

Taking RISC-V to Mainstream SoCs



Charlie Su, Ph.D.
CTO and SVP of R&D
2017/12/18

Agenda

- ❖ **Andes Corporate Overview**
- ❖ **RISC-V and AndeStar™ V5 Architecture**
- ❖ **V5 Processors NX25 and N25**
- ❖ **Andes-Enhanced Features**
- ❖ **Support for V5 Processors**
- ❖ **Concluding Remarks**

Agenda

- ❖ **Andes Corporate Overview**
- ❖ RISC-V and AndeStar™ V5 Architecture
- ❖ V5 Processors NX25 and N25
- ❖ Andes-Enhanced Features
- ❖ Support for V5 Processors
- ❖ Concluding Remarks

Andes Overview

Andes Mission

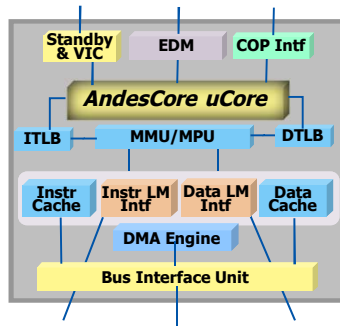
- Innovate **performance-efficient** processor IP Solutions

Andes Highlights

- Founded in **Hsinchu Science Park**, Taiwan (2005)
- EETimes' Silicon 60 **Hot Startups to Watch** (2012)
- **TSMC OIP Award** "Partner of the Year" for New IP (2015)
- A founding member of **RISC-V Foundation** (2016)
- **IPO on Taiwan Stock Exchange** (March 2017)

Andes Comprehensive Products

Processor Cores AndeStar™

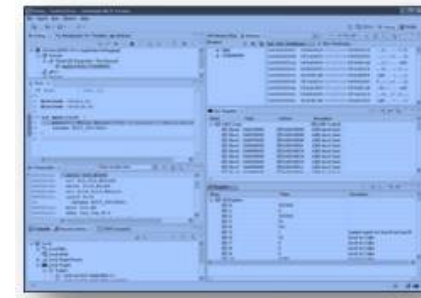


Processor Architecture AndeStar™

```

smw.adm $r1,[$sp],$r5,0x0
smw.adm $sp,[$sp],$sp,0x2
addi    $sp,$sp,-8
sethi   $r1,0x50a
lwi     $r1,[$r1+#0x98]
mov55   $r2,$r0
mov55   $r0,$r1
lwi     $r1,[$r1+#0x8]
addi    $r3,$sp,12
    
```

Development Tools AndeSight™



Development Platforms AndeShape™



Software Stacks AndeSoft™



Andes Business Highlights

❖ Over 130 commercial licensees

- Taiwan, China, Korea, Japan, US, Europe
- >2.3 Billion Andes-Embedded™ SoCs
- >200 agreements

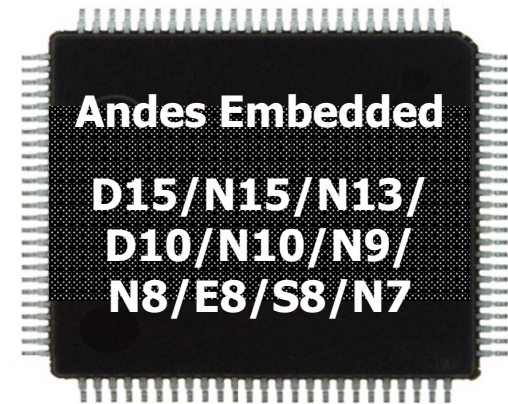
❖ AndeSight™ IDE:

- >11,000 installations

❖ Ecosystem

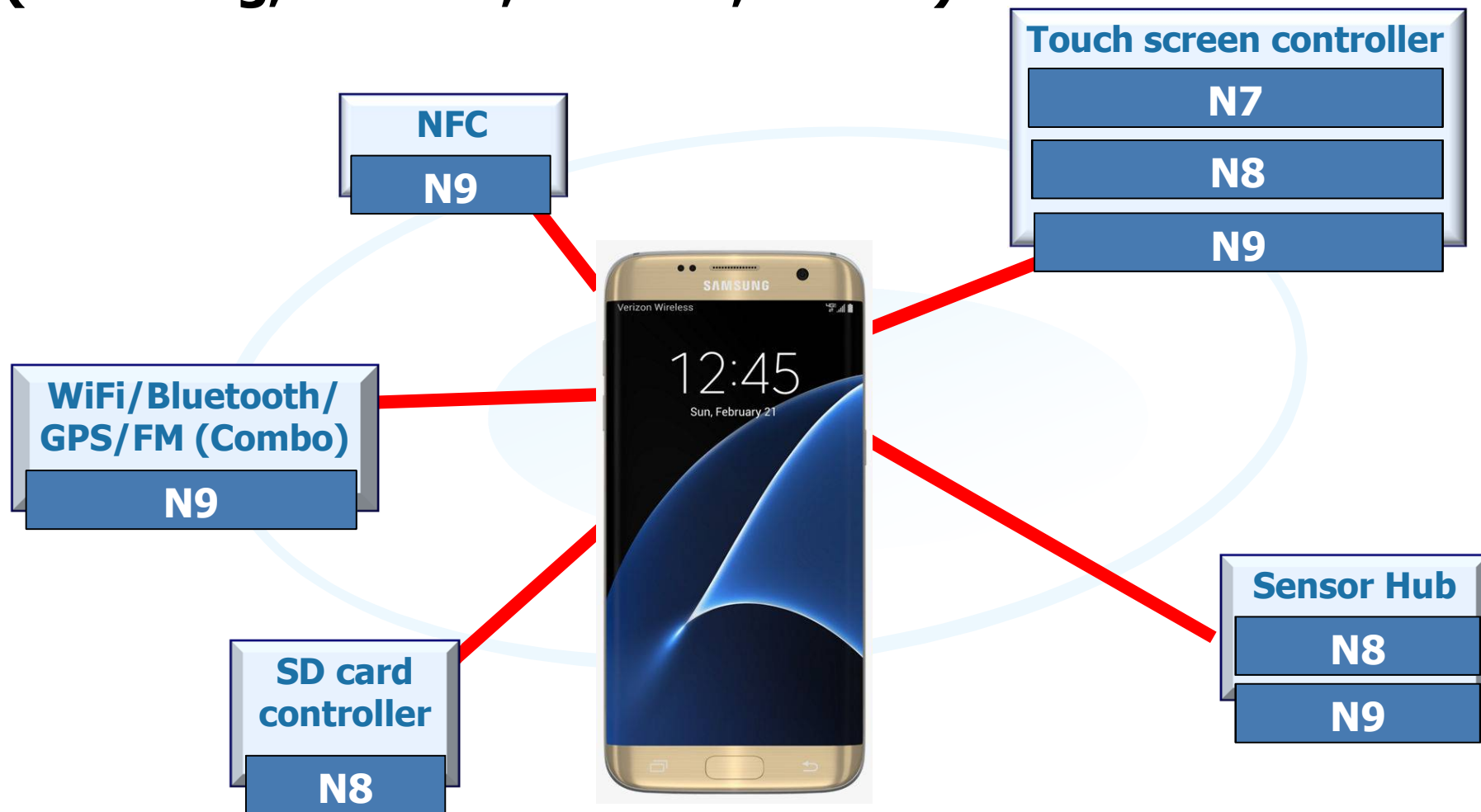
- >100 partners

❖ Diversified applications based on Bare Metal, RTOSes, and Linux



Andes Embedded in Smart Phones

**1 in 5 Smart Phones are with Andes Embedded
(Samsung, Huawei, Xiaomi, and ...)**



Andes Embedded in Consumer Devices, Cars and Datacenters



Switch:
MXIC Flash Ctrl



Echo Dot2:
Mediatek WiFi IoT



SD Card:
Phison Ctrl



X-Trail:
ADAS Ctrl



- ❖ In leading machine learning computers for datacenter
- ❖ In tier-one switch routers for datacenter

Andes Product Strength

- ❖ Architecture components beyond basic RISC kernel
 - Major extensions: DSP, Security, Custom Instruction
 - Architecture features: CoDense™, StackSafe™, PowerBrake
 - ➔ AndeStar™ architecture V1 to V3
- ❖ Efficient processor pipeline for leading PPA
 - 2-stage to 8-stage, dual-issue
- ❖ Highly-optimized compiler and libraries
 - Very compact code and very good performance
- ❖ Feature-rich AndeSight IDE
- ❖ >10 RTOS and Linux support
- ❖ Commercial-grade verification for all products
 - IP design: directed, directed random, and formal model checking
 - Compiler, debugger, GUI IDE:
 - ◆ SW test suites: open-source, commercial, and in-house
- ❖ Professional supporting infrastructure

Agenda

- ❖ Andes Corporate Overview
- ❖ **RISC-V and AndeStar™ V5 Architecture**
- ❖ V5 Processors NX25 and N25
- ❖ Andes-Enhanced Features
- ❖ Support for V5 Processors
- ❖ Concluding Remarks

RISC-V and AndeStar V5

- ❖ Andes joined RISC-V Foundation last year
 - ❖ The **1st established CPU IP vendor** adopting RISC-V
- ❖ What we like about RISC-V
 - growing **ecosystem**
 - **compact, modular and customizable** ISA
- ❖ Andes can help
 - grow the RISC-V ecosystem faster
 - **take RISC-V to the diversified mainstream SoC's**
- ❖ **AndeStar V5 adopted RISC-V as its subset**
 - With additional good features Andes developed in the past
- ❖ What Andes has been doing
 - A major contributor to RISC-V tools such as GCC, binutils, newlib, and recently LLVM and LLD
 - Started contributing architecture extensions too



AndeStar V5 and RISC-V

❖ **RISC-V ISA is very similar to AndeStar V3.**

RISC Characteristics	Andes V3	RISC-V
Instruction width	16,32	16,32
GPR number	16,32	16,32
Delayed branch	No	No
Predicated execution	No	No
Condition code	No	No
PC as GPR	No	No
\$R0 as hardwired 0	No	Yes

➔ **Migrating to V5 is straightforward in design philosophy**

Agenda

- ❖ Andes Corporate Overview
- ❖ RISC-V and AndeStar™ V5 Architecture
- ❖ **V5 Processors NX25 and N25**
- ❖ Andes-Enhanced Features
- ❖ Support for V5 Processors
- ❖ Concluding Remarks

V5 AndesCore™ NX25/N25

➔ **Fast and small cores for control tasks** in storage, networking, AI, and more.

❖ **NX25: 64-bit, N25: 32-bit**

- Designed from scratch

❖ **AndeStar V5m ISA**

- Superset of RISC-V IMAC ISA

❖ **5-stage pipeline**

- A full-cycle for critical SRAM

❖ **Configurable multiplier**

- Sequential: 1~8 bits per cycle

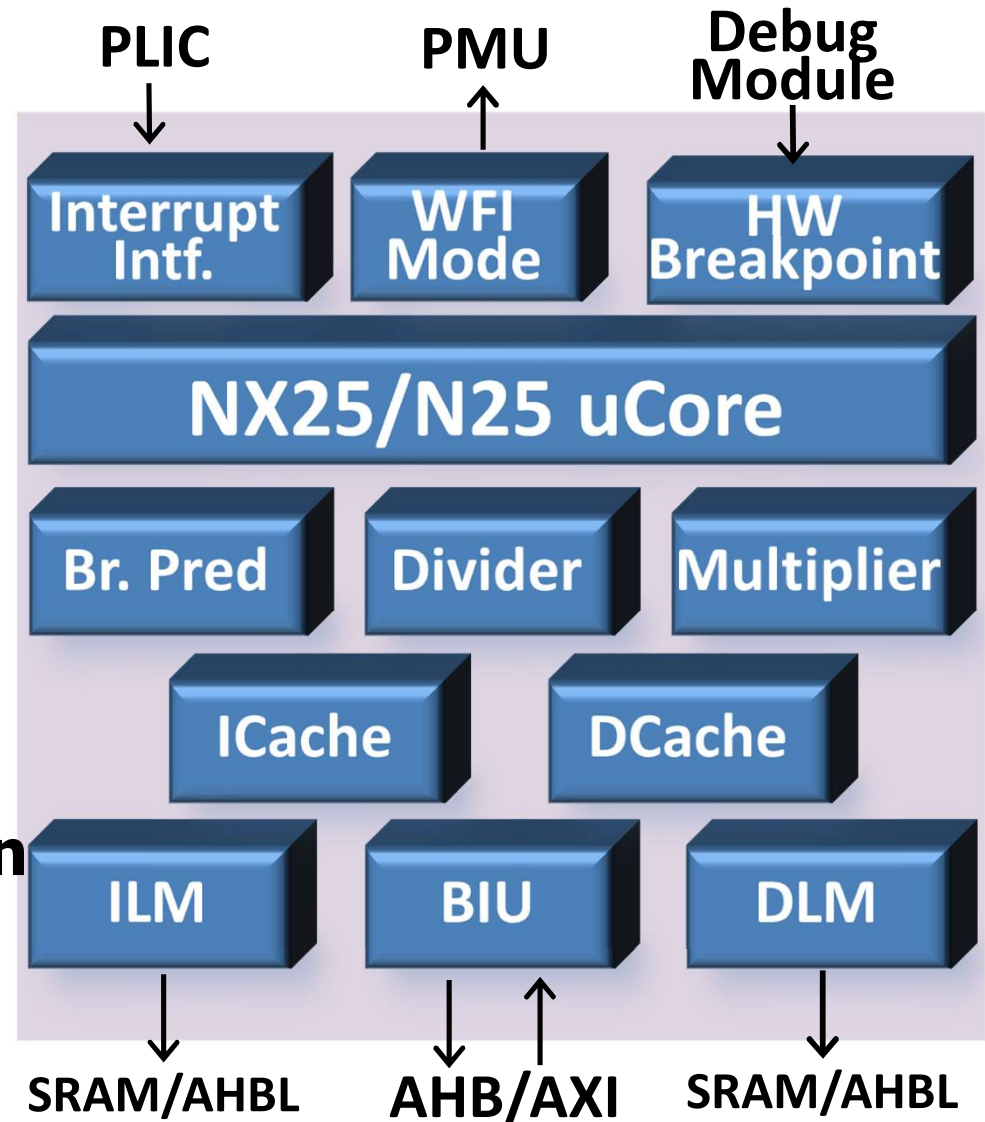
- Parallel: pipelined 2 cycles

❖ **Optional branch prediction**

- Branch Target Buffer (BTB)

- Branch History Table (BHT)

- Return Address Stack (RAS)



V5 AndesCore™ NX25/N25

❖ Memory subsystem

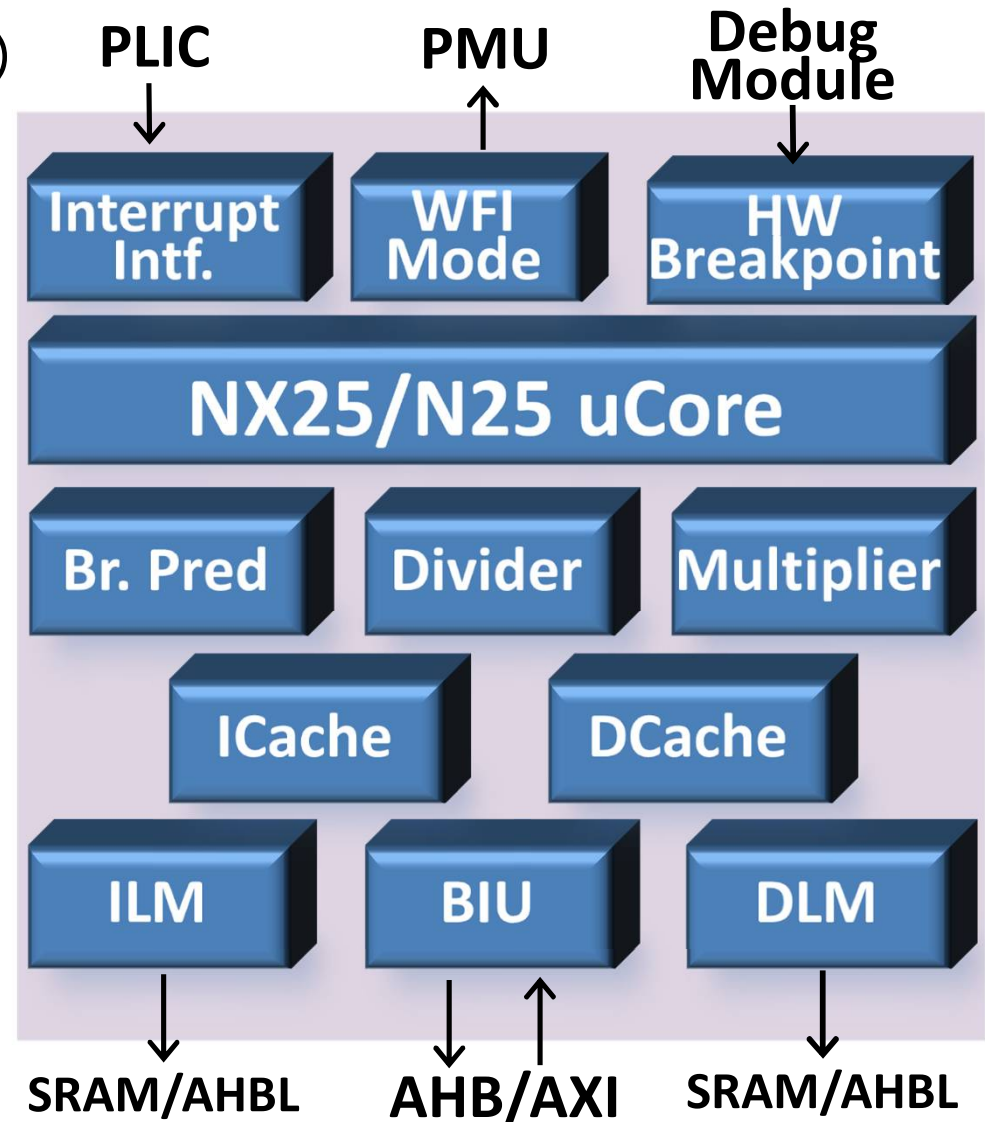
- Optional I/D Local Memory (LM)
 - ◆ Size: 4KB to 16MB
 - ◆ Interface: SRAM or AHB-lite
- Optional I/D caches
 - ◆ Size: 8KB to 64KB
 - ◆ Direct-map, 2-way, 4-way
- Optional error protection:
 - ◆ Parity or ECC (SECDED)

❖ Bus interface

- A master port
- An optional slave port

❖ JTAG debug module

- up to 8 triggers
(breakpoints/watchpoints)



N25/NX25: Performance Data

Features	N25 (32b)		NX25 (64b)	
AndeStar V5m ISA	RV32IMAC + Andes Ext.		RV64IMAC + Andes Ext.	
Bus Interface	Data: AXI64, AHB64/32 Addr: 32 bits		Data: AXI64 Addr: 32-64 bits	
Best Performance ¹	2.86 DMIPS/MHz 3.48 Coremark/MHz		3.20 DMIPS/MHz 3.45 Coremark/MHz	
Frequency (worst case) ²	1 GHz	50 MHz	1 GHz	50 MHz
Gate Count (K gates) ²	37	30	63	50
Die Area (mm ²) ²	0.029	0.024	0.049	0.039
Dynamic Power (uW/MHz) ²	11	7.8	13	9.4

1: Use AndeSight v3.1.0 for V5; both DMIPS/MHz don't apply no-inline option. Following Dhrystone's no-inline ground rules, the DMIPS/MHz becomes 1.97 and 2.19, resp.

2: TSMC 28HPC RVT 12-track library, M31 high-speed memory, die area: logic only with 85% utilization.

Agenda

- ❖ Andes Corporate Overview
- ❖ RISC-V and AndeStar™ V5 Architecture
- ❖ V5 Processors NX25 and N25
- ❖ **Andes-Enhanced Features**
- ❖ Support for V5 Processors
- ❖ Concluding Remarks

AndeStar V5 Extension

- ❖ **PLIC**: vectored dispatch with priority-based preemption
 - ➔ Address the limitation in the current scheme
 - **Fast**: important for the critical interrupts
 - ◆ Save >30 instructions (>50%) for dispatch and SW preemption overhead (before entering the actual C ISR)
 - **Easy to program**
 - ◆ SW initializes priorities once and never needs to deal with it again
 - **Scalable** in the number of interrupt sources
 - **Backward-compatible**:
 - ◆ Compatible on reset; turn on the extension by setting a mode bit
 - Flexible coving: up to 1023 sources, 255 priorities, 16 targets
- ❖ **StackSafe™**: HW supported stack protection
- ❖ **PowerBrake**: Stalling pipeline at a selected percentage of cycles to save power

Andes DSP ISA Extension

❖ Feature highlights:

- **> 150 efficient 32-bit DSP instructions using only GPRs**
- **saturation and/or rounding**
- Data types: integer (32b, 16b, 8b) and fractional (Q31, Q15, Q7)
- 16-bit and 8-bit **SIMD** instructions
 - ◆ 16-bit: +, -, x, min, max, abs, clip, compare, <<, >>, signed, unsigned
 - ◆ 8-bit: +, -, min, max, abs, unpack, compare, signed, unsigned
- **64-bit signed/unsigned accumulation with multiplication**
 - ◆ **E.g., $64 += 16 \times 16 + 16 \times 16$**
- **Zero-Overhead Loop**

❖ SW support:

- Compiler generation of SIMD instructions based on vector data type
- Intrinsic functions for using instructions in C
- >200 DSP-ISA-optimized DSP libraries
 - ◆ **110%** performance boost with DSP ISA (average of integer functions)

❖ Andes donated it as a basis of “P” extension

- High level semantics already posted on isa-dev mailing list in riscv.org
- We’re extending it for RV64

Performance of Andes DSP ISA

❖ Helix MP3 decoder (open source)

GCC Compiler	Decode (MCPS)
Baseline Compiler	22.64
DSP Compiler	11.40
Speedup	~ 2x

❖ G.729 voice codec (open source)

GCC Compiler	Encode (MCPS)	Decode (MCPS)
Baseline Compiler	69.03	21.70
DSP Compiler + DSP Intrinsic	13.75	4.03
Speedup	> 5x	> 5x

* MCPS, Millions of Cycles Per Second: the lower the better

Andes Custom Extension™ (ACE)



❖ **Benefit:**

- Allow application-specific acceleration and energy reduction

❖ The powerful tool **COPILLOT** automatically generates

- **RTL** for instruction decode, operand accesses (for registers, memory and ports), dependence checking, result gathering, etc.
- **SW tools:** compiler, debugger, and C or SystemC simulator
- Verification patterns/environment and reports errors

❖ Instruction types:

- scalar or vector, running in the foreground or background

❖ Operand types:

- Standard operands: GPR, memory
- Custom registers/memory with arbitrary width and number

Madd32: A Half-Page ACE Description



madd32.ace

```
insn madd32 {
  operand= {io gpr acc,
            in gpr dat, in gpr coef};
  csim= %{
    acc+=(dat & 0xffff) * (coef & 0xffff)
          + (dat >>16)   * (coef >>16);
  %};
  latency= "1";
};
```

madd32.v

```
//ACE_BEGIN: madd32
  assign acc_out = acc_in
    + dat[15:0] * coef[15:0]
    + dat[31:16] * coef[31:16];
//ACE_END
```

❖ File madd32.ace: ACE definition file

- **insn**: define instruction name, "madd32"
- **op(erand)**: operand names and attributes (imm, in/out/io gpr, etc.)
- **csim**: instruction semantics in C for instruction set simulator
- **latency**: estimated cycles spent on instruction execution; default is 1

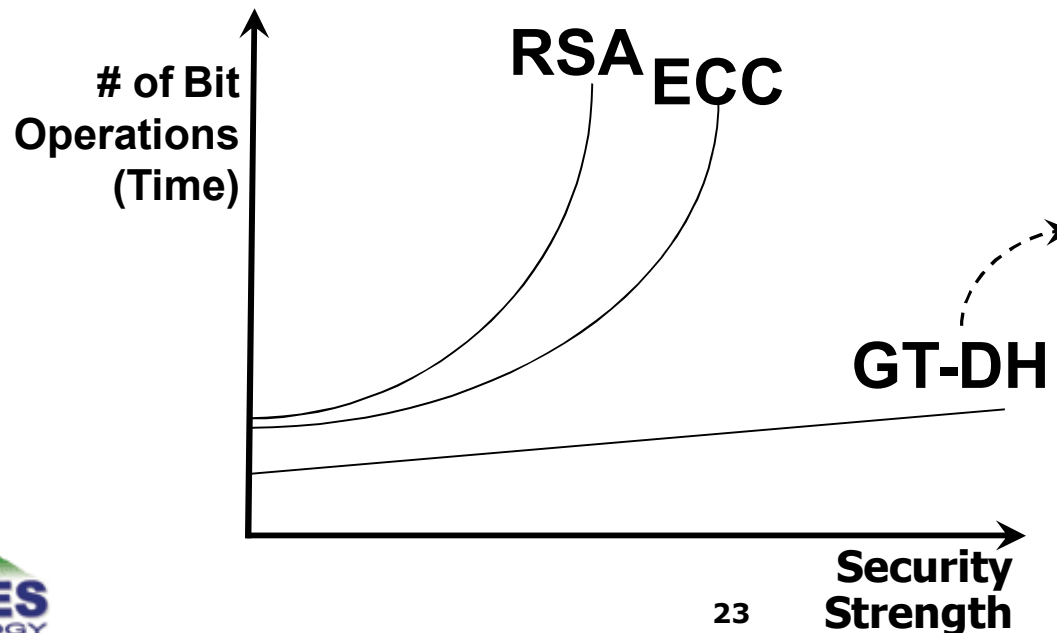
❖ File madd32.v: concise Verilog RTL

- **//ACE_BEGIN ... //ACE_END**: instruction-specific Verilog logic

SecureRF WalnutDSA (Digital Signature Algorithm)



- ❖ **IoT security: All agree it's important, but few actually implement it. Why?**
 - Due to the cost (gate count, memory, and/or power consumption) to support legacy security methods
- ❖ **SecureRF's asymmetric public-key WalnutDSA solutions** (using Group Theoretic Diffie-Hellman, or GT-DH) provide identification, authentication, and data protection **with much lower-resource requirements**



- ❖ **stronger security**
- ❖ **linear-in-time**
- ❖ **simpler: 80x faster than ECDSA, when measured by running both SW on AndesCore**

ACE: Speed up SecureRF WalnutDSA



```
insn multiply {
  op= {in gpr datA, in gpr datB};
  implied_op= {io MATRIX m};
  csim= %{
    // datA selects 3 out of 8 rows;
    // apply Galois Field mult-n-add on
    // datB and the 3 rows in parallel
  }%;
  . . .
};
```

```
reg MATRIX { //custom register
  num= 1;
  width= 320; //8 rows of 8 5-bit data
}
```

8 rows:

(1 row)
8 5-bit: **op datB**

WalnutDSA without ACE

Simulator Profiling				
Total Self Cycle Count: 163330				
Name	Calls	Self CycC	Time Percentage	
walnut_emul	66	152,436	93.33%	
walnut_verify_init	1	9,002	5.51%	
main + 52	1	799	0.49%	
walnut_verify_final	1	597	0.37%	
memset	4	377	0.23%	
walnut_verify_update	2	104	0.06%	
memset + 70	1	15	0.01%	

WalnutDSA with ACE

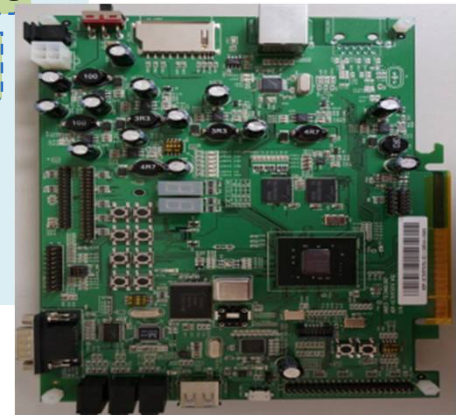
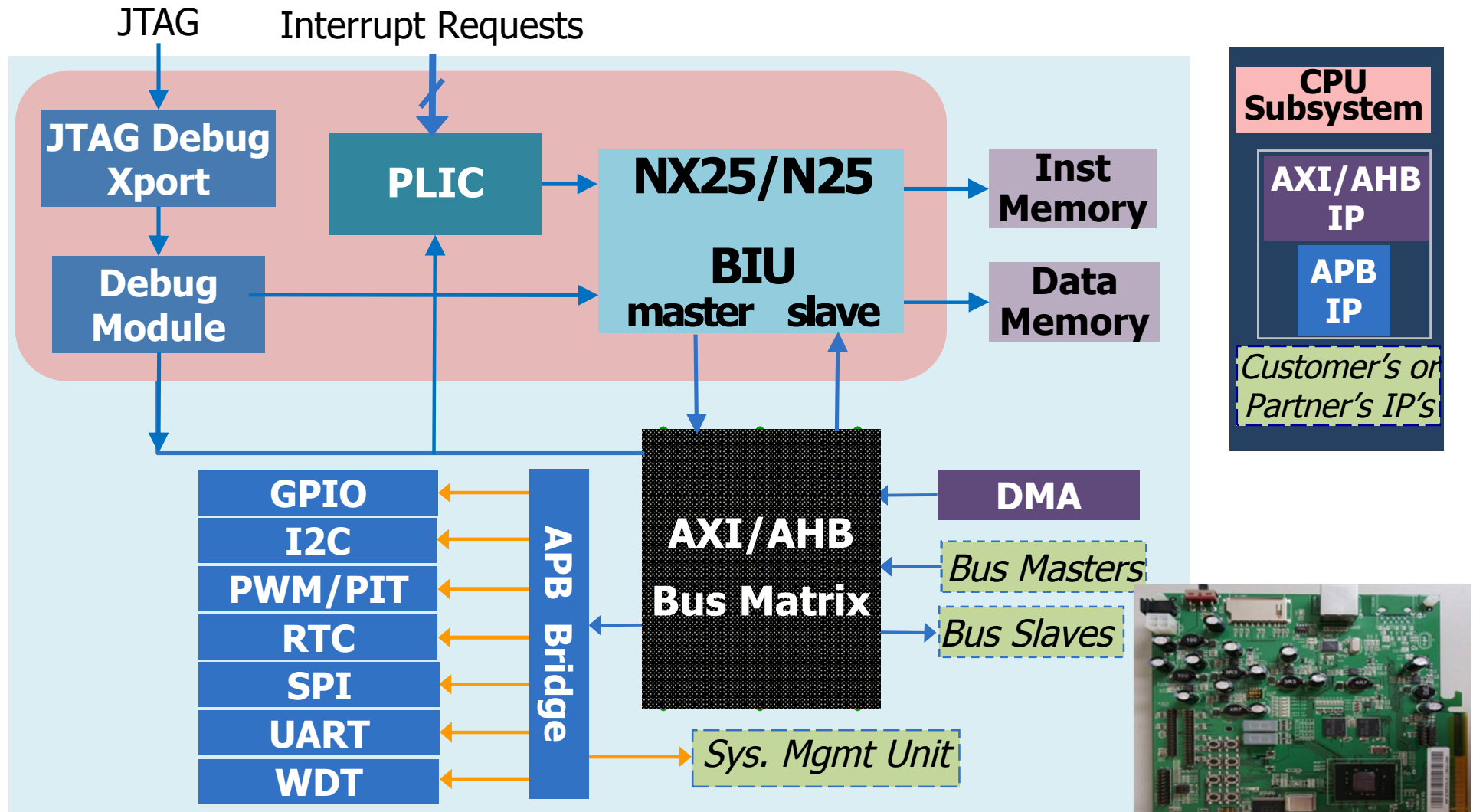
Simulator Profiling				
Total Self Cycle Count: 51420				
Name	Calls	Self CycC	Time Percentage	
walnut_verify_init	1	26,157	50.87%	
walnut_verify_update	2	23,060	44.85%	
main + 56	1	1,248	2.43%	
walnut_verify_final	1	601	1.17%	
memset	4	339	0.66%	
memset + 70	1	15	0.03%	

Overall Speedup: 163330/51420= 3.2x

Agenda

- ❖ Andes Corporate Overview
- ❖ RISC-V and AndeStar™ V5 Architecture
- ❖ V5 Processors NX25 and N25
- ❖ Andes-Enhanced Features
- ❖ **Support for V5 Processors**
- ❖ Concluding Remarks

NX25/N25 Pre-integrated Platform



Driving Innovations™

AndeSight IDE: Comprehensive Features

- ❖ Eclipse-based
- ❖ Project Setup:
 - Linker Script Editor
 - Flash ISP
- ❖ Debug Support:
 - RTOS-Awareness
 - Virtual Hosting
 - Register Bitfield View
 - Break on Exceptions
- ❖ Program Analysis
 - Function Profiling
 - Performance Meter
 - Code Coverage
 - Function Code Size
 - (Static) Stack Size
- ❖ Arduino Support
- ❖ Custom Plugin Intf
- ❖ Rich sample codes

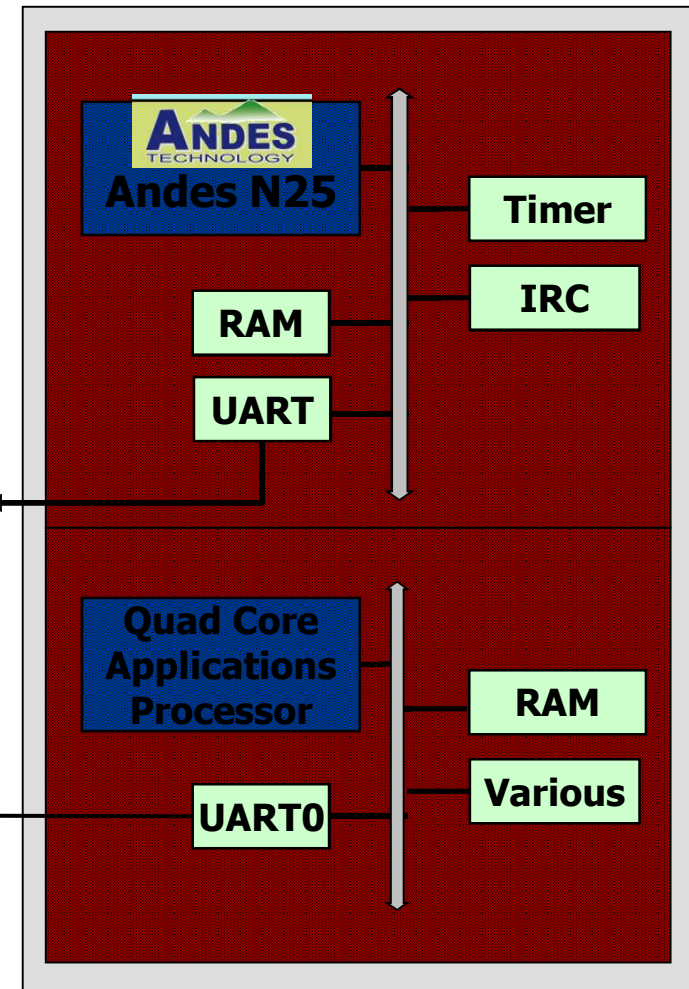
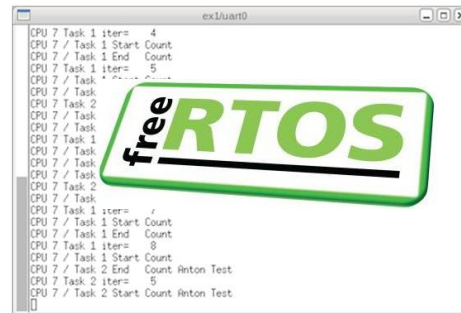
The collage displays several key features of the AndeSight IDE:

- SoC Registers:** A window showing a list of registers (cr2, cr3, VLPT, IVTB, NTPT, DE, HPTVK, TBLCK, EPSZ) and their values. A green arrow points to the 'cr3 (MMU_CFG)' register.
- Linker Script Editor:** A window showing the linker script configuration, including sections like USER_SECTIONS, LOAD_ROM, EXEC_ROM, and input sections like +ISR.
- gprof Profiler:** A window showing the execution profile of a program. It lists functions like jidctint.c, jpeg_idct_islow, jdhuff.c, decode_mcu, jpeg_fill_bit_buffer, jpeg_huff_decode, jpeg_make_d_derived_tbl, jidcolor.c, ycc_rgb_convert, and build_ycc_rgb_table, along with their sample counts and percentage of time spent.
- FreeRTOS Task List:** A window showing the status of tasks in a FreeRTOS system, including task name, number, priority, start of stack, top of stack, and status.
- FreeRTOS Event List:** A window showing the event list for a FreeRTOS system, including queue name, handler address, max length, item size, messages waiting, waiting Tx, and waiting Rx.
- Hardware Schematic:** A window showing a detailed hardware schematic diagram of a system, including components like the ADC, DAC, and various peripheral blocks.
- Code Editor:** A window showing the source code of a program, with a function code size analysis table at the bottom. The table lists functions like lddc_off, lddc_set framebuffer, main, make_runny_pointers, make_odither_array, master_selection, median_cut, merged_lv_upsample, and merged_2v_upsample, along with their size, file, and line numbers.

Virtual Platform with Imperas

❖ Imperas Extendable Platform Kits (EPKs)

- Virtual platforms for early SW development
- Rich tool support
- Heterogeneous multicore ready
- Very fast



Virtual Platform with Imperas

The screenshot displays the Imperas IDE interface during a virtual platform launch. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Imperas, Window, and Help. The main workspace is divided into several panes:

- Debug Console:** Shows the platform launch process. The selected item is "ID #1 [N25_CA15/N25/cpu0] N25 andes_riscv (Suspended : User Request)". The current instruction is "handle_reset() at entry.S:44 0x80000024".
- Registers:** A table showing the state of various registers.
- Source Code:** Displays the assembly code for the "handle_reset" function in "entry.S".
- Disassembly:** Shows the disassembled instructions corresponding to the source code.
- Console:** Displays the output of the platform launch, showing the execution of instructions 42, 43, and 44.

Name	Value	Description
General Registers		
zero	0	General Purpose and FPU
ra	0	
sp	0	
gp	0	
tp	0	
t0	2147483708	
t1	0	
t2	0	
fp	0	
s1	0	

```
entry.S:44 0x80000024
handle_reset:
    la t0, trap_entry
    csrw mtvec, t0
    csrw mstatus, 0
    csrw mideleg, 0
    csrw medeleg, 0
    csrw mie, 0
    # initialize global pointer
    la gp, _gp
```

```
41      csrw mstatus, 0
80000018: csrw mstatus, 0
42      csrw mideleg, 0
8000001c: csrw mideleg, 0
43      csrw mideleg, 0
80000020: csrw mideleg, 0
44      csrw mie, 0
80000024: csrw mie, 0
47      la gp, _gp
80000028: auipc gp, 0xf
8000002c: addi gp, gp, 1512 # 0x8000f610 < malloc av +792>
```

Platform Launch [Imperas - Connect to running simulator] mpd (99999999)

```
42      csrw mideleg, 0
43      csrw mideleg, 0
44      csrw mie, 0
```

Agenda

- ❖ Andes Corporate Overview
- ❖ RISC-V and AndeStar™ V5 Architecture
- ❖ V5 Processors NX25 and N25
- ❖ Andes-Enhanced Features
- ❖ Support for V5 Processors
- ❖ **Concluding Remarks**

Concluding Remarks

❖ **RISC-V has a great start**

- Open, compact, modular, extensible, and more

❖ **Advance beyond free (ISA) into risk-free (SoC)**

❖ **Serious SoC design teams demand**

- Faster time-to-market
- Quality, stability and support
- Lower total cost in product life cycle

➔ **Need experienced IP vendor**

❖ **Andes addresses it with AndeStar V5**

- Start with fast-n-small N25 and NX25 processors
- Bring rich extended features
- With good supporting infrastructure
- Ready to take RISC-V to mainstream SoCs

Thank You !!



AndesCore™



Driving Innovations™