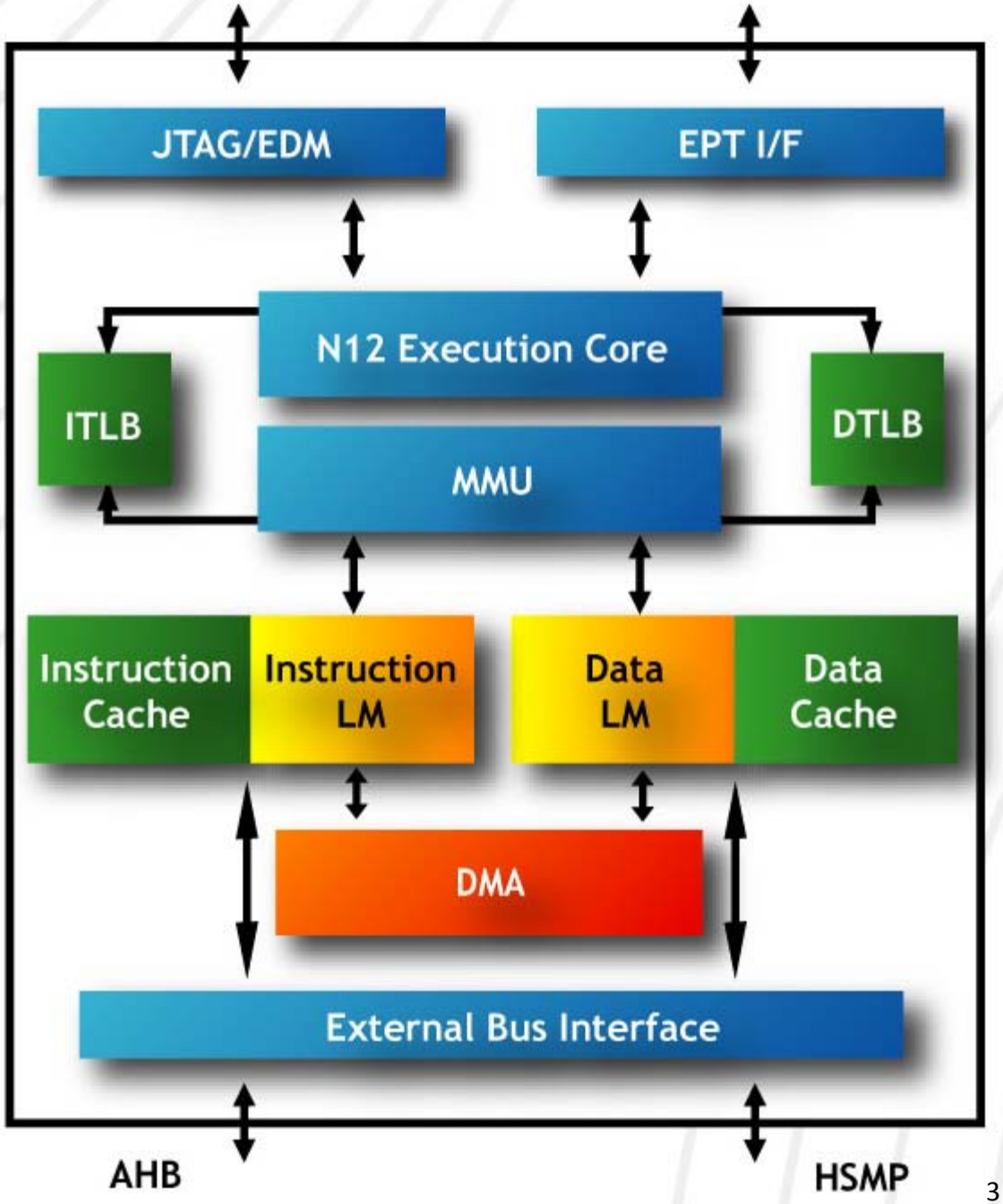
硬體架構與開發環境簡介

ARCHITECTURE

AndesCore N12



AndesCore

- CPU Core
  - 16-/32-bit mixable instruction format
  - 32 general-purpose 32-bit registers
  - 8-stage pipeline
  - Dynamic branch prediction
  - 32/64/128/256 BTB
  - Return address stack (RAS)
  - Vector interrupts for internal/external interrupt controller with 6 hardware interrupt signals
  - 3 HW-level nested interruptions
  - User and super-user mode support
  - Memory-mapped I/O
  - Address space up to 4GB
- Memory Management Unit
  - TLB
  - Optional hardware page table walker
  - Two groups of page size support
- Memory Subsystem
  - I & D cache
  - I & D local memory (LM)
  -
- Bus Interface
  - Synchronous/Asynchronous AHB bus: 0, 1 or 2 ports
  - Synchronous High speed memory port(HSMP): 0, 1 or 2 ports
  -
- Debug
  - JTAG debug interface
  - Embedded debug module (EDM)
  - Optional embedded program tracer interface
  -
- Miscellaneous
  - Programmable data endian control
  - Performance monitoring mechanism

Benefit

- Performance
  - Versatile memory access instructions
  - Burst support for uncached load multiple
  - Efficient atomic access synchronization without locking system bus
  - Low latency vectored interrupt improving real-time performance
  - Zero-wait-state local memory with 1D/2D DMA
  - MMU
    - Optional HW page table walker
    - TLB management instructions
- Flexibility
  - Memory-mapped IO space
  - PC-relative jumps for position independent code
  - JTAG-based debug support
  - Performance monitors for performance tuning
  - Bi-endian
- Power Management
  - Clock-gated pipeline
  - Low-power mode support instructions

Applications

- Digital TV
- Digital Home
- Set top Box
- MFP
- Switch/Router
- Communication
- Smart Phones

Development Tools

- **AndeSight**
  - Integrated development environment
- **AndESLive**
  - ESL integrated virtual environment

AndeStar ISA

- In order to achieve optimal system performance, code density and power efficiency, a set of mixed-length (32/16-bit) instructions has been implemented for Andes ISA
  1. The 32-bit and 16-bit instructions can be freely mixed in a program.
  2. The 16-bit instructions are a frequently used subset of 32-bit instructions.
  3. No 32/16-bit mode switching performance penalty when executing mixed 32-bit and 16-bit instructions.
  4. The 32/16-bit mixed-length ISA is in a big-endian format.
  5. The ISA is a RISC-style register-based instruction set.
  6. 5-bit register index in 32-bit instruction format.
  7. 5/4/3-bit register index in 16-bit instruction format.
  8. The ISA provides hardware acceleration for a mixed-endian environment.

32-Bit Baseline Instruction

- Data-processing instructions
  - ADDI, SUBI, ANDI, ORI, ...
- Load and Store Instructions
  - LW, LWI, SW, SWI, ...
- Jump and Branch Instructions
  - J, JAL, JR, RET, ...
- Privilege Resource Access Instructions
  - MFSR, MTSR, ...
- Miscellaneous Instructions
  - CMOVZ, CMOVN, MSYNC ...

16-Bit Baseline Instruction

- 32-bit Instruction Mapping
  - For example

Mnemonic	Instruction	32-Bit Operation
ADDI45 rt4, imm5u	Add Word Immediate	ADDI rt5, rt5, ZE(imm5u)
ADDI333 rt3, ra3, imm3u	Add Word Immediate	ADDI rt5, ra5, ZE(imm3u)
SUBI45 rt4, imm5u	Subtract Word Immediate	ADDI rt5, rt5, NEG(imm5u)
SUBI333 rt3, ra3, imm3u	Subtract Word Immediate	ADDI rt5, ra5, NEG(imm3u)

32-bit ISA Extension

- Performance Extension V1
  - The performance extension V1 instructions are used to optimize high level language (C / C++) performance.

  Ex.

  ABS, absolute with register

  AVE, average two signed integers with rounding

  MAX, return the lager

  MIN, return the smaller

32-bit ISA Extension

- 32-bit String Extension
  - The String Extension instructions are use to optimize text and string processing related algorithms.

  EX.

  FFB, find first byte

  FFBI, find first byte immediate

  FIMISM, find first mis-match

  FLMISM, find last mis-match

Memory Management Unit

- The Andes MMU's main component is a Translation Look-aside Buffer (TLB) structure.
  - It uses virtual page to physical page address translations managing physical memory resources
  - It provides address space protection among different processes.
  - It provides memory page attributes such as cacheability.
  - It can be completely managed by software or a hardware-assisted page table walker (HPTWK) can be enabled to improve the TLB fill performance and/or the TLB capacity.

Interruption Architecture

- an Interruption is a control flow change of normal instructionexecution generated by an Interrupt or an Exception
- An interrupt is a control flow change event generated by an asynchronous internal or external source
  - Hardware interrupt is any interrupt event generated from external agents or AndeScore™.
  - Software interrupt is an interrupt event generated by an instruction executing in the AndeScore™.
- An exception is a control flow change event generated as a by-product of instruction execution

Local Memory

- Andes architecture provides support for fast-accessed low latency local memories in the AndeScore™.
- The local memories are divided into instruction local memory (ILM) and data local memory (DLM) for increased performance and design simplicity.

Local Memory DMA

- The local memory DMA is provided to transfer blocks of data between AndeScore ™ local memory and external, off-core, memory in parallel with the AndeScore ™ execution pipeline.
- To transfer large amounts of data to or from local memory, it is more efficient to use the DMA engine instead of the AndeScore ™ load/store instructions.

DEVELOPMENT TOOLS

AndeSight / AndESLive

AndeSight

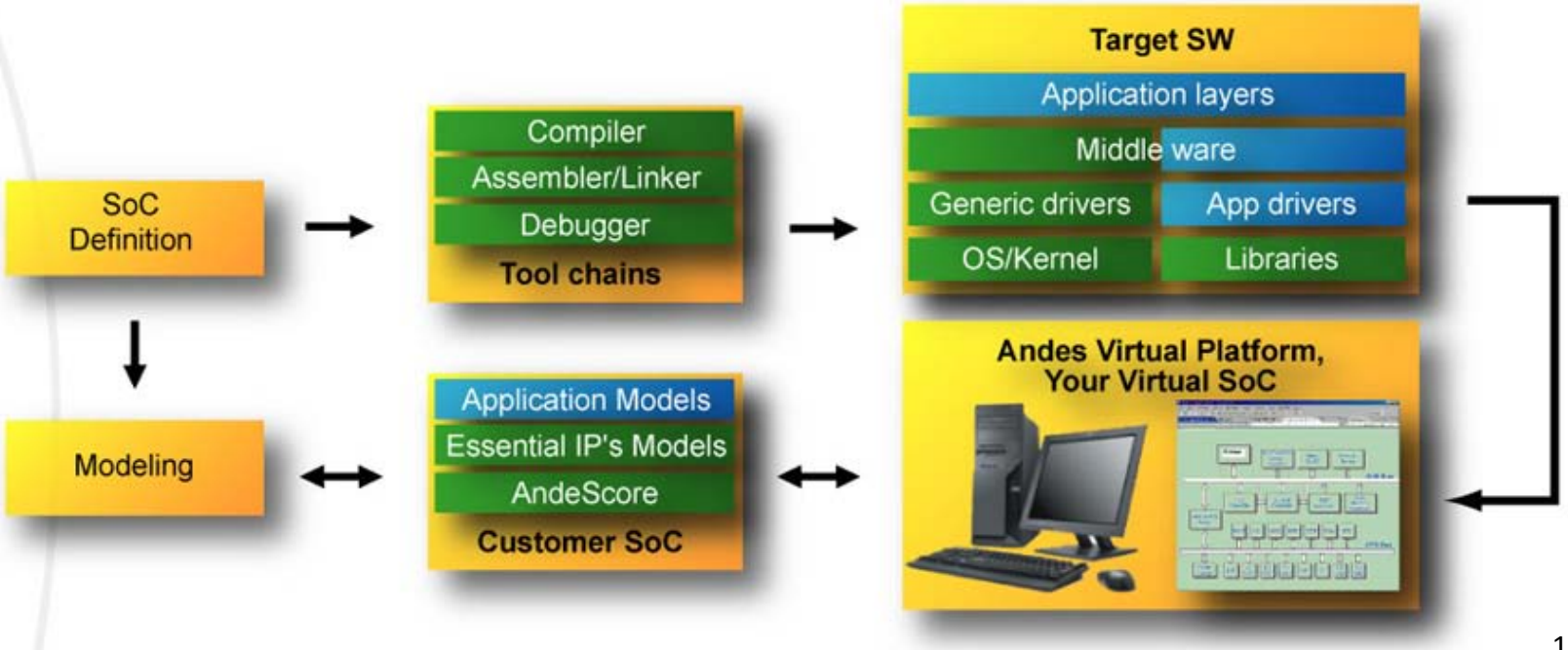
- Coder
- Debugger
- Profiler
- Target
- Manager

Toolchains

- Compiler
- Assembler
- Linker
- Debugger

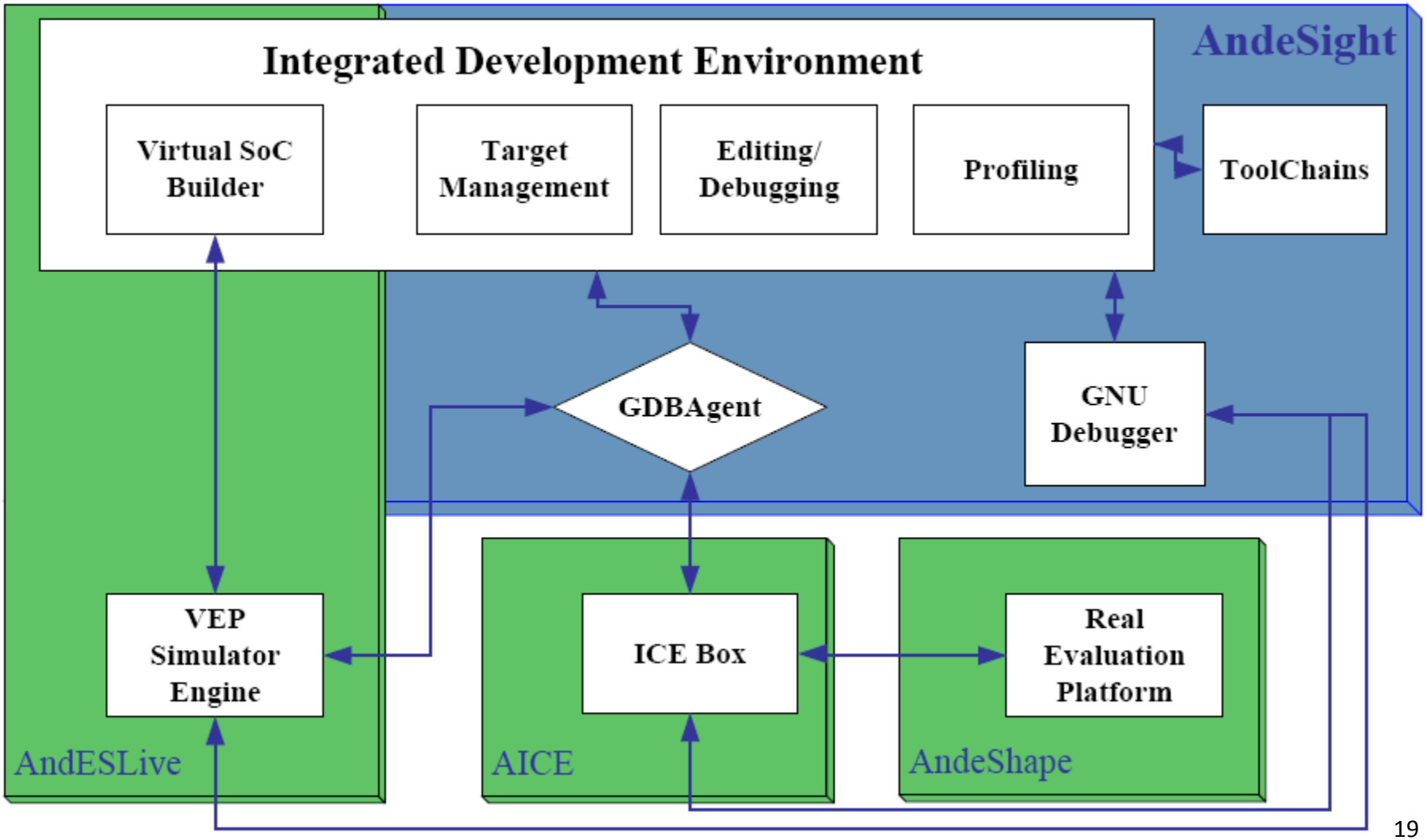
AndESLive

- Pre-defined models of AndesCore
- Peripheral IPs, and bus
- Virtual SoC Builder
- Visibility of debugging
- Simulation of I/O devices

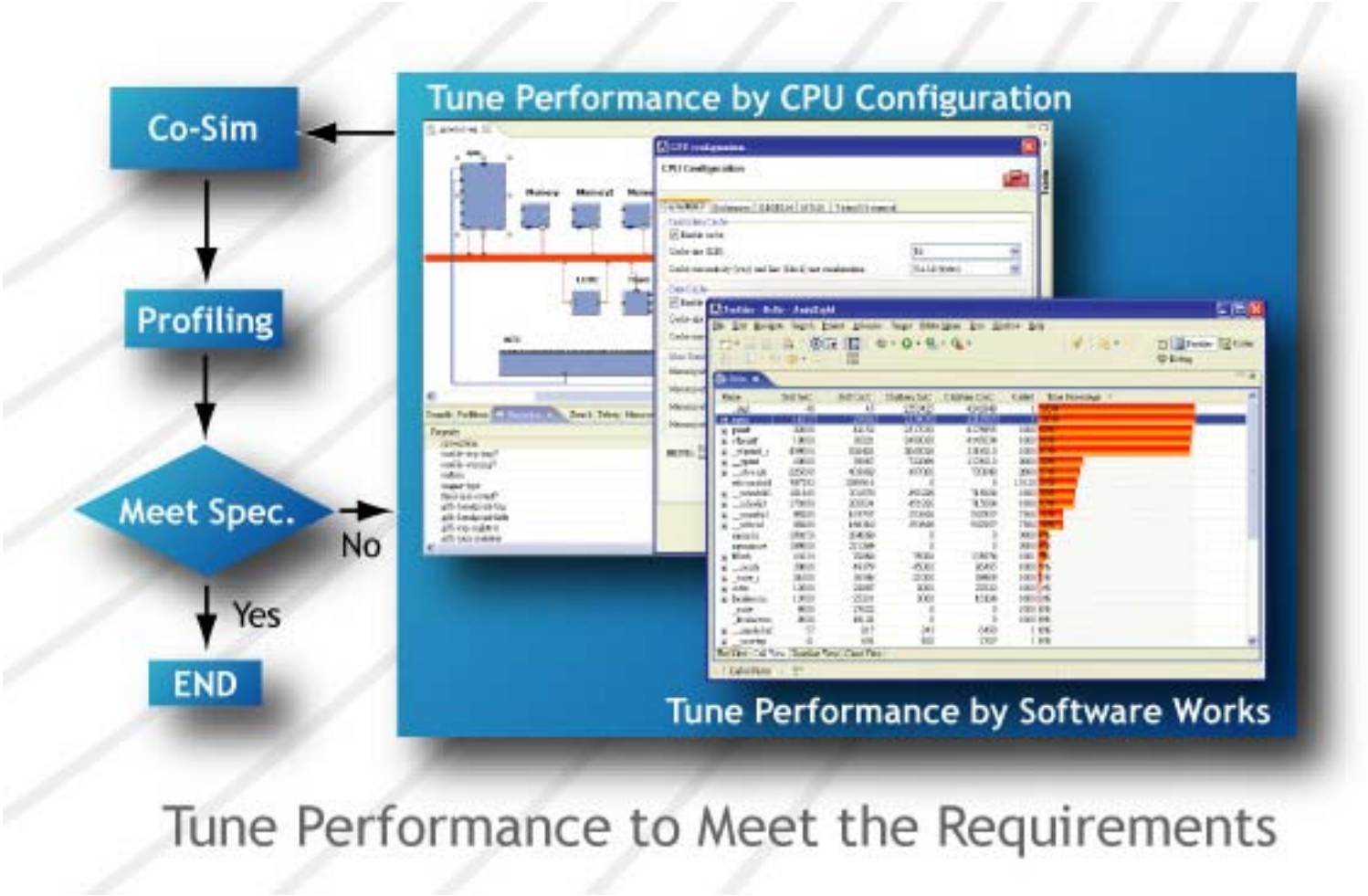


Virtual SoC and Software Design Flow

AndeSight


AndeSight
Integrated Development Environment
Virtual SoC Builder
Target Management
Editing/ Debugging
Profiling
ToolChains
GDBAgent
GNU Debugger
VEP Simulator Engine
ICE Box
Real Evaluation Platform
AndESLive
AICE
AndeShape

Software Profiling



AndESLive

- AndESLive consists of a SoC simulator engine and a Virtual SoC Builder where the user may:
  - construct a Virtual Evaluation Platform with AndESLive™ IP models
  - configure the existing AndESLive™ IP models, including the CPU model
  - configure new user defined IP models
  - Export and import CPU configuration file
  - Run, debug, and profile a program on a Virtual Evaluation Platform

TOOLCHAIN

The GNU Tool-Chain

- Why using the GNU Tool-chain :
  - GNU is a complete, Unix-like operating system that has been in development for just over twenty years
  - GNU software is known for it's stability and standard compliance.
  - GNU toolchain is open source which be easy to modify

The GNU Tool-Chain

- The GNU toolchain consists :
  - Compiler – gcc
  - Assembler – binutils : as
  - Linker-- binutils : ld
  - Library– glibc
  - Debugger-- gdb

- gcc : can be native to the host or a cross-compiler

GNU Compiler Collection

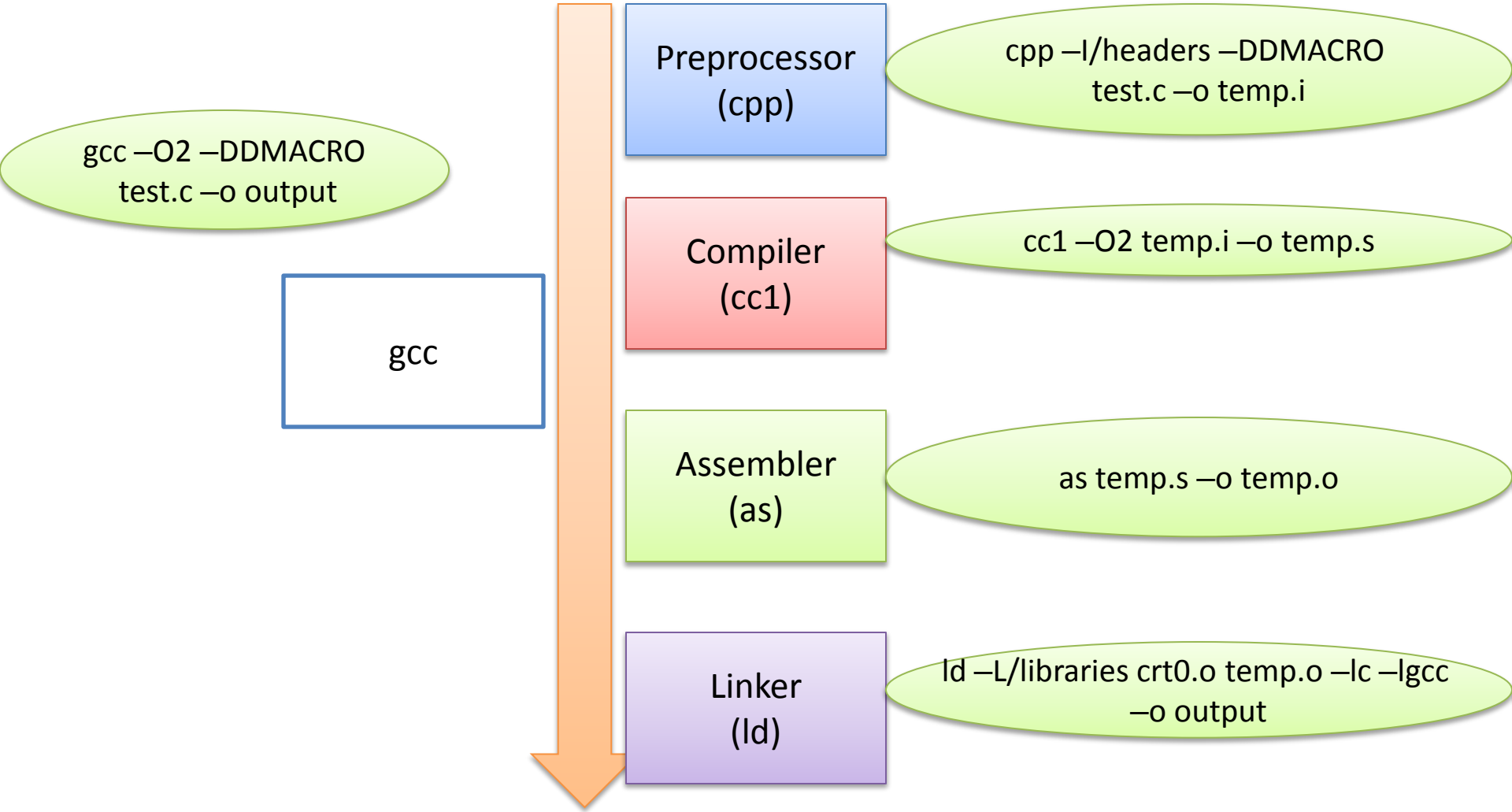
- gcc - GNU Compiler Collection
- gcc is a full-featured ANSI C compiler with support for K&R C, C++, Objective C, Java, and Fortran.
- GCC provides many levels of source code error checking , produces debugging information, and can perform many different optimizations to the resulting object code.

GNU Compiler Collection

- The GNU Compiler for Java is now integrated and supported: GCJ can compile Java source or Java bytecodes to either native code or Java class files.

- GCC now supports the Intel IA-64 processor, so completely free software systems will be able to run on the IA-64 architecture as soon as it is released

GCC Compilation Stage



GNU Binutils

- Binutils are a collection of binary tools like assembler, linker, disassembler … .
- Binutils is to create and manipulate binary executable files.
  - T he main tools for Binutils

    ld - the GNU linker.

    as - the GNU assemble

Binutils (1/2)

Name	Description
ld	the GNU linker.
as	the GNU assembler.
addr2line	Converts addresses into filenames and line numbers.
ar	A utility for creating, modifying and extracting from archives
c++filt	Filter to demangle encoded C++ symbols.
dlltool	Creates files for building and using DLLs.
gold	A new, faster, ELF only linker, still in beta test.
gprof	Displays profiling information.
nlmconv	Converts object code into an NLM.
nm	Lists symbols from object files.

Binutils (2/2)

Name	Description
objcopy	Copys and translates object files.
objdump	Displays information from object files.
ranlib	Generates an index to the contents of an archive.
readelf	Displays information from any ELF format object file.
size	Lists the section sizes of an object or archive file.
strings	Lists printable strings from files.
strip	Discards symbols.
windmc	A Windows compatible message compiler.
windres	A compiler for Windows resource files.

GNU C Library

- Any Unix-like operating system needs a C library (defines the “system calls” and other basic facilities such as open, malloc, printf, exit...)
- The GNU C library is used as the C library in the GNU system and most systems with the Linux kernel.
- The GNU C library is a large blob of glue code, which tries to give it's best to hide kernel specific functions from you(e.g. You can use the same function name and do not care the difference between different kernel)

GDB

- What is GDB
  - GDB : The GNU Project Debugger
  - Allows you to see what is going on `inside‘ another program while it executes
  - Allows you to see what another program was doing at the moment it crashed
  - GDB can run on most popular UNIX and Microsoft Windows variants.

GDB

- Function :
  - Start your program, specifying anything that might affect its behavior.
  - Make your program stop on specified conditions.
  - Examine what has happened, when your program has stopped.
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

GDB

- Supported Languages
  - C
  - C++
  - Pascal
  - Objective-C
  - Many other languages

Cygwin

- What is cygwin?
  - Cygwin is a Linux-like environment for Windows. It consists of two parts:
    - A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality.
    - A collection of tools which provide Linux look and feel.

- The Cygwin DLL currently works with all recent, commercially released x86 32 bit and 64 bit versions of Windows, with the exception of Windows CE.

Reference

- GNU Compiler Collection
  - http://gcc.gnu.org/
- GNU Binutils
  - http://sources.redhat.com/binutils/
- GNU C Library
  - http://www.gnu.org/software/libc/
- GNU Project Debugger : GDB
  - http://www.gnu.org/software/gdb/
- Cygwin
  - http://cygwin.com/