APPLICATION SPECIFIC PROCESSORS

ECE/CS 570

PROJECT FINAL REPORT

BY

YOUSEF QASIM

PRADYUMNA JANGA

SHARATH KUMAR

HANI ALESAIMI

Contents

Table of Figures

Table of Tables

1. Application Specific Processors

General purpose processors are designed to execute multiple applications and perform multiple tasks. General purpose processors can be quite expensive especially for small devices that are designed to perform special tasks. Also general purpose processors might lack high performance that a certain task required. Therefore, application specific processors emerged as a solution for high performance and cost effective processors. Application specific processors have become a part of our life's and can be found almost in every device we use on a daily basis. Devices such as TVs, cell phones, and GPSs they all have a form of application specific processors. An application specific processor combines high performance, low cost, and low power consumption.

1.1. Application Specific Processors Classification

Application specific processors can be classified into three major categories:

- Digital Signal Processor (DSP): Programmable microprocessor for extensive real-time mathematical computations.
- Application Specific Instruction Set Processors (ASIP): Programmable microprocessor where hardware and instruction set are designed together for one special application.
- Application Specific Integrated Circuit (ASIC): Algorithm completely implemented in hardware.

1.2. Application Specific Systems

Some of the typical approaches of building an application specific system or an embedded system are to use one or more of the following implementation methods: GPP, ASIC or ASIP.

- **GPP**

Functionality of the system is exclusively build on the software level. Although the biggest advantage of such system is the flexibility but it is not optimal in term of performance, power consumption, cost, physical space and heat dissipation.

- **ASIC**

Compared to GPP, ASIC based systems offer better performance and power consumption but at the cost of flexibility and extensibility. Although it is difficult to use the ASIC for tasks other than what they were designed for, but it is possible to use GPP to perform the more general less demanding tasks in addition to ASIC in the same system.

- **ASIP**

In this approach, and as defined in [9], [4], an ASIP is basically a compromise between the two extremes; The Application specific integrated circuit processors ASIC being designed to do mostly a very specific job with high performance but with minimal room for modifications and the general purpose processors which costs a lot more than ASIP but with extreme flexibility at what they do. Due to this flexibility and low price, ASIP are great to be used in embedded and system-on-a-chip solutions.

Figure 1.1 and Table 1.1 shows the comparison between the three approaches in term of performance and flexibility and other design considerations.

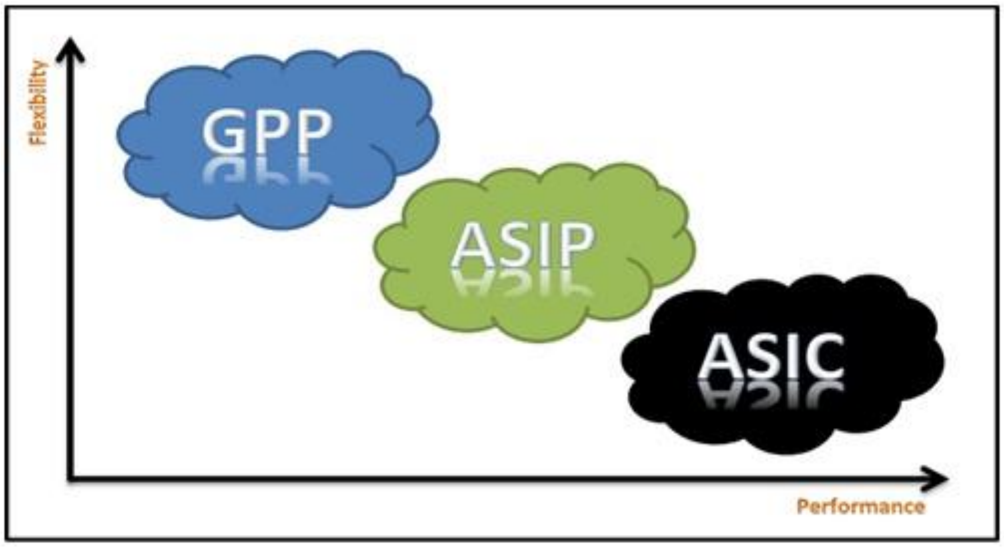

Figure 1.1: Compares the Processors Performance vs. Their Flexibility

	GPP	ASIP	ASIC
Performance	low	high	very high
flexibility	excellent	good	poor
HW design	none	large	very large
SW design	small	large	none
power	large	medium	small
reuse	excellent	good	poor
market	very large	relatively large	small
cost	mainly on SW	SOC	volume sensitive

Table 1.1: Compares between Different Approaches for Building Embedded Systems [1]

This project studies the varieties of different application specific processors represented in the three categories mentioned earlier. Also this project deal with Field Programmable Gate Array (FPGA) as a way to implement these embedded systems.

2. Digital Signal Processors

Digital signal processors are specialized microprocessors optimized for the need of performing digital signal processing. Digital signal processors gained their importance with the increased demand on data-intensive applications such as video and internet browsing on mobile devices. The digital signal processors satisfy the need for powerful processor while maintaining low cost and low power consumption. Size limitation prevented the possibility to include more than four processors on the board but due to the recent shrink in feature size allowed for single-chip DSPs to become multicore with decent amount of memory and I/O interface. The need for larger memory on chip and the power constrains driven these multicore DSPs to become system-on-chip. Using system-on-chip leads to more reduction in cost for its simplified board design. The move into multicore direction in the embedded systems allowed for higher performance, lower cost, and lower power consumption.

2.1. Background and History

The concept of DSP was introduced in mid 1970s. The rise in interest for DSPs is the result of the need to solve real world problems using digital systems. Few years later a toy by the name Speak and Spell was created using a single integrated circuit to synthesize speech opening the doors for new possibilities. These possibilities can be described by that digital signal processing is possible in real time and DSPs can be cost effective. DSPs differ from other microprocessors is that DSP intend to do complex math while guaranteeing real-time processing [2].

Prior to the advance in DSP chips, the DSP application was implemented using AMD 2901 bit slice chip. These bit slice architectures would sometimes include a peripheral multiplier chip. Later on Intel released the 2920 as an analog signal processing that has ADC-DAC with internal signal processor. But lack of hardware multiplier was the reason for it market failure. In 1980 the first dedicated digital signal processor was released. In 1983 Texas instruments released their first successful DSP [3].

Architecture properties such as dual/multiple data busses, logic to prevent over/underflow, single cycle complex instructions, Hardware multiplier, little to no support of interrupt, and support of special instructions to handle signal processing gave the DSP the ability to do complex math in real time.

As an example, the sample rate for voice is 8 kHz. First DSPs allowed around 600 instructions per sample period, barely enough to transcoding. As better DSPs become available functions such as noise cancelation and echo cancelation can be implemented. This was a result of using multiprocessors DSPs rather than improving performance of single DSPs. This example is depicted in Figure 2.1 (a). Nowadays, more than 68% of DSPs are used in the wireless sector precisely in the mobile handsets and base stations [2].

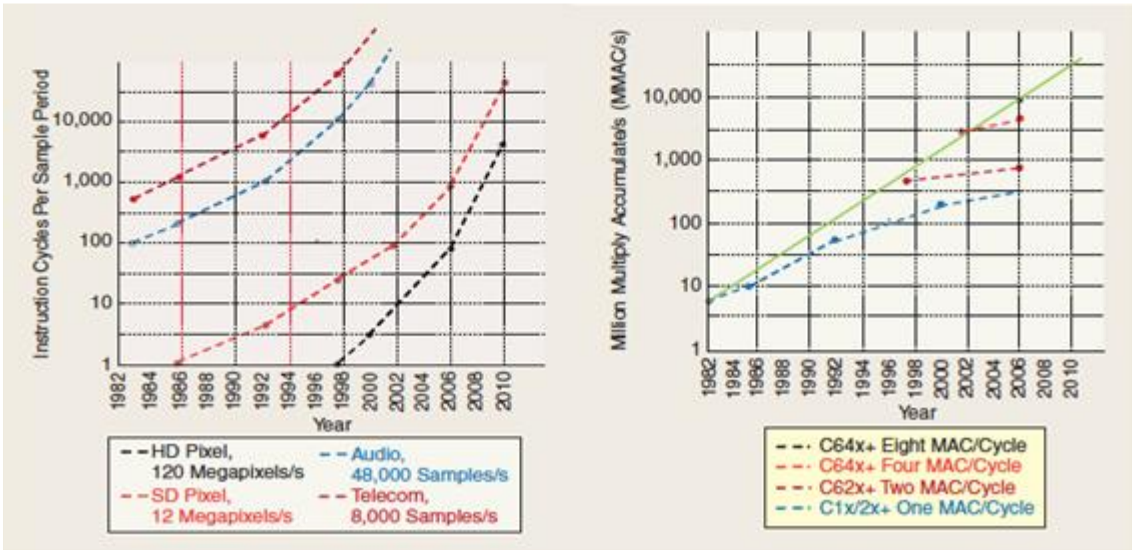

Figure 2.1: (a) Evolution of DSPs for Different Applications in Instruction/Cycle. (b) The Effect of Using Multiprocessing on DSPs performance [2]

2.2. General DSP Properties

2.2.1. Architecture:

Hardware features visible through DSP instruction sets commonly include:

- Hardware modulo addressing, allowing circular buffers to be implemented without having to constantly test for wrapping.
- Memory architecture designed for streaming data, using DMA extensively and expecting code to be written to know about cache hierarchies and the associated delays.
- Driving multiple arithmetic units may require memory architectures to support several accesses per instruction cycle
- Separate program and data memories (Harvard architecture), and sometimes concurrent access on multiple data busses
- Special SIMD (single instruction, multiple data) operations
- Some processors use VLIW techniques so each instruction drives multiple arithmetic units in parallel
- Special arithmetic operations, such as fast multiply–accumulates (MACs). Many fundamental DSP algorithms, such as FIR filters or the Fast Fourier transform (FFT) depend heavily on multiply–accumulate performance.
- Bit-reversed addressing, a special addressing mode useful for calculating FFTs
- Special loop controls, such as architectural support for executing a few instruction words in a very tight loop without overhead for instruction fetches or exit testing
- Deliberate exclusion of a memory management unit. DSPs frequently use multi-tasking operating systems, but have no support for virtual memory or memory protection. Operating systems that use virtual memory require more time for context switching among processes, which increases latency.

2.2.2. Program flow

- Floating-point unit integrated directly into the datapath
- Pipelined architecture
- Highly parallel multiplier–accumulators (MAC units)
- Hardware-controlled looping, to reduce or eliminate the overhead required for looping operations

2.2.3. Memory Architecture

- DSPs often use special memory architectures that are able to fetch multiple data and/or instructions at the same time:
  - Super Harvard architecture
  - Harvard architecture
  - Modified von Neumann architecture
- Use of direct memory access
- Memory-address calculation unit

2.2.4. Instruction set

- Multiply–accumulate (MAC, including fused multiply–add, FMA) operations, which are used extensively in all kinds of matrix operations, such as convolution for filtering, dot product, or even polynomial evaluation
- Instructions to increase parallelism: SIMD, VLIW, superscalar architecture
- Specialized instructions for modulo addressing in ring buffers and bit-reversed addressing mode for FFT cross-referencing
- Digital signal processors sometimes use time-stationary encoding to simplify hardware and increase coding efficiency.

2.3. Multicore DSPs

Multicore DSPs a raised as a solution for higher- performance applications such as third generation wireless communication, mobile TV, and mobile web browsing. The move from single core to multi core resulted in significant improvement in performance, power consumption, and space requirements. Figure 2.2 depicts a generic multicore DSP.

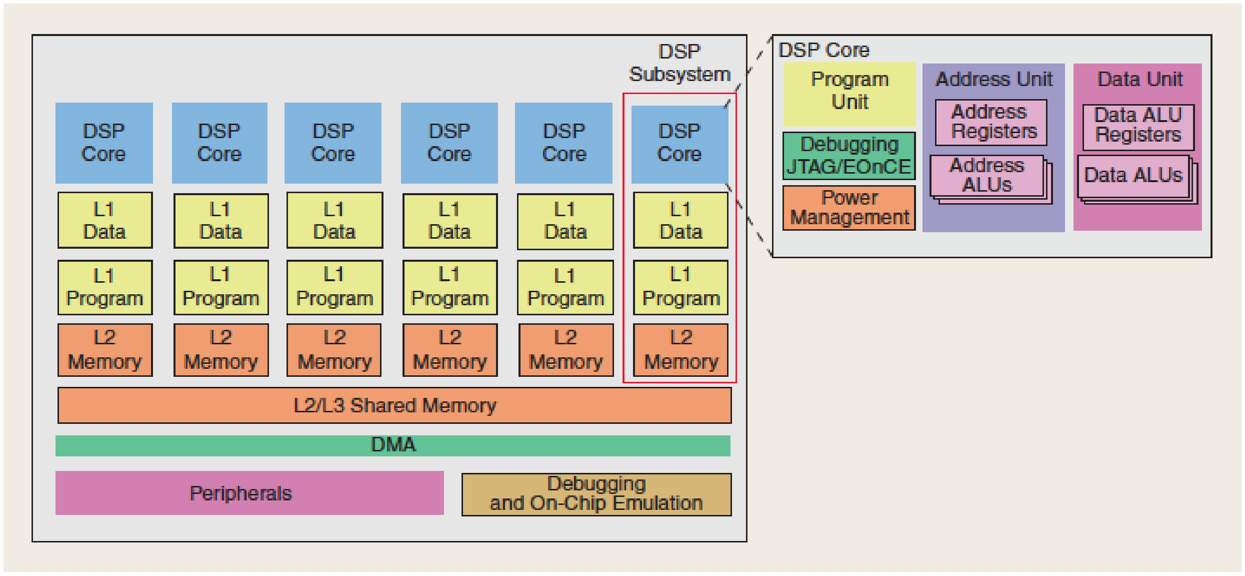

Figure 2.2: Generic Multicore DSP Architecture [2].

Multicore DSPs can be categorized in several ways. They can be categorized by the type of the cores on the chip into homogeneous or heterogeneous. Homogeneous multicore DSPs consist of the same type of cores, on the other hand heterogeneous DSPS consist of different cores. DSPs also can be categorized base on the way they are interconnected. DSPs can be categorized base on interconnection between cores as a hierarchal or as a mesh topology. In hierarchal topology cores communicate through switches. Cores that need to communicate with each other faster are placed close to each others. Cores can share memory or have individual memories. Cores usually have a dedicated level 1 memory, and a dedicated or a shared level 2 memory. The communication between cores is done through memory transfer between memories. This process is known as direct memory access (DMA). In mesh topology cores are placed on 2D array and the cores are connected through a network of buses and multiple simple switches. Memory is usually dedicated (local) to each processor. Mesh topology allows scaling to large number of cores without increasing the complexity. Both topologies are depicted in Figure 2.3.

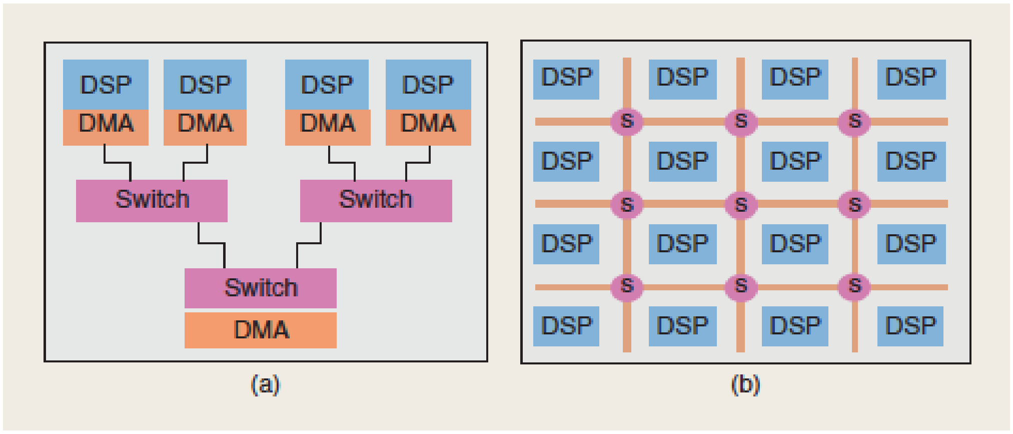

Figure 2.3: Multicore Hierarchal Topology, (b) Multicore Mesh Topology [2]

2.4. Examples of Multicore DSPs

Multicore DSPs are produced by several vendors these days. Company such as TI, Freescale, and Picochip provides different DSP platforms for several applications such as wireless communication, video, and VOIP. In this section several multicore DSPs are presented.

2.4.1. TI TNETV3020

This multicore DSP platform contains six TMS320C64x+ DSP cores each running at speed of 500 MHz. This platform is designed to fit high performance wireless voice and video applications. This platform is based on the hierarchal architecture. Figure 2.4 depicts this platform [4].

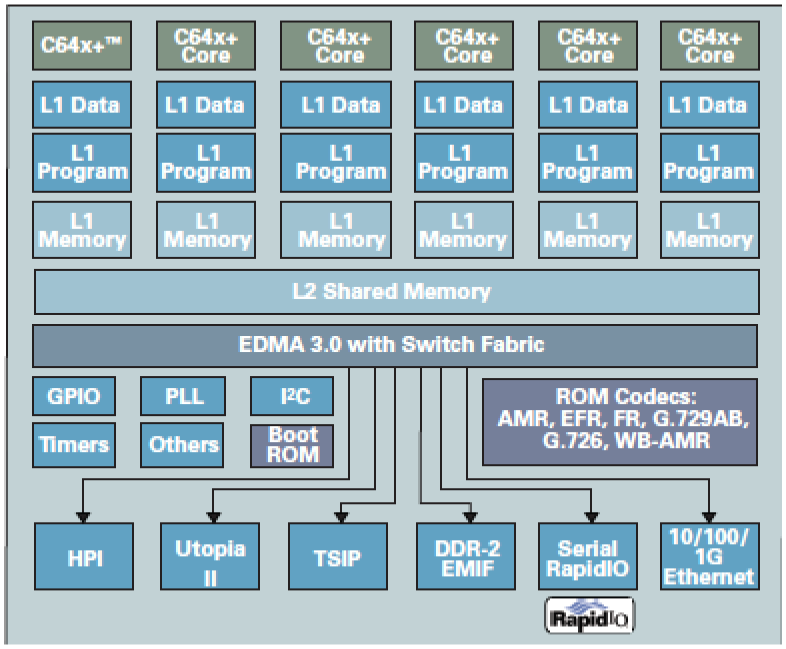

Figure 2.4: TI TNETV3020 DSP Platform [4]

2.4.2. Freescale MSC8156

This multicore DSP platform contain six starcore SC3850 DSP processors each running at speed of 1 GHz. This platform is designed for 3G/4G applications. Figure 2.5 depicts this platform [5].

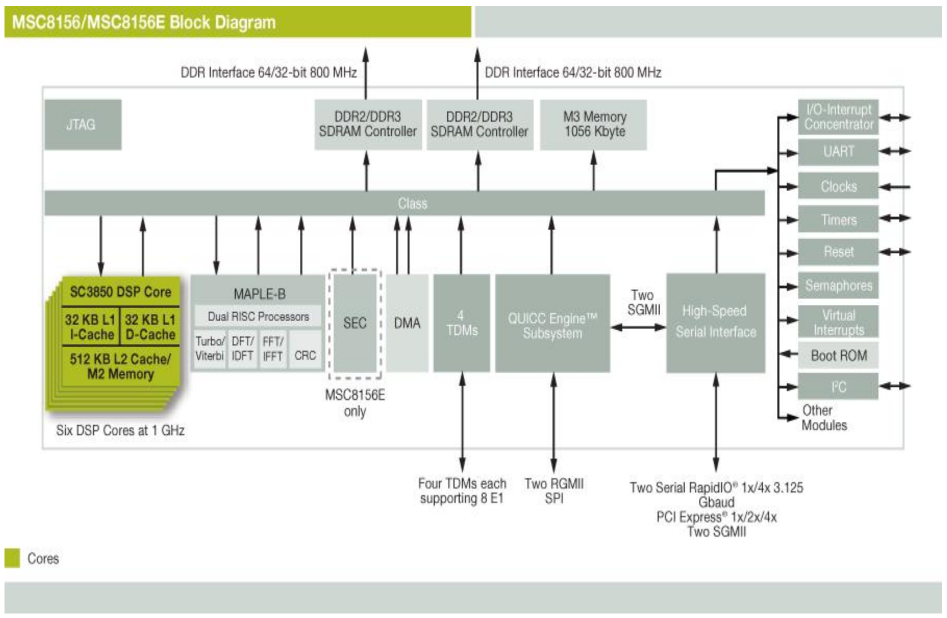

Figure 2.5: Freescale MSC8156 Platform [5]

2.5. Study Case: TI TMS320C6x

This DSP provides high performance and low power consumption in addition to low cost. The processor is used in multiple platforms which are used in several applications such as video and wireless communications. The TI TMS320C6x is a family of several processors which supports floating point and fixed point. This family of processors supports VLIW [6].

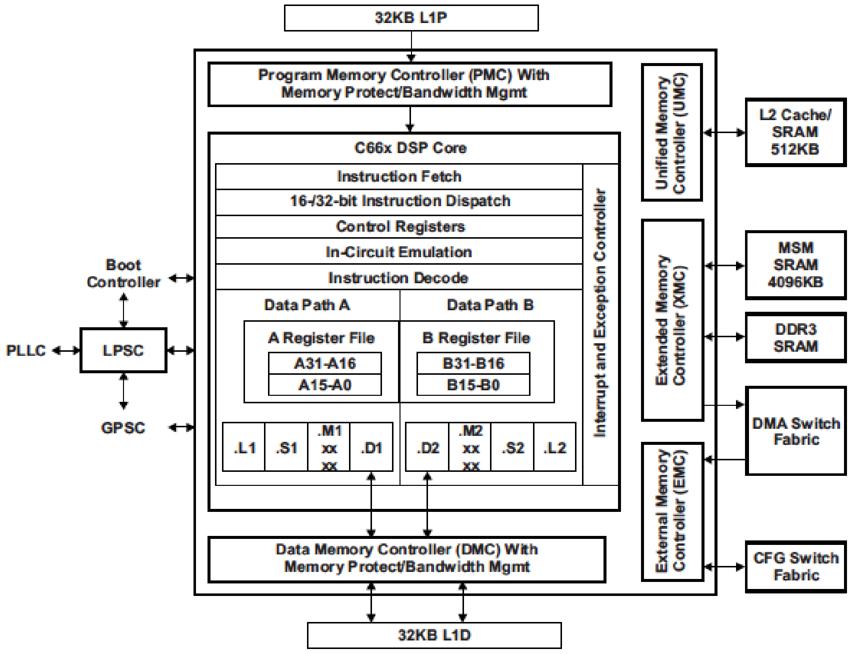

Figure 2.6: TI TMS320C66x Processor [7]

The TI TMS320C66x runs at speed of 1 GHz to 1.25 GHz. The chip also integrates a 64-bit wide main memory, supports DDR3 and queue management. Figure 2.6 depicts the TI TMS320C66x processor. The fetch stage is subdivided into 4 stages, decode stage is subdivided into 2 stages, and execution stage is subdivided into 10 stages. While every instruction have to go through all fetch and decode stages, it is not necessary for each instruction to go through every execution stage. Also each core have 2 multipliers, 2 store units, 2 load units, and 2 data address units. The TI TMS320C66x supports a data and instruction L1 cache memory and can range from 4KB to 32KB. Each processor support 4 ways set associate L2 cache ranging between 32KB to 1 MB [7], [8].

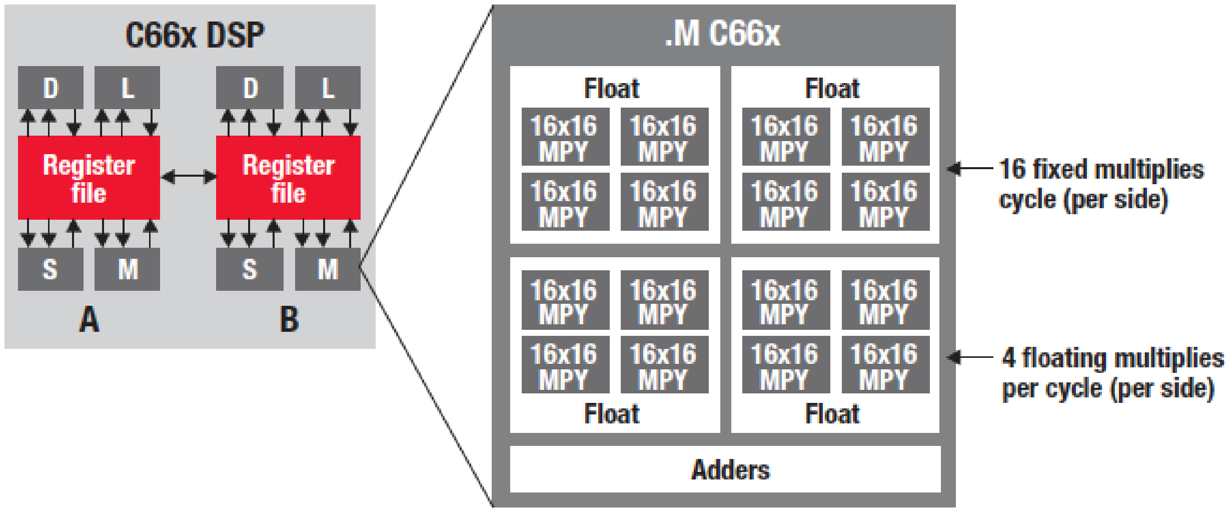
C66x DSP
D
L
Register file
S
M
A
B
.M C66x
Float
16x16 MPY
Adders
16 fixed multiplies cycle (per side)
4 floating multiplies per cycle (per side)

Figure 2.7: Inside TI TMS320C66x Multiplier [9]

3. Application-Specific Instruction Set Processors

3.1. Background

An ASIP is typically a programmable architecture that is designed in a specific way to perform certain tasks more efficiently. This extra efficiency is not exclusively associated with faster performance. Other factors like reduced production costs, simplified manufacturing process and less power consumption can all be considered efficiency qualities for ASIP. The term “Application” in ASIP is not necessarily related to software applications, it actually describe the class of tasks the ASIP platform was designed to efficiently accomplish. As the name suggests, the Instruction set seems to be the core characteristic of any ASIP based platform; but this is entirely not true. Considering a whole platform, other very important attributes like interfaces and micro-architecture do contribute a lot to the overall system performance.

Interfaces outline how the system and the architecture communicate with each other. Having only an extremely parallelized instruction-set ASIP in a data extensive processing platform doesn’t mean necessarily a more efficiently performing system. For example, if the load/store unit cannot handle data processing as quick, there will be a performance bottle-nick due to system interfaces. Similarly for the micro-architecture, the pipeline of the ASIP must be designed in a specific way that optimizes the performance of the whole system. A traditional RISC stages (Fetch, Decode, Execute, memory & write-back) might not be the optimal pipeline for the application. Specifically designing any of these aspects of ASIP will be always associated with trade-offs. Having an extremely parallelized instruction-set will propose issues with the size of the instruction-set and it will affect the program memory, interfaces and fetch & decode stages of the pipeline.

3.2. Instructions Set

An ASIP instruction is usually different than a normal instruction. It doesn't have to be composed of a mnemonic and register/memory operands. The design of the architecture –which is controlled by the application of the ASIP-, shapes the instruction-set format. An ASIP can use either general purpose registers or configuration registers. In a typical RISC processor, instructions trigger functional units along with general purposes register addresses, while ASIP can benefit also from configuration registers or utilizes specially designed data flow mechanisms that are hardwired in the system. Using configuration register with constant operands eliminate the necessity to encode their addresses them in the instruction word as with general purpose registers. In [1], the authors describe ASIPs as more of a hardwired block than a processor from an architectural and interface perspective.

Instructions set in ASIPs can be divided into two parts, Static logic which defines a minimum ISA and configurable logic which can be used to design new instructions. The configurable logic can be programmed in the field similar to FPGA or during the chip synthesis.

3.3. Network Processors

Douglas comer defines a network processor as "*A network processor is a special-purpose, programmable hardware device that combines the low cost and flexibility of a RISC processor with the speed and scalability of custom silicon (i.e., ASIC chips). Network processors are building blocks used to construct network systems.*"[13]

3.3.1. Networking Systems Evolution

Generally, network systems architecture evolution can be classified into three main generations:

- **First Generation (1980s):** Networking services are executed on the application layer on a general purpose processor based system. In another word, network services are basically software packages that are running on top of the operating system of a PC.

- **Second Generation (mid 1990s):** Few networking functions were assigned to more dedicated hardware units and a faster switching fabric that replaces the shared bus.

- **Third Generation (late 1990s):** Utilization of ASIC for a more distributed design with ASIC hardware and an exclusive processor on each network interface to accommodate the fast data plane requirements.

The complexity of latest network devices does benefit from the distributed approach in designing the platform. Activities like exchanging routing table entries among all routing interface in a network router need to be carried out smoothly without any interfering with the router forwarding tasks. [13]

But the move toward ASIC had its trade-offs. Some of those are the high production costs, the long delay of availability to markets time and the inflexibility of ASIC based platforms. These trade-offs made it clear that ASIC based systems are not efficient in term of scalability or upgradability.

Chip manufacturers began exploiting new approaches to design flexible yet fast performing network systems. That when programmable network processor came out. A processor with combined advantages of first and third generation devices; the flexibility of general purpose processors and the speed of ASIC was obvious.

It is very obvious that software based systems does not perform as fast as hardware based ones. So with the programmable approach in mind, networking core tasks must be exploited and assigned to dedicated functional units to guarantee the overall performance of these processors would be acceptable.

3.3.2. Network Applications

In order to understand how network processors are special and why they are designed accordingly, we need to understand the characteristics of network applications first. Most of network applications are based on pattern matching, address lookup and queuing management which can be classified into two main categories according to the processing types, Data Plane and control plane processing.[15]

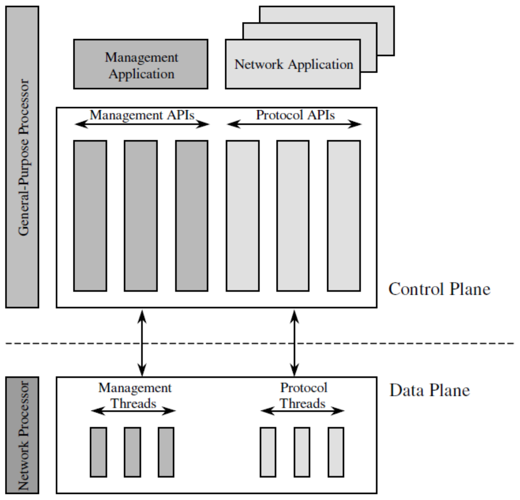

Figure 3.1: Network Processing

Data plane processing:

Basically the processor is moving data from one port to another without much interest in the payloads. Such tasks varies from less processor demanding applications like routing or forwarding to a heavier applications like firewall filtering, encryption or contents payload inspection.

Control plane processing:

Tasks like updating router lookup entries and establishing secured tunnel connections are some of the control plane tasks. Usually such tasks are more demanding processes although it deals with fewer amounts of data. Data plane tasks require minimal level of designing but they require a heavy processing power while on the contrary, the control plane processing doesn't require heavy processing power but it takes effort to design them properly.

3.3.3. Architecture:

Exploiting the architecture of network processor involve understanding how networking processes are carried out, what functional units needs to be added to support these tasks better, what types of interface, memory, instructions set, etc. Such exploitation was not done properly between researchers and chip vendors which yield to each vendor adopting different approaches in designing their network processors. By 2003, there were more than 30 different products being sold as network processors. Not having considered the universal designing blocks that would have worked on all network systems and across all protocol stack layers has led to this wide variety of network processors. Normally not all designs will be widely recognized which lead to some of the chip vendor to pull back from the network processors market. Figure 2.2 illustrate a generic architecture of a network processor.

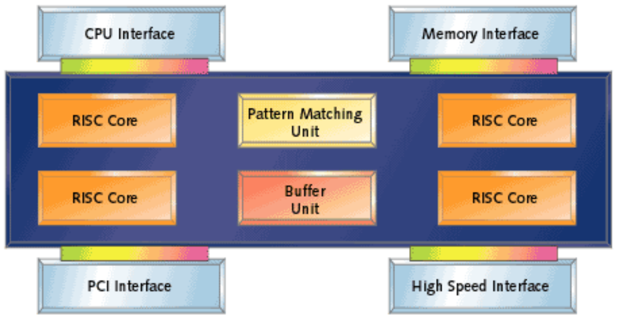

Figure 3.2: Generic Network Processor

Architects mainly utilize three approaches to improve performance of a chip. Higher clock rate for a single processor, parallel processing and hardware based pipelining.

Figure 2.3 displays the three architectural approaches in designing any embedded system processing unit. In (a) single processor approach, the processing unit handles one packet at a time in what is known as run-to-completion method. In (b) multiple processors approach, there are more than one packet processing unit and every processing unit can process its own packet independently of other units. Such approach requires a scheduling mechanism to determine which packet gets assigned to which unit and an ordering mechanism to assure packets at the egress port are being transmitted in the right order. In the last approach (c) every packet flows through the entire pipeline and in each stage, a part of the required process is performed. [13]

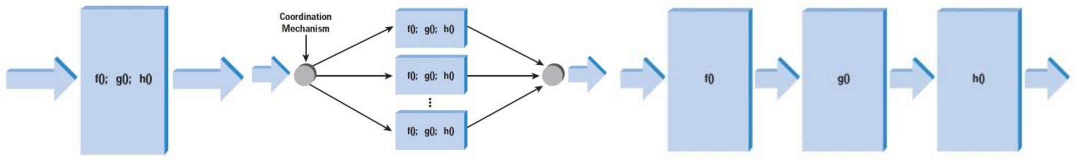

Figure 3.3: Typical architectural approaches. (a) Single processor (b) Multiple processors (c) Pipeline

3.3.4. Network processing pipeline:

Network traffic is basically multiple “packets” being received, processed then transmitted accordingly. A generic networking platform will undergo such a processing cycle in 4 stages:

- Receive Stage: When the packets enter through a port or an endpoint to the networking platform, they will be reassembled and moved to the memory.
- Packet processing stage: In this stage the platform could be performing multiple tasks that might involves packets validation, address lookup, system status updates and destination parameter determination.
- Quality of service stage: This is basically will prioritize transmitting of packets according to their importance in the network. Such tasks are usually based on queue management and priority scheduling.
- Transmitting stage: Some of the activities in this stage are such packets retrieval, segmentation and packets transmitting over physical or virtual ports.

3.3.5. Memory

In a typical network processor, there are three types of memory contents. Instructions, control data and packet payload. Instruction memory must be able to send instruction to all processing unit simultaneously and quickly. Usually such memory is implemented internally since it doesn't contain a large amount of information, but in more advanced network processors with more sophisticated capabilities it might be implemented externally. Control processing involves addresses lookups and such tasks require larger bandwidth for its memory. In packet payload processing, the packet memory is completely dependent on the characteristic of the packet processing unit. Some processing unit are design to implement tasks that doesn't deal with large amount of data while other units might intensively and repetitively process huge amount of information. Network processors memory architecture can be classified into three approaches, shared, distributed and hybrid.[15]

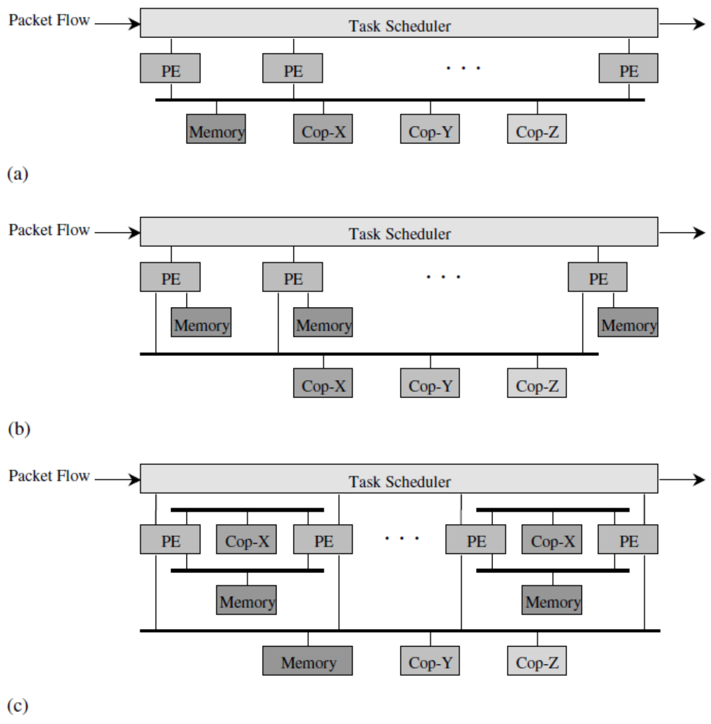

Figure 3.4: Network processor memory: (a) shared memory, (b) distributed memory, (c) hybrid memory.

In shared memory model the main advantage is that its easier to be implemented but it lacks scalability and performance when compared to distributed model. The hybrid model is basically a compromise between the two approaches. The hybrid approach is very useful in implemented a different model for every packet processing unit according to its tasks. Figure 2.4 illustrates these models.

3.3.6. Examples of network processors

Alchemy Semiconductor, Augmented RISC

This processor was basically augmented with special instructions and I/O interfaces to speed packet processing.

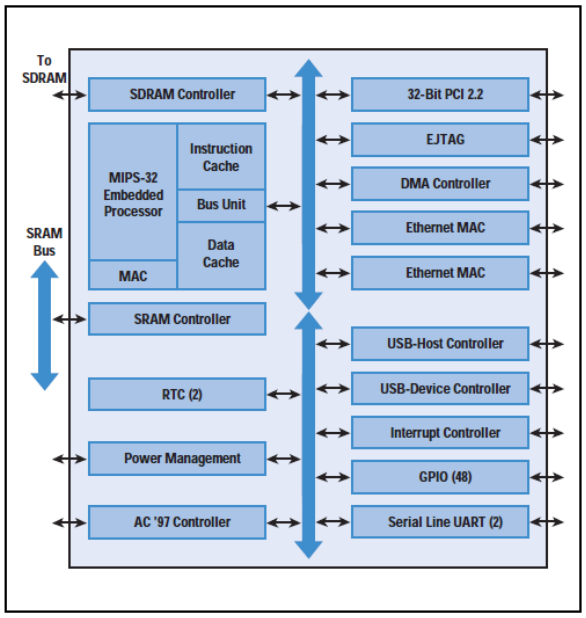

Figure 3.5: Alchemy Semiconductor, Augmented RISC

Parallel Processors Plus Coprocessors AMCC

Parallel processors and coprocessors dedicated for packets processing tasks. In this design, when a packet arrives, one of the cores handles it while the coprocessors are shared and can work with any of the parallel core processors.

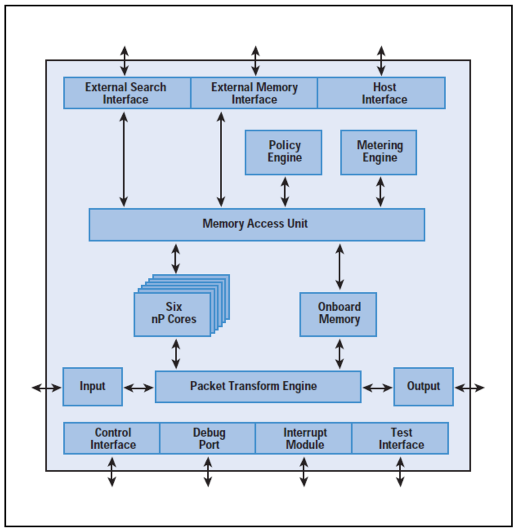

Figure 3.6: AMCC

Extensive and diverse processors (Rainier)

This network processor was initially designed by IBM but later on it was owned by Hifn Corp. this processor architecture implements parallelism and utilizes both, general and special purpose processors.

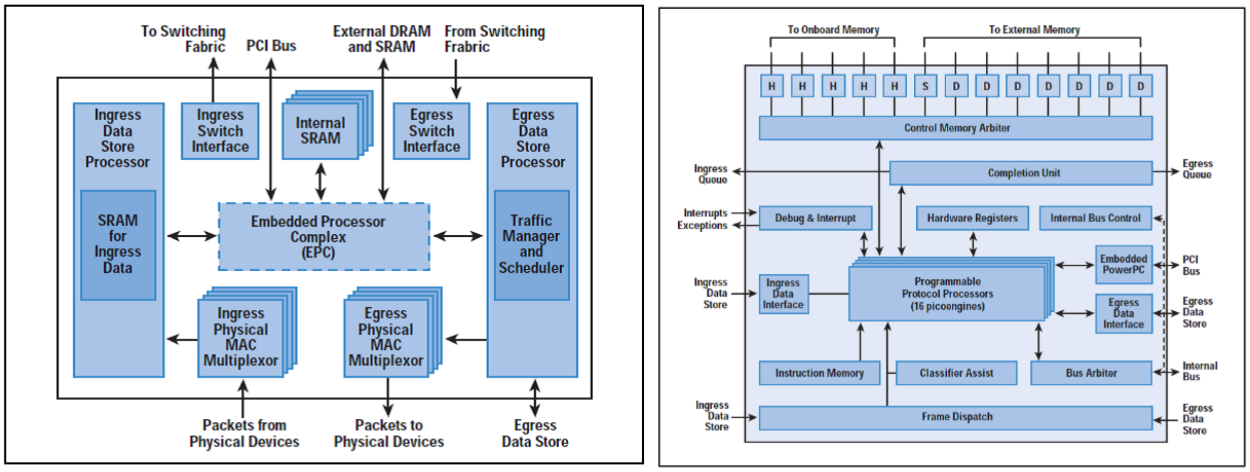

Figure 3.7: Extensive and diverse processors (Rainier)

The EPC unit has 16 programmable packet processors called picoengines and various coprocessors in addition to a PowerPC processor for control and management.

Parallel Pipelines of Homogeneous processors (CISCO)

A hybrid design of implementing multiple paralleled pipelines is implemented in this processor. When a packet arrives to the processor, it enters one of the pipelines and moves throughout the entire pipeline.

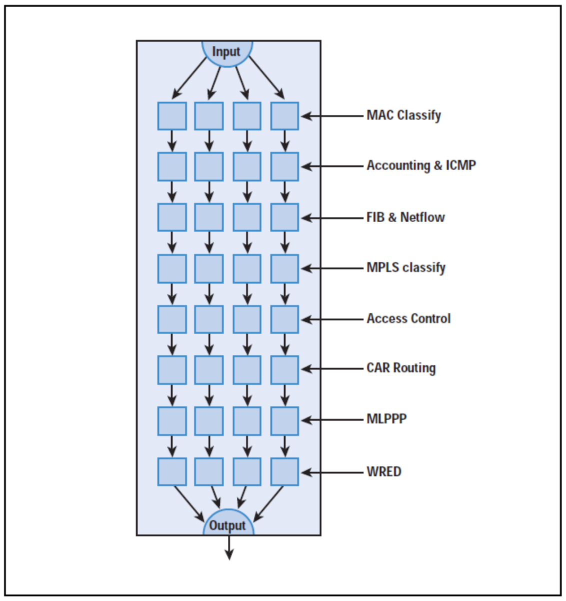

Figure 3.8: Parallel Pipelines of Homogeneous processors (CISCO)

Although the idea of programmability might sounds appealing, it is an extremely difficult task for programmers. In order to achieve high speed processing, programming must be done at a low level. Indeed many network processors are more of a microcontroller than a processor since they are programmed in microassembly. But manufacturers like Agere Systems (owned by LSI now) and IP Fabrics provided high-level programming languages for their systems.

3.4. Case Study (Cisco QuantumFlow Processor QFP)

3.4.1. Architecture

The QFP was announced by CISCO in 2008. It is a non-pipelined, multi core processor with centralized shared memory. The first generation comes into two separate chips, the processing engine which is responsible for flows processing and the traffic manger which handles queuing and scheduling functions for the system and I/O. QFP is capable of providing up to 100+ Gbps of packet processing bandwidth inside the chip itself (the entire payload and headers are available for processing). Figure 2.7 illustrate the architecture of CISCO QFP.

The QFB consists of three main components:

A. Multi-Core Packet processor
   - 40 customized multi-threaded Packet Processor Engines (PPE) consolidated into the same silicon.
   - Each PPE is running at a maximum speed of 1.2 GHz.
   - Ti 90nm, 8-layers of metal.
   - Die size 19.54^2 mm^2.
   - 307 million transistors.
   - 20 Mb SRAM
   - 1019 I/O including 800 MHz DDR memory.

B. Queuing engine that handles Buffering, Queuing and Scheduling (BQS) processing.
   - Runs at 400 MHz
   - Buffering, 200K queues, hardware scheduling
   - TI 90nm, 8-layers metal
   - 19.0 x 17.48 mm2
   - 522 million transistors
   - 70 Mb SRAM
   - 1318 I/O, including 800 MHz DDR memory.

C. Software architecture based on ANSI-C environment implemented in a true parallel processing environment.

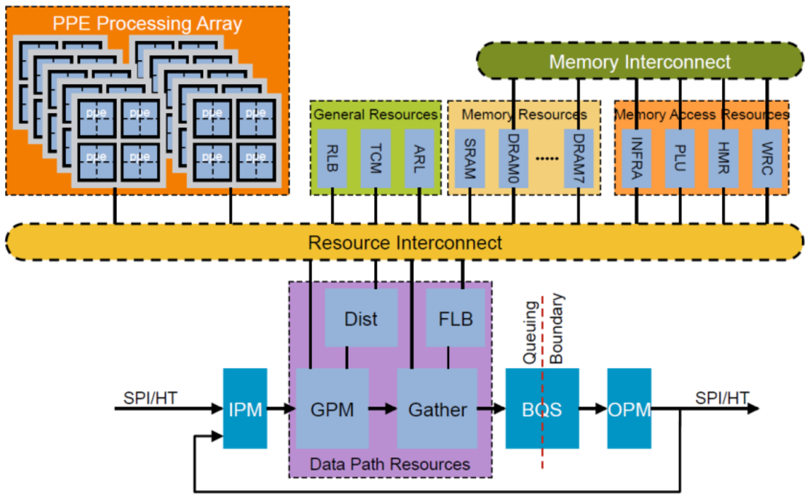

Figure 3.9: CISCO QuantumFlow Processor Architecture

3.4.2. PPE

Each PPE unit is a 4-way 32-bit RISC core which is capable of processing 4 threads at a time (4 independent parallel packets processing). Processing power is optimized and distributed among all 4 threads intelligently. Each PPE can execute 1200 millions of instructions per second (MIPS).

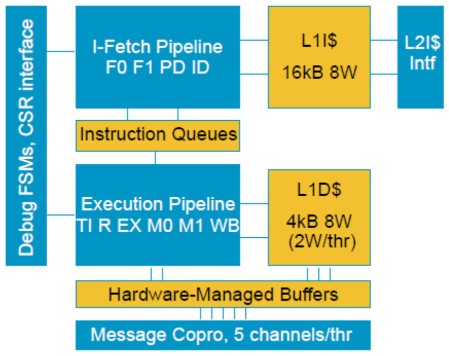

Figure 3.10: Packet Processor Engines (PPE)

3.4.3. Cache

Internal cache consists of two levels: -

- 16 KB of layer 1 cache which is shared across all 4 threads of the same PPE
- 2x256 KB layer 2 cache which are shared across all threads from all PPEs.

Each PPE thread has access to its own layer 1 cache, layer 2 cache of other PPE and to a reduced latency external memory that is already mapped to the address space of each PPE.

Cache coherence is implemented at hardware level so the software doesn't have to worry about it. Additionally processing 160 packets concurrently requires a very sophisticated flow management control to guarantee the correct order of packets outputting the system.

3.4.4. Internal resources

In addition to the 40 PPEs, an array of dedicated hardware engines that works together with every PPE's. These engines accelerate tasks like address and prefix lookups, hash lookups, traffic placer, access control list acceleration, cryptographic accelerator, etc. Any PPE can benefit from these accelerators while simultaneously processing its own packets.

4. Application Specific Integrated Circuits

4.1. History and Background

The term 'ASIC' stands for 'Application-Specific Integrated Circuit'. An ASIC is basically an integrated circuit designed specifically for a special purpose or application. An example of an ASIC is an IC designed for a specific line of cellular phones of a company, whereby no other products can use it except the cell phones belonging to that product line. The opposite of an ASIC is a standard product or general purpose IC, such as a logic gate or a general purpose microcontroller, both of which can be used in any electronic application by anybody. Aside from the nature of its application, an ASIC differs from a standard product in the nature of its availability. The intellectual property, design database, and deployment of an ASIC are usually controlled by just a single entity or company, which is generally the end-user of the ASIC too. Thus, an ASIC is proprietary by nature and not available to the general public. A standard product, on the other hand, is produced by the manufacturer for sale to the general public. Standard products are therefore readily available for use by anybody for a wider range of applications.

The first ASIC's were introduced in 1980. They used gate Array Technology known as uncommitted logic array or ULA's. They had few thousand gates; they were customized by varying the mask for metal interconnections. Thus, the functionality of such a device can be varied by modifying which nodes in the circuit are connected and which are not. Later versions became more generalized, customization of which involve variations in both the metal and poly silicon layers [24].

4.2 Design Flow

Figure 4.1 illustrates the overview of design flow of ASIC. The Design Flow is categorized into Software and Hardware Design. The Hardware Design involves the detailed layout design of the Integrated Circuit. Once it is designed, it is verified and tested. Once the Design passes thee test it is sent for PCB Design and fabricated. The Software Design involves the algorithm which gives the functionality of the IC. Once the designed software is tested, the fabricated hardware and the software algorithm will be merged and tested together before sending into the market.

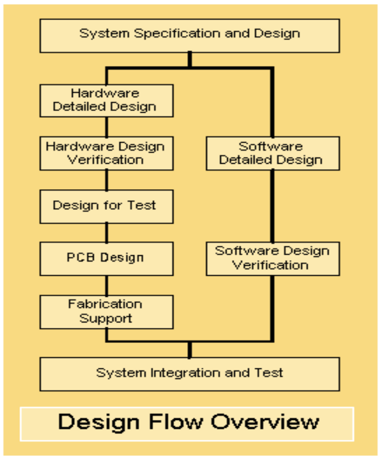
System Specification and Design
Hardware Detailed Design
Hardware Design Verification
Design for Test
PCB Design
Fabrication Support
Software Detailed Design
Software Design Verification
System Integration and Test
Design Flow Overview

Figure 4.1: VLSI Design Flow [25]

4.2. Classification

According to the technology used for manufacturing, ASIC's are usually classified into one of four categories: Full-custom, Semi-custom, Structured and Gate Array. Figure 4.2 depicts these classifications.

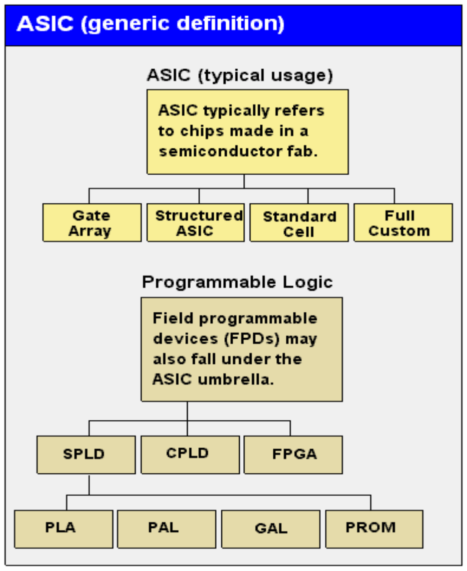
ASIC (generic definition)
ASIC (typical usage)
ASIC typically refers to chips made in a semiconductor fab.
Gate Array
Structured ASIC
Standard Cell
Full Custom
Programmable Logic
Field programmable devices (FPDs) may also fall under the ASIC umbrella.
SPLD
CPLD
FPGA
PLA
PAL
GAL
PROM

Figure 4.2: Classification of ASIC [24]

4.2.1. Full-custom ASIC's

These are those that are entirely tailor-fitted to a particular application from the very start. Since its ultimate design and functionality is pre-specified by the user, it is manufactured with all the photolithographic layers of the device already fully defined, just like most off-the-shelf general purpose IC's. The use of predefined masks for manufacturing leaves no option for circuit modification during fabrication, except perhaps for some minor fine-tuning or calibration. This means that a full-custom ASIC cannot be modified to suit different applications, and is generally produced as a single, specific product for a particular application only.

4.2.2. Semi-custom or Standard Cell ASIC's

Semi-Custom ASIC's on the other hand, can be partly customized to serve different functions within its general area of application. Unlike full-custom ASIC's, semi-custom ASIC's are designed to allow a certain degree of modification during the manufacturing process. A semi-custom ASIC is manufactured with the masks for the diffused layers already fully defined, so the transistors and other active components of the circuit are already fixed for that semi-custom ASIC design. The customization of the final ASIC product to the intended application is done by varying the masks of the interconnection layers, e.g., the metallization layers. Figure 4.3 shows the layout of a standard cell.

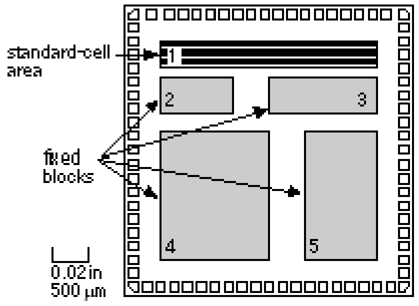

Figure 4.3: Standard Cell [25]

4.2.3. Structured or Platform ASIC's

All the relatively new ASIC classification comes under this category, which have been designed and produced from a tightly defined set of:

- Design methodologies.
- Intellectual properties (IP's)
- Well-characterized silicon, aimed at shortening the design cycle and minimizing the development costs of the ASIC.

A platform ASIC is built from a group of 'platform slices', with a 'platform slice' being defined as a pre-manufactured device, system, or logic for that platform. Each slice used by the ASIC may be customized by varying its metal layers. The re-use of pre-manufactured and pre-characterized platform slices simply means that platform ASIC's are not built from scratch, thereby minimizing design cycle time and costs [28].

4.2.4. Gate Array based ASIC's

Gate-array–based ASIC are transistors which are predefined on the silicon wafer. The predefined pattern of transistors on a gate array is the base array, and the smallest element that is replicated to make the base array is the base cell. Only the top few layers of metal, which define the interconnect between transistors, are defined by the designer using custom masks. To distinguish this type of gate array from other types of gate array, it is often called a masked gate array (MGA). The designer chooses from a gate-array library of predesigned and pre-characterized logic cells. The logic cells in a gate-array library are often called macros. The reason for this is that the base-cell layout is the same for each logic cell, and only the interconnection is customized, so that there is a similarity between gate-array macros and a software macro.

We can complete the diffusion steps that form the transistors and then stockpile wafers (sometimes we call a gate array a pre-diffused array for this reason). Since only the metal interconnections are unique to an MGA, we can use the stockpiled wafers for different customers as needed. Using wafers prefabricated up to the metallization steps reduces the time needed to make an MGA, the turnaround time , to a few days or at most a couple of weeks. The costs for all the initial fabrication steps for an MGA are shared for each customer and this reduces the cost of an MGA compared to a full-custom or standard-cell ASIC design [27].

4.3. Applications of ASIC

- An IC that encodes and decodes digital data using a proprietary encoding/decoding algorithm.
- A medical IC designed to monitor a specific human biometric parameter.
- An IC designed to serve a special function within a factory automation system.
- An amplifier IC designed to meet certain specifications not available in standard amplifier products.
- A proprietary System on Chip and
- An IC that's custom-made for a particular automated test equipment [24].

4.4. Crypto Processor

4.4.1. Introduction

Current applications demand high speed processors for a great amount of data that must be transformed in real-time terms. Software approaches could be a good choice since they have low cost and require a short development time. The low values of performance are a forbidden factor for possible software implementation. A Crypto Processor is microprocessor (an example of ASIC) which is used for cryptographic operations as the name specifies and is embedded in a packaging with multiple physical security measures. As far as the design parameters are concerned, it is designed in such a way that it is Cost-Effective, Secure and perform metrics meet the requirements. On the other hand, hardware alternatives could be selected for implementing Crypto-Processors architectures. Both Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs) can support high data rates, although such designs are more time consuming and expensive compared with the software alternative [28].

A detailed comparison of Hardware v/s Software solutions for implementing Crypto-Processors architectures is introduced in table 4.1 below. Based on the elements of the comparison, hardware solutions are better in most of the cases than the software alternatives. The main advantages of Software are the low cost and the short time to market. Low performance is a fundamental drawback of software integrations [28].

	Software	FPGA	ASIC
Performance	Low	Low	High
Power Consumption	Depends	Very High	Low
Logic Integration	Low	Low	High Integration
Tool Cost	Low	Low	Low
Test Development Complexity	Very Low	Very Low	High
Density	High	Very Low	High
Design Efforts	Low-Medium	Low-Medium	High
Time Consumed	Short	Short	High
Size	Small-Medium	Small	Large
Memory	Fine	Fine	Fine
Flexibility	High	High	-
Time to Market	Short	Short	High
Run Time Configuration	-	High	-

Table 4.1: A Comparison of Hardware vs. Software Solutions for Implementing Crypto-Processors Architectures [28]

A proposed Crypto-Processor for the WTLS implementation is presented in Figure 4.4. The introduced system has been designed like a typical processor with data path, memory, I/O interface, and control unit. Six different ciphers are supported by the proposed Crypto-Processor. DES and IDEA algorithms are selected for the Bulk Encryption Unit. The Reconfigurable Integrity Unit performs efficiently in two different operation modes, for SHA-1 and MD5 hash functions. The operations of both RSA and Diffie-Hellman are performed by the Reconfigurable Authentication Unit. An extra security scheme is also supported by the proposed Crypto-Processor. A Reconfigurable Logic block, in cooperation with the Modified DES Unit, implements the Authorized User Verification Unit. A common data bus of 64-bit and a 32-bit address bus are used for the internal data transfer purposes. Two different storage units have also been integrated. The appropriate for the algorithms keys are stored and loaded in the RAM blocks, while the transformed data are kept as long as necessary in the Transformed Data Registers. A Common Bus Interface Unit, which supports 32-bit input data and 32-bitaddress buses, has been implemented for the Crypto-Processor to communicate efficiently with the external environment. This environment may be a general purpose processor or a special CPU [28].

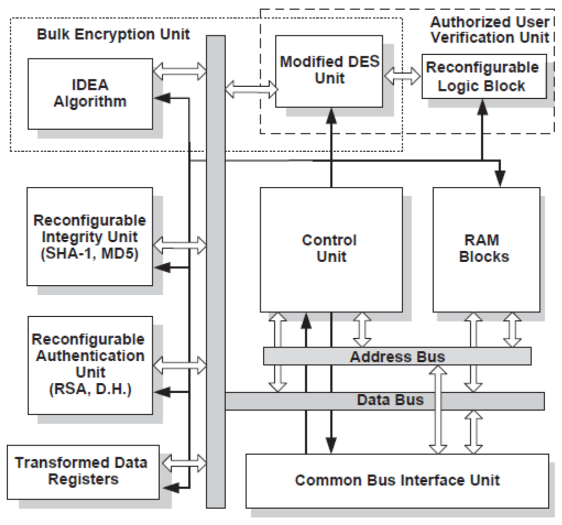
Bulk Encryption Unit
IDEA Algorithm
Modified DES Unit
Authorized User Verification Unit
Reconfigurable Logic Block
Reconfigurable Integrity Unit (SHA-1, MD5)
Control Unit
RAM Blocks
Reconfigurable Authentication Unit (RSA, D.H.)
Address Bus
Data Bus
Transformed Data Registers
Common Bus Interface Unit

Figure 4.4: A Proposed Crypto-Processor for the WTLS Implementation [28]

A proposed Crypto-Processor for SHA-2 hash family standard implementation is illustrated in Figure 4.5. It performs the three different SHA-2 functions (256, 384, and 512). The control unit is totally responsible for the system operation. It defines the proper constants and operation word length, manages the ROM blocks, and controls the proper algebraic and digital logic functions for the operation of SHA-2 hash functions. The Hash Computation Unit is the main data path component of the system architecture. The specified number of the data transformation rounds, for each one of the SHA-2 hash family functions, is performed in this component with the support of a rolling loop (feedback). The transformed data are finally modified in the last transformation, which operates in cooperation with the constants unit. In this way, the message digest is produced and is stored into the Message Digest Register [28].

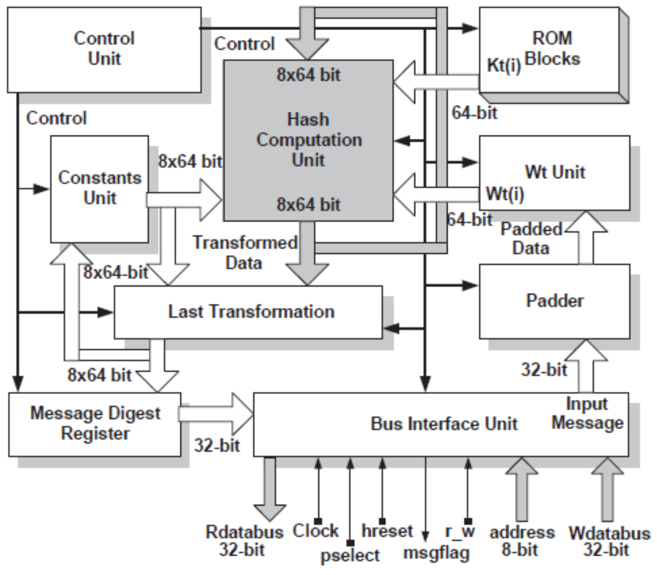

Figure 4.5: A proposed Crypto-Processor for SHA-2 Hash Family Standard Implementation [28]

4.4.2. Throughput Analysis

The performance metrics of different Crypto Processors are as follows

Crypto Chip	Throughput	Application
Cavium Nitrox II	10 GBPS	VPN Gateway, IPSEC & SSL Server connections, Server Load balancing, and Server backup.
HIFN 8350 & HIFN 4350	4 Gbps	Applications for High-end enterprise security services VPNs, Firewalls, Server security and secure storage.
Cavium Nitrox	3.2 Gbps	VPN Gateway, Router Gateways.
Broadcom BCM5840	2.4 Gbps	High performance (wire speed) security applications. This would be connected to the NPU to create a secure solution for an enterprise router or layer 3 switch.
HIFN 8154	2.3 Gbps	Applications for High-end enterprise security services in multi service appliances.

Table 4.2: The performance metrics of different Crypto Processors [28]

4.5. Case Study: Hifn 7902

The HiFn 7902 is a high performance pipelined security processor that integrates a math processor and a random number generator. These blocks provide additional features to support public key cryptography. The integrated algorithms support standard network security protocols. With a minimum amount of external logic, the HiFn 7902 can be interfaced with standard processors such as the RC32355. The internal block diagram of Hifn 7902 is as shown below in Figure 4.6.

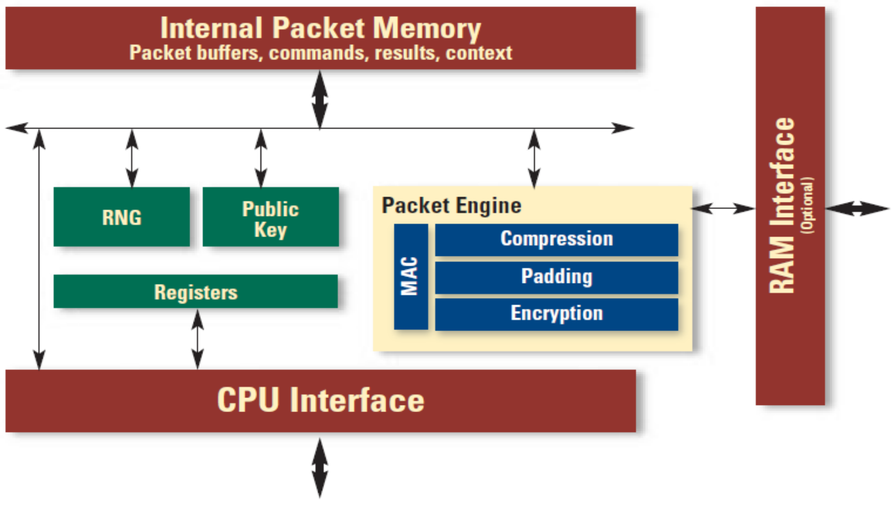

Figure 4.6: The Internal Block Diagram of HiFn 7902 [37]

4.5.1. Interfacing

This application note uses the generic names to show the connections between the two processors. The RC32355 and HiFn 7902 interface is similar to an SRAM in some aspects. Data can be transferred between the CPU and the security processor using the standard load and store commands. The Memory and peripheral bus interface of the RC32355 provides the necessary signals to connect to the HiFn 7902. Since the Burst mode is not recommended for the RC32355, the HiFn 7902 will be used in single-beat transfer mode, i.e., non-burst mode. Figure 4.7 below shows an example interface between the two processors using the non-burst mode. A typical Interface diagram between hifn 7902 and CPU is shown below.

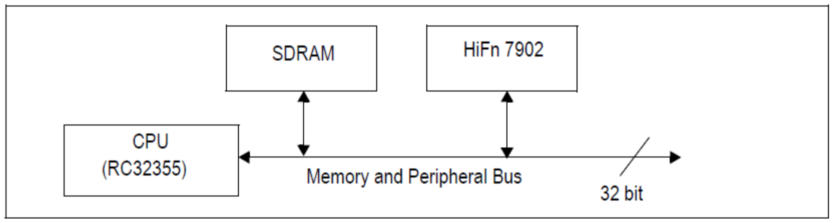

Figure 4.7: Interface Diagram between HiFn 7902 and CPU [38]

4.5.2. Features

• Single chip multi-algorithm acceleration (LZS, 3-DES, SHA, Public Key & more)
• Public Key processing unit (2048-bit key lengths) and true random number generator
• Concurrent symmetric and public Key processing
• Compression (LZS and MPPC)
• Software API and development support
• 32 Mbps processing
• Multi-protocol support: IPSec, IPPCP (IPCOMP), PPTP, L2TP, PPP, and IKE [37]

4.5.3. Advantages

• Off loads compute-intensive work from CPU
• Delivers top security protection and breaks performance bottlenecks
• Concurrent operations
• Improves system throughput
• Reduces time to market
• VPN at full broadband speeds
• Interoperates with all major vendor's equipment [37]

5. Field Programmable Gate Arrays

A field-programmable gate array (FPGA) is an integrated circuit that can be used to implement any logical function that an ASIC could perform. FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together". Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory [29].

5.1. History and Background

The XC2064 was the first commercially viable field programmable gate array in 1985 which was invented by Xilinx Co-Founders, Ross Freeman and Bernard Vonderschmitt. The XC2064 had programmable gates and programmable interconnects between gates. The XC2064 consisted of 64 configurable logic blocks (CLBs), with two 3-input lookup tables (LUTs) [29].

Adrian Thompson, a researcher working at the University of Sussex, merged genetic algorithm technology and FPGAs to create a sound recognition device. Thomson's algorithm configured an array of 10 x 10 cells in a Xilinx FPGA chip to discriminate between two tones, utilizing analogue features of the digital chip. This led to a glimpse of fame for FPGA's in 1997 [29].

5.2. FPGA Board

The Spartan®-6 FPGA SP605 Evaluation Kit shown in Figure 5.1 conveniently delivers all the basic components of the Xilinx Base Targeted Design Platform for developing broadcast, wireless communications, automotive, and other cost- and power sensitive applications that require transceiver capabilities in one package. Along with the development board, cables, and documentation, the new kit provides an integration of hardware, software, IP, and pre-verified reference designs so development can begin right out of the box. The spartan-6 FPGA SP605 Evaluation Kit provides a flexible environment for higher-level system design including applications which need to implement features such as high-speed serial transceivers, PCI Express®, DVI, and/or DDR3. The SP605 development board includes an industry-standard FMC (FPGA Mezzanine Card) connector for scaling and customization to specific applications and markets. The integration of Xilinx development tools help streamlines the creation of systems that adhere to complex requirements [35].

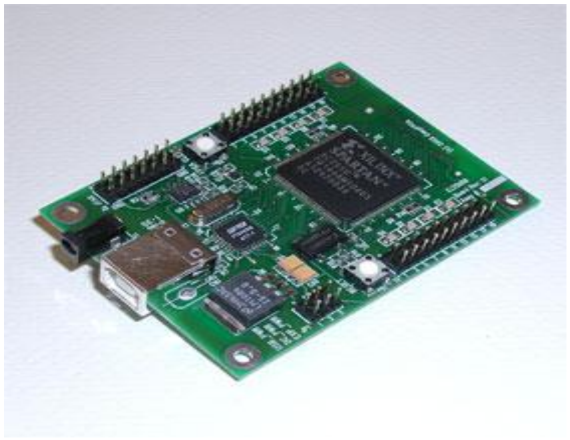

Figure 5.1: Xilinx Spartan 6 FPGA Evaluation board

5.3. FPGA Architecture

The Architectural Diagram describes FPGA as an array-style FPGA. It consists of an array of programmable logic blocks of potentially different types, including general logic, memory and multiplier blocks, surrounded by a programmable routing fabric that allows blocks to be interconnected. The array is surrounded by programmable input/output blocks that connect the chip to the outside world. A FPGA architecture is depicted in Figure 5.2. The "programmable" term in FPGA indicates an ability to program a function into the chip after silicon fabrication is complete [33].

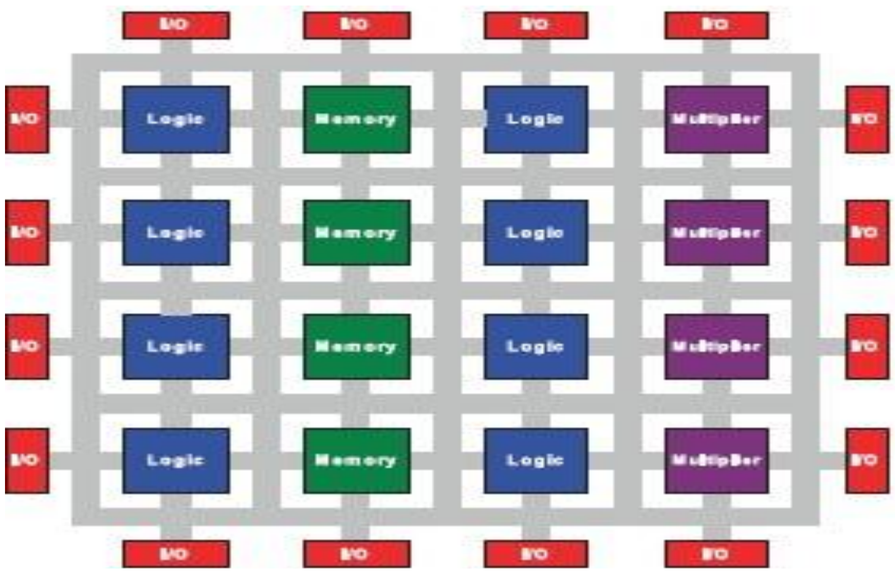

Figure 5.2: An Architectural diagram of FPGA

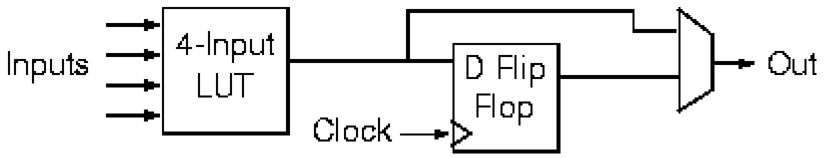

Figure 5.3: Logic Block Structure

The FPGA logic block consists of a 4-input look-up table (LUT), and a flip flop, as shown below. There is only one output, which can be either the registered or the unregistered LUT output. The logic block has four inputs for the LUT and a clock input [30]. An example of a logic block is depicted in Figure 5.3.

5.3.1. FPGA Interconnect

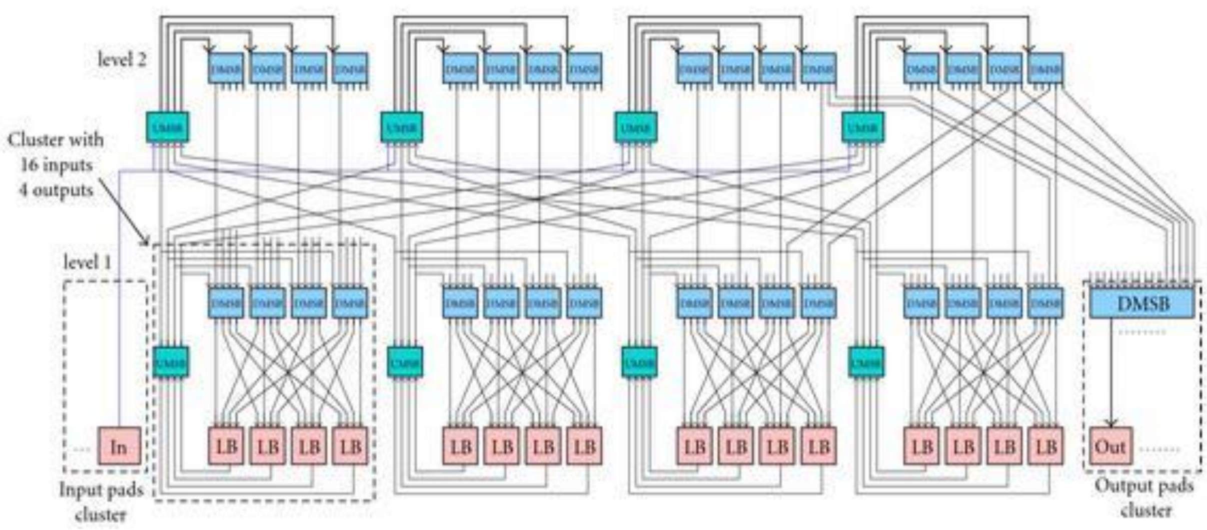

Figure 5.4: Tree-based interconnect: upward and downward networks

The Tree-based FPGA interconnect architecture unifies two unidirectional networks. The downward network uses a “Butterfly Fat Tree’’ topology to connect DMSBs to LBs inputs. The upward network connects LBs outputs to the DMSBs (Downward Mini-switch Blocks) at each level. We use UMSBs (Upward MSBs) to allow LBs outputs to reach a large number of DMSBs and to reduce fan-out on feedback lines. UMSBs are organized in a way allowing LBs belonging to the same “owner cluster’’ to reach exactly the same set of DMSBs at each level. Thus positions inside the same cluster are equivalent, and LBs can negotiate with their siblings [32]. This interconnection is shown in Figure 5.4.

5.4. Memory Control Block

The MCB is a dedicated embedded block multi-port memory controller that greatly simplifies the task of interfacing Spartan devices to the most popular memory standards. It provides significantly higher performance, reduced power consumption, and faster development times than equivalent IP implementations. The embedded block implementation of the MCB conserves valuable FPGA resources and allows the user to focus on the more unique features of the FPGA design [31]. This is depicted in Figure 5.5.

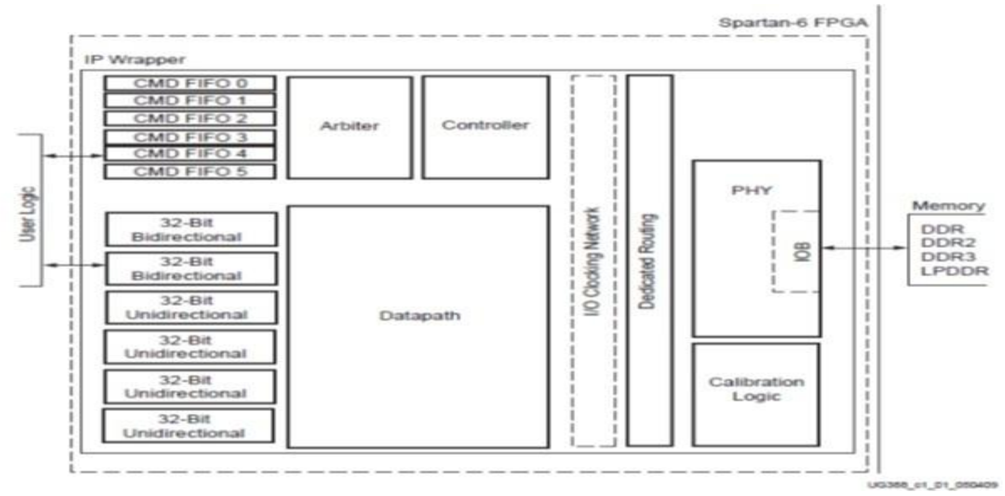

Figure 5.5: Spartan FPGA memory controllers block (MCB)

The single data rate (SDR) user interface to the MCB inside the FPGA can be configured for one to six ports, with each port consisting of a command interface and a read and/or write data interface.
The two 32-bit bidirectional and four 32-bit unidirectional hardware-based ports inside the MCB can be grouped to create five different port configurations. Other major components of the MCB include:

- Arbiter: Determines which port currently has priority for accessing the memory device.
- Controller: Primary control block that converts the simple requests made at the user interface into the necessary instructions and sequences required to communicate with the memory.
- Data path: Handles the flow of write and read data between the memory device and the user logic.
- Physical Interface (PHY): Converts the controller instructions into the actual timing relationships and DDR signaling necessary to communicate with the memory device.
- Calibration Logic: Calibrates the PHY for optimal performance and reliability.

5.5. FPGA in Embedded Processors

Embedding a processor inside an FPGA has many advantages. A variety of memory controllers enhance the FPGA embedded processor systems interface capabilities. FPGA embedded processors use general-purpose FPGA logic to construct internal memory, processor busses, internal peripherals, and external peripheral controllers (including external memory controllers). As more pieces (busses, memory, memory controllers, peripherals, and peripheral controllers) are added to the embedded processor system, the system becomes increasingly more powerful and useful. However, these additions reduce performance and increase the embedded system cost, consuming FPGA resources.

Xilinx publishes a Spartan-3 MicroBlaze benchmark of 65 DMIPs (Dhrystone MIPs) running at 85 MHz. This case study investigates the system design for which this benchmark was achieved and compares it to other, more real-world processor systems. The IP included in this embedded processor design are: MicroBlaze, hardware divider and barrel shifter included, no data or instruction cache, Data-side peripheral bus (no instruction-side), 8KB instruction local memory, 16KB data local memory, No debug hardware, RS232 [34]. Figure 5.6 represents Spartan-3 DMIPs System Block Diagram.

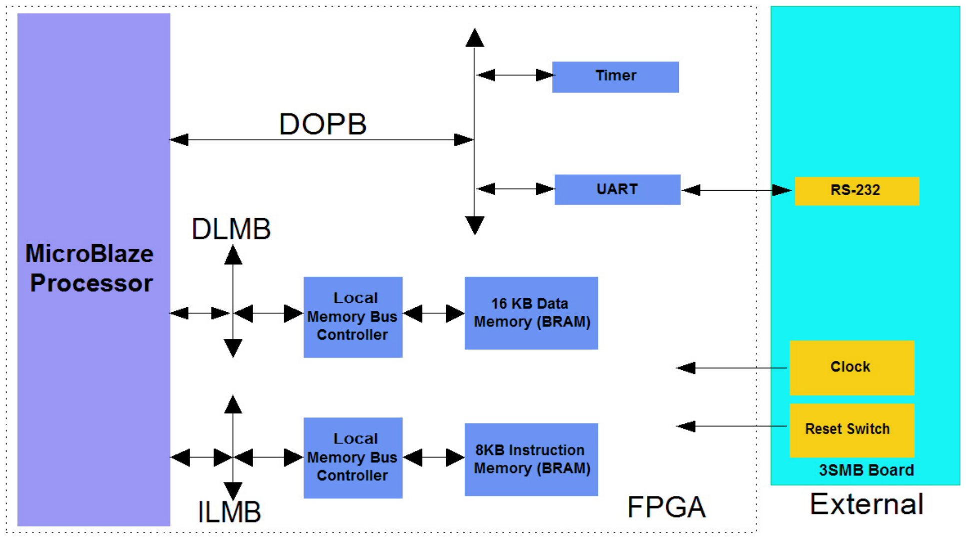

Figure 5.6: Spartan-3 DMIPs System Block Diagram

5.6. FPGA in Digital Signal Processors

Advancements in Field Programmable Gate Arrays (FPGAs) provide new options for DSP design engineers. When a design demands the use of a DSP, or time-to-market is critical, or design adaptability is crucial, then the FPGA may offer a better solution. The SRAM-based FPGA is well suited for arithmetic, including Multiply & Accumulate (MAC) intensive DSP functions. A wide range of arithmetic functions (such as Fast Fourier Transform's (FFT's), convolutions, and other filtering algorithms) can be integrated with surrounding peripheral circuitry.

When building a DSP system in an FPGA, the design can take advantage of parallel structures and arithmetic algorithms to minimize resources and exceed the performance of single or multiple general-purpose DSP devices. Distributed Arithmetic for array multiplication in an FPGA is one way to increase data bandwidth and throughput by several orders of magnitudes over off-the-shelf DSP solutions [36].

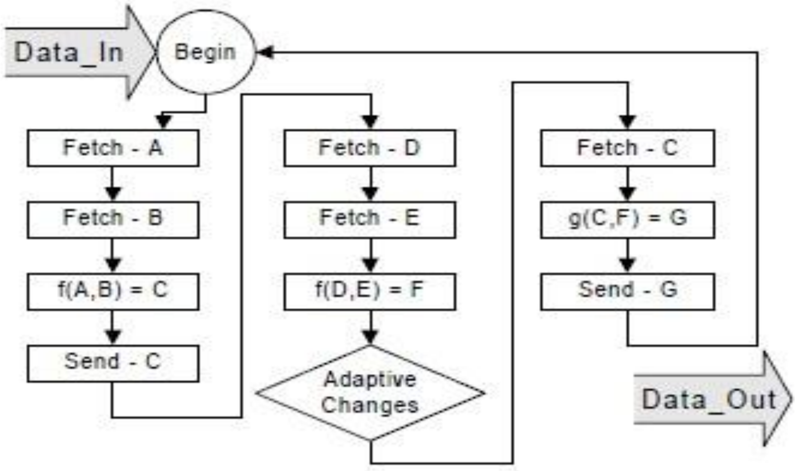

Figure 5.7: DSP Algorithm Process

The FPGA is well suited for many DSP algorithms and functional routines. The FPGA can be programmed to perform any number of parallel paths. These operational data paths can consist of any combination of simple and complex functions, such as; Adders, Barrel Shifters, Counters, Multiply and Accumulation, Comparators and Correlators just to mention a few. The FPGA can also be partially or completely reconfigured in the system for a modified or completely different algorithm. Figure 5.7 shows a DSP algorithm process.

5.6.1. Case Study: Viterbi Decoder

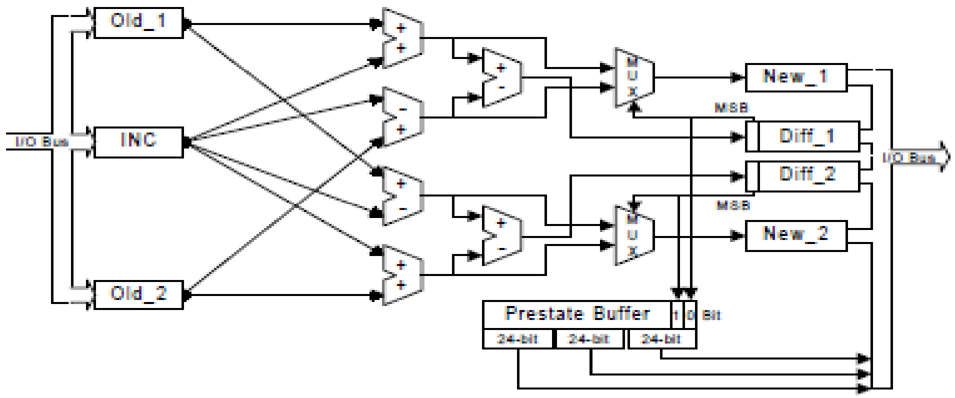

Figure 5.8: Block Diagram of a Viterbi Decoder

A Viterbi Decoder [36] is a good example of how the FPGA can accelerate a function. The Viterbi Decoder algorithm requires 360 nsec [(24-clock cycles) x (15 nsec)] of processing time. There are two limiting factors for this DSP-based design. First, the wait state associated with the DSP's external SRAM memory requires two 15 nsec clock cycles for each memory access. Hence, the data Bus transfer requires 30 nsec for each data transaction. This forced the I/O Bus speed to a maximum of 30 nsec. Secondly, each Add/Subtract and MUX stage has to be performed sequentially with additional wait-states. The Add/Subtract stages require four additional operations with multiple instructions. A diagram of a Viterbi decoder is depicted in Figure 5.8.

5.6.2. Case Study: FPGA Based Viterbi Decoder

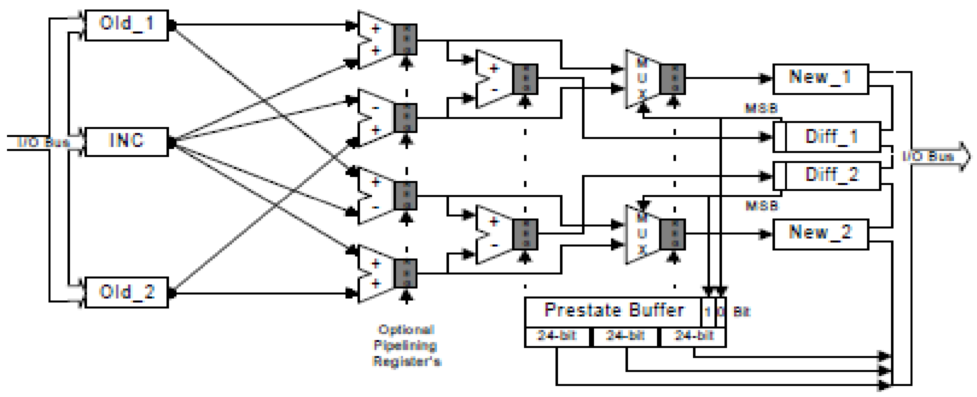

Figure 5.9: Block Diagram of an FPGA based Viterbi Decoder

The Viterbi Decoder is well suited for the FPGA. The ability to process parallel data paths within the FPGA takes advantage of the parallel structures of the four Add/Subtract-blocks in the first stage and the two Subtract-blocks in the second stage. The two MUXblocks take advantage of the ability to register and hold the input data until needed with no external memory or additional clock cycles. The FPGA based Viterbi Decoder requires 135 nsec [(9-clock cycles) x (15 nsec)] of total processing time compared to the 360 nsec required for the partial output data by the DSP. This enhancement equates to 37.5% of the original DSP processing time or 62.5% better processing performance [36]. Block diagram of an FPGA based Viterbi decoder is depicted in Figure 5.9.

References

[1] Nohl, A, Schirrmeister, F & Taussig, D. "*Application specific processor design: Architectures, design methods and tools*" Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on Nov. 2010.

[2] Karam, L.J.; AlKamal, I.; Gatherer, A.; Frantz, G.A.; Anderson, D.V.; Evans, B.L.,"*Trends in multicore DSP platforms,*" Signal Processing Magazine, IEEE, vol.26, no.6, pp.38-49, November 2009

[3] Digital Signal Processing, Wikipedia, http://en.wikipedia.org/wiki/Digital_signal_processor.

[4] TNETV3020 carrier infrastructure platform, Texas Instruments, http://focus.ti.com/lit /ml/spat174a/spat174a.pdf, January 2007

[5] MSC8156 product brief, Freescale, http://www.freescale.com/webapp/sps/site/prod_summary .jsp?code=MSC8156&nodeId=0127950E5F5699, December 2008

[6] Texas Instruments TMS320, Wikipedia, http://en.wikipedia.org/wiki/ Texas Instruments TMS320# C6000_Series.

[7] "*TMS320C66x DSP CPU and Instruction Set*", Texas Instruments, SPRUGH7, November 2010.

[8] "*TMS320C66x DSP CorePac*" Texas Instrument User Guide, SPRUGW0B, November 2010.

[9] "*TI's new TMS320C66x fixed- and floating-point DSP core conquer the need for speed*", Texas Instruments, SPRY147, November 2010

[10] Michael B. Rash, "*Intrusion prevention and active response: deploying network and host IPS Syngress*", Apr 12, 2005, page 63

[11] Niraj Shah, "*Understanding Network Processors*", September 2001.

[12] Jain, M.K.; Balakrishnan, M.; Kumar, A.; "*ASIP design methodologies: survey and issues*" VLSI Design, 2001. Fourteenth International Conference on 2001, Page(s): 76 – 81, 10.1109/ICVD.2001.902643

[13] Douglas Comer, "*Network Processors: Programmable Technology for Building Network Systems*", *The Internet Protocol Journal - Volume 7, Number 4*, December 2004 pages 2-12

[14] "*Application-specific instruction-set processor*"http://en.wikipedia.org/wiki/Application-specific_instruction-set_processor

[15] Mohammad Peyravian, Jean Calvignac, "*Fundamental architectural considerations for network processors*" Computer Networks, 41, 2003, page 587-600

[16] Keutzer, K.; Malik, S.; Newton, A.R.; "*From ASIC to ASIP The Next Design Discontinuity*" Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on 2002, Page(s): 84 – 90, 10.1109/ICCD.2002.1106752

[17] Peter Brink,Manohar Castelino, David Meng, Chetan Rawal,Hari Tadepalli, "*Network Processing Performance Metrics for IA- and IXP-Based Systems*" Intel Technology Journal, Volume 7, Issue 4, 2003

[18] Liem, C, May, T, & Paulin, P." *Instruction-set matching and selection for DSP and ASIP code generation*" European Design and Test Conference, 1994. EDAC, The European Conference on Design Automation. ETC European Test Conference. EUROASIC, The European Event in ASIC Design, Proceedings, 10.1109/EDTC.1994.326902, 1994 , Page(s): 31 – 37.

[19] Cox S., Goossens G., Brockmeyer E. "*ASIPs Programmable Accelerators for Multicore SoCs*", Target Compiler Technologies, SoC Conference, Newport Beach, CA , November 4, 2009

[20] Kohler, Mark. "*NP Complete*," Embedded Systems Programming, November 2000, pp. 45-60.

[21] *“The Cisco QuantumFlow Processor: Cisco’s Next Generation Network Processor”*, 2008 Cisco Systems.

[22] Will Eatherton, Don Steiss,James Markevitch "*The QFP Packet Processing Chip Set*" Cisco Systems, Cisco Development Organization

[23] M. Peyravian, J. Calvignac , Computer Networks 41 (2003) 587–600

[24] Application-Specific Integrated Circuits (ASIC's)http://www.siliconfareast.com/asic.htm

[25] “*Application-Specific Integrated Circuits*” by Michael John Sebastian Smith.

[26] Crypto Chips http://informatik.uibk.ac.at/teaching/ws2009/esa/crypto_slides.pdf

[27] Semiconductors: Overview & Glossary http://www.xtensiblesolutions.com/Papers/2008-04/SemiconductorsOverviewAndGlossary.htm

[28] Nicolas Sklavos “*On the Hardware Implementation Cost of Crypto-Processors Architectures*” Information Security Journal: A Global Perspective, 19:53–60, 2010.

[29] “*Field Programmable Gate Arrays*”, http://en.wikipedia.org/wiki/Field-programmable_gate_array.

[30] Vaughan, “*FPGA Architecture for Challenge*”.

[31] “*Spartan-6 FPGA Memory Controller*”, User Guide.

[32] Zied Marrakchi, Hayder Mrabet, Umer Farooq, and Habib Mehrez, “*FPGA Interconnect Topologies Exploration*”, 2009.

[33] Ian Kuon1, Russell Tessier2 and Jonathan Rose1, “*FPGA Architecture: Survey and Challenges*”, Foundation and trends in Electronic Design and automation, 2008.

[34] B. Fletcher, Memec, “*FPGA Embedded Processors: Revealing True System Performance*”, Embedded Systems Conference, San Francisco, 2005.

[35] Xilinx, “*Spartan – 6 FPGA Evaluation Kit*”, San Jose, 2009.

[36] Gregory Ray Goslin, “*A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance*” Xilinx Inc. 1995.

[37] 7902 Datasheet, http://www.digchip.com/datasheets/parts/datasheet/191/7902.php.

[38] Interfacing 7902 with RC 32334, http://www.datasheetarchive.com/RC3233x/Datasheet-082/DASF0050090.html