Massive parallel heterogeneous system will you be able to program it?

Dr. Avi Mendelson
Microsoft R&D Center, Israel (ILDC)
Personal background – Avi Mendelson

- 1999 – 2009 – Senior Architect at Intel and in charge of the First CMP implementation in Core Due and Core Due-2 families of processors

- 1996 – Present – Adjunct prof. in the EE and CS departments, Technion, Israel Institute of Technology

- 2009 – present – Microsoft Research and Development Center, Israel.
Disclaimer

Sole responsibility for all opinions and conclusions falls on the author 😞 but it doesn’t mean you cannot trust it😊
Agenda

- Why Heterogeneous systems
- Different types of heterogeneous systems
- Can we program these systems?
- Why it is important?
Motivation and trends in system development

For many years the computer industry was driven by the “Moore’s Law”

Two of the many versions of Moore’s Law
- Number of transistors on a die doubles every 18 months (the original form)
- Measured performance of computer systems doubles every two years (one of many variations)

Implications:
- Enables adding value to the user
- Enables innovation
- Enables new applications and markets
- Allows to maintain prices and revenue for both the HW and the SW industry.
- Prevent from becoming commodity (allow massive investment in new technologies)
Motivation and trends – the Turning point

For decays, the computer industry depends on improving single thread performance. It allows increasing benefit for the consumer without changing the programming model.

Due to process and power limitations the improvements of single threaded application was severely slowed.

The main possibility to continue improving performance is through parallel processing.

Few important observations:

- For modern architectures Power == performance.
- For the same area, as long as enough parallelism exists, in most cases, it is more power efficient to increase the number of cores than increase the frequency in order to achieve the same performance.
- The optimal number of cores is usually depends more on the SW characteristics than on the HW capabilities.
Different types of parallel systems

**Multi-cores** – Small number of “complex core”
- Maintain backward performance computability for single thread applications.
- Aims at supporting existing (General Purpose) SW

**Many-cores**: many small and simple cores on Die
- Very performance and power efficient for massive parallel systems.
- Can hardly support code with insufficient number of parallel processes/threads.

**Heterogeneous Processors**: Combine different types of parallel computing elements
- Aims to achieve the best of all worlds
- Uses two different SW methodologies and developing environments
- May have low die utilization (cost)
- Can we program such systems?
Taxonomy

- Single threaded systems
- Cores + FPGA
- Component based SoC
- Many + Multi Cores
- Heterogeneous-Core
- Multi-Core
- Many-Core
Different types of Heterogeneity

- **Same ISA**
  - Intel to integrate array of Atoms together with i7.

- **Different ISA**
  - AMD puts CPU and GPU on the same die.

- **SoC**
  - Smart phones integrate IP from different companies.
  - TTM (Time to market) of HW, SW and OS is the main motivations for this trend.
Deja-vu -- we have been there before in the 80’s and in the 90’s

Multi-cores:

Intel Paragon
• Meiko
• SGI
• IBM SP1, SP2,...
• Multi More

Many cores:

• CM1- CM4
• Vector machines
• iWRAP –systolic arrays
• Transputers
• Many more

Very few of them became commercially successful
Too many companies went bankrupt because of these ideas.
What stopped the previous “parallel wave”

SW issues
- Parallel systems were too difficult to program and debug.
- Luck of tools and OS support

Parallel systems where relative expensive

Each time technology hits a single thread performance wall, a miracle happened
- When TTL runs out of steam, CMOS appears
- When frequency seemed not to scale anymore, new lithography and process technologies were developed
- When in-order machine could not deliver significant more performance, OOO was invented

Since no miracle is scheduled soon, have we solved the basic SW issues that stopped the past “parallel waves”?
Programming Many-cores and Multi cores

(Out of the scope of our main discussion, but in short)

- TM (Transaction memory) suppose to ease the difficult process of developing parallel systems. But still not mature enough for prime time
- New programming languages – take too long to be adopted.
- Auto parallelization and auto vectorization – can find parallel operations out of sequential code – limited success
- Back to Data-flow (Task level parallelism) – still a niche
- Lock-free algorithms – hard to implement
- Cloud contains very large number of computers, but a single instance is still very small.

All these directions may take too long to become mainstream and save the SW from becoming commodity.
Programming Heterogeneous systems

- CPU + FPGA is being used for many years
  - Very difficult to program and debug
  - Few environment can translate C -> VHDL.
    - Great success for domain specific applications. Still based on excellent and well trained programmers

- CUDA, OpenCL (GPGPU)
  - Easy to write, hard to debug and optimize
  - Useful mainly for limited number of algorithms and data structures.
  - Only experts can develop efficient GPGPU code.
  - Shared virtual address space should help to solve some of the problems

Avi Mendelson -- Paris SW Summit, April, 13 2011
Other possible directions
Heterogeneous OpenMP

The system is defined as a collection of coherent islands. Each island is cache coherent and can be manage by an “OpenMP” like component.

Island can use CUDA, OpenCL or any language that preserve “pure interfaces” (no side effects)

Not HW coherency between islands. System manage the communication, at that level, using “MPI like” layer

MPI is complicated enough for most of the programmers, but the need to understand both MPI and OpenMP may be too high hurdle to overcome.

Thus will be significantly used only by small community
Component based extension - Ct

Ct is a library based extension that aims to parallelize certain data structures for multi-cores, many cores and asymmetric cores.
So far HW made a significant progress in its integration methodology. SW needs to catch-up.

- We are still in early stages of the research on how to develop SW that can be easily integrate and support fast TTM (time to market) for SoC
- We need to extend the OS to support different simultaneous running Operating systems
  - Current virtualization is not cost effective for that purpose
  - Barrellfish made a significant progress toward the right direction.
- New HW/SW interfaces may be needed
- New verification and testing techniques are needed. Most likely based on formal verification
Summary and (my) conclusions

- The computer industry moved to parallel systems mainly due to Power and Thermal related issues.
- Single thread performance is not expected to significantly improve anymore.
  - The benefit of future generations of (same) software packages is reduced and threat to become commodity; i.e., open Office, increases.
- The effect of “commodity SW” is already happening; e.g., most of SW for cellular phones is being sale for 1-2$.
  - Technology companies such as Apple, IBM, Microsoft, etc. may start gaining much more money from infrastructure and services instead of developing new applications.
- Bad sign for large companies that contribute much of the research money
Summary -- heterogeneous systems.

I believe that the trend to build heterogeneous systems mainly to provide better power/performance will not last too long.

- At some near future the different technologies will be converged to uniform approach.

SoC will last if we found the way to build SW to support fast TTM.

Some of the main challenges:
- Portability of modules
- Interfaces
- Testing and verification
- OS and resource management

SoC will continue to depend on high quality experts at given fields. Will large companies be flexible enough to react to the change?
Questions?