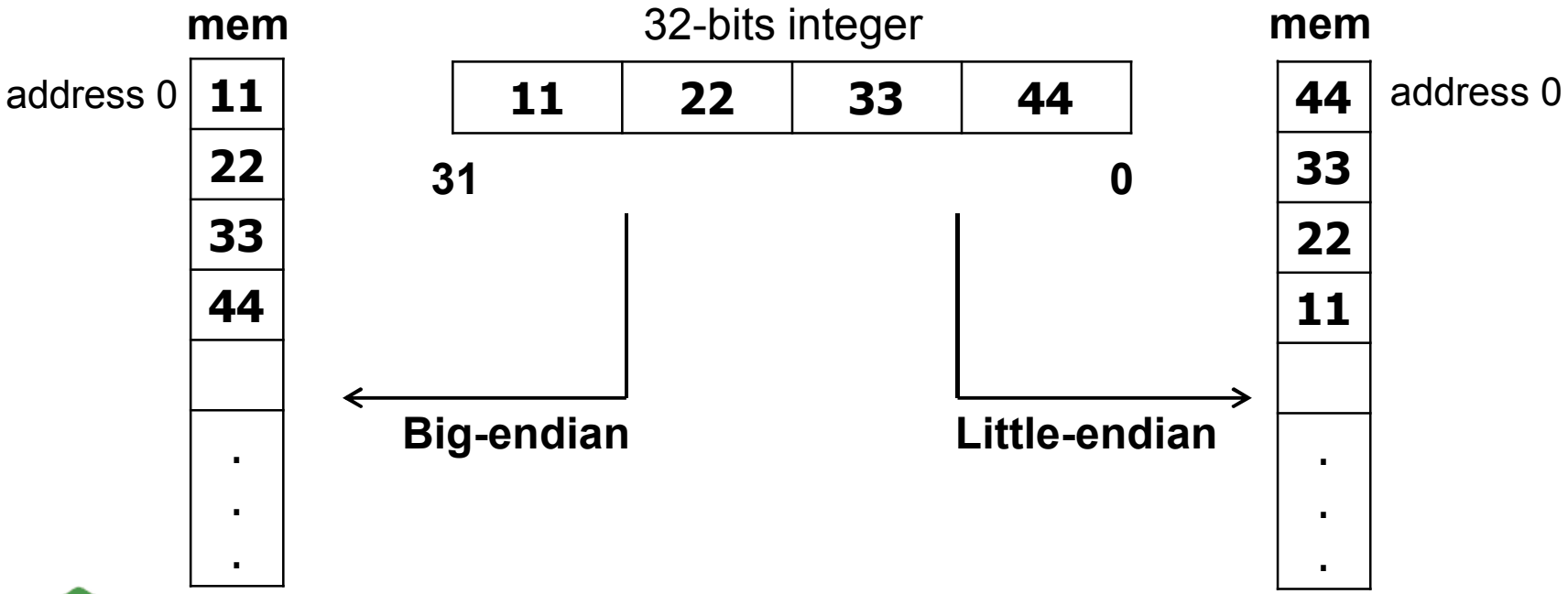
Andes Endian Support

Driving Innovations™

Definition of Endianness

- Big-endian
  - Store the most significant byte of a word in the smallest address
- Little-endian
  - Store the least significant byte in the smallest address



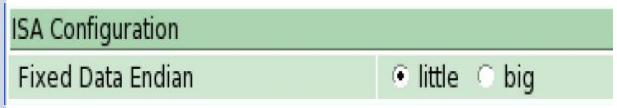
Endian Support of AndesCore™

- Instruction side
  - Big-endian (fixed)
  - Instruction transfer (EILM bus/ front-side bus) are always big-endian

Endian Support of AndesCore™

- Data side
  - Big-endian/Little-endian (configurable)
  - Data transfer (EDLM bus/ front-side bus)can be configured

	N7	N8	N9	N10	N13
Config. Option	√	√			
Input Signal			√	√	√
cr3.DE	RO	RO	RO	RO	RO



default_endian
1:big, 0:little

Endian Support of AndesCore™

- Device side
  - Big-endian/Little-endian (configurable)
  - Setting by input signal
    - default_device_endian (N9/N10, 1: big, 0:little)
    - default_drde (N13, 1: big, 0:little)
    - device_endian_enable

	N7	N8	N9	N10	N13
Device endian support			√	√	√
cr3.DRDE	Reserved	Reserved	RO	RO	RO
mr0.DREE	Reserved	Reserved	RW	RW	RW

Endian Support of AndesCore™

- Device side
  - The behavior of device endian can be summarized as

while (C==0 | NTC==0)

If (device_endian_enable == 1)

the endian behavior will determine by ***PSW.DRBE*** *bit (the default value is copied from cr3.DRDE bit)*

else if (device_endian_enable == 0)

the endian behavior will determine by ***PSW.BE*** *bit*

Endian Support of Andes Toolchain

- Tool chain
  - "be"= big endian; "le"= little endian
  - "elf"=newlib/mculib
  - Implementation version: v2, v2j, v3, v3m
    - Little Endian: nds32le-elf-mculib-v3m
    - Big Endian: nds32be-elf-mculib-v3m

Hardware Behavior of AndesCore™

- All registers inside AndesCore are little-endian
- In *lsu* module
  - Swap bytes based on the configuration *NDS_BIG_ENDIAN (n7, n8)* or input signal *default_endian (n9, n10, n13)*

Example Code of Endian Behavior

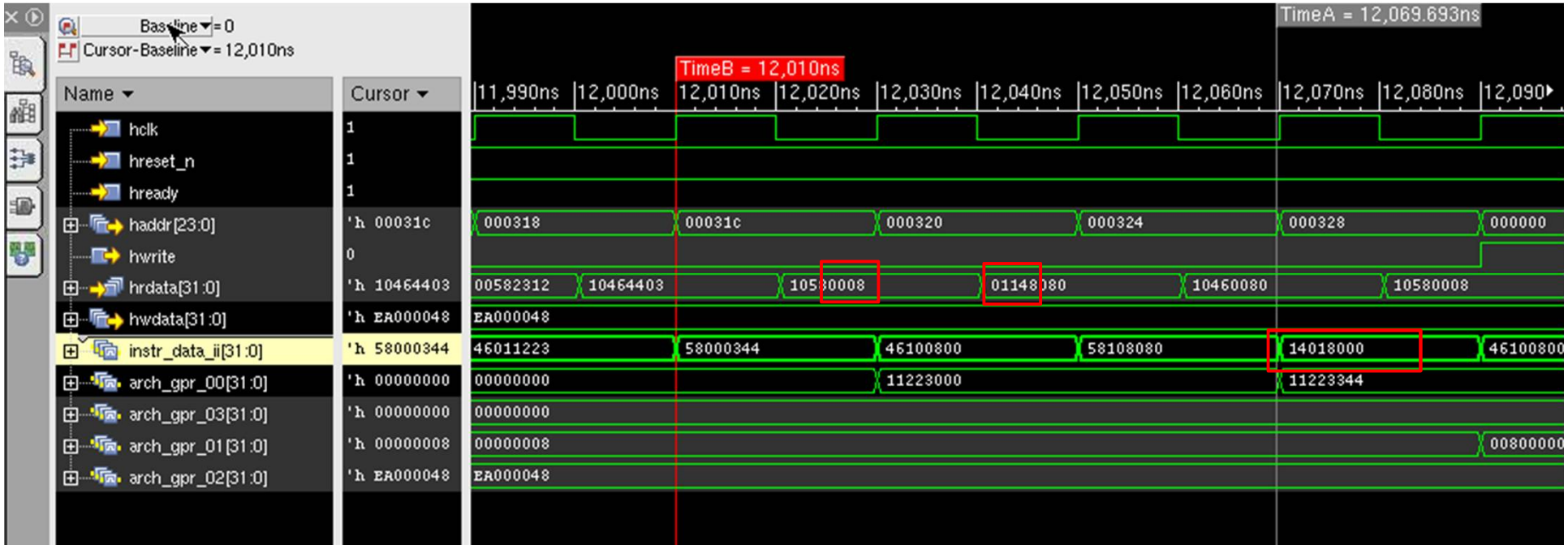
- Simple test code
  - Store/load 0x11223344 to address [$r3+#0x0]

```
30c:   ba 83          swi37 $r2,[$fp+#0xc]
30e:   46 01 12 23    sethi $r0,#0x11223
312:   58 00 03 44    ori $r0,$r0,#0x344
316:   46 10 08 00    sethi $r1,#0x800
31a:   58 10 80 80    ori $r1,$r1,#0x80
31e:   14 01 80 00    swi $r0,[$r3+#0x0]
322:   46 10 08 00    sethi $r1,#0x800
326:   58 10 80 80    ori $r1,$r1,#0x80
32a:   04 21 80 00    lwi $r2,[$r3+#0x0]
```

Waveform View of Example Code

- n801 w/ little-endian configuration

- Instruction fetch
  - hrdata big-endian, instr_data_ii little-endian



Waveform View of Example Code

- n801 w/ little-endian configuration

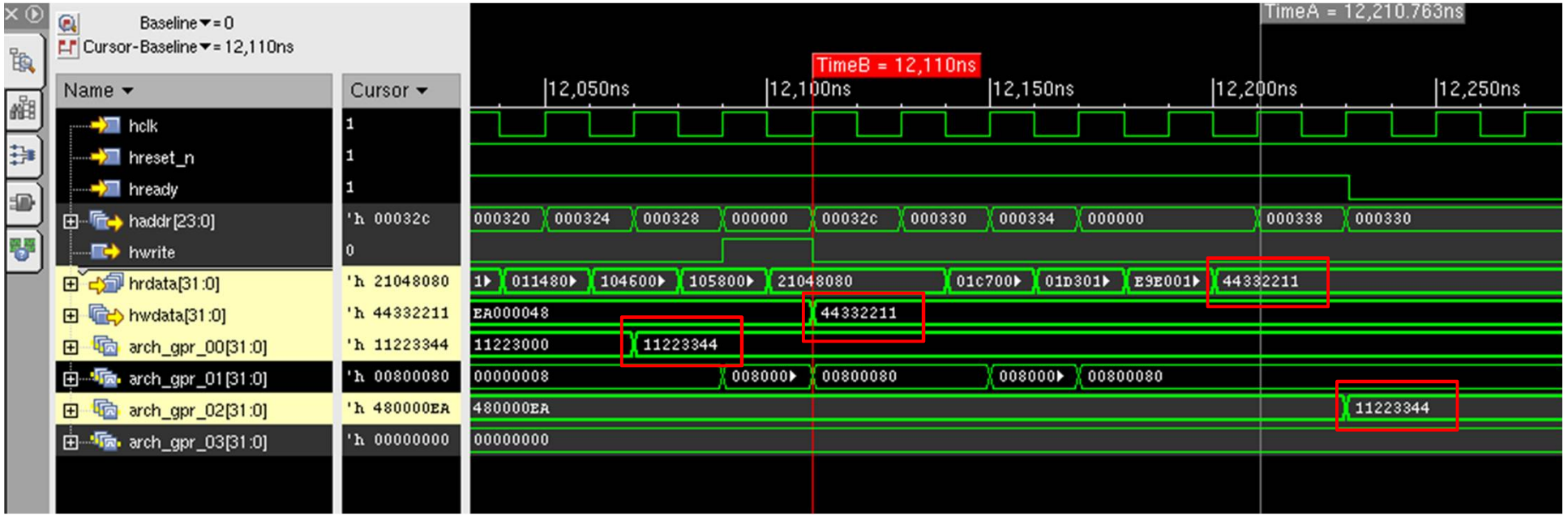
- Data load/store
  - hrdata/hwdata little-endian, gpr little-endian

Waveform View of Example Code

- n801 w/ big-endian configuration
- Instruction fetch
  - hrdata big-endian, instr_data_ii little-endian

Waveform View of Example Code

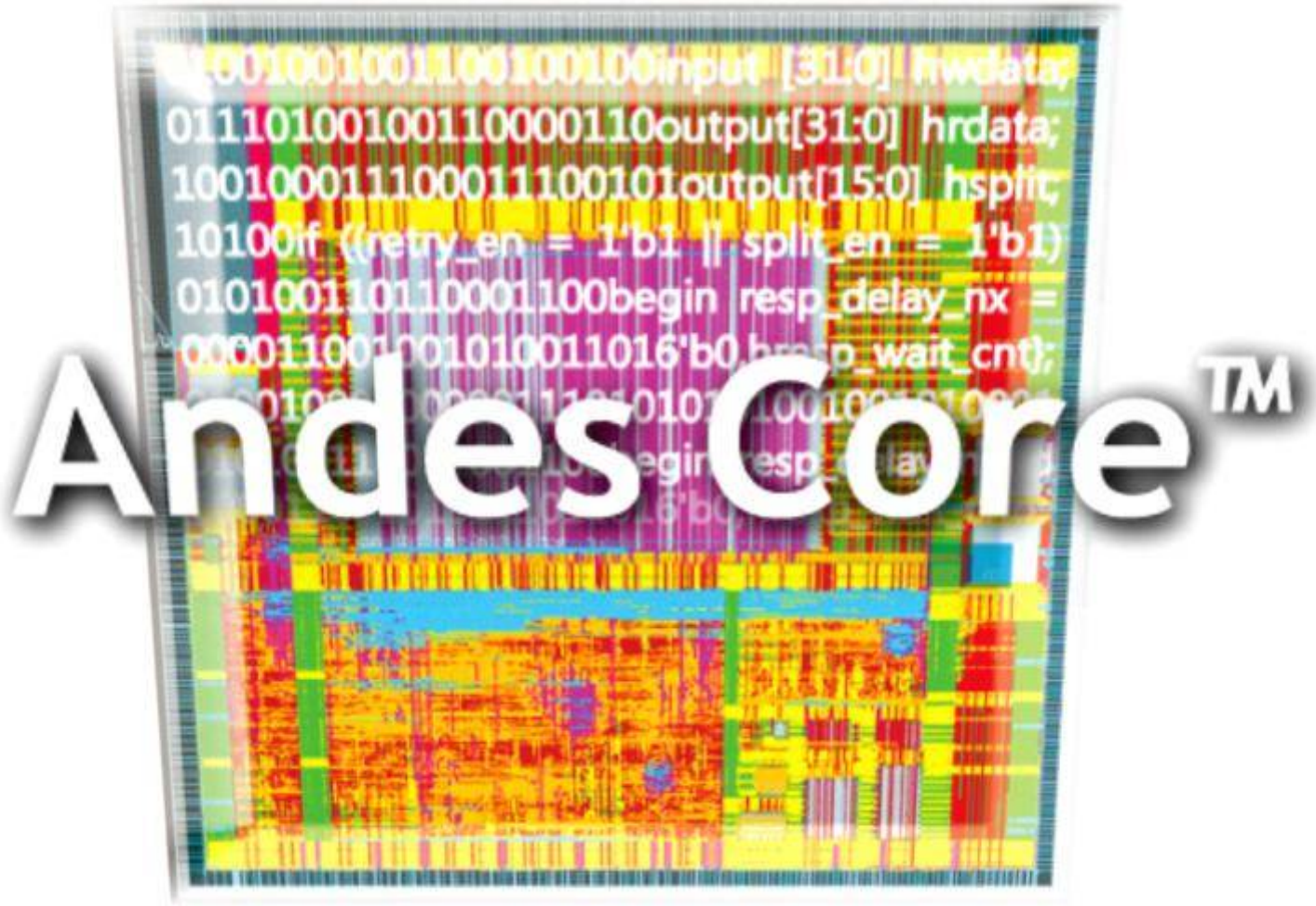
- n801 w/ big-endian configuration
- Data load/store
  - hrdata/hwdata big-endian, gpr little-endian



Summary

- Instruction bus is always big-endian
- Data bus is based on user's device & memory model, user needs to select a suitable configuration
- Choose a suitable toolchain when develop your software code
- Tool will cover endian translation automatically

Thank You



www.andestech.com