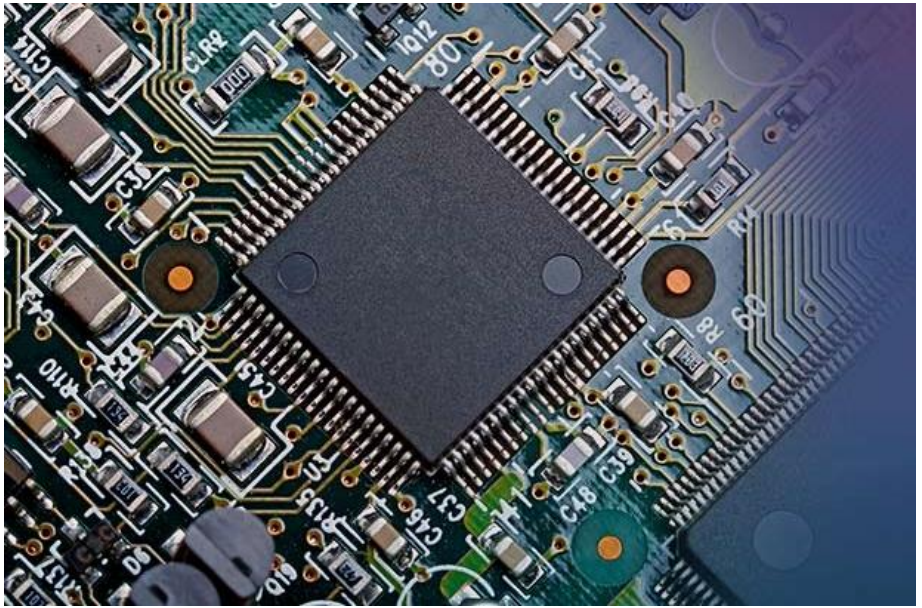
Co-Processor

Introduction

Secure coprocessors enable secure distributed applications by providing safe havens where an application program can execute (and accumulate state), free of observation and interference by an adversary with direct physical access to the device. However, for these coprocessors to be effective, participants in such applications must be able to verify that they are interacting with an authentic program on an authentic, untampered device. Furthermore, secure coprocessors that support *general-purpose* computation and will be manufactured and distributed as *commercial products* must provide these core sanctuary and authentication properties while also meeting many additional challenges.



History of Coprocessor

Coprocessors for floating-point arithmetic first appeared in desktop computers in the 1970s and became common throughout the 1980s and into the early 1990s. Early 8-bit and 16-bit processors used software to carry out floating-point arithmetic operations. Where a coprocessor was supported, floating-point calculations could be carried out many times faster. Math coprocessors were popular purchases for users of computer-aided design(CAD) software and scientific and engineering calculations.

Overview

- Extends the processing features of a core by extending the instruction set.
- More than one coprocessor can be added to the ARM core via the coprocessor interface.
- ARM processor uses coprocessor 15 registers to control the cache, TCMs, and memory management.
- ARM instruction set to process vector floating-point (VFP) operations.
- Extend the instruction set by providing a specialized group of new instructions.
- The architecture is fully compatible across generations with backward binary compatibility.

Properties of Coprocessor

- Without a primary microprocessor, the coprocessor cannot function.
- Main processor has to identify and segregate computationally intensive instructions in a program.
- The instructions which have an intensive amount of calculations are performed by a coprocessor.
- The main processor handles all other activities.

Advantages

- Save register usage
- Fetch data immediately when it is ready at WB stage
- Easy coprocessor task generation; basic block grouping
- Full control of instruction synthesis
- Maximize parallelism; address calculations are also performed
- Memory I/O task given to base processor
- No branch calculations

Disadvantages

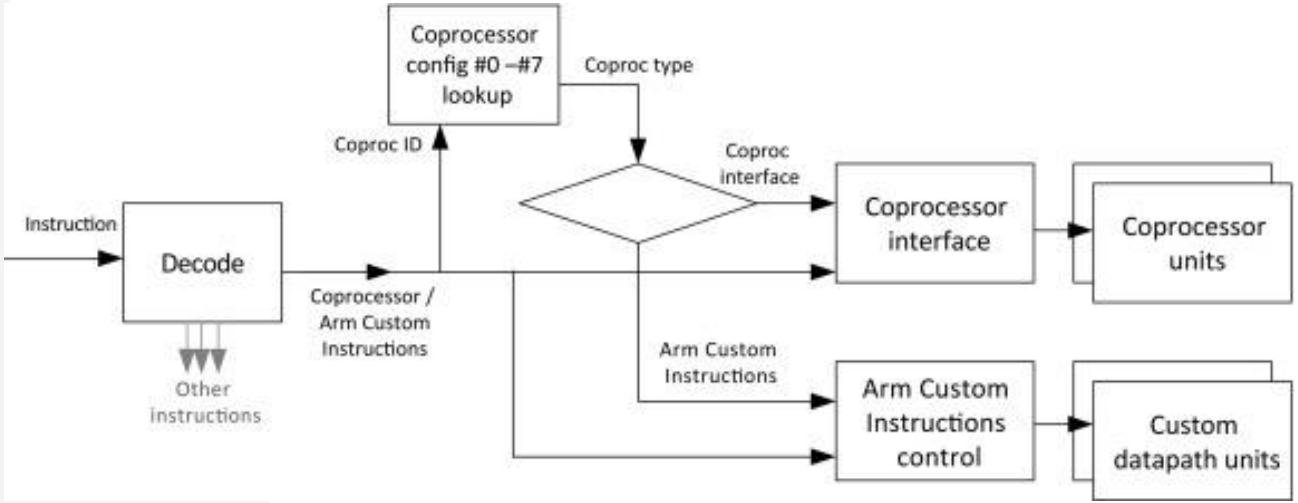
- The disadvantage (besides just materials cost) is that you will have to figure out how to power it, mount it, and interface with it.
- It’s a significant step up in the complexity of your control system and introduces new potential points of failure.
- Many co-processors don’t like being turned off suddenly, require configuration to start your program at boot, can’t take a ton of vibration, etc.

Architecture

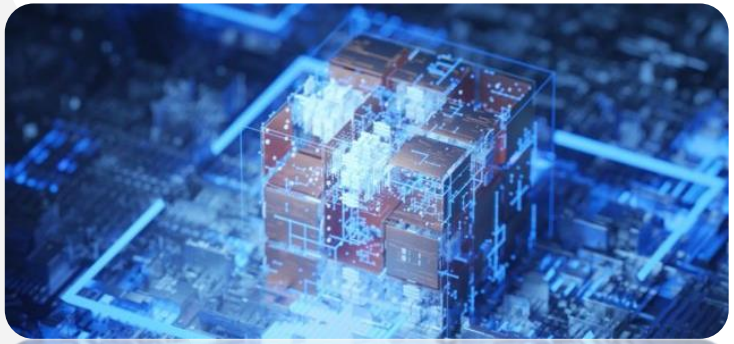
- The coprocessor interface and the Arm Custom Instructions features, both use the concept of a coprocessor ID value. The instruction set architecture uses a bitfield (4-bit) to define which coprocessor is accessed, meaning, in theory, that there could be up to 16 coprocessors attached to a processor. However, only coprocessors ID #0–#7 are allocated for custom-defined solution(s), with these eight units being either:
- coprocessors connected via the coprocessor interface, or
- custom data path units inside the processor.
- The coprocessor ID #0–#7 are shared between the coprocessor interface and Arm custom instructions.

Architecture

- For each coprocessor ID, chip designers need to decide, which way the instruction should be handled, that is, as a coprocessor or as a custom data path unit. This is a decision that needs to be made during the chip design stage.



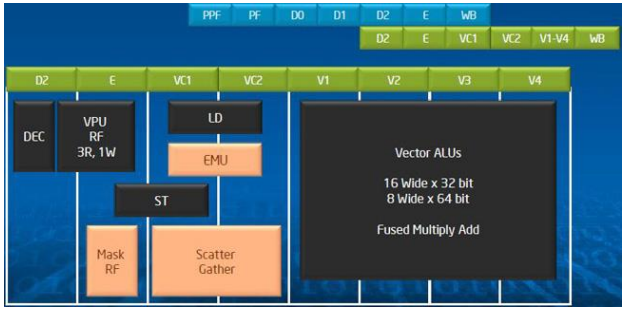
Structure of Coprocessor



- RISC machine
- 64-bit addressing and 64-bit data
- Increased bandwidth
- Fault tolerance
- Nine stage pipeline; can do up to 4 instructions per cycle
- On-chip 16Kb data and instruct. Caches with 2Mb external cache

Coprocessor Architecture (Example)

The Intel Xeon Phi coprocessor is connected to an Intel Xeon processor, also known as the “host”, through a PCI Express (PCIe) bus. Since the Intel Xeon Phi coprocessor runs a Linux operating system, a virtualized TCP/IP stack could be implemented over the PCIe bus, allowing the user to access the coprocessor as a network node.



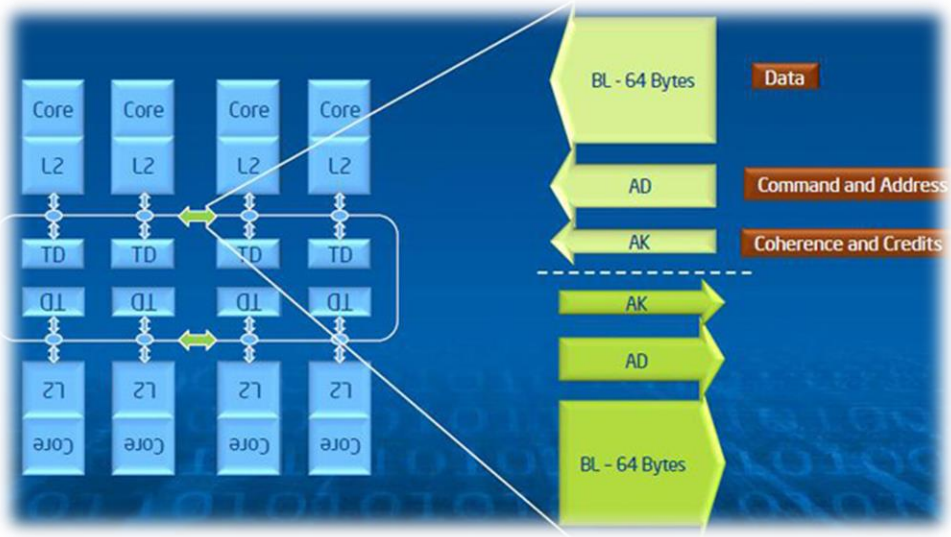
Vector Processing Unit: An important component of the Intel Xeon Phi coprocessor’s core is its vector processing unit (VPU), shown in Figure. The VPU features a novel 512-bit SIMD instruction set, officially known as Intel® Initial Many-Core Instructions (Intel IMCI).

Coprocessor Architecture

The Interconnect: The interconnect (Figure 6) is implemented as a bidirectional ring. Each direction is comprised of three independent rings. The first, largest, and most expensive of these is the data block ring. The data block ring is 64 bytes wide to support the high bandwidth requirement due to a large number of cores.

Caches:

The Intel MIC architecture invests more heavily in L1 and L2 caches compared to GPU architectures.



Coprocessor Architecture

The Intel Xeon Phi coprocessor implements a leading-edge, very

high bandwidth memory subsystem. Each core is equipped with

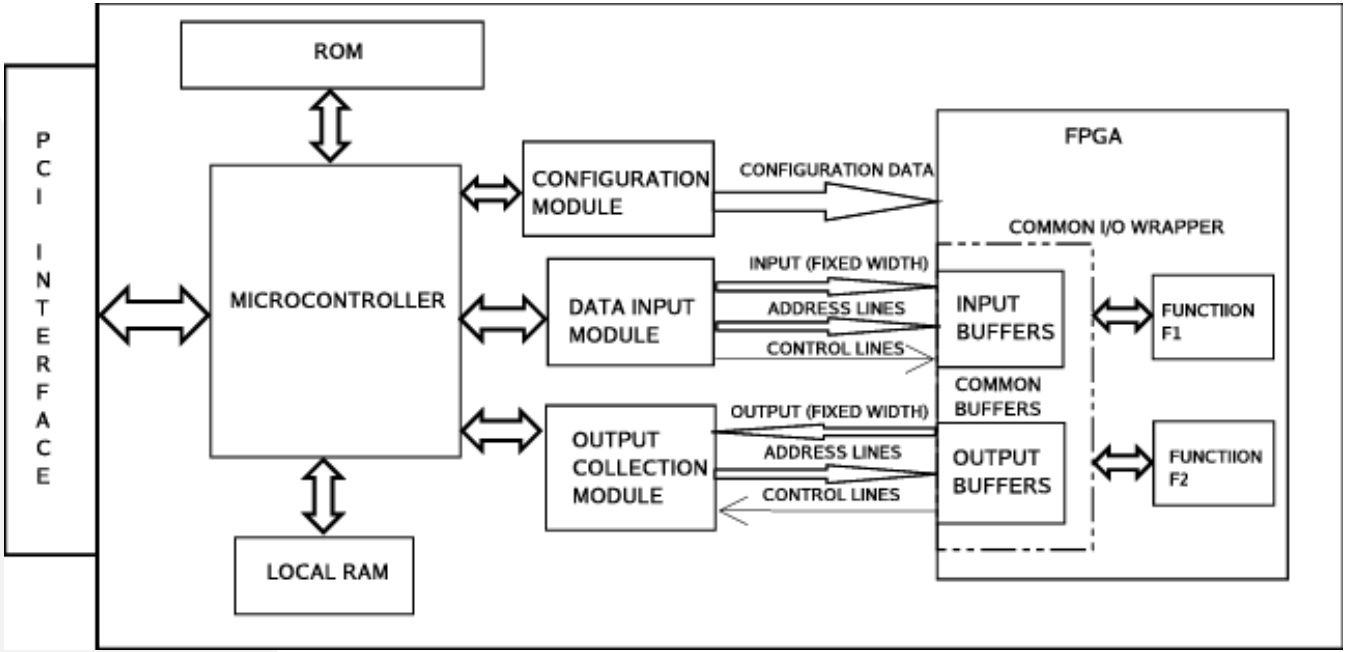
a 32KB L1 instruction cache and 32KB L1 data cache and a 512KB unified L2 cache. These caches are fully coherent and implement the x86 memory order model.

Stencils:

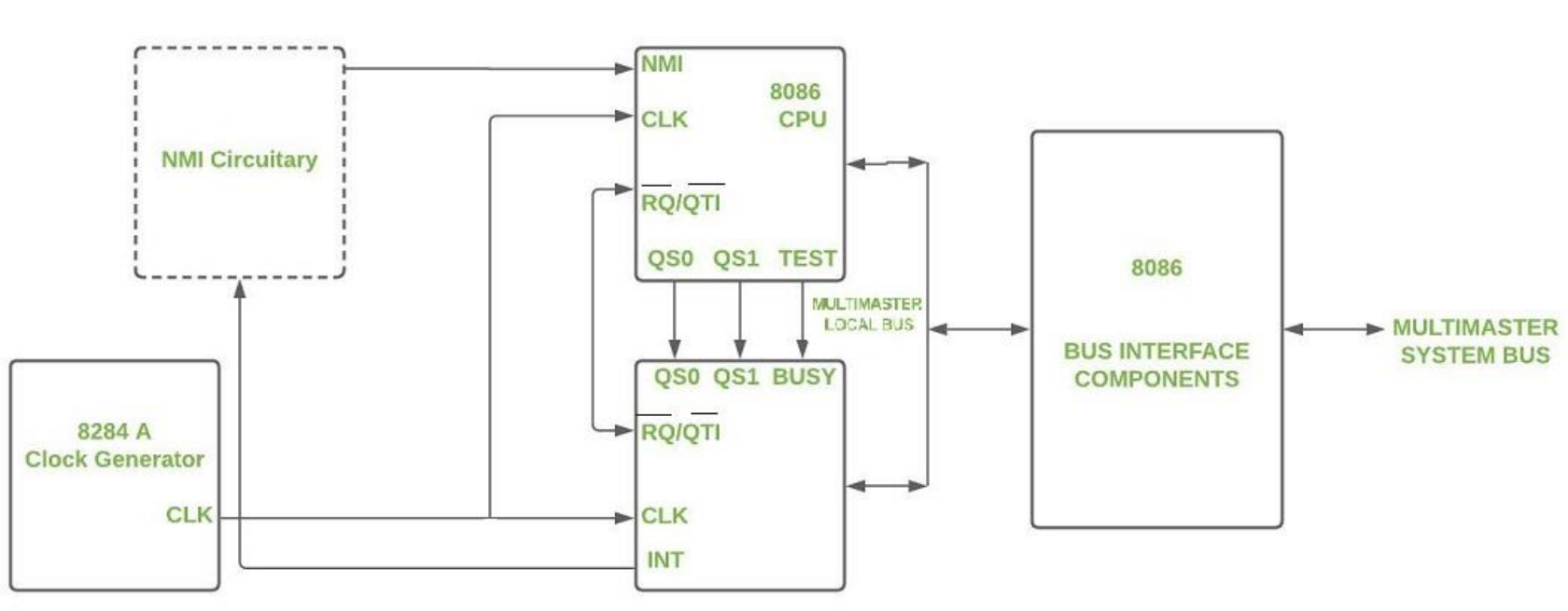
Stencils are common in physics simulations and are classic examples of workloads that show a large performance gain through

efficient use of caches

Block Diagram of Coprocessor



Functional Diagram of Co-processor



Conclusion

- To achieve independent access to memory data for high utilization of cores.
- History of Coprocessor is shown.
- The characteristics of Coprocessor are presented.
- The advantages and disadvantages are Coprocessor presented.
- The basic modules of the coprocessor, namely the vector processing unit, and caches units are studied, in detail.
- The internal operation and capabilities of Coprocessor have been given.
- The Block Diagram of the coprocessor and Functional Diagram Coprocessor is shown.

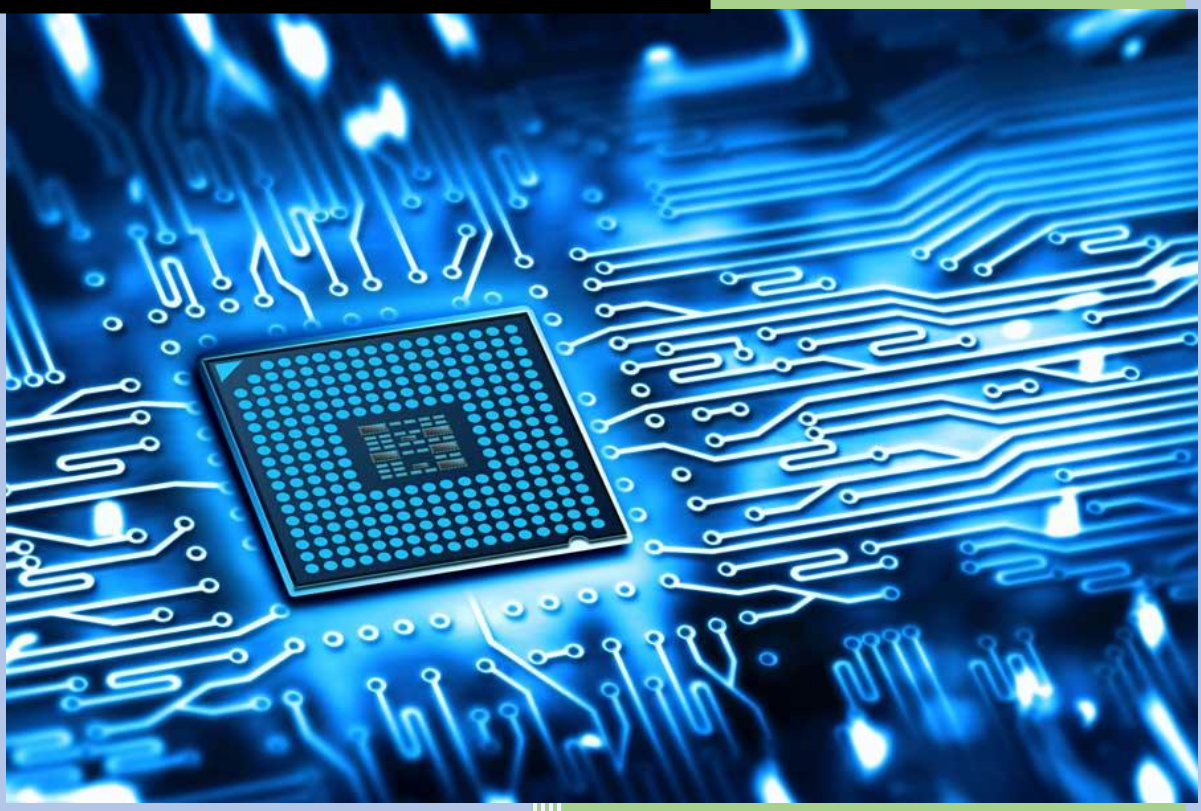
REFERENCES:-

- https://www.geeksforgeeks.org/co-processor-computer-architecture/
- https://en.wikipedia.org/wiki/Coprocessor
- https://www.sciencedirect.com/topics/engineering/coprocessor
- https://www.techopedia.com/definition/2856/coprocessor
- https://www.researchgate.net/publication/220760609_Case_Study_of_Integration_of_Reconfigurable_Logic_as_a_Coprocessor_into_a_SCI-Cluster_under_RT-Linux

THANK YOU !

CO CASESTUDY

CO-PROCESSOR



Submitted By:

Om Prakash Mohanta

What is a Co-processor?

A coprocessor can be attached to the ARM processor. A coprocessor extends the processing features of a core by extending the instruction set or by providing configuration registers. More than one coprocessor can be added to the ARM core via the coprocessor interface.

Overview:

- Extends the processing features of a core by extending the instruction set.
- More than one coprocessor can be added to the ARM core via the coprocessor interface.
- ARM processor uses coprocessor 15 registers to control the cache, TCMs, and memory management.
- ARM instruction set to process vector floating-point (VFP) operations.
- Extend the instruction set by providing a specialized group of new instructions.

Abstract:

Secure coprocessors enable secure distributed applications by providing safe havens where an application program can execute (and accumulate state), free of observation and interference by an adversary with direct physical access to the device. However, for these coprocessors to be effective, participants in such applications must be able to verify that they are interacting with an authentic program on an authentic, untampered device. Furthermore, secure coprocessors that support *general-purpose* computation and will be manufactured and distributed as *commercial products* must provide these core sanctuary and authentication properties while also meeting many additional challenges, including:

- The applications, operating system, and underlying security management may all come from different, mutually suspicious authorities;
- Configuration and maintenance must occur in a hostile environment while minimizing disruption of operations;
- The device must be able to recover from the vulnerabilities that inevitably emerge in complex software;
- Physical security dictates that the device itself can never be opened and examined; and
- ever-evolving cryptographic requirements dictate that hardware accelerators be supported by reloadable on-card software.

Co-Processor:

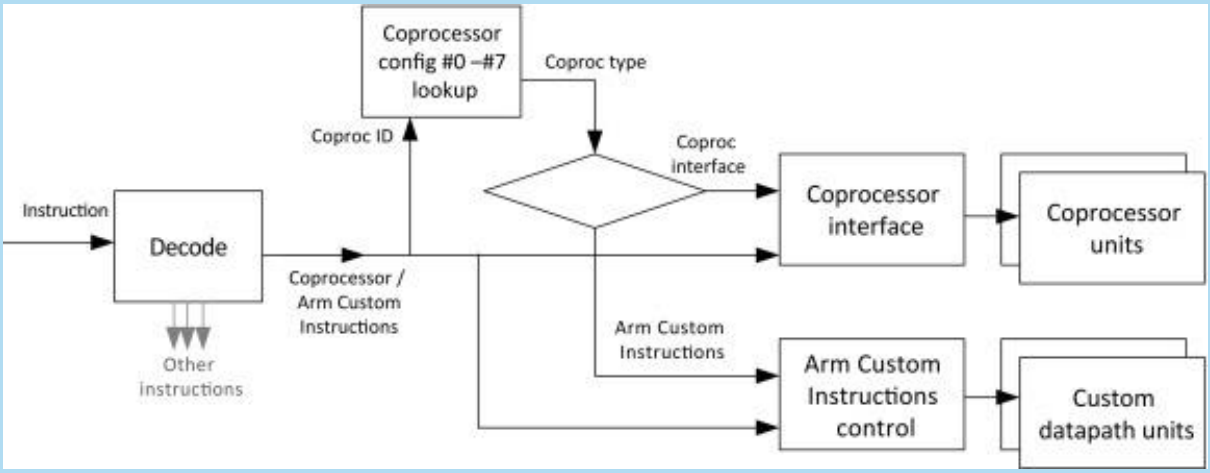
The coprocessor can be accessed through a group of dedicated ARM instructions that provide a load-store type interface. Consider, for example, coprocessor 15: The ARM processor uses coprocessor 15 registers to control the cache, TCMs, and memory management.
The coprocessor can also extend the instruction set by providing a specialized group of new instructions. For example, there are a set of specialized instructions that can be added to the standard ARM instruction set to process vector floating-point (VFP) operations.
These new instructions are processed in the decode stage of the ARM pipeline. If the decode stage sees a coprocessor instruction, then it offers it to the relevant coprocessor. But if the coprocessor is not present or doesn't recognize the instruction, then the ARM takes an undefined instruction exception, which allows you to emulate the behavior of the coprocessor in software.

Overview of the architecture:

The coprocessor interface and the Arm Custom Instructions features, both use the concept of a coprocessor ID value. The instruction set architecture uses a bitfield (4-bit) to define which coprocessor is accessed, meaning, in theory, that there could be up to 16 coprocessors attached to a processor. However, only coprocessors ID #0–#7 are allocated for custom-defined solution(s), with these eight units being either:

- coprocessors connected via the coprocessor interface, or
- custom data path units inside the processor.

Other coprocessor numbers are reserved for internal use by the Arm processor (e.g., CP10 and CP11 are used by the FPU and by the Helium processing unit for the Cortex-M55 processor).
Please note, that the software cannot change how each coprocessor ID is used.
The coprocessor ID #0–#7 are shared between the coprocessor interface and Arm custom instructions. For each coprocessor ID, chip designers need to decide, which way the instruction should be handled, that is, as a coprocessor or as a custom data path unit. This is a decision that needs to be made during the chip design stage.
The instruction encoding of the coprocessor interface operation and Arm Custom Instructions overlaps, and, as such, the instruction decode logic inside the processor needs to take account of the coprocessor hardware configuration that was set up by the chip designer.



Parallel Computer Architectures Using Fiber Optics:

The Warp processor was designed in a project at Carnegie Mellon University (CMU). It was designed for computationally intensive parallel processing on large data sets as is common in signal and image processing. The name of the processor, Warp, refers to an image processing operation.
The processors were designed to operate in a pipeline, each processor receiving data from the previous processor and sending processed data on to the next processor in the system. This kind of processing avoids many synchronization and communication problems that occur in more complex network topologies.
Such processing sometimes has problems with latency (the time necessary to fill or exhaust the pipeline) but in the data-intensive processing that the Warp was designed for latency issues are small relative to the size of the data set and the required computation per data point.
Each Warp processor had a state-of-the-art floating-point coprocessor (pipelined also) and fast integer arithmetic. Each processor was controlled by a VLIW (very long instruction word) microprocessor that allowed maximal internal parallelism because the powerful components of this processor could be addressed and controlled separately.
This machine is an early example of a powerful MIMD processor. However, the topology of this machine and the processing expected of it are such that optical components and in particular optical buses are not necessary. It was designed to avoid the issues optical components are designed to address.

Floating Point Operations:

CPACR register:

The CPACR register allows you to enable or disable the FPU. It is located at address 0xE000ED88 and can be accessed as “SCB->CPACR” in CMSIS-Core. Bit 0 to bit 19 and bit 24 to bit 31 are not implemented and are reserved (Figure 13.12).

Co-processor access control register (SCB->CPACR, 0xE000ED88)

This programmer’s model of this register provides enables control for up to 16 co-processors. On the Cortex®-M4, the FPU is defined as co-processors 10 and 11. Since there is no other co-processor, only CP10 and CP11 are available and both are for the FPU. When programming this register, the settings for CP10 and CP11 must be identical. The encoding for CP10 and CP11 is shown in Table.

CP10 and CP11 Settings

Bits	CP10 and CP11 Setting
00	Access denied. Any attempted access generates a Usage fault (type NOCP – No Co-processor)
01	Privileged Access only. Unprivileged access generates a Usage fault
10	Reserved – result unpredictable
11	Full access

Low Power and System Control Features:

Co-processor access control register

The Co-processor Access Control Register is available in Cortex®-M4 with the floating-point unit for enabling the floating-point unit. This register is located at address 0xE000ED88 (privileged accesses only). In C language programming you can access this registry using the "SCB->CPACR" symbol. By default, the floating-point unit is turned off to reduce the power consumption.

Co-processor Access Control Register (SCB->CPACR, 0xE000ED88)

Bits	Name	Type	Reset Value	Descriptions
31:24	Reserved	–	–	Reserved. Read as Zero. Write ignore
23:22	CP11	R/W	0	Access to the floating-point unit
21:20	CP10	R/W	0	Access to the floating-point unit
19:0	Reserved	–	–	Reserved. Read as Zero. Write ignore

The encoding for CP10 and CP11 is shown in the table below, and the value 01 or 11 must be set to use the floating-point unit.

CP10 and CP11 Settings

Bits	Setting
00	Access denied. Any attempted access generates a Usage Fault (type NOCP – No Co-processor)
01	Privileged Access only. Unprivileged access generates a Usage Fault
10	Reserved – result unpredictable
11	Full access

The settings for CP10 and CP11 must be identical. Usually, when the floating-point unit is needed, you can enable the floating-point unit using the following code:
SCB->CPACR|= 0x00F00000;

COPROCESSOR 15 AND CACHES:

There are several coprocessor 15 registers used to specifically configure and control ARM cached cores. The table below lists the coprocessor 15 registers that control cache configuration. Primary CP15 registers *c7* and *c9* control the setup and operation of the cache. Secondary CP15:c7 registers are write-only and clean and flush cache. The CP15:c9 register defines the victim pointer base address, which determines the number of lines of code or data that are locked in the cache.

Coprocessor 15 registers that configure and control cache operation

Function	**Primary register**	**Secondary registers**	**Opcode 2**
Clean and flush cache	*c7*	*c5, c6, c7, c10, c13, c14*	0, 1, 2
Drain write buffer	*c7*	*c10*	4
Cache lockdown	*c9*	*c0*	0, 1
Round-robin replacement	*c15*	*c0*	0

The ARM Vector Floating Point Coprocessor:

The ARM VFP coprocessor adds a great deal of power to the ARM architecture. The register set is expanded to hold up to four times the amount of data that can be held in the ARM integer registers. The additional instructions allow the programmer to deal directly with the most common IEEE 754 formats for floating-point numbers. The ability to treat groups of registers as vectors adds a significant performance improvement. Access to the vector features is only possible through assembly language. The GCC compiler is not capable of using these advanced features, which gives the assembly programmer a big advantage when high-performance code is needed.

Conclusion:

- To achieve independent access to memory data for high utilization of cores.
- History of Coprocessor is shown.
- The characteristics of Coprocessor are presented.
- The advantages and disadvantages are Coprocessor presented.
- The basic modules of the coprocessor, namely the vector processing unit, and caches units are studied, in detail.
- The internal operation and capabilities of Coprocessor have been given.
- The Block Diagram of the coprocessor and Functional Diagram Coprocessor is shown.

Name:-Om Prakash Mohanta

Regd No:-20010448

Roll No:-CSD20032

Branch:-CSE(Data Science)