TMS320C55x Technical Overview

Literature Number: SPRU393
February 2000

TEXAS INSTRUMENTS

Contents

Figures

Tables

Chapter 1

Introduction

Topic | Page

1.1 Introduction to the TMS320C55x

The TMS320C55x digital signal processor (DSP) represents the latest generation of ’C5000 DSPs from Texas Instruments. The ’C55x is built on the proven legacy of the ’C54x and is source code compatible with the ’C54x, protecting the customer’s software investment. Following the trends set by the ’C54x, the ’C55x is optimized for power efficiency, low system cost, and best-in-class performance for tight power budgets.

With core power dissipation as low as 0.05 mW/MIPS at 0.9V, and performance up to 800 MIPS (400 MHz), the TMS320C55x offers a cost-effective solution to the toughest challenges in personal and portable processing applications as well as digital communications infrastructure with restrictive power budgets. Compared to a 120-MHz ’C54x, a 300-MHz ’C55x will deliver approximately 5X higher performance and dissipate one-sixth the core power dissipation of the ’C54x.

The ’C55x core’s ultra-low power dissipation of 0.05mW/MIPS is achieved through intense attention to low-power design and advanced power management techniques. The ’C55x designers have implemented an unparalleled level of power-down configurability and granularity coupled with unprecedented power management that occurs automatically and is transparent to the user.

The ’C55x core delivers twice the cycle efficiency of the ’C54x through a dual-MAC (multiply-accumulate) architecture with parallel instructions, additional accumulators, ALUs, and data registers. An advanced instruction set, a superset to that of the ’C54x, combined with expanded busing structure complements the new hardware execution units.

The ’C55x continues the standard set by the ’C54x in code density leadership for lower system cost. The ’C55x instructions are variable byte lengths ranging in size from 8 bits to 48 bits. With this scalable instruction word length, the ’C55x can reduce control code size per function by up to 40% more than ’C54x. Reduced control code size means reduced memory requirements and lower system cost.

This overview describes the CPU architecture, low-power enhancements, and embedded emulation features of the TMS320C55x. For detailed information, refer to the documentation listed in Appendix A, *Related Documents from Texas Instruments*.

Unless otherwise specified, all references to the ’C5000 refer to the TMS320C5000 platform of DSPs; ’C55x refers to the TMS320C55x fixed-point DSPs in the ’C5000 platform; and ’C54x refers to the TMS320C54x fixed-point DSPs in the ’C5000 platform.

1.2 Applications and Benchmarks for the ’C55x

The TMS320C55x delivers an optimal combination of ultra-low power, best-in-class performance, and low system cost to fuel the continued digitization and miniaturization of personal and portable applications.

The ’C55x architecture and design was developed with four interrelated objectives:

- ❑ **Industry-leading ultra-low power**

  Allows increased battery life for portable applications or greater channel density for power-efficient infrastructure systems.

- ❑ **Efficient DSP performance**

  Allows more functionality to be added to systems or faster processing time for existing algorithms.

- ❑ **Leadership in code density**

  Requires less memory to perform given functions which translates to lower system cost and/or smaller system size. All this as a result of more tightly packed code.

- ❑ **Complete code compatibility with the ’C54x**

  Maintains the customer’s software investment and large existing code base.

Typical Applications for ’C55x

Industry-leading power efficiency coupled with the tremendous processing performance and low system cost of the TMS320C55x spawn the next generation of the following digital applications:

- ❑ Wireless handsets and personal communication systems
- ❑ Portable audio players
- ❑ Personal medical devices (hearing aids, etc.)
- ❑ Digital cameras
- ❑ Internet/information appliances
- ❑ Power-efficient multichannel telephony systems (e.g. RAS, VOP)

Generally speaking, the ’C55x is broadly targeted at the consumer and communication markets which utilize DSP algorithms such as:

- ❑ Speech coding and decoding
- ❑ Line or Acoustic Echo cancellation; noise cancellation
- ❑ Modulation and demodulation
- ❑ Image and audio compression and decompression
- ❑ Speech encryption, decryption
- ❑ Speech recognition, speech synthesis

Low Power, Low System Cost, High Performance

The ’C55x supports four basic application categories that value low power, low system cost, and high performance with different levels of importance. Those categories include:

1) Applications that require **much longer battery life while maintaining or slightly increasing performance**. Examples include extending the battery life of today’s digital cellular handsets, portable audio players, or digital still cameras from hours to days, or days to weeks, while maintaining the same level of functionality.

2) Applications that require **much higher performance while maintaining or slightly increasing battery life**. Examples include tomorrow’s 3G wireless handsets or internet appliances which may converge audio, video, voice, and data into a single multifunction mobile product. Consumers have come to expect a certain level of battery life in standby and active modes and will not be willing to sacrifice this for more functionality.

3) Applications that require **very small size, ultra-low power consumption, and low-to-medium levels of DSP performance**. Examples include the personal medical market whereby new advances in hearing aids and medical diagnostics require DSP capability but with battery life measured in weeks or months.

4) Power-efficient infrastructure applications (RAS, VOP, multiservice gateways, etc.) that need **increased channel density while meeting stringent board-level power and space budgets**.

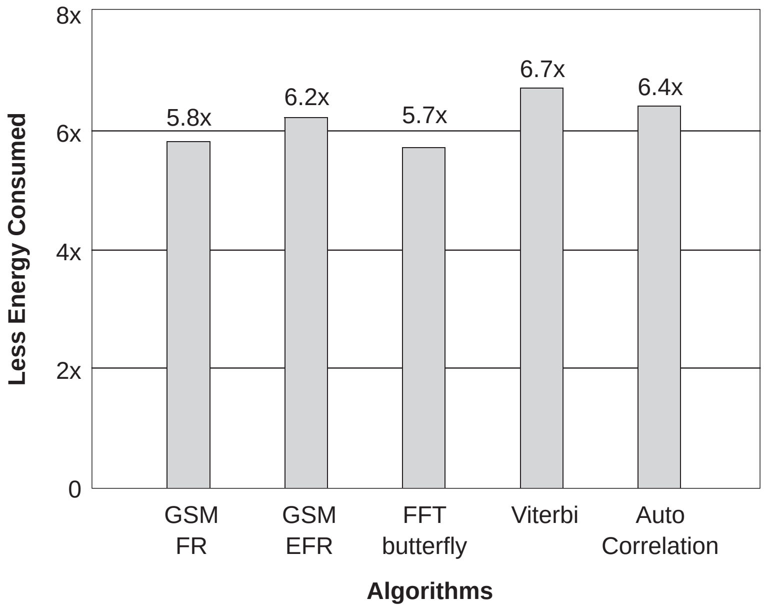
’C55x Benchmarks

Figure 1–1, Figure 1–2, and Figure 1–3 compare the ’C55x to the ’C54x based on the leadership position and prevalence of ’C54x in communication markets today. When compared to a 120-MHz, 1.8V ’C54x; a 300-MHz, 0.9V ’C55x exhibits up to:

- ❑ 6X lower core power
- ❑ 5X higher performance
- ❑ 30% less code size

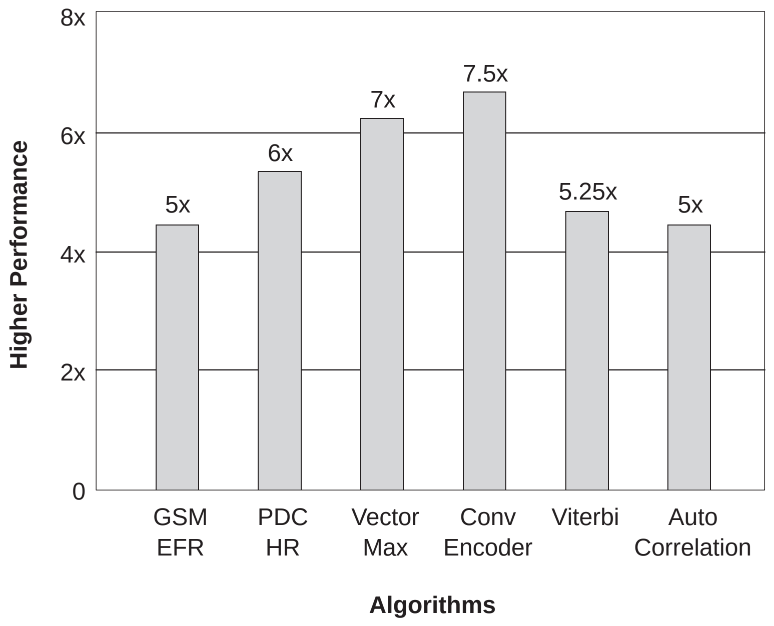
The figures also illustrate improvements in the execution of algorithms for some of the previously discussed target applications. These improvements are a result of architectural enhancements to the ’C55x.

Figure 1–1. Energy Consumption (W-us) 300-MHz, 0.9V ’C55x[†]



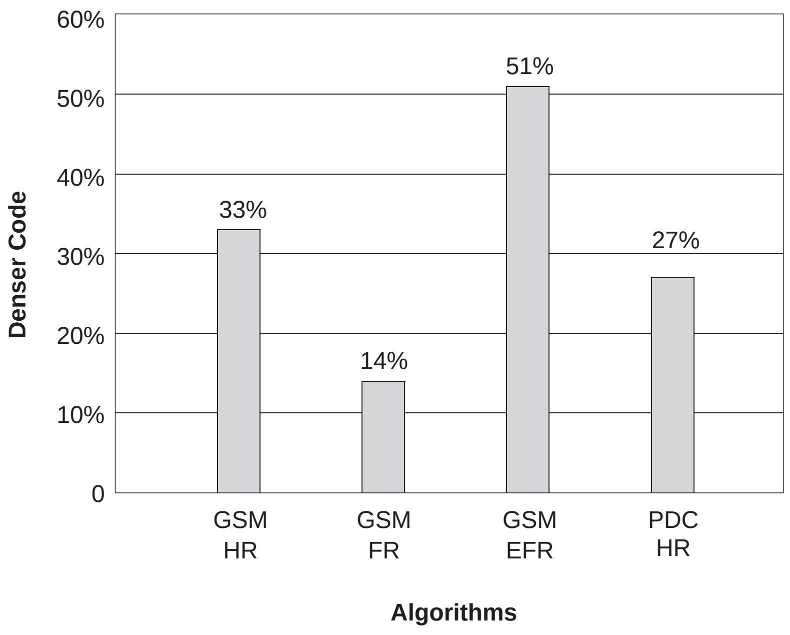
[†] ’C55x results based on comparison of a 300-MHz, 0.9V ’C55x to a 120-MHz, 1.8V ’C54x.

Figure 1–2. Performance (MIPS) 300-MHz ’C55x[†]



[†] ’C55x results based on comparison of a 300-MHz, 0.9V ’C55x to a 120-MHz, 1.8V ’C54x.

Figure 1–3. Code Density (Bytes) ’C55x[†]



[†] ’C55x results based on comparison of a 300-MHz, 0.9V ’C55x to a 120-MHz, 1.8V ’C54x.

1.3 Key Features of the ’C55x

The ’C55x incorporates a rich set of features that provide processing efficiency, low-power dissipation, and ease of use. Some of these features are listed in Table 1–1.

Table 1–1. Features and Benefits of the ’C55x

Feature(s)	Benefit(s)
A 32 x 16-bit Instruction buffer queue	Buffers variable length instructions and implements efficient block repeat operations
Two 17-bit x17-bit MAC units	Execute dual MAC operations in a single cycle
One 40-bit ALU	Performs high precision arithmetic and logical operations
One 40-bit Barrel Shifter	Can shift a 40-bit result up to 31 bits to the left, or 32 bits to the right
One 16-bit ALU	Performs simpler arithmetic in parallel to main ALU
Four 40-bit accumulators	Hold results of computations and reduce the required memory traffic
Twelve independent buses: – Three data read buses – Two data write buses – Five data address buses – One program read bus – One program address bus	Provide the instructions to be processed as well as the operands for the various computational units in parallel —to take advantage of the ’C55x parallelism.
User-configurable *IDLE* Domains	Improve flexibility of low-activity power management

Within the ’C5000 DSP platform, the ’C55x is the latest generation building on the proven legacy of the ’C54x generation. The ’C55x is completely source code compatible with the ’C54x, which means that programs developed on the ’C54x can be re-assembled and executed on the ’C55x with identical, bit-exact results. Table 1–2 outlines a comparison of the hardware features between the ’C54x and the ’C55x.

Table 1–2. ’C54x/’C55x Comparison

	’C54x	**’C55x**
MACs	1	2
Accumulators	2	4
Read buses	2	3
Write buses	1	2
Program fetch	1	1
Address buses	4	6
Program word size	16 bits	8/16/24/32/40/48 bits
Data word size	16 bits	16 bits
Auxiliary Register ALUs	2 (16-bit each)	3 (24-bit each)
ALU	1 (40-bit)	1 (40-bit) 1 (16-bit)
Auxiliary Registers	8	8
Data Registers	0	4
Memory Space	Separate Program/Data	Unified space

Chapter 2

’C55x CPU Architecture

Topic **Page**

2.1 Instruction Set Architecture and Implementation Highlights

The TMS320C55x is a low-power, general-purpose signal processing architecture with an instruction set optimized for efficiency, ease of use, and compactness. Although the ’C55x instruction set is much more powerful and flexible than that of previous generations, the architecture is completely compatible with TMS320C54x instructions. This allows programs developed on the ’C54x to be re-assembled and executed on the ’C55x with bit-exact results. A highly parallel architecture complements the ’C55x instruction set and enables increased code density while reducing the number of cycles required per operation. The union of an efficient, compact instruction set with a highly parallel architecture provides a high-performance signal processing engine while minimizing code size and power consumption.

The ’C55x instruction set includes a flexible set of orthogonal features to enhance ease of use and program efficiency. Powerful addressing modes which include the absolute addressing mode, the register-indirect addressing mode, and the direct addressing mode (also known as displacement) greatly reduce the instruction count required for signal processing algorithms. A three-operand instruction format, with support for both memory and register references, also provides excellent instruction density. All ’C55x instructions that move data support any of the major addressing modes and operand formats. This regularity is conducive to efficient high level language compiler use and simplifies programming in assembly. The instruction set also includes syntax that allows the programmer or compiler to schedule multiple instructions for parallel execution. These instruction set features simplify the task of the programmer and optimize the efficiency of ’C55x code resulting in shorter product development time.

A key to the processing power and superior code density of the ’C55x is its efficient implementation. This implementation uses variable length instruction encoding to achieve optimal code density and efficient bus usage. Multiple computational units are included to carry out computations in parallel, thereby reducing the number of cycles required per operation. The dual multiply-and-accumulate (MAC) units can perform two 17-bit x 17-bit MAC operations in a single cycle while the 40-bit ALU can be used to operate on 32-bit data, or can be split to perform dual 16-bit operations. A second 16-bit ALU for general-purpose arithmetic further increases parallelism and adds flexibility. Based on a modified Harvard architecture, the ’C55x includes one program bus and three independent read data buses that can simultaneously bring data operands to the various computational units. The high degree of parallelism and efficient instruction encoding maximize the overall processor efficiency without sacrificing performance.

This chapter describes the ’C55x CPU implementation in terms of the four units highlighted in Figure 2–1. The four units are:

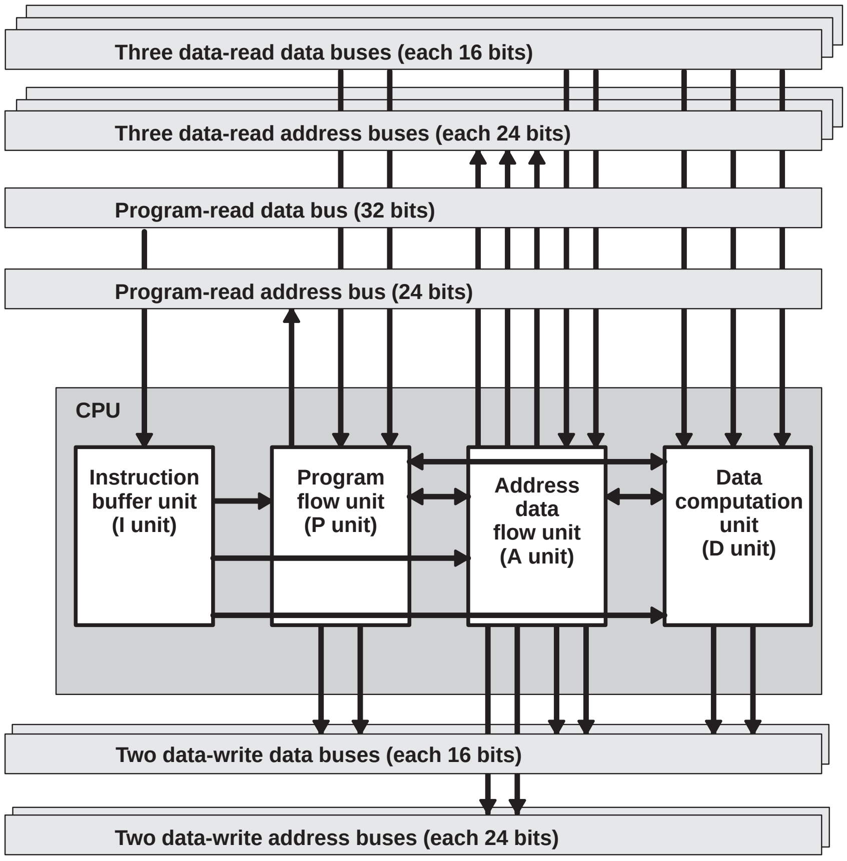
1) **Instruction buffer unit** – This unit buffers and decodes the instructions that make up the application program. In addition, this unit includes the decode logic that interprets the variable length instructions of the ’C55x. The instruction buffer unit increases the efficiency of the DSP by maintaining a constant stream of tasks for the various computational units to perform.

2) **Program flow unit** – The program flow unit keeps track of the execution point within the program being executed. This unit includes the hardware used for efficient looping as well as dedicated hardware for speculative branching, conditional execution, and pipeline protection. This hardware is vital to the processing efficiency of the ’C55x as it helps reduce the number of processor cycles needed for program control changes such as branches and subroutine calls.

3) **Address data flow unit** – This unit provides the address pointers for data accesses during program execution. The efficient addressing modes of the ’C55x are made possible by the components of the address data flow unit. Dedicated hardware for managing the five data buses keeps data flowing to the various computational units. The address data flow unit further increases the instruction level parallelism of the ’C55x architecture by providing an additional general-purpose ALU for simple arithmetic operations.

4) **Data computation unit** – This unit is the heart of the DSP, and performs the arithmetic computations on the data being processed. It includes the MACs, the main ALU, and the accumulator registers. Additional features include a barrel shifter, rounding and saturation control, and dedicated hardware for efficiently performing the Viterbi algorithm, which is commonly used in error control coding schemes. The instruction level parallelism provided by this unit is key to the processing efficiency of the ’C55x.

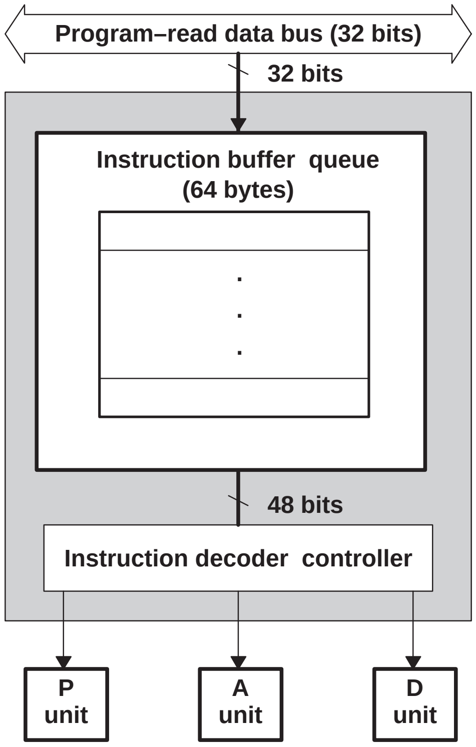
Figure 2–1. CPU Diagram


Three data-read data buses (each 16 bits)
Three data-read address buses (each 24 bits)
Program-read data bus (32 bits)
Program-read address bus (24 bits)
CPU
Instruction buffer unit (I unit)
Program flow unit (P unit)
Address data flow unit (A unit)
Data computation unit (D unit)
Two data-write data buses (each 16 bits)
Two data-write address buses (each 24 bits)

2.2 Instruction Buffer Unit (I Unit)

The instruction buffer unit of the TMS320C55x handles the task of bringing the instruction stream from memory into the CPU. During each CPU cycle, the I unit receives four bytes of program code from the 32-bit program bus, and decodes one to six bytes of code that were previously received in the queue. The I unit then passes the decoded information to the P unit, the A unit, and the D unit for execution of the instructions. Figure 2–2 shows a block diagram of the I unit.

Figure 2–2. Instruction Buffer Unit (I Unit) Diagram


Program–read data bus (32 bits)
32 bits
Instruction buffer queue (64 bytes)
48 bits
Instruction decoder controller
P unit
A unit
D unit

During the prefetch phase of the pipeline, the CPU fetches 32 bits of code from program memory and places it in the instruction buffer queue. When the CPU is ready to decode instructions, up to six bytes are transferred from the queue to the instruction decoder. The instruction buffer queue can hold up to 64 bytes of code at a time, optimizing the performance of the CPU by maintaining a continuous program flow.

The instruction buffer queue is also used in conjunction with the local repeat instruction to *repeat* or *loop* a block of code stored in the queue. This method of looping is extremely efficient in both performance and power dissipation because once the code is loaded into the queue, no additional memory fetches are required to execute the loop.

Another benefit of the instruction buffer queue is that it can perform speculative fetching of instructions while a condition is being tested for conditional program flow control instructions (conditional call, conditional return, or conditional goto). This capability minimizes overhead due to program flow discontinuities by preventing the need to flush the pipeline. Cycles that otherwise would have been lost to a pipeline flush are converted to useful processing cycles.

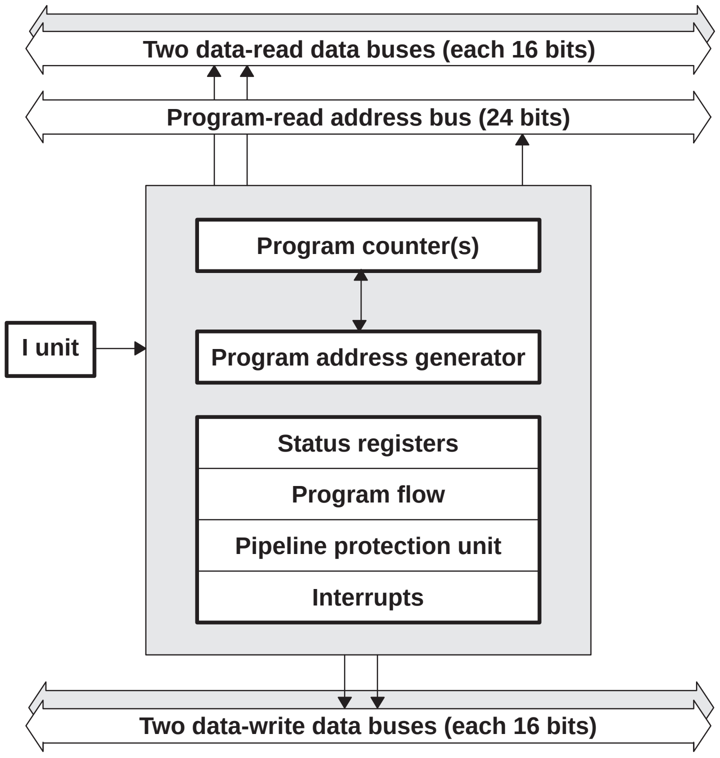
In the decode phase of the pipeline, the instruction decoder accepts up to six bytes of program code from the instruction buffer queue and decodes those bytes. Instructions are decoded in the order that they are received in the instruction buffer queue—the I unit does not perform dynamic scheduling. This results in predictable execution time, which is essential for designing real-time embedded systems.

The ’C55x instruction set has a variable length encoding, with instruction lengths varying from one to six bytes. Instead of encoding all instructions with the same number of bits, simple instructions are encoded with fewer bits than complex instructions. The instruction decoder identifies the boundaries of instructions so that it can decode 8-, 16-, 24-, 32-, 40- and 48-bit instructions. This encoding method results in very high-density program code and optimal use of program memory.

2.3 Program Flow Unit (P Unit)

The ’C55x program flow unit, or P unit, controls the sequence of instructions executed in a program. It generates the addresses for instruction fetches from program memory, and directs operations such as hardware loops, branches, and conditional execution. This unit also includes the logic for managing the instruction pipeline, and the four status registers used to control and monitor various features of the CPU. The components of the P unit enable the superior cycle efficiency of the ’C55x. Figure 2–3 shows a block diagram of the P unit.

Figure 2–3. Program Flow Unit (P Unit) Diagram


Two data-read data buses (each 16 bits)
Program-read address bus (24 bits)
Program counter(s)
I unit
Program address generator
Status registers
Program flow
Pipeline protection unit
Interrupts
Two data-write data buses (each 16 bits)

Within the P unit, the program address generation logic generates 24-bit addresses for instruction fetches from program memory. There are no alignment restrictions on code placement within memory, because the P unit supports byte addressing. The 24-bit address gives the ’C55x a program reach of 16M bytes to accommodate large programs.

The P unit normally generates sequential addresses using the program counter to keep track of the execution point within a program. However this logic also generates nonsequential addresses for program control operations such as:

- branches
- calls
- returns
- hardware looping (repeats)
- conditional execution
- interrupt servicing

The P unit is highly optimized for efficient execution of program flow operations with minimal impact on pipeline performance. The address generation logic of the P unit is completely independent of the other units within the CPU. Because of this, the target address of a branch can be calculated and the condition for a conditional branch can be tested early in the pipeline to minimize branch latency. This parallelism also enables the execution of program-control instructions in the same execute phase of the pipeline as data processing instructions. This greatly enhances ’C55x performance over previous architectures that only allow delay slots as a means for improving branch performance. Other features of the P unit that enhance program control performance include speculative branching logic, and a separate program counter dedicated for fast returns from subroutines or interrupt service routines.

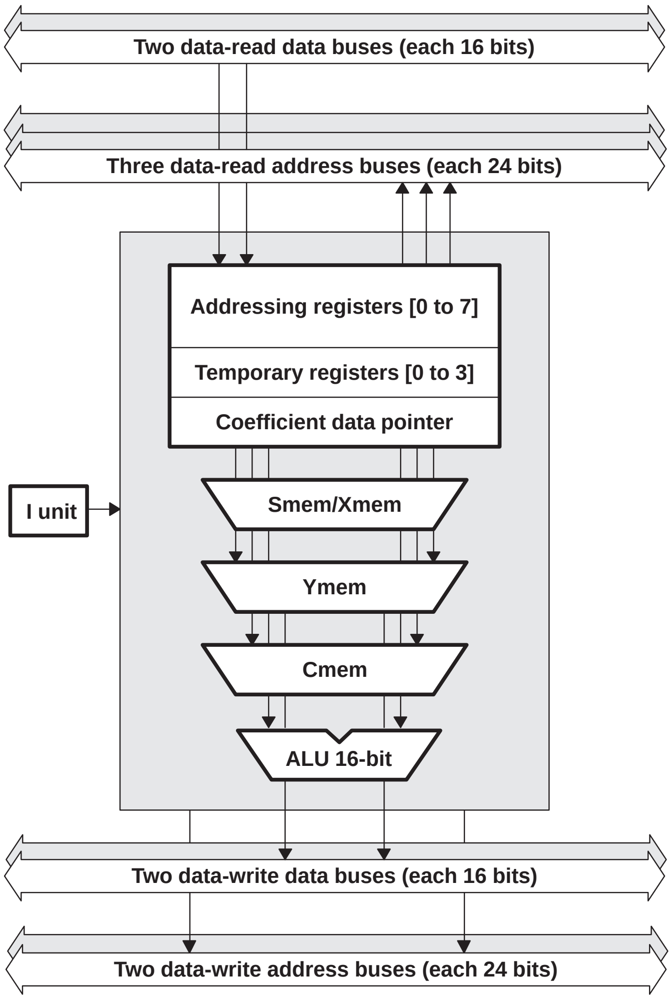
The looping capabilities provided by the P unit include repetition of a single instruction or a block of instructions. Three levels of hardware loops are possible on the ’C55x by nesting a block repeat operation within another block repeat operation, and including a single repeat in either or both of the repeated blocks. The P unit also includes hardware to support conditional repeats.

A major benefit that the P unit provides is dedicated logic for pipeline protection. In addition to handling control hazards, the P unit provides full protection against write-after-read (WAR) and read-after-write (RAW) data hazards. When such data hazards occur in a ’C55x instruction stream, the pipeline protection logic inserts cycles to maintain the intended order of operations and correct execution of the program.

2.4 Address Data Flow Unit (A Unit)

The address data flow unit generates the addresses for read and write accesses to data space. This unit contains all the logic and registers necessary to generate the addresses for the three data-read address buses and the two data-write address buses. It also contains a general-purpose 16-bit arithmetic logic unit (ALU) with shifting capability. Figure 2–4 shows a block diagram of the A unit.

Figure 2–4. Address Data Flow Unit (A Unit) Diagram


Two data-read data buses (each 16 bits)
Three data-read address buses (each 24 bits)
Addressing registers [0 to 7]
Temporary registers [0 to 3]
Coefficient data pointer
I unit
Smem/Xmem
Ymem
Cmem
ALU 16-bit
Two data-write data buses (each 16 bits)
Two data-write address buses (each 24 bits)

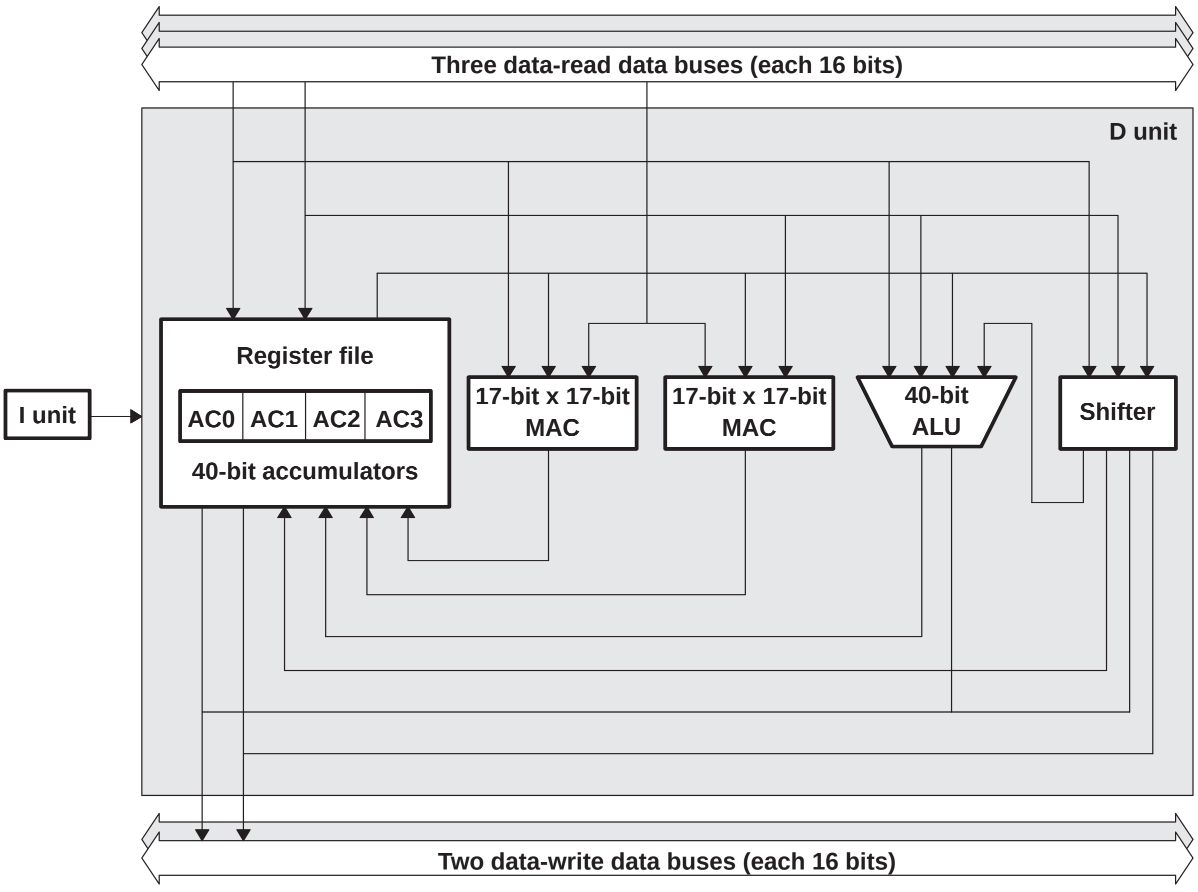
The 16-bit ALU allows simpler arithmetic operations to be performed in parallel with more complex operations performed in the D unit. It accepts immediate values from the I unit and communicates bidirectionally with memory, the A-unit registers, the D- unit registers, and the P-unit registers. Within the A unit, the ALU can manipulate four general-purpose 16-bit registers, or any of the address-generation registers. The four general-purpose registers enable improved compiler efficiency and minimize the need for memory accesses.

Either the general purpose ALU, or one of the three Addressing Register ALUs (ARAUs) can modify the nine addressing registers used for indirect addressing. The three ARAUs provide independent address generators for each of the three data-read buses of the ’C55x. This parallelism allows two 16-bit operands and a 16-bit coefficient to be read into the D unit during each CPU cycle. The A unit also includes dedicated registers to support circular addressing for instructions that use indirect addressing. Up to five independent circular buffer locations can be used simultaneously with up to three independent buffer lengths. There are no address alignment constraints for these circular buffers.

2.5 Data Computation Unit (D Unit)

The data computation unit is the primary part of the CPU where data is processed. Three data-read buses feed the two multiply-and-accumulate (MAC) units and the 40-bit ALU, and intermediate results can be stored in one of four 40-bit accumulator registers. The parallelism of this unit minimizes the cycle count required per task to provide efficient execution of signal processing algorithms. Figure 2–5 shows a block diagram of the D unit.

Figure 2–5. Data Computation Unit (D Unit) Diagram



The key to the computational power of the ’C55x is the dual MAC architecture. A MAC unit consists of a multiplier and a dedicated adder with saturation logic. In a single cycle, each MAC unit can perform a 17-bit by 17-bit multiplication and a 40-bit addition or subtraction with optional 32-/40-bit saturation. The three data-read buses can be used to carry two data streams and a common coefficient stream to the two MAC units. The results from the MAC units can be placed in any of four 40-bit accumulators within the D unit. This dual MAC capability greatly increases the ’C55x performance for executing block filtering algorithms as well as for other signal processing applications.

The D unit also includes a 40-bit arithmetic logic unit (ALU) that is completely separate from the MAC units. The D unit ALU can perform arithmetical or logical operations on 40-bit values from the accumulators, or it can be used to perform dual 16-bit arithmetic operations simultaneously. In addition to accepting inputs from the 40-bit accumulator registers of the D unit, the ALU can accept immediate values from the I unit, and communicates bidirectionally with memory, the A-unit registers, or the P-unit registers.

A powerful barrel shifter complements the MACs and ALU of the D unit. This shifter can shift 40-bit accumulator values up to 31 bits to the left, or up to 32 bits to the right. It accepts immediate values from the I unit and communicates bidirectionally with memory, the A-unit registers, and the P-unit registers. In addition, it can supply a shifted value to the D-unit ALU as an input for further calculation.

The results of computational operations in the D unit are written to memory by the two 16-bit data-write buses. These buses, together with the A-unit address generation logic can perform two 16-bit writes, or a single 32-bit write to memory in one CPU cycle. This data throughput is essential for supporting the real-time processing speed provided by the D unit.

Chapter 3

Low-Power Enhancements

Topic **Page**

3.1 Enhancements for Low-Power Dissipation

With the addition of the ’C55x generation of processors, Texas Instruments furthers its position as the vanguard in low-power dissipation DSPs. The ’C55x generation builds on the architecture foundation of the ’C54x generation of processors—already the lowest power DSPs in the industry. A series of process, design, and architectural enhancements collectively enable new levels of power reduction. These design enhancements to the ’C55x not only achieve ultra-low power but greatly increase performance.

3.1.1 Architecture

Increased Parallelism Minimizes Cycles Per Task

The ’C55x architecture expands on the ’C54x architecture to provide higher performance and lower power dissipation through increased parallelism. Increased data processing throughput per cycle is provided by:

- ❑ Two multiply-accumulate (MAC) units
- ❑ Two arithmetic logic units (ALUs)
- ❑ Three read buses
- ❑ Two write buses

These enhancements allow processing of two data streams, or one stream at twice the speed, without the need to read coefficient values twice. Minimizing memory access for a given task improves power and performance.

The ’C55x instruction architecture also provides the capability for two instructions to be executed in a single cycle. The presence of two internal write buses provides the capability to perform two writes, or a single double-word write, or a double stack push in one cycle, reducing cycle time per task. Less time per task means more time spent in a power-down (IDLE) mode and more power saved.

Alternate Computational Hardware Use Provides a Low-Power Option for Many Tasks

The ’C55x architecture provides flexibility in performing computational tasks. Two arithmetic/logic units (ALUs) can be used:

- ❑ One 40-bit ALU (Standard on the ’C54x)
- ❑ One 16-bit ALU (Added on the ’C55x)

The 40-bit ALU is consistent with the architecture of the ’C54x generation, and is used for primary computational tasks. The ’C55x architecture implements an additional 16-bit ALU which can be used for smaller arithmetic and logic tasks. The flexible instruction set provides the capability to direct simpler computational or logical/bit-manipulation tasks to the 16-bit ALU which consumes less power. This redirection of resources also saves power by reducing cycles per task since both ALUs can operate in parallel.

Memory Accesses Minimized

Memory accesses, both internal and external, can be a major contributor to power dissipation. Minimizing the number of memory accesses necessary to complete a given task furthers the goal of minimizing power dissipation per task. The ’C55x generation reduces the number of fetches necessary to provide instructions to the CPU. On the ’C55x, program fetches are performed as 32-bit accesses (extended from 16-bit on the ’C54x). In addition, the variable-byte-length instruction set means that each 32-bit instruction fetch can retrieve more than one instruction. Variable length instructions **improve code density** and conserve power by scaling the instruction size to the amount of information needed. This alliance of instruction set design and architecture minimizes the power necessary to keep the application running at top performance.

The flexible ’C55x instruction cache also provides a configurable cache capability that can be used to optimize the cache operation for different types of code. Improving the cache hit ratio means fewer external accesses and less system power consumed. The burst-fill capability of the instruction cache can minimize external memory accesses and their associated loss of performance and power efficiency.

Automatic Low-Power Mechanisms for Peripherals and On-Chip Memory Arrays

The ’C55x core processor actively manages power consumption of on-chip peripherals and memory arrays. This resource power management is fully automatic and transparent to the user. It is performed without any impact on the software or the computational performance of the application. It is another contributor to power reduction without impact on performance.

When individual on-chip memory arrays are not being accessed, they are automatically switched into a low-power mode. When an access request arrives, the array returns to normal operation, without latency in the application, and completes the memory access. If no further accesses to that array are requested, the array returns to a low-power state until it is needed again.

The processor provides a similar control to on-chip peripherals. Peripherals can enter low-power states when they are not active and the CPU does not require their attention. The peripherals also respond to processor requests and exit their low-power states without latency. This power management occurs in addition to the software controllable low-power states provided by the IDLE domain control of the peripherals.

Configurable Functional (IDLE) Domains Provide Greater Power-Down Flexibility

A critical component of power conservation is minimizing the power used when an application is in an idle or low-activity state. The ’C55x generation improves the flexibility of low-activity power management through the implementation of user-controllable *IDLE domains*. These domains are sections of the device which can be selectively enabled or disabled under software control. When disabled, a domain enters a very low-power IDLE state in which register or memory contents are still maintained. When the domain is enabled, it returns to normal operation. Each of the domains can be separately enabled or disabled providing the application the capability to manage low-activity power situations as efficiently as possible. On initial ’C55x devices, the sections of the device configured as separate IDLE domains are: the CPU, the DMA, the peripherals, the external memory interface (EMIF), the instruction cache, and the clock generation circuitry.

3.1.2 Process

Advanced Lower-Voltage Process Technology

In addition to the power dissipation reductions achieved by the architectural and instruction set enhancements, the ’C55x generation of processors will further challenge the barriers to power reduction through advanced low-voltage CMOS technologies. Initial ’C55x devices will be based on a power-efficient CMOS technology that supports devices running at 1.5V and 0.9V. These low-voltage processors still maintain the capability to interface directly to other standard 3.3V CMOS components.

Chapter 4

Embedded Emulation Features

Topic **Page**

4.1 Basic Emulation Features and Enhancements

The ’C55x generation of processors support enhanced emulation and debug capabilities providing an emulation environment that more closely models the actual application environment than ever before. Features allowing real-time operation of the application during emulation and a faster, more efficient debug environment combine to minimize product development effort and time-to-market.

Enhanced emulation features on the ’C55x development tools include:

- ❑ Non-intrusive real-time debug with watchpoint/breakpoint capability
- ❑ Faster screen updates
- ❑ Better control of functional code execution during emulation *Halt* events
- ❑ Trace capability
- ❑ Real-Time Data Exchange (RTDX)

The integration of these enhanced capabilities provides the software/system developer with tools that allow greater visibility of hardware operation without stopping the CPU or consuming MIPS for emulation purposes. The result is an emulation environment capable of utilizing the full performance of the DSP.

Non-Intrusive Real-Time Debug

Emulation capability on TI DSPs is provided through a scan-based system that exchanges data between the emulator/debugger and the DSP through a serial test access port. On earlier DSP generations, exchange of data requires that the CPU be stopped while data was scanned in or out. On the ’C55x generation, this limitation is eliminated through the addition of dedicated on-chip emulation hardware that orchestrates data exchange between the CPU and the emulator without the need to halt the processor. The result is an environment where debug information is available to the developer during emulation while the application continues to run at full performance.

During application operation, the normal CPU timing and interaction with peripherals is maintained by causing the emulator to delay access to on-chip resources until they are not in use by the CPU for the application. For example, the emulation hardware may be allowed to use the internal buses to access on-chip memory only when the CPU is not using the same buses to support the application. Restriction of emulation access to periods when on-chip resources are available also limits impact on interrupt latency creating a more realistic emulation environment.

In the event, that the emulation data exchange is more critical that preserving the CPU environment, the developer has the capability to configure the DSP to allow the emulator to share on-chip resources to gain quicker access. As an example, the emulator may hold off CPU access to memory for a cycle to gain access to data that needs to be exported. This flexibility allows the developer to choose the emulation environment that best supports his/her debug needs.

Faster Debug Screen Updates

Data from the DSP displayed and used by the emulator is exchanged through a serial scan chain inside the DSP. On the ’C55x generation, the scan chain length is limited to the on-chip emulation hardware block instead of the entire CPU and peripheral system. This structure greatly reduces the length of the scan chain and consequently improves the speed of data exchange between the device and the debugger. The emulation hardware block is responsible for managing the on-chip resources for data movement and relieves the emulator of this additional overhead. The result is faster, more efficient visibility into the operation of the application.

Better Control of Functional Code Execution During Emulation Halt Events

In addition to the ability to exchange data between the CPU and the emulator without halting the processor, the ’C55x emulation environment is also capable of allowing the DSP to service interrupts when the CPU is halted. Often debugging interrupt latency and performance can be difficult because other devices in the system (such as data converters or codecs) interfacing to the DSP via on-chip peripherals cannot be controlled by the emulator. Since these other devices may be free running, debugging such a system is difficult due to the loss of synchronization with these events when the DSP is halted (for example, to update the screen on the debugger). The ’C55x devices support two separate interrupt environments: one for interrupts requested when the DSP is not interacting with the emulator (the normal application state) and a separate environment for when the DSP is involved with emulation. These two states are under program control of the DSP.

The CPU can execute interrupt service routines (ISR) while the main program is halted during debug. Interrupt sequencing automatically preserves the debug capabilities so that they can be re-enabled at the end of the ISR. Also, certain debug registers can be accessed by the DSP to control debug functionality from within the application. Certain critical portions of code can disable emulation capabilities, then re-enable them at completion of the ISR.

Instructions in the CPU pipeline also can complete execution before the emulation halt stalls the pipeline. This feature improves the readability of expected results from registers in the debug window during halt.

4.2 Trace Capability

Another enhancement of the ’C55x on-chip emulation hardware is Program Counter (PC) Trace capability, which provides greater visibility into application program flow. The PC Trace capability addresses the need to reconstruct program flow by exporting enough information to completely reconstruct program sequencing with an off-line program. Multiple capabilities will be selected for export as a runtime user option to control what information is exported when, and in which format. The PC Trace hardware in concert with the emulator is capable of exporting:

- ❑ Trace of the last 32 PC values, or
- ❑ Trace of the the last 16 PC discontinuities

Trace of the last 32 PC values provides the ability to observe recent program flow history. For example, a given subroutine may be called from many different locations in the main program. By placing a breakpoint in the subroutine, the PC trace capability can be used to determine the location in the main program from which the subroutine was called.

Trace of the last 16 PC discontinuities provides the ability to observe long-term history of the program flow. This function becomes more valuable in code that is highly dependent on conditional branches and calls.

4.3 Real-Time Data Exchange (RTDX)

Real-Time Data Exchange (RTDX) functions of future ’C55x derivatives will offer the capability to exchange data between the target and the emulation host running the debugger. This feature will be enabled by on-chip real-time debug hardware providing a shared path with the debug control. Allowing target data to be sent from and received by the host at a rate of up to 2M bytes per second will open new emulation possibilities including:

- ❑ Simulation of real-time inputs to the target
- ❑ Update of target system performance graphs on the host in real time

Appendix A

Related Documents from Texas Instruments

The following books describe the TMS320C55x devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU. This book also describes how to make individual portions of the DSP inactive to save power.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the mnemonic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the algebraic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x Optimizing C Compiler User's Guide (literature number SPRU281) describes the 'C55x C compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Reference Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and includes application program examples.

Code Composer Studio™ contains the following online guides:

TMS320C55x DSP Instruction Sets Online Reference Guide describes the algebraic and mnemonic instructions individually. It also includes the parallelism features and rules of the instruction sets , a summary of the instruction sets, a list of the instruction opcodes, and a cross-reference of the mnemonic instruction sets.

TMS320C55x DSP Registers Online Guide describes the registers inside the TMS320C55x DSPs and shows the addresses for DSP registers that are mapped to memory.

TMS320C55x DSP CPU Online Guide describes the architecture and operation of the CPU inside the TMS320C55x DSPs. This guide also describes how to make individual portions of the DSP inactive to save power.

Trademarks

Code Composer Studio is a trademark of Texas Instruments Incorporated.

Appendix B

Glossary

A

address: The location of program code or data stored; an individually accessible memory location.

ALU: See *arithmetic logic unit*.

arithmetic logic unit (ALU): The hardware of the CPU that performs arithmetic and logic functions.

C

cache: A fast storage buffer in the central processing unit of a computer.

central processing unit (CPU): The unit that coordinates the functions of a processor.

circular addressing: An address mode in which a finite set of addresses is reused by linking the largest address back to the smallest address.

clock cycles: A periodic or sequence of events based on the input from the external clock.

code: A set of instructions written to perform a task; a computer program or part of a program.

compiler: A computer program that translates programs in a high-level language into their assembly-language equivalents.

CPU: See *central processing unit.*

crosspath: A link between register files to provide communication between the CPU units.

D

data memory: A region of memory used for storing or manipulating data, separate from the region used for storing program code.

direct memory access (DMA): Memory access that does not use the CPU; used for data transfer directly between memory and a peripheral.

direct memory access (DMA) controller: Specialized circuitry that transfers data from memory to memory without using the CPU.

DMA: See *direct-memory access.*

E

external interrupt: A hardware interrupt triggered by a pin.

external memory interface (EMIF): Microprocessor hardware which is used to read from and write to off-chip memory.

F

fixed-point processor: A processor which does arithmetic operations using integer arithmetic with no exponents.

floating-point processor: A processor capable of handling floating-point arithmetic where real operands are represented using exponents.

H

hardware interrupt: An interrupt triggered through physical connections with on-chip peripherals or external devices.

I

IDLE: A power-down mode.

IDLE domain: Sections of a device which can be selectively enabled or disabled under software control. When disabled, a domain enters a very low-power state in which register or memory contents are still maintained.

indirect addressing: An addressing mode in which an address points to another pointer rather than to the actual data.

interrupt: A signal sent by hardware or software to request a processor's attention. An interrupt tells the processor to suspend its current operation, save the current task status, and perform a particular set of instructions. Interrupts communicate with the operating system and prioritize tasks to be performed.

L

latency: The delay between the occurrence of a condition and the reaction of the device. Also, in a pipeline, the necessary delay between the execution of two potentially conflicting instructions to ensure that the values used by the second instruction are correct.

M

million instructions per second (MIPS): A measure of the execution speed of a computer.

multiplier: A CPU component that multiplies the contents of two registers.

P

parallelism: Sequencing events to occur simultaneously. Parallelism is achieved in a CPU by using instruction pipelining.

peripheral: A device connected to and usually controlled by a host device.

pipeline: A method of executing instructions in which the output of one process serves as the input to another, much like an assembly line. These processes become the stages or phases of the pipeline.

pipeline processing: A technique that provides simultaneous, or parallel, processing within the computer. It refers to overlapping operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously.

program cache: A fast memory cache for storing program instructions allowing for quick execution.

program fetch unit: The CPU hardware that retrieves program instructions.

program memory: A memory region used for storing and executing programs, separate from the region used for storing data.

R

register: A small area of high speed memory, located within a processor or electronic device, that is used for temporarily storing data or instructions. Each register is given a name, contains a few bytes of information, and is referenced by programs.

reset: A means of bringing the CPU to a known state by setting the registers and control bits to predetermined values and signaling execution to start at a specified address.

S

saturation: A state where any further input no longer results in the expected output.

shifter: A hardware unit that shifts bits in a word to the left or to the right.

W

word: A set of bits that is stored, addressed, transmitted, or operated on as a unit.

Index