



aJile Java™ Processor Core JEMCore™

Introduction

The JEMCore is aJile's Java processor core, which is based on the proven JEM2 from Rockwell Collins. JEMCore directly executes Java Virtual Machine™ (JVM) bytecodes, real-time Java threading primitives and a number of extended bytecodes for embedded operations. The JEMCore improves Java execution efficiency by eliminating the Java interpreter (software translation layer) and the RTOS kernel layer. Since JVM bytecodes are executed as native instructions, the JEMCore's Java performance is similar to RISC processors executing compiled C. In addition, Java threading primitives (wait, yield, notify, monitor enter/exit) are implemented as extended bytecodes, eliminating the need for a traditional RTOS. The result is extremely low executive overhead with thread to thread context switch times of less than 1µsec. The JEMCore can be synthesized and optimized for power, die size or speed. The JEMCore can be deployed as the main Java native processor or as a coprocessor with a legacy processor to power 3G cellphones, Java Cards™ and STBs. This flexibility gives the price/performance ratio necessary for a wide range of Java based smart Internet mobile and consumer appliances.

Features

32-bit Direct Execution Java Processor Core

- Native JVM bytecodes
- Extended bytecodes for I/O and threading support
- Floating-point arithmetic
- Writeable control store supports custom extended bytecodes

Native Java Threading Support

- Hard real-time, multi-threading kernel in hardware
- Threading operations are atomic, including true Java synchronization
- Built-in deterministic scheduling queues
- Directly supports the Real Time Specification for Java (RTSJ)
- Thread to thread yield in less than 1µsec
- Eliminates traditional RTOS layer

Register file

- 24 registers by 32-bit wide
- Cache of top six accumulator elements
- JTAG communication port

Datapath

- 32-bit ALU
- Floating point disassembly/assembly
- 32-bit barrel shifter
- Count leading/trailing zeros
- Arithmetic overflow/and NaN detection

Bus interface

- Dynamic bus sizing from 8-, 16- to 32-bit
- Write buffer
- HLD/HAK bus arbitration
- Easily customized for other on-chip peripherals

Test access unit

- Low intrusion memory access
- Two on-chip hardware breakpoints
- Software breakpoints
- External breakpoint pin
- Register access
- Single stepping and "step to"
- 1149 interface
- Pseudo-random BIST mode

Optional Multiple JVM Manager (MJM™)

- Supports two independent JVMs
- Programmable deterministic time slicing between JVMs
- Watchdog timers ensure time allocations are maintained
- External signals allow configurable memory protection
- Full Java threading support available to each application including selectable memory management policies.

Designed for ultra-low-power operation

- Less than 1mW/MHz power consumption
- Fully static operation up to 100 MHz
- Implemented in 3.3V and 0.25µm CMOS process

Gate count

- 25K gates
- 4K by 56 ROM
- 10K gates for optional MJM



aJile Java™ Processor Core JEMCore™

System Development Support

A JEMCore based system can be configured to execute in real-time and/or dynamic environments to support a wide range of applications. Using commercial Java IDEs, application developers can create standalone real-time Java applications totally in Java with the performance and memory efficiency of systems programmed in C and assembly. The dynamic runtime supports the CLDC Mobile Information Device Profile (MIDP) to allow Java "MIDlets" to be downloaded and executed dynamically in separate JVM environments. Multiple "MIDlets" can be run simultaneously under the control of the Java Applications Manager (JAM) that maintains memory and execution time allocations. A JEMCore based system can also be configured to execute both real-time and dynamic applications in deterministic time sliced schedule.

The primary components of the development and runtime environments are summarized as follows:

Optimizing Linker/Application Builder

- GUI based application build configuration and control tool - JEM Builder
- Utilizes standard JVM class files generated by commercial Java IDEs
- Statically resolves class files and eliminates unused methods and fields
- Performs bytecode optimizations
- Performs method substitutions (method invokes replaced by extended bytecodes)
- Builds boot tables, class initialization code, and assigns interrupt and trap handlers
- Configures JVM's and memory layout

Java Runtime System

- Java run-time environment based on a J2ME CLDC
- Includes networking classes, storage classes, and Java communications API
- Dynamic runtime includes the JAM (class loader, verifier, scheduler) and GC components
- Device drivers for integrated peripherals and generic physical device interfacing in Java

Application Debugging Tools

- Host-target communications via an IEEE 1149 (JTAG) interface
- Host-based full featured low-level debugger - Charade (Target level debugger threads and routines are not required)
- Host-based JDI provided to interface to commercial JDI compliant source-level debuggers

aJ-100EVB Evaluation System

- 100 MHz aJ-100 based on the JEMCore
- Standard configuration has 4MB Flash and 1MB SRAM
- 10BaseT ethernet
- LCD controller, touch screen controller, USB port, MMC socket, Dallas Semi One Wire™ port
- 2 serial I/O ports (IrDA option)
- 40 GPIO ports
- 4 SPI ports
- Compact Flash and local bus connector

Deliverables for aJile Java processor core

- Reference manual
- JEMCore Microarchitecture
- Verification guide
- Microarchitecture simulator (cycle accurate)
- Validated RTL model (VHDL)
- aJ-100EVB Development system

aJile Java™ Processor Core JEMCore™

aJile processor core architecture

The JEMCore is based on a microprogrammed architecture using both ROM and RAM control stores. The architecture includes a 32-bit ALU, a barrel shifter, a 24-element 32-bit wide register file, a test interface unit and a processor bus interface. The processor bus interface consists of a 32-bit address, 32-bit data bus and control signals. The bus can be dynamically programmed to interface to 32-bit, 16-bit and 8-bit memory subsystems. The bus interface also employs a simple request/acknowledge protocol, which can be seamlessly interfaced to a legacy CPU in a shared memory and I/O architecture. Figure 1 is a simplified block diagram of JEMCore.

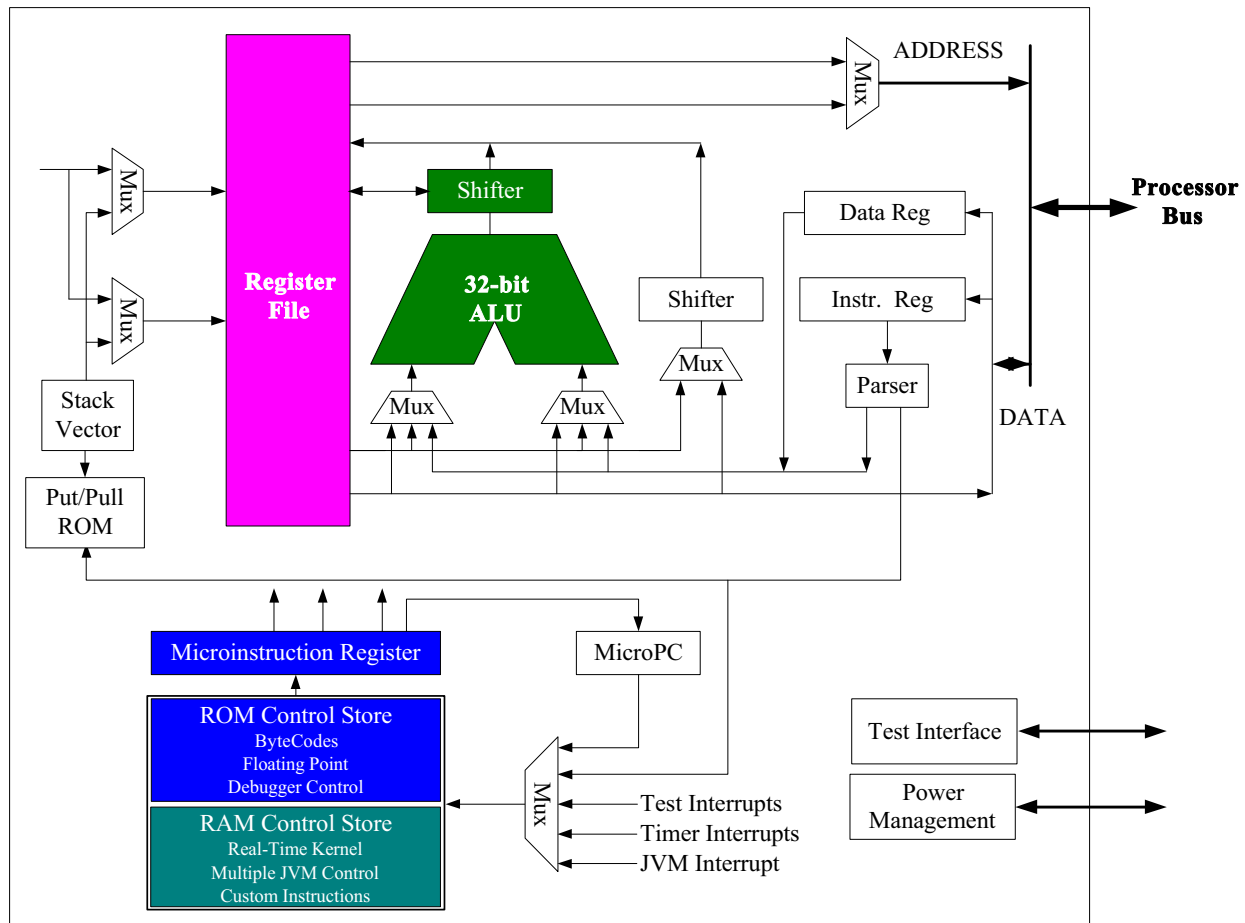


Figure 1: aJile Java Processor Core Architecture

Custom Instructions

JEMCore enables the use of custom microcode to implement new instructions. The new instructions can significantly increase the performance of frequently used algorithms. The power of custom instructions is reflected in the threading instructions of the aJile Java processor core. For example, the yield instruction results in a thread-to-thread switch of one microsecond while a typical RTOS written in a high level language may take several milliseconds.

For an example of where a custom instruction could be used consider the square root method in java.lang.Math. This method could be implemented with microcode and initiated via an instruction. When processing a class file, the aJile tools will replace invokes of java.lang.Math.sqrt with the custom sqrt instruction. The performance advantage of a custom instruction varies from 5X for simple algorithms to 50X for complex algorithms.

aJile Java™ Processor Core JEMCore™

Multiple JVM Manager (MJM)

The multiple JVM feature allows up to two independent Java applications to execute with a deterministic, time-sliced schedule and with full memory protection. Within its bounded execution interval and memory space, each JVM environment can employ its own multi-threading and memory utilization policies without threat of intervention by faulty or malicious applications.

The Multiple JVM Manger (MJM) provides timing resources and interrupt logic to ensure that no JVM (applications) can interfere with the processing needs of other JVMs. The MJM provides a timer to maintain the time slices allotted to each logical JVM. A separate timer is provided for each JVM (total of four) to schedule threads for that JVM. Figure 2 illustrates the MJM.

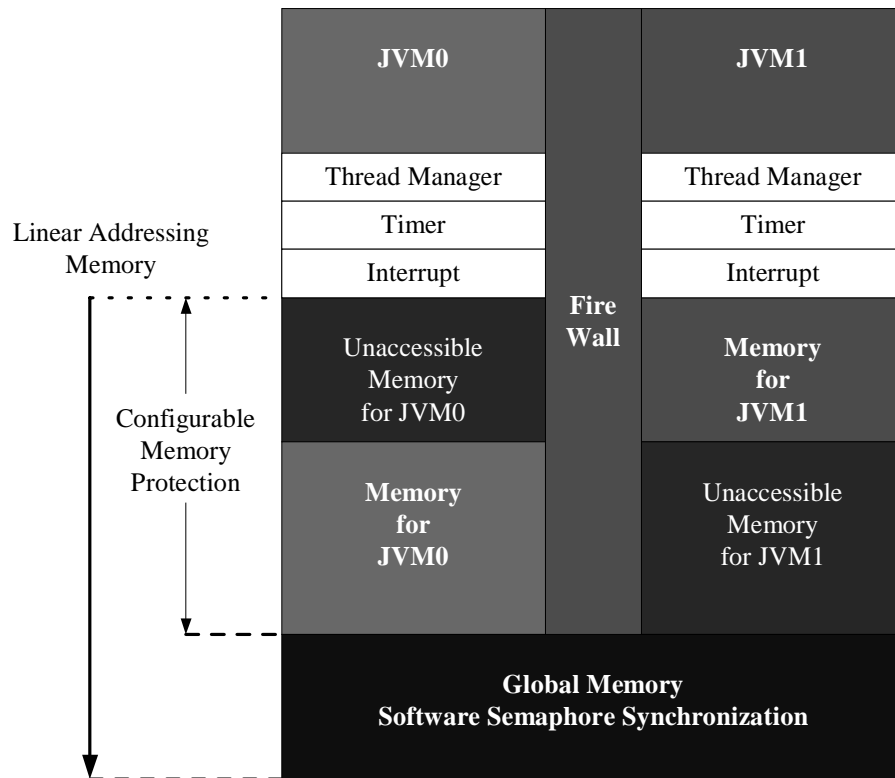


Figure 2: Multiple JVM Management (MJM)

Target Market

JEMCore is a licensable core that can be deployed as a native Java processor or coprocessor. It can be integrated with application specific I/O functions to design a complete embedded processor for:

- 3G Cellphones
- Java Cards
- STBs