CoreMP7

DirectCore

Product Summary

Key Features

- FPGA Optimized ARM7™ Family Processor
- Compatible with ARM7TDMI-S™
- 32/16-Bit RISC Architecture (ARMv4T)
- 32-Bit ARM® Instruction Set
- 16-Bit Thumb® Instruction Set
- 32-Bit Unified Bus Interface
- 3-Stage Pipeline
- 32-Bit ALU
- 32-Bit Memory Addressing Range
- Static Operation
- Embedded ICE-RT Real-Time Debug Unit
- JTAG Interface Unit

Intended Use

- Personal Audio (MP3, WMA, and AAC players)
- Personal Digital Assistants
- Wireless Handset
- Pagers
- Digital Still Camera
- Inkjet/Bubble Jet Printer
- Monitors

Benefits

- Fully Implemented in FPGA Fabric
- All Microprocessor I/Os Available to User
- Unified Bus Interface Simplifies System-on-a-Chip (SoC) Design
- ARM and Thumb Instruction Sets Can Be Mixed

ARM Supported Families

- ProASIC®3 (M7A3P)
- Fusion (M7AFS)

Synthesis and Simulation Support

- Directly Supported within the Actel Libero® Integrated Design Environment (IDE)
- Synthesis: Synplify® and Design Compiler®
- Simulation: Vital-Compliant VHDL Simulators and OVI-Compliant Verilog Simulators

Verification and Compliance

- Compliant with ARM7 Instruction Set Architecture (ISA)

Core Version

- This Datasheet Defines the Functionality for CoreMP7 Version 1.0

Introduction

The CoreMP7 soft IP core is an ARM7 family processor optimized for use in Actel ARM-ready FPGAs and is compatible with the ARM7TDMI-S. Users should refer to the *ARM7 Technical Reference Manual* (DDI0234A-7TMIS-R4.pdf), published by ARM Corporation, for detailed information on the ARM7. The *ARM7 Technical Reference Manual* (TRM) is available for download from the ARM website at www.arm.com.

CoreMP7 is supplied with an Advanced Microcontroller Bus Architecture (AMBA) Advanced High-Performance Bus (AHB) compliant wrapper for inclusion in an AMBA-based processor system such as the one generated by the Actel CoreConsole IP deployment platform.

ARM7 Family Processor

CoreMP7 is a general purpose 32-bit ARM7 family microprocessor that offers high performance and low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core. Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. The CoreMP7 processor also implements the Thumb instruction set, which is suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The Thumb instruction set's 16-bit instruction length allows it to approach twice the density in memory of standard 32-bit ARM code while retaining most of the ARM performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because Thumb code operates on the same 32-bit register set as ARM code. Thumb code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.

CoreMP7 Signals

Table 1 gives the signals for the core.

Table 1 • **CoreMP7 Signal Descriptions**

Name	Type	Description
ABORT	Input	Memory abort or bus error
CFGBIGEND	Input	Big/little endian configuration
CLK	Input	Clock
CLKEN	Input	Clock enable
CPA	Input	Coprocessor absent
CPB	Input	Coprocessor busy
DBGBREAK	Input	EICE breakpoint / watchpoint indicator
DBGEN	Input	Debug enable
DBGEXT[1:0]	Input	EICE external input 0
DBGnTRST	Input	Test reset
DBGRQ	Input	Debug request
DBGTCKEN	Input	Test clock enable
DBGTDI	Input	EICE data in
DBGTMS	Input	EICE mode select
nFIQ	Input	Interrupt request
nIRQ	Input	Fast interrupt request
nRESET	Input	Reset
RDATA[31:0]	Input	Read data bus
ADDR[31:0]	Output	Address bus
CPnI	Output	Coprocessor instruction (asserted low)
CPnMREQ	Output	Memory request (asserted low)
CPnOPC	Output	Op-code fetch (asserted low)
CPnTRANS	Output	Memory translate (asserted low)
CPSEQ	Output	Sequential address
CPTBIT	Output	Processor in Thumb mode
DBGACK	Output	Debug acknowledge
DBGCOMMRX	Output	EICE communication channel receive
DBGCOMMTX	Output	EICE communication channel transmit
DBGnEXEC	Output	Executed (asserted low)
DBGnTDOEN	Output	TDO enable (asserted low)
DBGRNG[1:0]	Output	EICE rangeout
DBGTDO	Output	EICE data out
DBGINSTRVALID	Output	ETM Instruction valid indicator
DMORE	Output	Set when next data memory access is followed by a sequential data memory access
LOCK	Output	Locked transaction operation
PROT[1:0]	Output	Indicates code, data, or privilege level
SIZE[1:0]	Output	Memory access width
TRANS	Output	Next transaction type (i, n, s)
WDATA[31:0]	Output	Write data bus
WRITE	Output	Indicates write access

CoreMP7 is available with the native ARM7 bus interface or with an AHB wrapper which changes or transforms some of these signals. This is detailed in Table 2 on page 4.

Programmer's Model

The CoreMP7 processor implements ARM architecture v4T, which includes the 32-bit ARM instruction set and the 16-bit Thumb instruction set.

Processor Operating States

The CoreMP7 processor has two operating states: ARM state and Thumb state.

ARM State

32-bit, word-aligned ARM instructions are executed in this state.

Thumb State

16-bit, halfword-aligned Thumb instructions are executed in this state.

In Thumb state, the Program Counter (PC) uses bit 1 to select between alternate half words.

Note: Transition between ARM and Thumb states does not affect the processor mode or the register contents.

Users can switch the operating state of CoreMP7 between ARM state and Thumb state using the BX instruction. This is described fully in the ARM Architecture Reference Manual (available online from the book-selling websites).

All exception handling is performed in ARM state. If an exception occurs in Thumb state, the processor reverts to ARM state. The transition back to Thumb state occurs automatically on return.

Memory Formats

The CoreMP7 processor supports either little endian or big endian memory formats. The core supports word (32-bit), halfword (16-bit), and byte (8-bit) data formats.

The designer must align these as follows:

- Word quantities must be aligned to four-byte boundaries.
- Halfword quantities must be aligned to two-byte boundaries.
- Byte quantities can be placed on any byte boundary.

Operating Modes

The CoreMP7 processor has seven operating modes:

- User mode is the usual ARM program execution state. It is used for executing most application programs.
- Fast interrupt (FIQ) mode supports a data transfer or channel process.
- Interrupt (IRQ) mode is used for general-purpose interrupt handling.
- Supervisor mode is a protected mode for the operating system.
- Abort mode is entered after a data or instruction prefetch abort.
- System mode is a privileged user mode for the operating system.
- Undefined mode is entered when an undefined instruction is executed.

Modes other than user mode are collectively known as privileged modes. Privileged modes are used to service interrupts, exceptions, or access protected resources.

Registers

The CoreMP7 processor has a total of 37 registers:

- 31 general-purpose 32-bit registers
- 6 status registers

These registers are not all accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

ARM State Register Set

In ARM state, 16 general registers and one or two status registers are accessible at any one time. In privileged modes, mode-specific banked registers become available.

The ARM state register set contains 16 directly accessible registers, r0 to r15. An additional register, the Current Program Status Register (CPSR), contains condition code flags, and the current mode bits. Registers r0 to r13 are general purpose registers used to hold either data or address values. Registers r14 and r15 have special functions as the subroutine link register and the program counter.

Link Register

Register 14 is used as the subroutine Link Register (LR).

R14 receives a copy of r15 when a Branch with Link (BL) instruction is executed. At all other times you can treat r14 as a general-purpose register. The corresponding banked registers (r14_svc, r14_irq, r14_fiq, r14_abt, and r14_und) are similarly used to hold the return values of r15 when interrupts and exceptions arise, or when BL instructions are executed within interrupt or exception routines.

Program Counter

Register 15 holds the Program Counter (PC).

In ARM state, bits [1:0] of r15 are zero. Bits [31:2] contain the PC.

In Thumb state, bit [0] is zero. Bits [31:1] contain the PC.

In privileged modes, another register, the Saved Program Status Register (SPSR), is accessible. This contains the condition code flags, and the mode bits are saved as a result of the exception that caused entry to the current mode.

Thumb State Register Set

The Thumb state register set is a subset of the ARM state set. The programmer has direct access to the following:

- Eight general registers, r0–r7
- The PC
- A Stack Pointer (SP)
- A Link Register (LR)
- The CPSR

There are banked SPs, LRs, and SPSRs for each privileged mode.

Relationship between ARM State and Thumb State Registers

The Thumb state registers relate to the ARM state registers in the following way:

- Thumb state r0–r7 and ARM state r0–r7 are identical.
- Thumb state CPSR and SPSRs and ARM state CPSR and SPSRs are identical.
- Thumb state SP maps onto ARM state r13.
- Thumb state LR maps onto ARM state r14.
- The Thumb state PC maps onto the ARM state PC (r15).

Note: Registers r0–r7 are known as the low registers. Registers r8–r15 are known as the high registers.

Program Status Registers

The CoreMP7 core contains a CPSR and five SPSRs for exception handlers to use. The program status registers handle the following functions:

- Hold the condition code flags
- Control the enabling and disabling of interrupts
- Set the processor operating mode

Condition Code Flags

The N, Z, C, and V bits are the condition code flags. You can set these bits by arithmetic and logical operations. The flags can also be set by MSR and LDM instructions. The CoreMP7 processor tests these flags to determine whether to execute an instruction.

All instructions can execute conditionally in ARM state. In Thumb state, only the Branch instruction can be executed conditionally. For more information about conditional execution, see the *ARM Architecture Reference Manual*.

AHB Wrapper

The AHB wrapper interfaces between the CoreMP7 and the AHB bus. The module translates access from the core to AHB accesses when the core is the current master. The external interface signals from the wrapper are described in Table 2.

Table 2 • **AHB Wrapper External Interface**

Signal	Direction	Description
Ext. Master I/F		
HCLK	Input	Bus clock. This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn	Input	Reset. The bus reset signal is active LOW and is used to reset the system and the bus. This is the only active LOW AHB signal.
HREADY	Input	Transfer done. When HIGH, the HREADY signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
HRESP[1:0]	Input	Transfer response. Indicates an OKAY, ERROR, RETRY, or SPLIT response.
HGRANT	Input	Bus grant. Indicates that the CoreMP7 is currently the highest priority master. Ownership of the address/control signals changes at the end of a transfer when HREADY is HIGH, so a master gains access to the bus when both HREADY and HGRANT are HIGH.
HADDR[31:0]	Output	This is the 32-bit system address bus.
HTRANS[1:0]	Output	Transfer type. Indicates the type of the current transfer.

Table 2 • **AHB Wrapper External Interface (Continued)**

Signal	Direction	Description
HWRITE	Output	Transfer direction. When HIGH, this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0]	Output	Transfer size. Indicates the size of the transfer, which can be byte (8-bit), halfword (16-bit), or word (32-bit).
HBURST[2:0]	Output	Burst type. Indicates if the transfer forms part of a burst. The CoreMP7 performs incrementing bursts of type INCR.
HPROT[3:0]	Output	Protection control. These signals indicate if the transfer is an opcode fetch or data access, and if the transfer is a supervisor mode access or user mode access.
HWDATA[31:0]	Output	32-bit data from the MASTER.
HRDATA[31:0]	Input	32-bit date written back to the MASTER.

CoreMP7 Variants

CoreMP7 is available with and without debug for use in each ARM-enabled device. These variants (core only or core plus debug) are available in CoreConsole and are easily selected from the core configuration menus. The utilization and performance of the variants for each device are shown in Table 3.

Table 3 • **CoreMP7 Utilization and Performance**

Device Variant	Performance (MHz)	Tiles	RAM Block	Utilization (%)
M7A3P1000				
Core Only instead of CoreMP7S to Core Only	28.548	6,397	4	26.0
Core Plus Debug instead of CoreMP7Sd to Core Plus Debug	22.714	8,522	4	34.7
M7AFS600				
Core Only instead of CoreMP7S to Core Only	26.499	6,350	4	45.9
Core Plus Debug instead of CoreMP7Sd to Core Plus Debug	23.165	8,243	4	59.6

Core Plus Debug

This variant of CoreMP7 is configured with all features of the ARM7TDMI-S. The variant incorporates the full debug functionality of the ARM7TDMI-S and is fully compliant with RealView RVDS, RVDK, and other ARM software debug tools.

Core Only

This variant of CoreMP7 has the same features as the Core plus debug variant, except that it does not include the ICE-RT debug block or the TAP controller, which reduces the size of the core. This means that the standard debug tools cannot be used with this variant of CoreMP7.

No Debug

This means that standard software debug tools cannot be used when this variant of the CoreMP7 is instantiated. Users of this variant can employ the Core plus debug variant for development, and when the application has been fully tested and debugged, the Core only variant can be instantiated to reduce area in the final shipping product.

No Co-Processor Interface

The co-processor interface is a rarely used feature in ARM7 family microprocessors and has been removed from the CoreMP7 variant to minimize area.

Little Endian Only

Most microprocessor-based systems use little endian byte ordering. The option of selecting big endian has been removed from the CoreMP7 variant to minimize area.

On-Chip RAM Consumed by Register Block

To minimize the area, the CoreMP7 variants map the processor register block into on-chip RAM. RAM blocks used to implement CoreMP7 registers are no longer available for use in designs.

Delivery and Deployment

The CoreMP7 is delivered in CoreConsole, and can be instantiated in design projects created in Libero IDE. The files included with CoreMP7 consist of the Bus Functional Model (BFM) files and test wrapper, AHB wrapper, and the A7S secured CDB file, which is the placed and routed CoreMP7 core (this is actually instantiated on the user device). This deployment flow is implemented to ensure that the design is kept completely secure at all times, and allows CoreMP7 to be easily used with the standard design flow through the Libero IDE tool suite.

Bus Functional Model (BFM)

Introduction

During the development of an FPGA-based SoC, there are a number of stages of testing which may be undertaken. This may involve some, or all, of the following approaches:

- Hardware simulation using Verilog or VHDL
- Software simulation using a host-based Instruction Set Simulator (ISS) of the SoC's processor
- Hardware and software co-verification using a full-functional model of the processor in Verilog, VHDL, or SWIFT form (or using a tool such as Seamless® from Mentor Graphics).

BFM Usage Flow

The BFM acts as a pin-for-pin replacement of the CoreMP7 in the simulation of the SoC subsystem. It initiates bus transactions on the native CoreMP7 bus, which are cycle-accurate with real bus cycles that the CoreMP7 would produce. It does not have the ability to implement real CoreMP7 instructions. The BFM may be used to run a basic test suite of the SoC subsystem using the skeleton system testbench.

The developer may edit the SoC Verilog/VHDL to add new design blocks. The system-level testbench may also be filled out by the developer to include tasks which test any newly added functionality or add stubs to allow more complex system testing involving the IP cores. The BFM input scripts may also be manually enhanced so the user can test out access to register locations in newly added logic. In this way, the user can provide stimuli to the system from the inside (via the CoreMP7 BFM), as well as from the outside (via testbench tasks).

Timing Shell

There is a timing shell provided for each CoreMP7 variant wrapped around the BFM. Therefore, the BFM is bus cycle accurate, and it performs setup/hold checks to model output propagation delays.

Debug

The ARM Debug Architecture uses a protocol converter box to allow the debugger to talk via a JTAG port directly to the core. In effect, the scan chains in the core that are required for test are reused for debugging. The core uses the scan chains to insert instructions directly into CoreMP7. The instructions are executed on the core and depending on the type of instruction that has been inserted, the core or the system state can be examined, saved, or changed. The architecture has the ability to execute instructions at a slow debug speed or to execute instructions at system speed.

www.actel.com

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court, Meadows Business Park
Station Approach, Blackwater
Camberley Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
www.jp.actel.com

Actel Hong Kong
Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn