includes a hard bit DSP processor before the complex domain processing using the baseband DSP.

Several other components are used in the baseband subsystem: NOC interconnect, AXI bus interfaces to MAC, memory controllers and HARQ buffer, memory interfaces to L2 cache, an RF Interface block, and an interface to APB for relatively low rate peripherals.

X. MULTICORE CLUSTERS

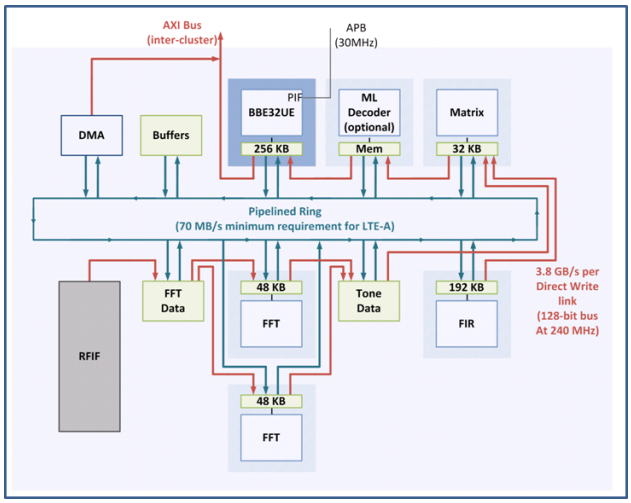

Figure 3. Multicore Cluster

A more advanced subsystem concept has been developed that advances the idea of a single-core subsystem into a multicore cluster. Such clusters can range in complexity from two cores to many cores (Fig. 3). A multicore cluster consists of a processor core, DMA component, local data memories and a variety of tightly bound accelerators. Figure 3 shows a DSP with several accelerators: FFT/iFFT/DFT blocks; a FIR accelerator; a Matrix Accelerator; and an optional ML decoder. Thus the complexity of a cluster can range quite widely.

A multicore cluster is responsible for part of the signal processing in the receive and transmit channels of a baseband modem. Data transfer between cores is very specific, high bandwidth and low latency, and is optimized for power, needing optimized interconnect on the local memory bus. Interconnect between clusters uses the system bus, and can usually tolerate the bus protocol overhead, in exchange for easier integration with the rest of the SoC.

XI. DESIGNING, MODELLING AND VERIFYING MULTICORE CLUSTERS

Design and verification issues begin to be interesting at the cluster level. At this level, several system-oriented issues become important, including:

- Resets – does each core get reset separately, or do we have a master DSP in a cluster which is reset, holds the other accelerators in runstall, dynamically loads the reset vector for every other core and resets them one by one? The second approach allows the local memory space for reset vectors on the slaves to be re-used during normal operation.
- Synchronisation and interrupts – strategies include using Memory-Mapped IOs (MMIO) devices which trigger interrupts for synchronization, memory-based using semaphores, or synchronizing via FIFO queues.
- Basic programming models – in dataflow systems we want to use run-to-completion semantics as much as possible and avoid interrupts, and their context switch overhead. With counter-based synchronization cores can check for work before going to sleep, waiting on an interrupt, maximizing useful work. This approach was used earlier in a multicore video system [10].

Verifying a multicore cluster in isolation from the complete subsystem needs a "Level One" testbench for interconnect sanity and basic transaction functions, as described earlier. In addition, a basic performance testbench at Verilog or SystemC levels, that checks bandwidth and latency of transactions using abstract task models and real communications, all modeled with accurate latencies, is used to validate and iterate system performance assumptions. In particular, shared memory arbitration and timing is important to get right early.

It is important in cluster design to constrain the component/core configuration space to allow integration. Not all elements of core configurability should be used, so that components or nodes in a cluster can be configured to work together based on core parameters and the integration process, including RTL and model generation, can become automated.

XII. COMPLETE BASEBAND SUBSYSTEM COMPOSED OF MULTICORE CLUSTERS

A complete baseband subsystem can be built with multicore clusters, as illustrated in Fig. 4 for user equipment Category 7 LTE-Advanced. This subsystem is composed of three clusters: for receive a complex domain processing cluster (Fig. 3), with a 3GPP Despread accelerator for multistandard support. Complex samples from the RF sections are converted into a soft-bit stream. The soft-bit cluster uses a soft-bit DSP (SSP16) and three accelerators for Viterbi, data shuffling (interleaving and transpose), and turbo decoding.

Transmit uses a hard bit stream cluster, and reuses the complex domain cluster, since processing requirements for receive are much greater than transmit. The hard-bit cluster has a specialized DSP (BSP3) and a turbo interleaver.

Cluster-based subsystems require three levels of configuration and build:

1. Configure DSPs and accelerators.
2. Configure ancillary components such as micro-DMA and build clusters.
3. Configure and interconnect cluster interfaces and system interfaces, e.g. to global L2 memory.

Configurable technology is being developed to automate the building of such complex subsystems, and to reduce the risk and effort involved.

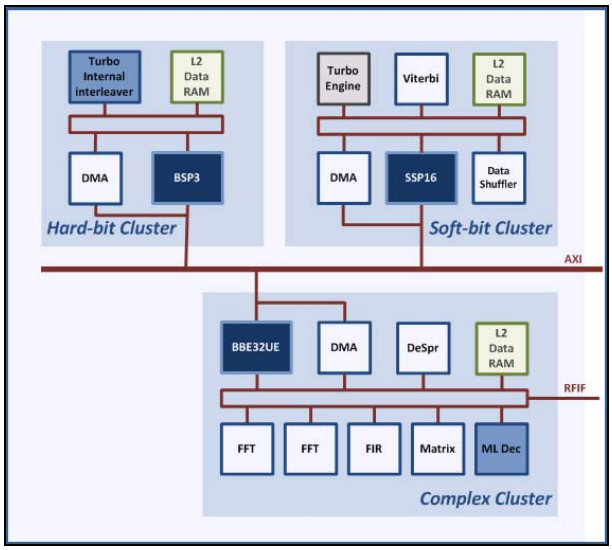

Figure 4. Complete baseband subsystem composed of multicore clusters

XIII. SELECTING, INTEGRATING AND VERIFYING COMPLETE SUBSYSTEMS

When designing complete subsystems, new issues come to the fore. We must select the full subsystem interconnect. IP choices include NOC and traditional bus based styles from a variety of vendors. The IP approach to interconnect is both configurable and pre-verified in many ways; it should lower risk and effort compared to internal design. The quality and degree of technical support offered by interconnect IP providers is an important criterion in selection, especially in first time usage. In general, third-party interconnect IP still needs additional work – lack of starting templates, first level testbenches, and useful physical design flows are all hindrances to effective first-time use.

The "Level One" testbench for sanity hookup and basic transaction function verification is important at the full subsystem level. It is also vital to evolve Verilog and SystemC testbenches to Level Two. This adds instrumentation and bus functional models, and uses software as traffic generators, perhaps abstract tasks, to generate interconnect traffic and measure realistic bandwidth and latencies. This is required for several purposes:

- Architectural exploration of the system and iterating configuration parameters for cores, clusters, and interconnect.
- Making tradeoffs of low power vs. maximal performance, and optimizing key characteristics such as the interconnect network size. The initial design tendency is to oversize everything.
- Developing system level models, usually SystemC, for third party software developers, to port and optimize PHY software stacks on the subsystem.

FPGA prototyping boards for demonstrations, evaluations and software development are another target. Capacity limits on FPGAs may require a "thinner" version of the system. As subsystems grow larger, larger FPGAs are used until it is necessary to partition across multiple FPGAs. In general, this is a heavily manual process.

XIV. CONCLUSIONS

IP subsystems, and the degree of configurability they require, are a hot topic in the industry [11]. Configurable, extensible processor technology is a fundamental capability to create DSPs and accelerators for baseband. When combined with other components, these DSPs can be built into single-core subsystems. Composing them allows support of various baseband standards. A more modern approach is to build multi-core clusters of DSPs and accelerators which can be composed into full subsystems.

One might expect that IP subsystems are always fixed solutions for complete system functions, offered in a pre-integrated, pre-verified form. But customers have widely varying goals, and a strong need to differentiate. IP-based subsystems that are totally fixed are not useful. Especially when the base components are highly configurable, customers require a very high degree of flexibility in the subsystems they use - even when working in a relatively known domain such as wireless baseband.

REFERENCES

[1] Paolo Ienne and Rainer Leupers, Editors, Customizable Embedded Processors, Burlington, MA: Morgan-Kaufmann Publishers, 2007.

[2] Prabhat Mishra and Nikil Dutt, Editors, Processor Description Languages. Burlington, MA: Morgan-Kaufmann Publishers, 2008.

[3] C. Rowen, Engineering the Complex SOC: Fast, Flexible Design with Configurable Processors. Upper Saddle River, NJ: Prentice Hall, 2004.

[4] Chris Rowen, Peter Nuth, Stuart Fiske, Marcus Binning and Sam Khouri, "A DSP architecture optimised for wireless baseband", International Symposium on System-on-Chip. Tampere, Finland, 2009.

[5] Chris Rowen, "Power/Performance Breakthrough for LTE Advanced Handsets", 2012 Linley Mobile Conference, April 16, 2012.

[6] http://www.synopsys.com/IP/SoCInfrastructureIP/DesignWare/AMBA/Pages/default.aspx

[7] Drew Wingard, "MicroNetwork-Based Integration for SOCs", DAC 2001, 673-677.

[8] Philippe Martin, "Design of a Virtual Component Neutral Network-on-Chip Transaction Layer", DATE 2005, 336-337.

[9] Marines Puig-Medina, Gulbin Ezer and Pavlos Konas, "Verification of Configurable Processor Cores", DAC 2000, 426-431.

[10] Gulbin Ezer, "MPSOC flow for multiformat video decoder based on configurable and extensible processors", GSPX, 2006.

[11] Eric Esteve, "Subsystem IP, myth or reality?", SemiWiki.com, 7 December 2012, http://www.semiwiki.com/forum/content/1884-subsystem-ip-myth-reality.html.