The Cell Processor

- A short Introduction -

Torsten Hoefler
htor@cs.tu-chemnitz.de

28th November 2005

Abstract

Mainstream processor development is mostly targeted at compatibility and continuity. Thus, the processor market is dominated by x86 compatible CPUs since more than two decades now. Several new concepts tried to gain some market share, but it was not possible to overtake the old compatibility driven concepts. A group of three corporates tries another way to come into the market with a new idea, the cell design. The cell processor is a new try to leverage the increasing amount of transistors per die in an efficient way. The new processor is targeted at the game console and consumer electronics market to enhance the quality of these devices. This will lead to a wide spreading, and if everybody has two or more cell processors in TV, game console or PDA, the interesting question comes up: what can I do with these processors? This paper gives a short overview of the architecture and several programming ideas which help to exploit the whole processing power of the cell processor.

1 Introduction

The development of the processor began in 2000, when the three companies Sony, Toshiba and IBM funded a research group to develop a new processor to fit the demands of multimedia applications. Sony had experience in processor design and programming due to the Play Station 1&2. Their vision was to make the Play Station 3 (PS3) around 100 times faster than its predecessor the PS2. Toshiba offered experiences in the field of development and high volume manufacturing and

IBM as the "traditional" processor designer and manufacturer (especially the 90nm SOI technology with copper wiring). They started their research project in a design center in Austin, Texas in 2001 with an investment of more than 400 Mio$. All three companies had future plans for their use of the new technology, Sony wanted to incorporate the Cell into the PS3, Toshiba into HDTV TVs and IBM in server or workstation systems. Their target was to build a high clock-rate and high-throughput multi purpose processor. There are several rationales not to choose x86 but a new design. The memory-latency is quite high in the traditional x86 design, due to the von Neumann bottleneck between CPU and memory. Another negative thing is the increased instruction latency due to the complex instruction scheduling logic of the out of order (ooo) execution and the compatibility layer which translates the ISA (Instruction Set Architecture) CISC (x86) commands to the internal RISC instruction set. The implicit memory hierarchy (caches) costs also much chip space and is hard to control for programmers. All these properties of modern CPUs lead to relatively long control logic paths which limit the clock rate.

1.1 Moore's Law

Moore's Law[1] states that the number of transistors doubles every 18 months. This arises the question, what to do with all these new transistors? Can they make a processor faster? The answer is: no, they can make it "broader". Many processor vendors try to enhance the computing speed by doing work in parallel, essentially doing CPU instructions in parallel. But there are data dependencies which prevent this (if one CPU instruction processes data which is returned by the previous one, it cannot be done in parallel - this is especially bad for streaming codes). There are several "tricks" to find more parallelism in the code, such as super-scalar out of order execution which mainly uses the Tomasulo scheme or speculative execution. But all these schemes introduce higher complexity into the codes and enlarge the critical path. This limits the clock frequency (the longest path has to be reloaded every cycle) and increases the instruction latency. Pipelining helps to mask the instruction latency and increase the clock cycle by splitting the critical path into multiple pipeline stages, but this introduces the problem of pipeline stalls, flushes and startup overhead. All these problems are well known to CPU architects today, and they lead to the effect, that the double amount of transistors leads only to 1.4 times performance (P. Gelsinger, Intel). This is effectively a big waste of transistors

[1] actually not Moore's Law, but it is commonly cited as his law ;)

every year. New concepts such as dual core have been introduced to do something more useful with these transistors and giving the task of parallelizing the code to the programmer. But this concept is also limited if it is based on the x86 architecture. The Cell processor is more flexible and can efficiently be used as a stream processing engine (that's why IBM calls the Cell processor "Broad Band Engine") and a general purpose CPU. The design is simplified to reach very high clock rates and decrease the "transistor waste".

2 Cell Architecture

The Cell architecture was patented by Ken Kuturagi (Sony) in 1999. He differentiated between software and hardware cells. A software cell is a program with the associated data, and the hardware cell is an execution unit with the capability to execute a software cell. He stated no fixed architecture (the software cells could float around and can be processed at any available hardware cell), which is very interesting combined with the ubiquitous computing idea. Several of these hardware cells can create a bigger cell computer (Ad Hoc). But this is only the architectural vision, a real life cell processor is bound to a single chip and floating cells are still only fiction. But the general principle remains the same. A cell processor consists of a so called PPE (PowerPC Processing Element) which acts as the main processor to distribute tasks (software cells), a MFC (Memory Flow Controller) which interfaces between the computing and memory units, many SPE's (Synergistic Processing Elements) which are the hardware cells with their own memory. The Cell CPU is essentially a combination of a Power Processor with 8 small Vector Processors (cmp. Co-Processors). All these Units are connected via an EIB (Element Interconnect Bus) and communicate with peripheral devices or other CPU's via the FlexIO Interface. Unfortunately, the Cell design incorporates hardware DRM features (Sony?), but the positive aspects outbalance this easily. A single Cell, essentially a Network on Chip, offers up to 256 GFlop single precision floating point performance. A block-diagram of the processor is shown in figure 1.

2.1 Prototype

A prototype was produced with 90nm silicon on insulator (SOI) technology with 8 copper layers (wiring). It consists of 241 Million Transistors on 235 mm^2 and

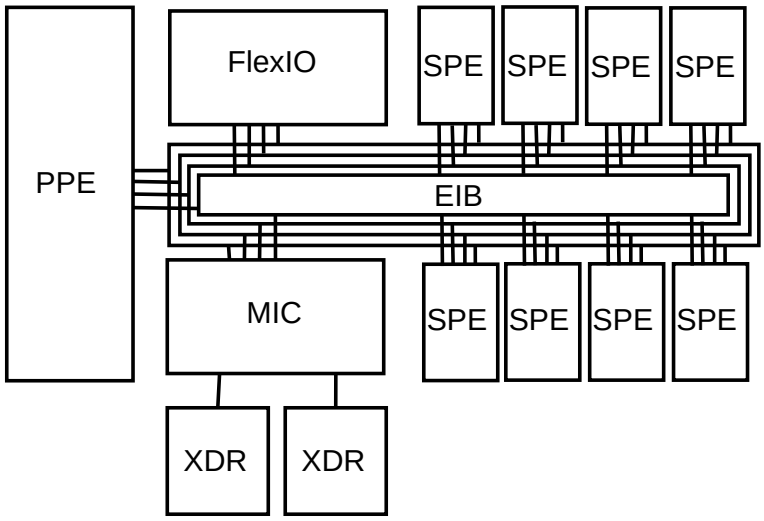

Figure 1: Cell Architecture

consumes 60-80W. IBMs virtualization technology is incorporated in the PPE. The CPU is clocked with 4GHz at 1.1V.

2.2 The Power Processing Element

The Power Processing Element (PPE) offers the normal PowerPC (PPC) ISA. It is a dual threaded 64 bit power processor which includes VMX (aka Altivec which is comparable to SSE). Its architecture is very simple to guarantee high clock rates. Thus, it uses only in order execution with a deep super scalar 2-way pipeline with more than 20 stages. It offers a 2x32kB L1 split cache, a 512kB L2 cache and virtualization. Altogether the PPE can be seen as a simplified Power processor.

2.3 The Synergistic Processing Element

The SPE is essentially a full blown vector CPU with own RAM. Its ISA is not compatible to VMX and has a fixed length of 32 Bit. Current SPEs have about 21 Million Transistors where 2/3 of them are dedicated to the SRAM (memory). The processor has no branch prediction or scheduling logic, and relies on the programmer/compiler to find parallelism in the code. As the PPE, it uses two independent

pipelines and issues two instructions per cycle, one SIMD computation operation and one memory access operation. All instructions are processed strictly in-order and each instruction works with 128 Bit compound data items. 4 single precision floating point units and 4 integer units offer up to 32GOps each. The single precision floating point units are not IEEE754 compliant in terms of rounding and special values. The single precision units can also be used to compute double precision floating point numbers which are compliant to the IEEE754 standard. But their computation is rather slow (3-4GFlops). The schematic architecture of a single SPE is shown in figure 2. The memory layout of the SPE is also quite special, each SPE has it's

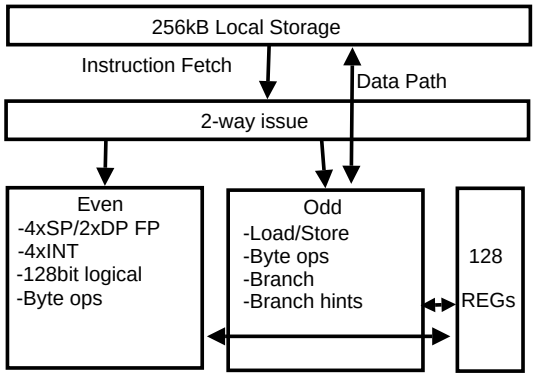

Figure 2: SPE Architecture

own 256kB RAM which is called Local Storage (LS). This SRAM storage can be accessed extremely fast in 128 bit lines. Additionally, each SPE has a large register file of 128 128 bit registers which store all available data types. There is no cache, virtual memory support or coherency for the Local Storage, and the data can be moved with DMA from/to main memory via the EIB. The Memory Flow Controller (MFC) acts like a MMU in this context and provides virtual memory translations for main memory access.

2.4 The Element Interconnect Bus

The EIB is the central communication channel inside a Cell processor, it consists of four 128 bit wide concentric rings. The ring uses buffered point to point communication to transfer the data and is therewith scalable. It can move 96 bytes per cycle and is optimized for 1024 bit data blocks. Additional nodes (e.g. SPEs) can be added easily and increase only the maximal latency of the ring. Each device has

a hardware guaranteed bandwidth of 1/numDevices to enhance the suitability for real time computing.

2.5 The I/O Interconnect - FlexIO

The I/O Interconnect connects the Cell processor (the EIB) to the external world, e.g. other cell processors :). It offers 12 uni-directional byte-lanes which are 96 wires. Each lane may transport up to 6.4GB/s, which make 76.8 GB accumulated bandwidth. 7 lanes are outgoing (44.8 GB/s) and 5 lanes incoming (32 GB/s). There are cache coherent (CPU interconnect) and non coherent links (device interconnect) and two cell processors can be connected glueless.

2.6 The Memory Interface Controller

The MIC connects the EIB to the main DRAM memory, which is in this case Rambus XDR memory which offers a bandwidth of 25.2 GB/s. The memory is ECC protected which shows that the cell will be used for more than game consoles and consumer electronics (this could also be for better EMC protection). The MIC offers virtual memory translation to the PPE and the SPEs. The memory itself is not cached, only the PPE has an own cache hierarchy.

3 Cell Programming

The programming of the cell will be as special as the architecture. The big advantage is that there is no abstraction layer between an external ISA and the internal core (cmp. x86). But the strict RISC design moves the effort to generate optimal code up, to the programmer or compiler. And the general problems of parallel or streaming application development stay the same as for multi-core or multi processor machines. The SPEs are programmed in a direct manner, as own autonomous processors with their 256kB Local Storage and 128 Registers. An Assembly language specification is available from IBM but higher level languages such as C/C++ are also supported for application development. The task distribution and allocation of SPEs is fully done in software. The operating system could use them as a shared resource and virtualize them for many tasks (each task sees their own SPEs, in the whole more

than available). The PPE is programmed like a standard PPC970 and Linux is running as is (without SPE support, but a patch is available from IBM). The SPEs can be used in different scenarios. A job queue can be used to processes a fixed amount of independent jobs as fast as possible, the SPEs could also be used in a self multitasking manner, as if the cell were a multi-core or multi CPU system. Stream processing (Pipelining) is also supported and especially very reasonable for media data (e.g. HDTV). The Local Storage could be used as a cache, but has to be managed by the software for this task. Additionally, MPI support is thinkable, where each SPE is a single node. All these different programming models are just ideas, the future will show which models will be used on the new Cell processors.

4 Conclusions

The Cell processor has a new design and consists of a single Power CPU and 8 additional vector processors. The Single Instruction, Multiple Data approach is used together with a strict RISC instruction set. The extremely fast I/O and memory subsystems allow high bandwidth communication with memory and other processors or devices. The hardware offers also real time capabilities which can be used in combination with HDTV stream programming. Altogether, the new architecture could be the standard of tomorrow and is (hopefully) integrated into the PS3 and TVs in 2006.

4.1 Disclaimer

Everything written in this paper is derived from the official IBM documentation (http://www.ibm.com/developerworks/power/cell/), different articles at www.anandtech.com or www.realworldtech.com or personal chat with IBM employees. Which means that all this information is about the current state of art and without any warranty. Most of the technological aspects mentioned are patented.