

Processor Technology Evaluation

Version 1.0
October 15, 2002
By Aaron Kobayashi

1. Introduction

The microprocessor business is a very competitive one requiring constant innovation and self-reflection. In order for our company to be successful, we need to stand on the shoulders of those who have come before us. To do this, we will be examining two very different architectures—the Transmeta Crusoe and the Intel Itanium paying special attention to their features and mechanisms. The goal of this paper is to compare and contrast these processors from both a performance and practical standpoint while simultaneously gaining a rudimentary knowledge of the mechanisms that make each processor possible.

2. Design Goals and Motivations

Upon examination of these processors, it was apparent that the design goals were different in many ways. The primary design goal of the Crusoe was to design a high performance x86 processor with low power requirements. Furthermore the Crusoe was designed to prove that software-hardware hybrid processors are not only a feasible alternative to the traditional approach, but in many cases much more effective than “throwing hardware” at common problems faced by micro-architects.

The Intel Itanium was designed from the ground up to be a high performance 64 bit processor capable of supporting both workstation and server workloads. This processor was designed to use the EPIC (Explicitly Parallel Instruction Computing) design style to let the software exploit all compiler time information and effectively deliver it to the hardware greatly increasing the instruction throughput. The Itanium was also the first processor to implement the IA-64 ISA.

3. Summary of the Crusoe

The Crusoe chip was designed around the idea that merging software and hardware in the creation of a processor could yield both high performance and low power consumption. The power savings comes from the reduction in the transistor count of the chip but has many interesting side effects such as dynamic recompilation and reduced cooling requirements.

3.1 The Hardware:

At the core of this processor is a simple VLIW execution unit that executes one molecule at a time where a molecule is defined as a group of up to four RISC-like instructions known as atoms. The execution unit is fed molecules by Transmeta’s code morphing software and executes molecules in order to save on complex reorder hardware. Preciseness of interrupts is guaranteed by shadowing all x86 registers. Upon exception, shadows can be restored and instructions can be executed serially to discover which instruction raised the exception. This reduction in hardware requirements frees up transistors on a chip to be used for additional Cache that is not possible on a more traditional processor. Additionally the reduction in transistors means that processors can be implemented on a smaller die raising the processor yield and can substantially lower or remove the cooling requirements!

3.2 Code Morphing

The Crusoe uses software as a solution to many fundamental architectural problems. The code morphing software approach is interesting in that it abstracts the underlying hardware from the compiled program. This abstraction removes the pressure for backwards compatibility that can often be a problem, and allows the hardware designers the freedom to radically change the

underlying hardware without affecting the ISA in any way. Furthermore this software can reorder and optimize code on the fly effectively recompiling and reoptimizing a program at runtime. This saves software creators the hassle of recompiling code to take advantages of a new processor with new hardware features. The Crusoe takes these dynamic optimizations one-step further by caching and further optimizing repeated sections of code. As an added bonus, the Crusoe caches the results of the optimizer for fast access on the next iteration. This cache actually makes the processor seem to get faster with repeated executions of a segment of code. One final example of the advantages of taking a software approach to the problem of microprocessors can be seen in aliasing. Aliasing enables code morphing to move loads above stores safely. This is implemented by saving information about the load or store and checking at a later point to find out if any data dependencies were violated at which point an exception can be raised. Aliasing can be further extended to eliminate redundant loads by using the additional information stored by the hardware.

3.3 Additional Information

This radically different design gives the Crusoe some very unique features that are not possible in other chips of its kind. Due to the fundamentally software approach to the processor, one major enhancement is the ability to upgrade a processor through software. It could be possible to flash your processor in the field if a new algorithm or bug is discovered in processors that have already shipped. The savings could be substantial if a bug is discovered in a processor that has already shipped like the Intel Pentium that had a floating-point problem. A final feature of this chip is its ability to change its clock frequency on the fly to meet the requirements of an application. Traditional processors simply turn themselves off periodically to save power. The Crusoe can dynamically adjust its clock speed and voltage to maximize usage of the power source. This can produce a cubic reducing in power that is significantly greater than the linear savings of more traditional approaches.

4. Summary of the Itanium

The Itanium was designed to be a highly reliable, scalable 64-bit high performance processor that could attain high performance in both server and workstation environments. The key to its power was in the use of the EPIC design style to allow the compiler to communicate hints about program flow to the processor reducing the branch mispredictions incurred.

4.1 EPIC (Explicitly Parallel Instruction Computing)

The Intel Itanium is the first implementation of the IA-64 ISA, and was designed to provide the highest possible instruction throughput by encompassing the EPIC design style. EPIC is essentially a VLIW (Very Large Instruction Word) design that enables compilers to deliver dependency information to the processor with the intention of increasing throughput. This takes shape in the form of branch and memory hints that can be sent to the processor. When a compiler exposes known dependencies, it can explicitly tell the processor how to handle the branch or load. Once an explicit instruction is received it can immediately affect the branch predictor or prefetch instructions from L2 caches into the instruction buffer. The effect of these mechanisms is an increase in throughput through the reduction of branch misses and load stalls. Intel was aiming to get maximum instruction throughput out of this processor and consequently went to great lengths to create hardware support systems to enable high throughput. The Itanium is designed to fetch, issue, execute, and retire six instructions per

clock cycle. It is clocked at 800 MHz and has a 10-stage pipeline to support such high frequencies and has aggressive instruction prefetch and branch prediction. A decoupling buffer allows the front end to speculatively fetch ahead.

4.2 Advanced Speculation and Predication

The Itanium was designed to be a high performance processor that utilizes the EPIC design style to achieve maximum throughput. In order for this goal to be achieved, a processor needs to have advanced hardware support. In the world of microprocessors, speculative techniques are extensively used to speed up processors. To minimize the possibility of branch mispredictions, multiple branch predictors are implemented and a selector mechanism is implemented to track the most accurate predictor. Some predictors take time to “warm up”. Through the use of branch hint directives that can be sent by the compiler, the “warm up” time can also be reduced. Branch predictors are further extended with a multiway branch prediction table that allows the predictor to better handle multidirectional branches. A return stack buffer is used to provide predictions for return instructions. It is eight entries big and stores return addresses along with register stack information. Predicated instructions are another feature of the Itanium and the IA-64 ISA. It allows higher performance by eliminating branches and their associated misprediction penalties. This is very important since the execution core uses stall based strategy to handle data dependencies. In this strategy, instructions can move to the execution stage where they stall while waiting for an instruction. The combination of prediction and this stall-based strategy can also increase overall performance.

4.3 Memory Support Systems

With all of these amazing innovations and high performance support structures, a robust cache and memory system is a necessity. L1 cache is split into an instruction and data cache, each 16K in size and 4-way set associative. The data cache has a load latency of two cycles. The L2 cache is 96K in size and is 6-way set associative. It can handle 2 requests per clock cycle and adheres to a four state MESI protocol for multiprocessor coherence. The L3 cache is 4 Mbytes in size and is 4-way set associative. The L3 cache also is MESI compliant. The chip registers consist of two registers—one integer and one floating point each 128 entries large. The integer register has 8 read ports and 6 write ports and can support two memory and two integer instructions per cycle. To enable low overhead for procedure calls, register rotation is used, a form of register remapping is used.

4.4 Additional Features

The Itanium was designed to be a scalable processor. It has built in support for up to for processors using its multidrop bus to share resources without bridges or other external support hardware. The communication systems implemented are quite advanced and include a cache-to-cache transfer mechanism that includes the ability to allow out-of-order data and transaction completion on its 64-bit 2.1Gbytes/s wide system bus.

5. Comparison of Technology

In this section we will investigate the commonalities and differences between these technologies.

Common Features:

- High performance processors
- VLIW design
- Predication
- Speculative loading
- Compatible with the x86 ISA (IA-32)
- Attempt to utilize software to improve performance

Differences:

- The Crusoe performs with significantly less power and provides real time power adjustment to maximize efficiency.
- Crusoe does not always require active cooling due to significantly less power usage
- The Crusoe translates code on the fly effectively recompiling a program every time it is executed.
- The Crusoe can have more cache while still maintaining a smaller die due to simplified hardware.
- Itanium core executes 3 instructions per word while the Crusoe executes 4
- Itanium uses complex hardware that relies on the compiler to uncover optimizations. The Crusoe uses code-morphing software to optimize at runtime.
- The fundamentally software approach allows possible processor upgrades in the field as well as the ability to retarget the processor to run code from another architecture such as Java bytecode
- The Crusoe “aliases” registers to provide precise exceptions and are extended to allow load instructions to be safely hoisted above store instructions.

6. Summary

In conclusion, it may not be fair to compare these two processors. They were designed with two very different goals in mind. However I believe that while it is still a relatively new technology, the advantages of the hardware-software model can exceed the more developed hardware only approach offered by the Itanium core. Furthermore the ability to upgrade our processors in the field and the higher yields (due to reduced die size) will extend the benefits even further. This design model will also allow us to easily upgrade or change the underlying architecture without having to work around backward compatibility in hardware. It will also allow us to attack a wide variety of markets by enabling scaling of the processor for size and speed requirements while simultaneously making it possible to change the ISA through a simple software change.

References:

- A. Klaiber, "The Technology Behind Crusoe Processors," *Transmeta Corporation*, January 2000.

- H. Sharangpani, and K. Arora, "Itanium Processor Microarchitecture," *IEEE Micro*, September-October 2000, pp. 24-43.