"Distributed Processors allow revolutionary Hardware / Software partitioning"

Jean-Reynald Mace
XILINX
Espace Jouy Technologies
21, Rue Albert Calmette
78353 – Jouy-en-Josas
France

Jean-Louis Brelet
XILINX
CICA
2229, route des Cretes
06560 – Sophia-Antipolis
France

Email: jean-reynald.mace@xilinx.com
Email: jean-louis.brelet@xilinx.com

Executive Summary:

Future system designs will use, in addition to traditional embedded processors, innovative approaches based on flexible distributed processors, enabled by digital programmable logic devices like complex FPGA.

The complexity of many of today's system architectures can be likened to highly integrated microelectronic chip design. Traditional system partitioning is generally done at the early stage of system architecture, by defining the tasks to be implemented on the embedded processor(s), and the tasks to be implemented on the hardware.

Evolution of system-on-chip, or integration of an application in one of few devices becomes a revolution when the technology enables innovative system architectures. The key to this new approach is high-bandwidth communications between distributed processors, and a flexible hardware / software partitioning which can be adapted to the evolution of the system in real time, and during the complete product life cycle.

This study targets the Xilinx Virtex-II Pro^{TM} platform FPGA, and is illustrated by comparative examples of hardware or software implementation. Another example shows how distributed processors can optimize interrupt management.

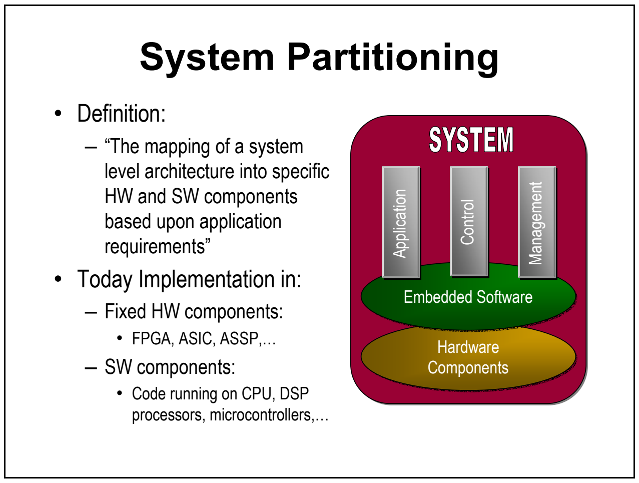
I. INTRODUCTION

All digital systems typically consist of some form of hardware components, processors running software, and memory. The system architect maps a particular system into these various components, taking into account a wide variety of factors, including component availability, upgradeability requirements, legacy hardware IP and software, cost, technical feasibility and more. A typical decision process for system partitioning is to implement heavy data path algorithms in hardware and control and decision making processes in software.

II. SYSTEM PARTITIONING

Each module of a complex system is optimized for cost, performance, bandwidth, etc without first deciding the specific hardware or embedded software implementation. System partitioning is defined as the mapping of a system level architecture into specific hardware and software components based upon application requirements, as illustrated in figure 1.

All systems typically consist of some form of hardware components, processors running software and memory. It is the System Level designer's job to map a particular system into these various components, taking into account a wide variety of factors. (Resources available, Technical feasibility, Upgradeability requirements, …)



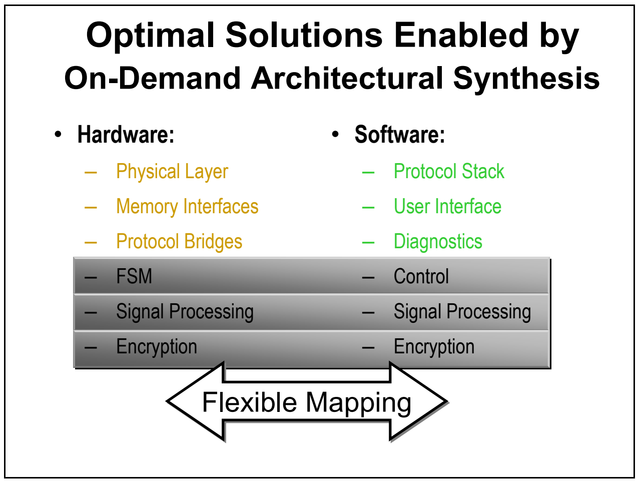
System Partitioning

- Definition:
  - "The mapping of a system level architecture into specific HW and SW components based upon application requirements"
- Today Implementation in:
  - Fixed HW components:
    - FPGA, ASIC, ASSP,…
  - SW components:
    - Code running on CPU, DSP processors, microcontrollers,…

Figure 1 – System Partitioning Definition

Typically functions like a physical layer, memory interfaces, protocol bridges, finite states machine (FSM), signal processing (filters,…) algorithms, encryption functions are designed in HW e.g. a FPGA. On the other hand, protocol stack, user interface (GUI), diagnostics module, control blocks, signal processing sequential blocks and encryption algorithms are traditionally coded in SW.

Some of these functions could be implemented in either software or hardware. The choice of which is very application dependent and the choice of which solution to deploy may not be straightforward. For example: Should a control algorithm be implemented in software or would a Finite State Machine running in hardware be a better option? The decisions will probably be between hardware area required vs software processor overhead required. Another example is encryption design: There are numerous standards for encryption, and change is common. Like the control algorithm, a hardware or software implementation will depend on the nature of the algorithm to be employed. This flexible mapping is shown in figure 2.



Optimal Solutions Enabled by On-Demand Architectural Synthesis

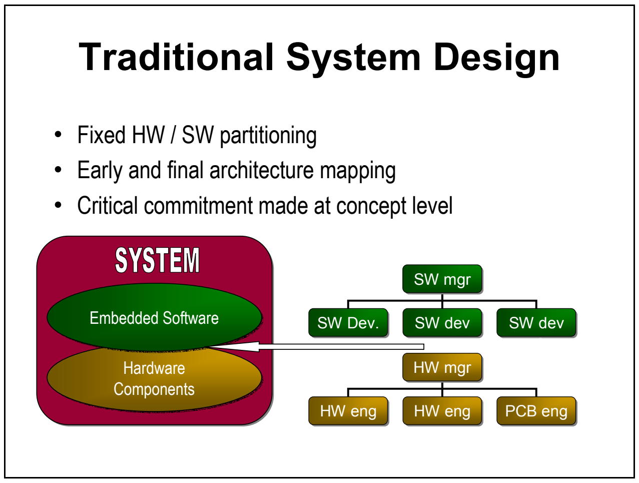
- Hardware:
  - Physical Layer
  - Memory Interfaces
  - Protocol Bridges
  - FSM
  - Signal Processing
  - Encryption
- Software:
  - Protocol Stack
  - User Interface
  - Diagnostics
  - Control
  - Signal Processing
  - Encryption

Figure 1 – HW / SW flexible mapping

Once this split has been decided however, then the system architect or lead engineer allocates tasks to the various teams. This method though has some inherent problems:

- The partitioning is done in the earliest stages of the design; at the moment where there is the greatest possibility for changes
- The system partition is now fixed: If requirements change in any way during the design process, it can be very difficult to incorporate these changes optimally, as the system architect is restricted to the existing partition.

Another consequence of this method is that it also separates software and hardware engineering teams, as shown in figure 3. Tasks are assigned to each of the groups and from the design perspective there is a fixed interface between the two. This often leads to barriers between the two teams.

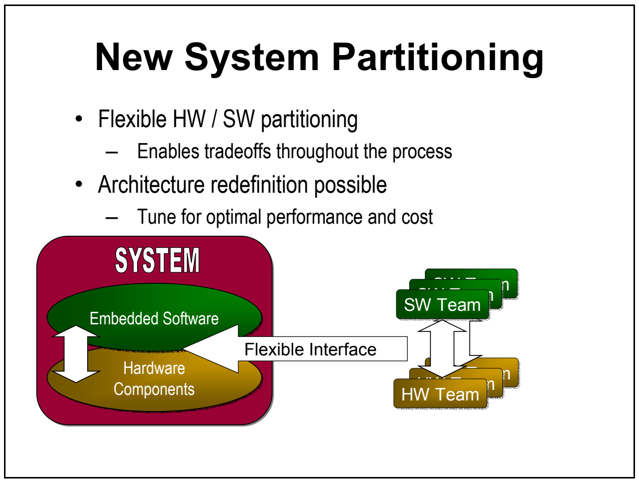


Traditional System Design

- Fixed HW / SW partitioning
- Early and final architecture mapping
- Critical commitment made at concept level

Figure 3 – HW Traditional System Design

The proposed new system partition allows a system architecture that is fully flexible – that is able to be changed throughout the design cycle. Such a method provides new capabilities, including the possibility of continually optimising a design for a particular application. The HW / SW partition is no longer fixed but can be changed to respond to required changes in the system design.



New System Partitioning

- Flexible HW / SW partitioning
  - Enables tradeoffs throughout the process
- Architecture redefinition possible
  - Tune for optimal performance and cost

Figure 4 – New System Partitioning

This innovative approach, as illustrated in figure 4, enables non-traditional system architectures, where SW modules can be implemented in HW, and HW modules can be moved to SW at any time in the design cycle. A scalable and flexible platform is required to optimize HW / SW integration. In addition, a co-design methodology allow to evaluate and modify the design attributes during development (Performances, resource usage,…). The SW developers and HW engineers can now create solutions at module level for optimal systems

III. DES ENCRYPTION ALGORITHM

Communications security has rapidly become one of the most discussed topics in the networking and telecommunications industries, as well as in the general media. With the advent of pervasive networks it has become essential to protect the privacy of the data passing over those networks. Numerous encryption mechanisms exist to enforce security, a good discussion of which may be found in the handbook of applied cryptography [i].

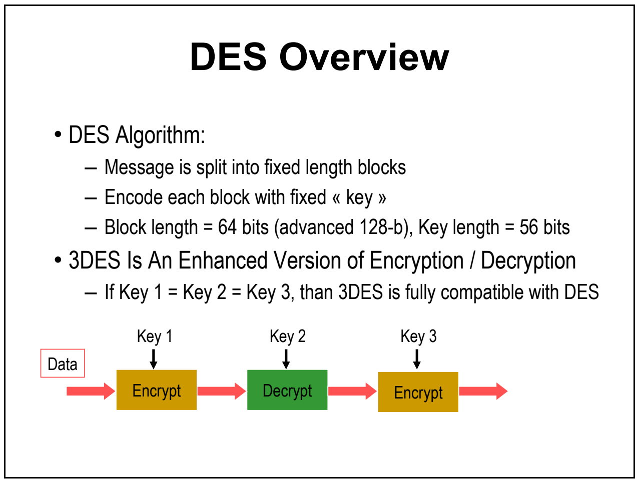
As these networks develop and data rates increase, the implementation of these mechanisms must inevitably change also, to cope with the increased data throughput and to thwart the attempts of those who would try to break the encryption schemes.

In recent times, the most popular encryption/decryption algorithm used has been the "Data Encryption Standard" (DES) and subsequent "Triple Data Encryption Standard" (3DES), as adopted by the US government in 1977 and later standardized by the American National Standards Institute (ANSI) as standard X3.92-1981/R1987.

DES Overview:

DES is a block cipher algorithm, where a message is split into fixed length blocks and then each block encoded using a fixed 'key', known only to the sender and recipient. In DES, the block length is 64 bits and the key is 56 bits in length. A full description of the basic DES algorithm may be found in Xilinx white paper WP115 [ii]

3DES is an enhanced version of this encryption/decryption algorithm, built to be compatible with the standard DES implementation, but offering higher data security. Encryption is performed by passing data block through a sequence of 3 modules as follows, each with their own 56 bit key, like in figure 5.



DES Overview

- DES Algorithm:
  - Message is split into fixed length blocks
  - Encode each block with fixed « key »
  - Block length = 64 bits (advanced 128-b), Key length = 56 bits
- 3DES Is An Enhanced Version of Encryption / Decryption
  - If Key 1 = Key 2 = Key 3, than 3DES is fully compatible with DES

Data
Key 1
Encrypt
Key 2
Decrypt
Key 3
Encrypt

Figure 5 – Triple DES Algorithm

This implementation was selected such that, if all keys are set to be the same, then the output of the 3DES encryption would be fully compatible with single DES encryption. Likewise, decryption would occur in a similar but complementary manner.

System Integrator's Dilemma

3DES is a relatively simple algorithm by today's standards. However it can be used to illustrate the dilemma that a System Integrator can face when building a system that should contain such an algorithm.

The systems Engineer must evaluate all cost and performance constraints imposed on the system to be developed and marry those with the technology that is available to him at that time: For example:

– If this algorithm was to be implemented in software, what performance processor would be needed?
– Would a specific processor be required or can an existing processor in the system be used?
– Does that 'shared' processor have enough 'horsepower' or should the processor be upgraded?
– Perhaps the performance requirements warrant a more dedicated solution?
– Do ASSPs exist and are they cost effective?
– Do they meet the full requirements of the system or is some customization required?

These are just some of the questions to be addressed by the System Integrator, Even if all of these questions can be satisfactorily answered, there is one over-riding question that remains:

– Can this algorithm, and thereby implementation, be fixed or is there a possibility it may need to change in future?

This latter question can lead the system integrator back into the loop of trading off a software and hardware solution to enable flexibility while still meeting performance & cost requirements. This situation is commonly referred to as the "System Integrator's Dilemma" – at some point, a commitment to a particular solution must be made.

As a case in point, when considering 3DES, the likelihood of needing to change the algorithm can be seen with the recent adoption of the AES algorithm by the US government[iii] – the days of 3DES may be numbered. Further, 3DES is a good example of the variety of implementations and architectures that a System Integrator can face.

Architectural Options

When considering the implementation of a 3DES, a number of solutions are immediately apparent.

The popularity of DES has meant numerous hardware and software open source implementations are publicly available. Also, full open source hardware implementations are also available from organization such as OpenCores (www.opencores.org)

In attempting to address the system Integrator's dilemma, more and more engineers are turning to Programmable Logic as the solution to allow fully customizable and modifiable system implementations but which offer far higher performance than purely software implementations.

As an example of this capability, in April 2000, Xilinx announced the record-breaking hardware implementation of a 3DES implementation that achieved a throughput of 10.7Gbits per second throughput using first generation Virtex architecture FPGA. Tests have not yet been done using the second generation Virtex architecture from Xilinx, but clearly this performance may already far exceed that of many system requirements, so it is quite apparent that some smaller in area or hybrid software/hardware solution may be more appropriate.

For example, if we consider the transmit/encryption path only, given the DES encryption module is called twice, as shown in Figure 5 – , it may prove more advantageous to implement solely DES encryption in hardware for acceleration purposes and decryption is maintained as a software solution. This is shown in figure 6

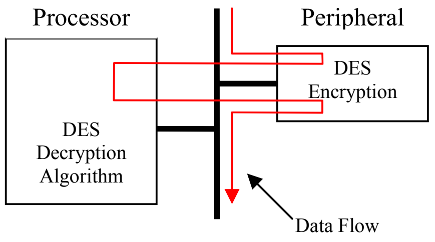


Figure 6 – HW Implementation of Encryption

However, decryption requires more compute power than encryption and so a solution where decryption is implemented in hardware and encryption in software is used, as shown in figure 7, may be more optimal.

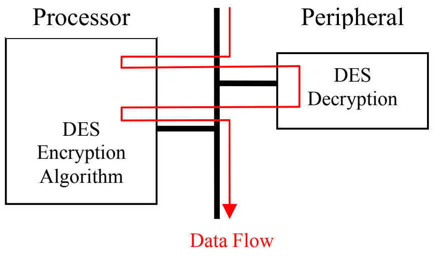


Figure 7 – HW implementation of decryption

If neither solution is appropriate, both encryption and decryption could be implemented in hardware – two such methods are shown in figures 8 and 9.

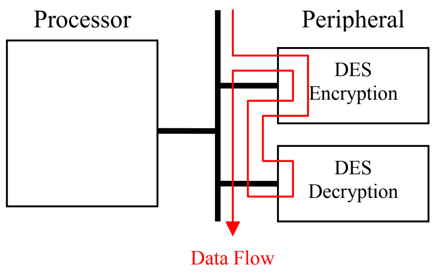


Figure 8 – Full HW Implementation; shared encryption

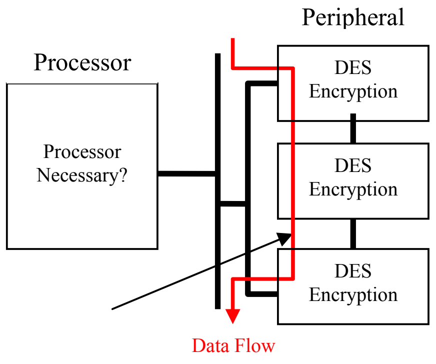


Figure 9 – Full HW Pipelined solution

Clearly there are many choices of hardware/software partition, each offering relative advantages and disadvantages to the system engineer. However, the choice of solution will inevitably be limited by the availability of suitable components and so the actual implementation will always be a compromise that most closely meets cost and performance requirements.

It is at this point that we can consider a totally new approach that would allow the System engineer to develop the exact solution for his needs, without the requirement to compromise.

In addition, the programmable systems solution can be changed on-demand. It is often the case that the true capability of a given system architecture is not fully understood until after significant development has been done. As a consequence, making changes to that architecture would prove costly, if even possible, and major systems engineering decisions must be committed to early in the design cycle in the absence of much needed data. Programmable systems greatly reduce this risk and allow "what if?" scenarios to be played out in the knowledge that change can be dealt with, even after a product has shipped to a customer. This is the enablement of true on-demand architectural synthesis.

This first example of a DES encryption algorithm offers an excellent case study for this paper to compare hardware and embedded software solutions, and the advantages of the flexibility in the choice beyond a simple partitioning decision. Both software and hardware can be co-designed to optimize the attributes of each, and thereby of the whole system. Software and hardware engineers should be able to work together as a cohesive team, focusing at the module level, to ensure that each module in the design is optimal.

IV. WIRELESS LAN

A second example entails the implementation of complex processing functions in real time. The application requires handling of very large tables with dynamically variable priority and lifetime events.

An efficient design solution requires a sophisticated architecture with optimal HW/SW partitioning.

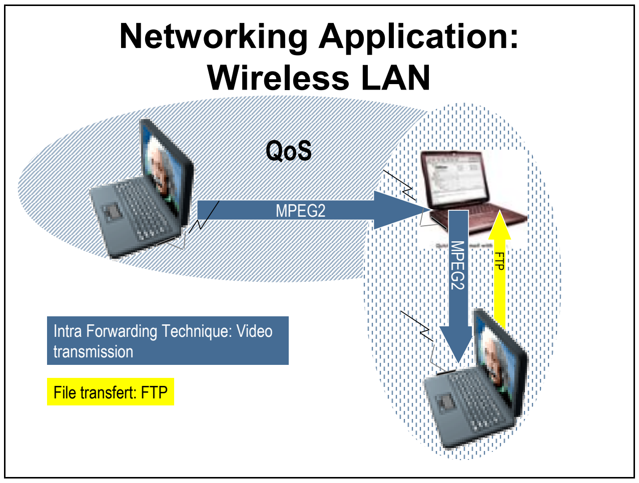

Networking Application: Wireless LAN
QoS
MPEG2
MPEG2
FTP
Intra Forwarding Technique: Video transmission
File transfert: FTP

Figure 10 – Wireless LAN example

The Wireless Local Area Network (WLAN) segment is an emerging market combining data connectivity with user mobility. WLANs represent an attractive connectivity alternative for a broad range of consumers and business customers [vi].

Wireless LAN technology focuses on the PHYsical (PHY) layer and data-link layer within the Medium Access Control (MAC) and Logical Link Control (LLC) sub-layers of the OSI model.

The PHY defines the electrical, mechanical, and procedural specifications, and handles the transmission of bits over a communication medium or channel. WLAN PHY layer technologies include narrowband radio, infrared, Orthogonal Frequency division Multiplexing (OFDM), Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS) and others technologies. The MAC layer handles error control and synchronization between physically connected devices communicating over a channel. It also determines priority and allocates accesses to the channel.

Some popular WLAN technologies are IEEE 802.11[vii]. (a and b), HiperLAN 1 and HiperLAN 2[viii].

In real time applications, the global system should include a specific Quality of Service (QoS) in order to guarantee an application dependent minimum bandwidth from the network. For example when the network is shared among multiple access point, the audio connections and the streaming video must received a minimum bandwidth regarding the lowest application – as file transfer as shown on figure 10.

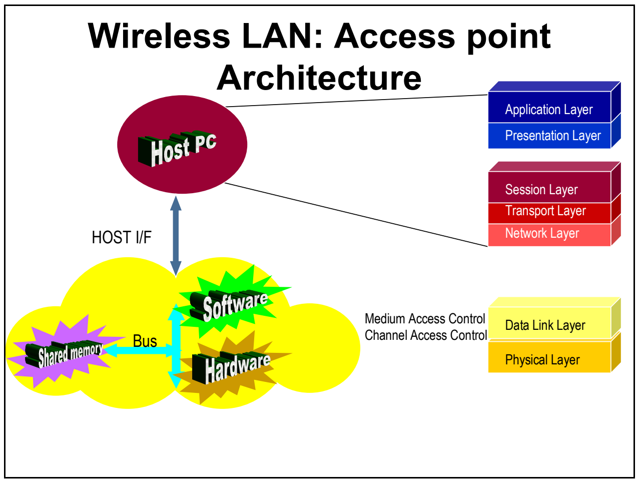


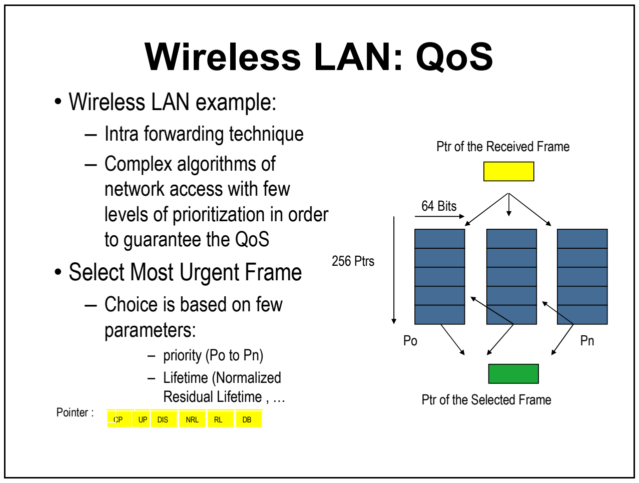
Figure 11 – Wireless Local Area Network or 802.11

The architecture of the Embedded System is composed of a hierarchy of subsystem. Hardware and Software modules must be partitioned with respect to the global specifications in order to guaranty sufficient Bandwidth between the Host and the external world.

A complete structure with adequate interfaces (multiplexed bus, dedicated bus, size,…) between SW modules (one or few processors, RTOS or Firmware, …) and HW modules (ASSPs, FPGAs, analog component, ….) must be chosen and optimized. (See figure 11)

- A key element is the type and size of the shared memory (size, transfer cycles, speed….) between the Software and the Hardware.
- The SW module is defined in terms of key features (performance, Mips, , cache,…).
- The HW FPGA module is sized in terms of frequency, LUTs, Block Ram…

The worst case specifications (maximum number of frames to transmit, concurrent Interrupts, concurrent events,..) is then used to optimize the features in all the modules.



Wireless LAN: QoS

- Wireless LAN example:
  - Intra forwarding technique
  - Complex algorithms of network access with few levels of prioritization in order to guarantee the QoS
- Select Most Urgent Frame
  - Choice is based on few parameters:
    - priority (Po to Pn)
    - Lifetime (Normalized Residual Lifetime , …

Figure 12 – Prioritization to guarantee QoS

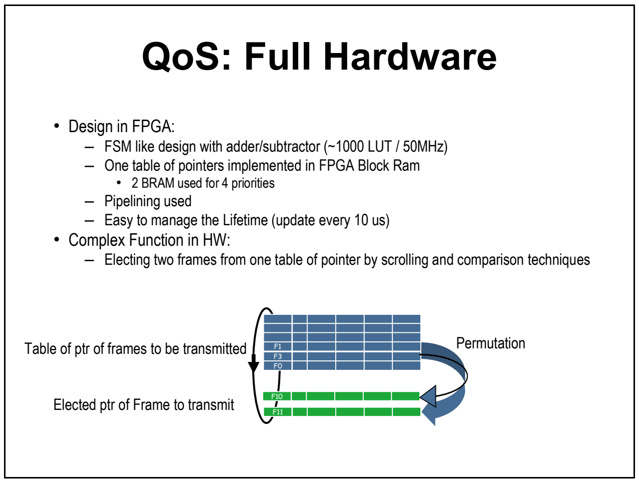
WLAN network access point such as Hiperlan 1, should support an intra forwarding function and complex algorithms of levels of prioritization in order to guarantee the Quality of Service. Each Frame has a Lifetime directly related to the application.

Due to the real nature of the system necessities a carefully crafted frame sorting algorithm which ensures the highest priority data is transmitted first. All the complete frames are memorized in the shared memory. To simplify the computations, only a pointer made up of all the lifetime parameters is managed.

The pointers are split in a priority list and each incoming frame is inserted in the appropriate list. All the Lifetime parameters (residual Lifetime,...) are updated and imply the election of the most urgent frame to transmit, as shown in figure 12.

Several architectural approaches can satisfy these requirements:

- Full HW implementation,
- Full SW implementation,
- Mixed solution.



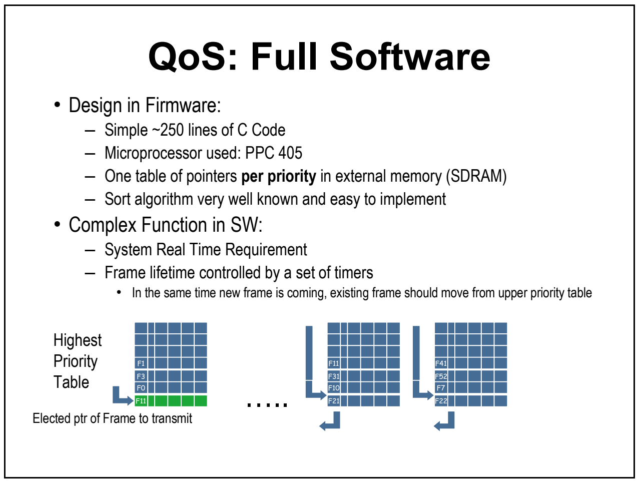
QoS: Full Hardware

- Design in FPGA:
  - FSM like design with adder/subtractor (~1000 LUT / 50MHz)
  - One table of pointers implemented in FPGA Block Ram
    - 2 BRAM used for 4 priorities
  - Pipelining used
  - Easy to manage the Lifetime (update every 10 us)
- Complex Function in HW:
  - Electing two frames from one table of pointer by scrolling and comparison techniques

Table of ptr of frames to be transmitted
Elected ptr of Frame to transmit
Permutation
F1
F3
F0
F10
F11

Figure 13 – Full HW implementation

The full HW solution is based on a FPGA.
The design uses some Finite State Machine (FSM) with adder/subtractors. One global table is stored in the Block RAM of the FPGA.
Election of the most urgent frame is done by scrolling through all tables and comparing each parameter to its neighbours.
All the features and flexibility of the FPGA HW are leveraged to provide an efficient implementation: parallelism, pipelining, Dual port RAM, and illustrated in figure 13.

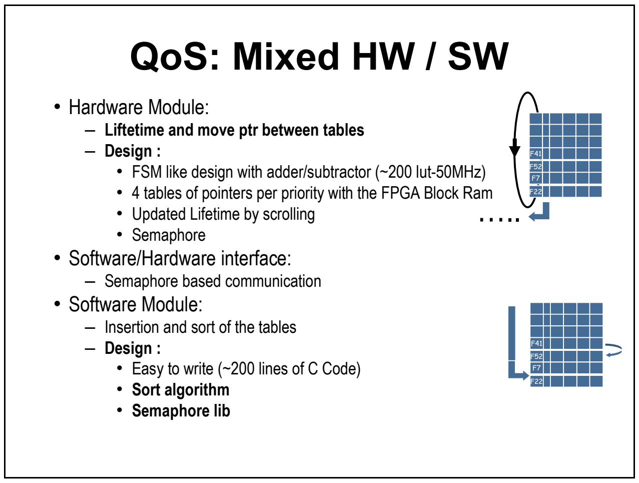


QoS: Full Software

- Design in Firmware:
  - Simple ~250 lines of C Code
  - Microprocessor used: PPC 405
  - One table of pointers **per priority** in external memory (SDRAM)
  - Sort algorithm very well known and easy to implement
- Complex Function in SW:
  - System Real Time Requirement
  - Frame lifetime controlled by a set of timers
    - In the same time new frame is coming, existing frame should move from upper priority table

Figure 14 – Full SW solution

The full SW solution is based on a PPC 405 running "C code". For each priority a table is created and memorized in an external memory.
A standard, well known sort algorithm is employed for sorting each table. (See figure 14)
Most urgent frame selection is just a matter of sorting all the tables. Upgrading the Lifetime parameters is accomplished via a set of timers.
A disadvantage of this pure SW approach is that the real time behaviour is complex to manage. Computational throughput is enhanced by increasing the microprocessor clock frequency.



QoS: Mixed HW / SW

- Hardware Module:
  - **Liftetime and move ptr between tables**
  - **Design :**
    - FSM like design with adder/subtractor (~200 lut-50MHz)
    - 4 tables of pointers per priority with the FPGA Block Ram
    - Updated Lifetime by scrolling
    - Semaphore
- Software/Hardware interface:
  - Semaphore based communication
- Software Module:
  - Insertion and sort of the tables
  - **Design :**
    - Easy to write (~200 lines of C Code)
    - **Sort algorithm**
    - **Semaphore lib**

Figure 15 – Mixed HW / SW solution

The mixed solution is based on the advantages of both implementations, as shown in figure 15.

- HW: handles the Lifetime and the moved request of the pointers between tables. The scrolling technique is used,
- SW: manages the sorting, insertion and movement of the pointers tables. Standard algorithm is used,
- I/F HW/SW: the synchronisation and the communication between the SW and HW is done by semaphores and Interrupt requests (IRQ).

V. ENABLING TECHNOLOGY: VIRTEX-II PRO

In March, 2002 Xilinx announced the availability of the first Programmable System solution in the form of Virtex-II Pro^{TM}. Based upon the award winning Virtex-II Architecture[iv,v], Virtex-II Pro is the first device family to fully immerse both High Performance microprocessors and High Speed serial channels into a fully programmable architecture. Figure 16 represents the platform FPGA architecture.

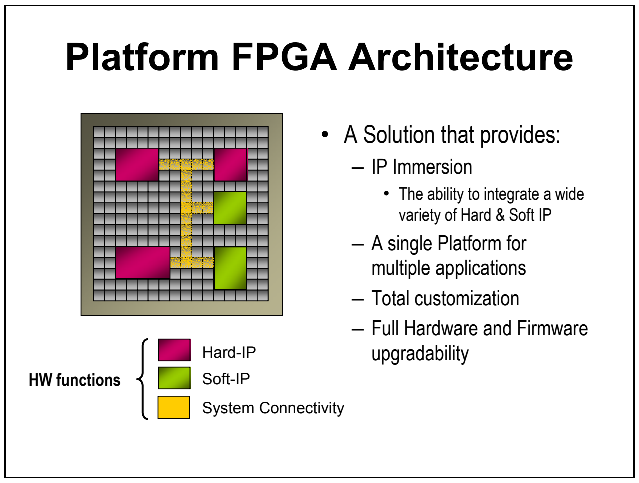

Platform FPGA Architecture
• A Solution that provides:
– IP Immersion
• The ability to integrate a wide variety of Hard & Soft IP
– A single Platform for multiple applications
– Total customization
– Full Hardware and Firmware upgradability
HW functions
Hard-IP
Soft-IP
System Connectivity

Figure 16 – Platform FPGA architecture

There are two kinds of challenges faced by the designers. The first one is to accelerate the performance of a single but large task. The solutions may be parallel processing using multiple processors. This is a first approach to distributed processing. Multiple processors run the task in parallel. The second challenge is to accelerate the performance of multiple different tasks in a system. By using multiple processors - one processor per task - each processor is dedicated at executing that specific task. Virtex-II Pro is illustrated in figure 17.

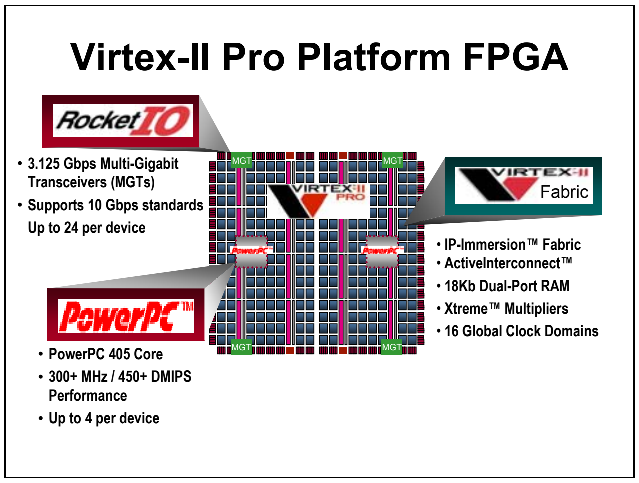


Figure 17 – Virtex-II Pro Platform FPGA

On-Chip memory (OCM) technology delivers the unique internal bandwidth of 6.4 Gbps between the Instruction or Data cache and the adjacent block RAM, as shown in figure 18. To illustrate this point, the example of the time required for changing from one task context to another due to hardware or software interrupt is often a critical parameter. Sometimes the interrupt task to be performed is small in relation to the time context change. If the number of resources and the number of interruptions become important then the timing budget for the Processor for the main task becomes too small. With the possibility to use several small soft processors around the PowerPC allow the distribution in an efficient manner the processing of the different tasks.

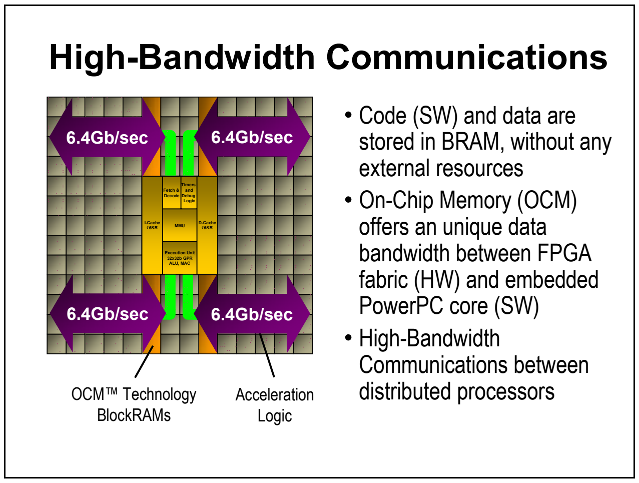
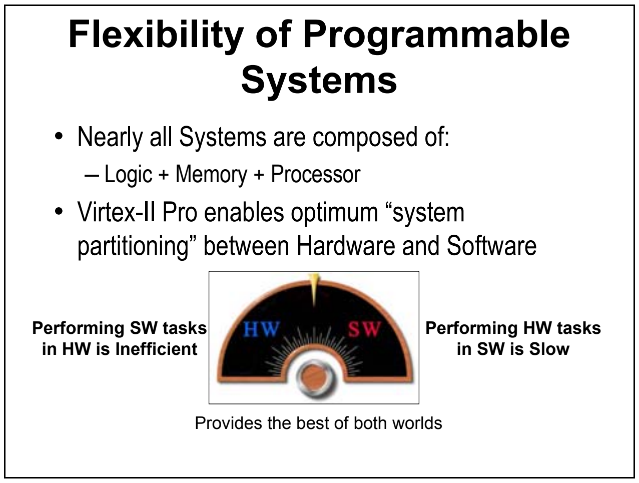

High-Bandwidth Communications
6.4Gb/sec
6.4Gb/sec
6.4Gb/sec
6.4Gb/sec
OCM™ Technology BlockRAMs
Acceleration Logic
• Code (SW) and data are stored in BRAM, without any external resources
• On-Chip Memory (OCM) offers an unique data bandwidth between FPGA fabric (HW) and embedded PowerPC core (SW)
• High-Bandwidth Communications between distributed processors

Figure 18 – High Bandwith through OCM

The Virtex-II Pro platform FPGA solution from Xilinx is the most technically sophisticated silicon and software product today. The goal is to revolutionize system architecture "from ground up", and offer a unique flexible platform to design distributed processors in a system, and high-speed connections. The Virtex-II Pro devices incorporate 3.125

Gbps full-duplex transceivers, and up to four PowerPC (IBM PPC 405) processor cores. With the addition of soft cores like the MicroBlaze^{TM} (32-bit RISC processor) and the PicoBlaze (8-bit microcontroller) from Xilinx, the Virtex-II Pro is the solution to design hierarchical distributed processors systems with the innovative hardware / software partitioning trade-offs explain in this paper. The previous examples show that complex system architectures now have a simple, flexible and reprogrammable solution.

Virtex-II Pro is a silicon enabler that allows a System Integrator to explore quickly and easily many different architectures for a given problem, to assess both hardware and software implementations, and so make the decision "which is best?". Even with an algorithm as simple as 3DES, many options are possible. However with Programmable Systems technology, for the first time an engineer is able to build precisely the solution required and without requiring to employ prohibitively costly and complex SoC type development – this technology is available to all engineers.


Flexibility of Programmable Systems

- Nearly all Systems are composed of:
  – Logic + Memory + Processor
- Virtex-II Pro enables optimum "system partitioning" between Hardware and Software

Performing SW tasks in HW is Inefficient
HW
SW
Performing HW tasks in SW is Slow

Provides the best of both worlds

Figure 19 – HW Flexibility of programmable systems

But above all it can also address one of the most complex issue facing the System Integrator – how to cope with change? The figure 19 shows HW / SW trade-off. Risk can be reduced as architectural changes can be accommodated late in a design cycle, even to the extent of deploying completely different implementations employing different algorithms. New algorithms will require alternative architectures – those architectures can be accommodated for and built, even after a product has shipped to the end customer. Whether software or a hardware implementation is "best" – an engineer can always change his mind.

VI. CONCLUSION

Virtex-II Pro platform FPGA provides an application-specific mix of logic, memory, integrated processors, and high bandwidth I/O.

Embedded and distributed processors allow flexible HW / SW partitioning with optimal mapping at the module level, to design with best solution of both worlds. Virtex-II Pro is the first programmable system to enable true architectural Synthesis, with unique bandwidth between embedded processors and HW.

References

All details about the Virtex-II Pro product can be found at:
http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Virtex-II+Pro+FPGAs

[i] Handbook of Applied Cryptography - Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone - CRC Press
ISBN: 0-8493-8523-7
October 1996

[ii] Data Encryption using DES/Triple-DES Functionality in Spartan-II FPGAs – Amit Dhir, Xilinx, WP115, March 2000

[iii] Federal Register / Vol. 66, No. 235 / Thursday, December 6, 2001 / Notices

[iv] Electronic Products Magazine, January 2002 – Virtex-II wins Product of the Year Award

[v] Virtex-II Pro Handbook, www.xilinx.com

vi "The Home Networking Revolution" – Amit Dhir Xilinx, a designer guide 2001

vii IEEE : 802.11 Wireless local area networks,

viii ETSI : Radio Equipement and System- High PERformance Radio Local Area Network (HIPERLAN) – type 1 - EN 300 652