USING INTEL® ARCHITECTURE FOR IMPLEMENTING SDR IN WIRELESS BASESTATIONS

John Mangan (Intel, Shannon, Ireland; john.mangan@intel.com)
Rajesh Gadiyar (Intel, Portland, OR; rajesh.gadiyar@intel.com)
Kannan Babu Ramia (Intel, Chandler, AZ; kannan.babu.ramia@intel.com)
Niall Power (Intel, Shannon, Ireland; niall.power@intel.com)
Terence Nally (Intel, Shannon, Ireland; terence.nally@intel.com)

ABSTRACT

The requirements for Wireless Basestations are rapidly evolving as more and more devices are becoming connected and Mobile Broadband has become a reality. Enabled by standards such as LTE and WiMax and technologies like OFDM and MIMO the throughput and capacity needs are increasing exponentially. At the same time Wireless Basestations are required to simultaneously support multiple radio standards and meet many different deployment and use case scenarios. It is becoming increasingly difficult for current solutions based on multiple application specific architectures to meet these demands while maintaining software and engineering investment

Software Defined Radio (SDR) has become a key element in introducing flexibility in recent solutions, however many obstacles still remain towards a single architecture and a 100% reuse nirvana.

This paper describes some of the Intel® Architecture developments and how it is becoming a practical solution towards running all Wireless Basestation workloads on the same platform. In particular we will focus on the elements that have evolved in the architecture that now allows SDR workloads to match and in many cases exceed the performance of current solutions. We provide proof points for some of the latest LTE signal processing workloads and also show how the development cost can be significantly reduced and maintained across multiple platforms and follow-on silicon generations.

1. INTRODUCTION

The pressures on modern Wireless networks to carry more and more data at a more competitive price point is increasing as subscribers demand a mobile internet experience similar to what is available from their home network. The telecommunication industry is challenged to respond to the explosive increase of data carried on modern wireless networks while dealing with a steady reduction in revenue earned per data bit.

The growth of these networks and evolution to higher capacity does not have a single path, with service providers moving from one generation to another or skipping generation's altogether, sometimes deploying overlay networks or planning on consolidation on a single network. Software Defined Radio has a critical part to play in offering flexible basestation deployment and the ability to adapt to the dynamic networks requirements in the future.

Over the years Telecommunication Equipment Manufactures (TEMs) have turned to multiple different silicon architectures to meet the specific workload demands of Basestations. For example we could have Intel® Architecture handling application and control workloads, a Network Processor running packet processing and signal processing on a number of DSPs. Though providing high performance to isolated workloads, this approach increases the complexity and overall cost of both the basestation hardware and software and limits portability and reuse between generations. How do you best design a basestation supporting more than one service provider's 2G, WCDMA and LTE services?

In 1991 Joseph Mitola coined the term "Software Defined Radio", describing it as "A software radio is a radio whose channel modulation waveforms are defined in software…" he goes on to define the extraction, down-conversion and demodulation of the channel waveform is done "using software on a general purpose processor"[1]. The concept of moving all digital workload to a general processor provides the greatest flexibility for SDR design.

In this paper we will discuss the evolutionary Signal Processing capabilities of Intel® Architecture and how the architecture which has a strong history in Application and Control (Packet processing capabilities are discussed elsewhere) workloads has become a real option for executing signal processing workloads. There are many advantages to consolidating the workloads on a single standard architecture such as the support of a large ecosystem, reduction in design complexity, rapid time to market and software reuse. These advantages of designing a basestation on a single scalable architecture provide the platform for fully functional Software Define Radio design. The latest Intel® Microarchitecture (Nehalem) allows the design process to be significantly streamlined and cost optimized.

2. INTEL® MICROARCHITECTURE (NEHALEM)

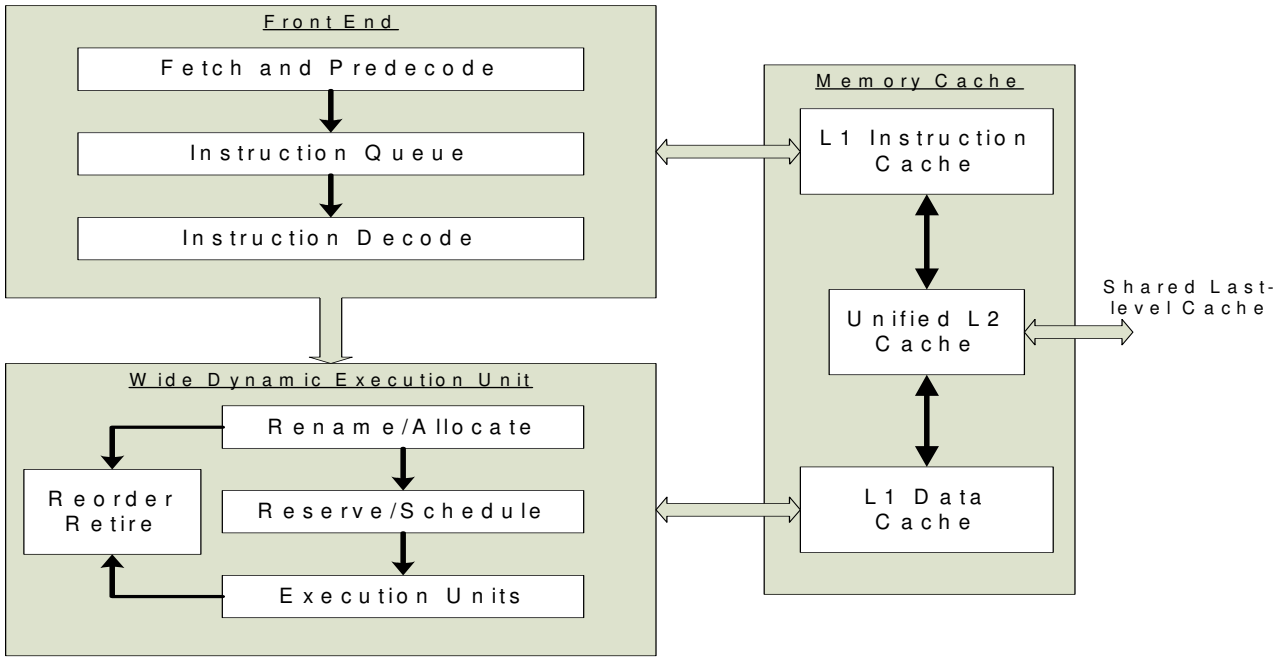

Figure 1 Intel Microarchitecture (Nehalem) simplified block diagram

The latest generations of Intel® Architecture processors are produced on 45nm and 32nm process technologies and are based on the Intel® Microarchitecture (Nehalem). This architecture includes many features that suit high performance and power efficient execution of signal processing workloads. To help understand how this architecture is effective a simplified block diagram is shown in Figure 1.

The front end fetches and decodes instruction streams through the L1 cache and can provide a sustained throughput of 4 decoded micro-operations (uops) per cycle. Up to 16 bytes of instructions can be fetched per cycle and the unit can support decoding instruction streams for 2 hardware threads in alternate cycles (known as Simultaneous Multithreading – SMT). Multiple performance and power enhancements are integrated such as branch prediction, loop caching & streaming, op code fusion and stack pointer tracking.

The uop stream then enters the first stage of the Intel Wide Dynamic Execution unit. This is an out-of-order superscalar execution core that can issue up to 6 uops per cycle and can have up to 128 uops in flight at any moment. The reservation station schedules which uops are issued for execution and an entry per uop in the reorder buffer of the retirement unit ensures the processing of the operations and architectural states are updated according to the specified program order.

Of the six issued uops per cycle, 3 can be related to computational operations and 3 to memory operations (up to 128-bits each). Please refer to Figure 2 for breakdown. For signal processing workloads this means that if SIMD (single instruction, multiple data) operations are used, the architecture is capable of supporting in a single cycle up to 12 compute and 3 memory I/O operations. These operations may be a combination of 4 x 32bit FP MULTIPY, 4 x 32 bit FP ADD, an SIMD shuffle or integer ALU op, a load, an address store and a data store.

To support high instruction throughput the Intel® Microarchitecture (Nehalem) contains a sophisticated memory sub-system. Each core contains a first level instruction cache (32KB 4way), a first level data cache (32KB 8way), a second-level unified cache (256KB 8way) and a last level cache of up to 8MB 16way that is shared amongst all the processor cores. With 3 DDR3 memory controllers the processor can provide a peak memory bandwidth of 25.6 GB/s. This high throughput capability is required to support the multi-gigabit rates for the processing of the sample streams from modern Wireless multi-antennae standards.

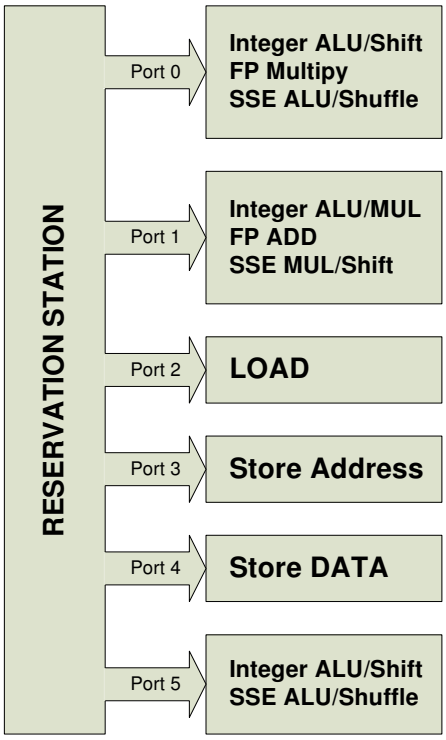

Figure 2 Execution Units

3. INTEL® ARCHITECTURE INSTRUCTION SET CAPABILITY

This section looks at some of the details of the Intel® Instruction Set Architecture (ISA) in relation to efficient implementation of signal processing requirement for SDR. Support for high throughput signal processing is based on the SSE (Streaming SIMD Extensions) which are part of the standard Intel® ISA [2][3]. Up to the current generation (4.2) more than 300 new instructions have been added to the ISA. SSE operations work from a set of 16 128-bit wide XMMx registers. The main categories covered for optimized parallel execution are:

- Floating point (single and double precision)
- Data Transfer
- Packing and unpacking
- Comparison and Logical
- Shuffle & Streaming
- Data Conversion

It should be noted that for many signal processing algorithms the IA implementation in floating point is faster and more accurate than the performance of a fixed point implementation due to high FP capacity and avoidance of range checking.

Some of the SSEn instructions applicable to signal processing are shown in Table 1.

Sub Group	Instructions	Usability Description
Load and Store Floating Point (single/double) precision	MOVAPS, MOVAPD, MOVHPS, MOVHPD, MOVLPS, MOVLPD	Data Movement instructions. Load and store packed single/double precision floating point value from/to memory/registers. They are used extensively in FFT and other single processing algorithms
Arithmetic Floating Point (single/double) precision	ADDPS/PD SUBPS/PD MULPS/PD SQRTPS/PD, ADDSUBPS, ADDSUBPD	These Arithmetic instructions are heavily used in signal processing algorithm like FFT, FIR etc.
Floating Point Dot Product	DPPS, DPPD	Improved performance for AOS(Array of structures) data processing through support for single and double precision dot products. Usages can be found in channel estimation algorithms.
Floating Point Round	ROUNDPS, ROUNDSS, ROUNDPD, ROUNDSD	Efficiently rounds the scalar and packed single and double-precision operands to integers, with enhanced support for various language requirements.
Packed Test and Set	PTEST	Faster branching from SIMD decisions to support conditionally vectorized code.
Accelerated searching and pattern recognition of large data sets	POPCNT	Calculates the number of bits set to 1 in the given operand. Often used for schedulers and buffer/memory management
Thread synchronization	MONITOR, MWAIT	Places Processor in an optimized state until write to the monitored address range occurs. SDR performance is dependent on efficient threading
Memory Barriers	SFENCE, LFENCE, MFENCE	Insures a performance efficient way of store and load memory ordering. Benefits multitasking programming where tasks execute in the out of order core. Typical use cases are implementing the inter thread/process communication mechanisms like queues and shared memory.

Table 1 Common SSE instructions used in SDR applications.

4. SIGNAL PROCESSING – EXAMPLE

This section describes the performance of two of the most common signal processing algorithms, the Fast Fourier Transform (FFT) and the Finite Impulse Response (FIR) filter.

The FFT implementation used in this example is a version that is included in the Intel Performance Primitive (IPP) library. The Intel IPP libraries contain a wide range of functions that are optimized implementations of common algorithms. and take advantage of Intel® ISA features such as SSE. Many signal processing functions are included such as filtering, convolution, transforms, windowing, sampling and array operations. The latest implementation is optimized for Intel® Microarchitecture (Nehalem) [4]. Other optimized software is available from the ecosystem such as the open source FFTW libraries. A large range of intrinsics may also be used to access specific instruction extensions.

This example uses 32-bit single precision complex floating point samples. The FFT is implemented for different sizes and the number of cycles per sample has been measured. Figure 3 shows the profiled results using a single thread on an Intel® Microarchitecture (Nehalem) core running at 2.67 GHz (Core i7 platform).

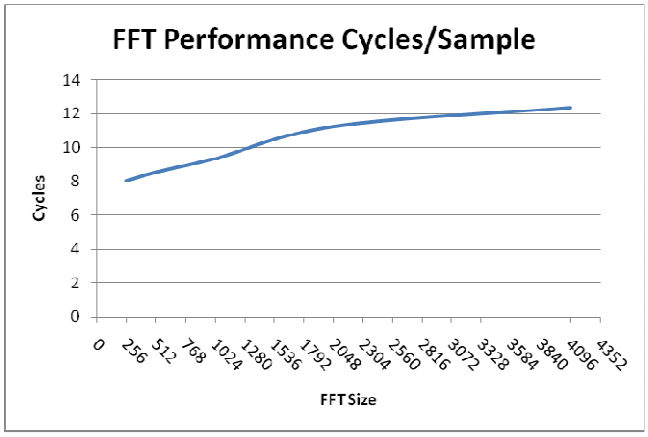

Figure 3 FFT Cycles per sample

The complexity of a N point FFT can be described as an order of $O(N.\log_2 N)$ complex multiplications and additions. The IPP implementation uses a complex multiplication taking 6 operations (2MUL & 2ADD) and a complex addition takes 2 operations (2ADD) for each point (Note: a MUL takes 4 operations). This amounts to $8N.\log_2 N$ floating point operations (FLOPs). By calculating the number of FLOPs per cycle the sustained GFLOP performance can be derived. Figure 4 shows that a single core is capable of between 20 and 30 GFLOPS for FFT execution which is up to more than 90% of theoretical capability.

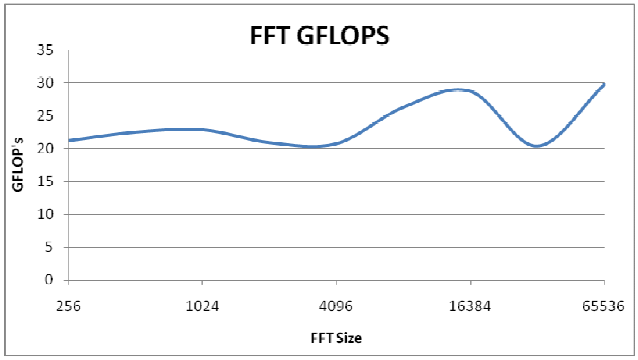

Figure 4: FFT Performance GFLOP's

The second Signal Processing example focuses on the FIR filter. Similar to the FFT example the FIR example is implemented using the Intel IPP functions. The results shown are for a 64 tap FIR filter using single precision 32bit complex floating point samples and coefficients. Figure 5 shows the profiled results using a single thread on an Intel® Microarchitecture (Nehalem) core running at 2.67 GHz (Core i7 platform).

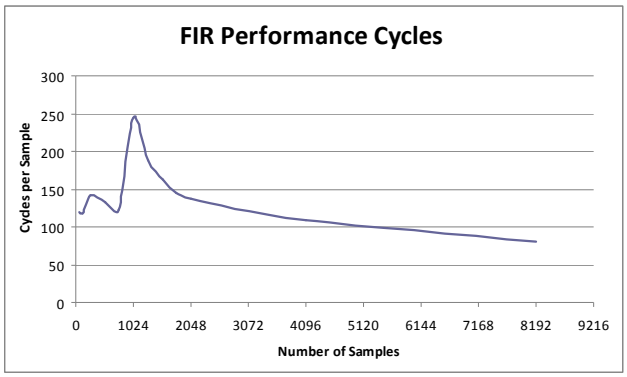

Figure 5 FIR filter, cycles per sample

The implemetation of the FIR filter for 64 taps requires 64 additions and 64 multiplications. This equates to a total of 512 floating point operations per sample. By calcualting the number of FLOP's per cycle the sustained GFLOPS Performance can be derived.

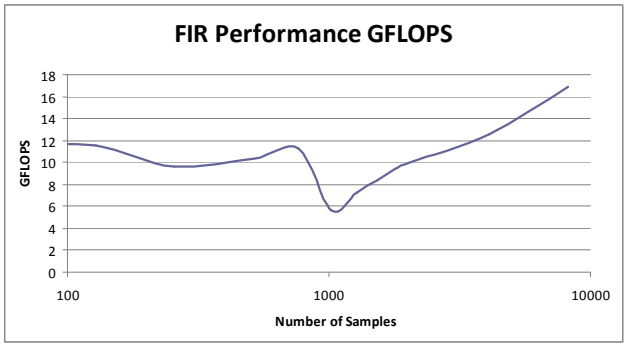

Figure 6 FIR filter GFLOP performance

5. PARALLELIZATION

Effective SDR implementation requires efficient use of all the resources on a processor platform. It is essential to be able to parallelize algorithms across multiple cores in a linear manner. This section presents an example of the parallelized scaling across multi-cores of an Intel® Microarchitecture (Nehalem) based platform. The example used here is complex multiplication, a common operation used in signal processing. The same approach can be applied to the parallelizing of more computationally intense algorithms. The implementation used in this example is based on the IPP library implementation of complex floating-point multiplication.

A threading model was used to implement the algorithm with parallel execution. The input data is divided into blocks, and each block (or number of blocks, depending on data size) is executed in full in separate parallel threads. This method is assumes no interdependence between blocks.

Some processor architecture parameters are required to be taken into account to optimize performance. These are cache size, cache line alignment and thread affinity. Inter-thread dependencies need to be minimized or avoided altogether.

In the example a single processor Core i7 platform was used to execute the complex floating point multiplications. The processor includes 4 cores and from Figure 7 we see a linear scaling result for the first 4 threads which are assigned to run on separate cores. Going from 4 to 8 threads takes advantage of the symmetrical multithreading (SMT) capabilities of the Intel® Microarchitecture (Nehalem) which supports 2 independent hardware threads per core. While performance here does not scale linearly the benefits are evident.

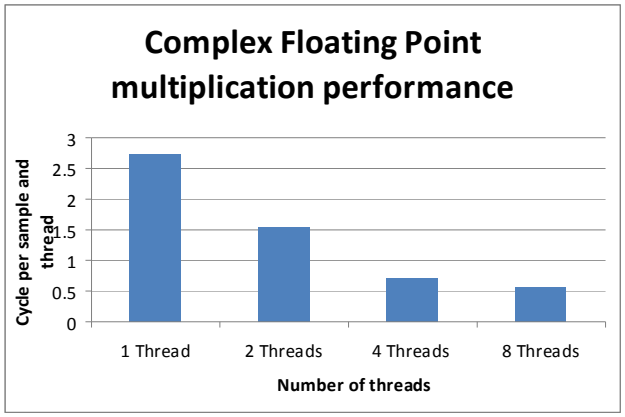

Figure 7: Complex floating point multiplication performance

The complexity of a single floating point complex multiplication is 6 FLOPs, 4 multiplications and 2 additions. C = A.B where C(real)=A(real).B(real)-A(img).B(img) and C(img)=A(real).B(img)+A(img).B(real).

The number of FLOPs per cycle can be calculated and Figure 8 shows the sustained GFLOPS performance based on number of threads used on the processor running at 2.67Ghz. Again, we see linear performance scaling across the 4 cores and an additional benefit when SMT is included.

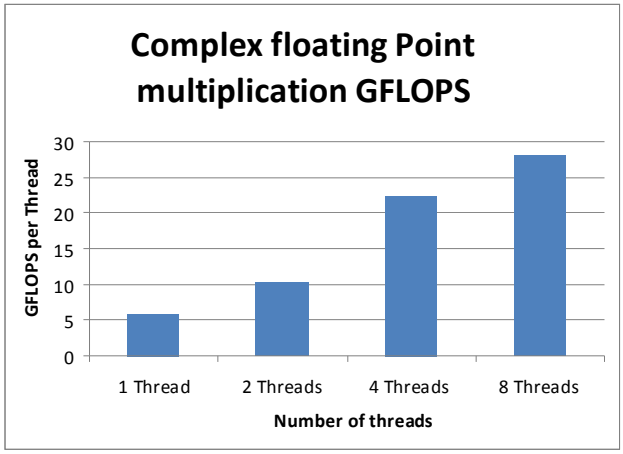

Figure 8: GFLOPS Per Thread

6. WIRELESS WORKLOAD EXAMPLE

In April 2009 Aricent, a global innovation, technology and services company focused exclusively on communications announced that they would create a complete Long Term Evolution(LTE) eNodeB reference solution including Layer 1 signal processing and Layer 2 protocols stacks for Intel® Architecture. The example used in this section is the downlink physical layer processing for LTE 2x2 MIMO. The uplink processing and other protocol stacks have also been completed.

Figure 9 shows the pipeline of the layer 1 downlink implementation. Based on the protocol stacks supplied by Aricent the code was profiled and optimized for Intel® Microarchitecture (Nehalem). The configuration used as the basis for these measurements is the downlink PDSCH channel. This is the highest bandwidth channel used to send data to the user equipment. A 20 MHz channel was used with 2x2 MIMO in transmit diversity mode. The highest modulation 64-QAM was coded and the output of the system's is 4 antenna channels each of 30.72 Msamples/sec of 32-bit (I and Q samples) ready for analog conversion. The signal consists of 10ms radio frames which are broken into 1ms sub-frames and 0.5ms slots. Each slot is divided into 15 kHz sub-carriers which carry 7 OFDM symbols each. For a 20Mz channel 1200 sub-carriers are available which corresponds to 8400 symbols per slot or just over 100Mb/s of raw data when 64-QAM coding is used.

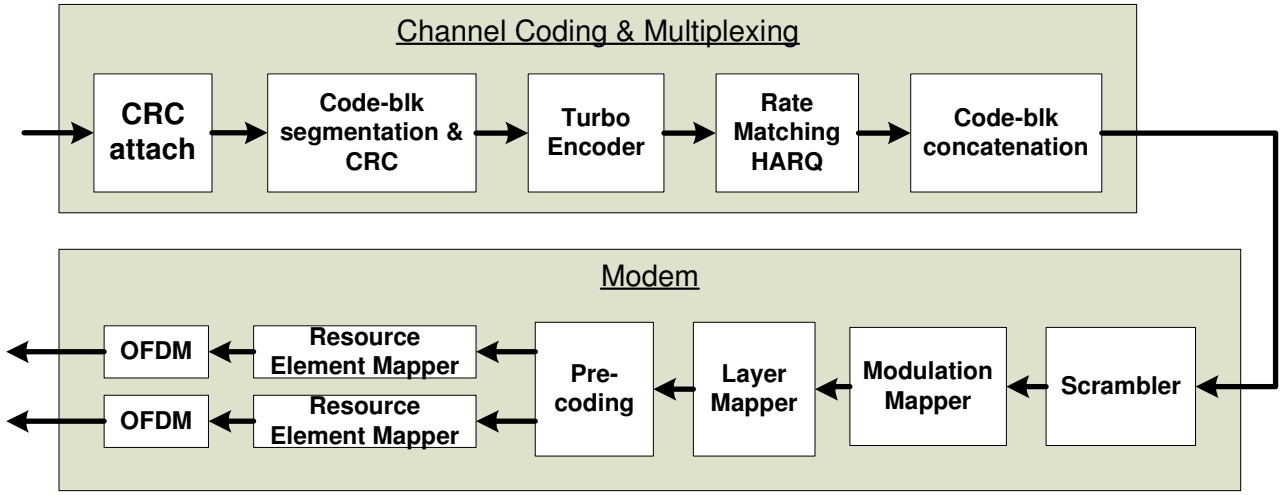

Figure 9 LTE Downlink PHY processing using SDR

The 2 subsystems of the pipeline are the Modem and the Channel Coding & Multiplexing. They consist of a mix of many sophisticated DSP algorithms working either at bit level (ex. Turbo Encoder) or 32-bit single precision floating point (ex. IFFT in OFDM signal generation). The pipeline executes on a single core of an Intel Core i7 platform with hyper-threading enabled. The complete processing of a sub-frame for all 2 antennae is taking 1.87M cycles. This corresponds to 0.7 milliseconds with the processor frequency at 2.67 GHz. With a budget of 1ms to achieve sustained throughput the result shows that a single IA processor core is capable of meeting the requirement with headroom.

The overall results show that the complete uplink and downlink processing for LTE 20MHz + 20MHz 2x2 MIMO can be performed on 3 IA processor cores (ex. Turbo Decoding) with scope to reduce this to just 2 cores in the future.

7. FUTURE TECHNOLOGY

The processors using Intel® Architecture are evolving at a regular yearly beat rate. Known as the tick/tock model the major evolutions of the Microarchitecture are interleaved with process technology evolution. The 2010 'tick' sees the introduction of products based on 32nm technology while the 2011 'tock' introduces the new 'Sandy Bridge' architecture which contains the instruction extensions (AVX [5]) doubling the FLOP performance.

Another technology that is relevant to signal processing is emerging from the Intel Visual Computing Group. Codenamed 'Larabee' it is initially targeting high-performance GPU products. The Intel Embedded Group is evaluating the technology for Wireless Signal Processing. The Larabee 'core' is an IA based processor that contains a new 512-bit wide vector processing unit (VPU). The VPU can handle 16 parallel 32-bit operations (integer or FP) or 512-bit level operations and has a very flexible and comple te set of instructions. Future embedded solution for signal processing may integrate this technology.

8. CONCLUSION

Intel Architecture has evolved significantly over the past few years and has incorporated several micro-architectural, platform and Instruction Set Architecture (ISA) enhancements for much improved Signal Processing. In this paper a selection of SSE instructions have been described and the performance of 2 common Signal Processing algorithms (FFT and FIR) benchmarked. The Wireless LTE Workload example epitomizes how these architectural improvements combined with software parallelization techniques can be used to implement a high performance SDR solution. The prowess of IA for Applications, Control and Data Plane processing is already well established. The Telecom Equipment Manufacturers (TEMs) now have the unique opportunity to consolidate all their workloads – Applications, Control, Data Plane and Signal Processing on a single architecture (IA) thereby achieving a scalable design, reduced software investment and time to market and improved total cost of ownership (TCO).

9. REFERENCES

[1] http://www.sdradio.eu/sdradio/

[2] R.M. Ramanathan, "Extending the World's Most Popular Processor Architecture," *Intel Corporstion White Paper*, http://download.intel.com/technology/architecture/new-instructions-paper.pdf, 2006

[3] http://www.intel.com/products/processor/manuals/

[4] http://software.intel.com/en-us/intel-ipp/

[5] http://software.intel.com/en-us/avx/