



## Getting the most out of your professional RISC-V compiler and debugger

Ryan Sheng / 盛磊, [ryan.sheng@iar.com](mailto:ryan.sheng@iar.com), 021-63758658  
IAR Systems (China)  
2019.11.13

# Highlight

- Meet the demand of quality & time-to-market for your RISC-V project
  - Easy code reuse and widest customers base from IAR Embedded Workbench, the complete IDE toolchain
  - Fit the needs of both memory size and necessary performance by the outstanding C/C++ compiler
  - Improve the code quality and find potential issues earlier by the integrated C-STAT analysis
  - Identify low level bugs and provide graphical visibility to all SoC resource by the powerful debugger

# IAR Embedded Workbench

## Complete C/C++ compiler and debugger toolchain



Most widely used development tools for embedded applications

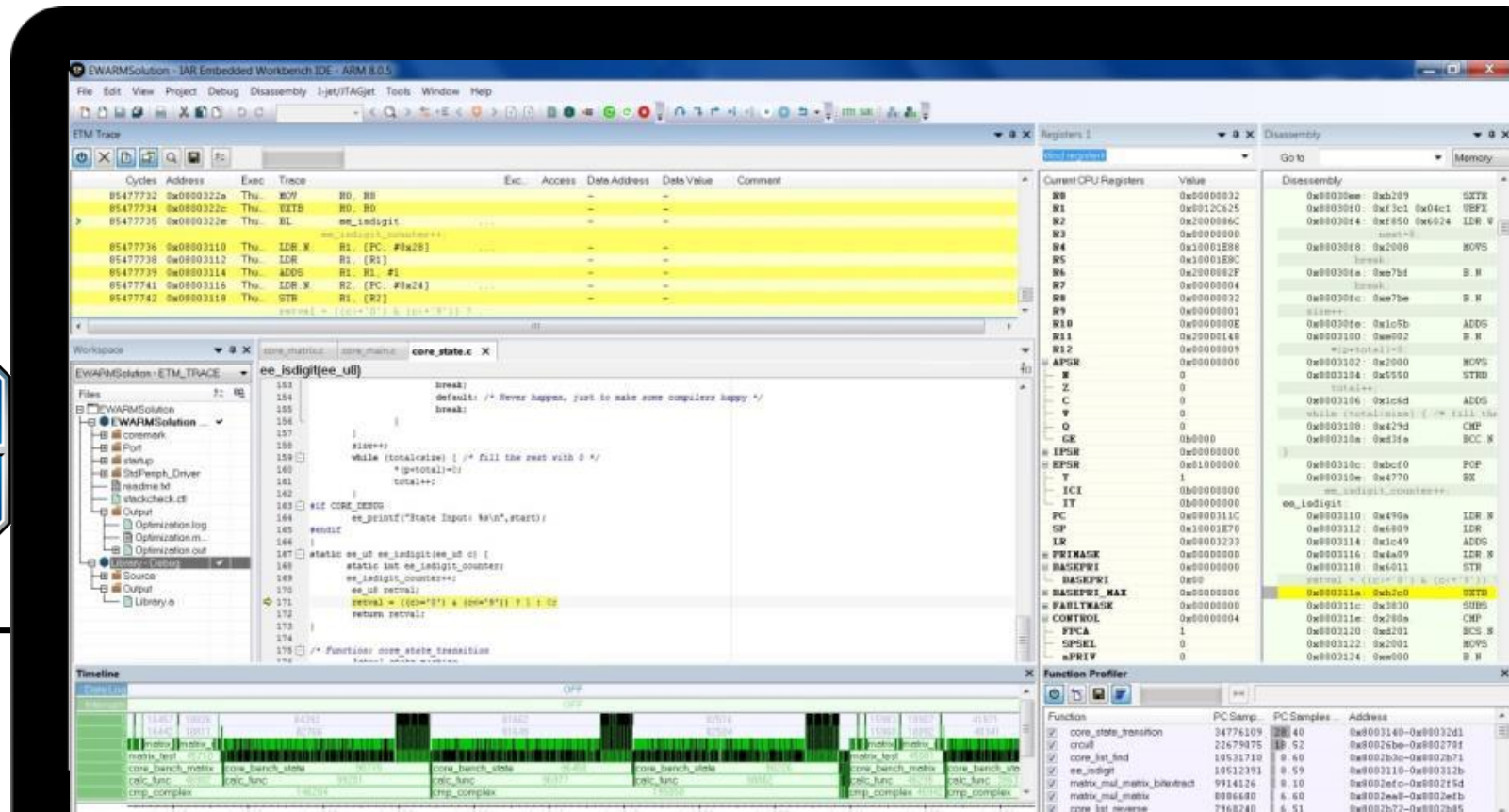
User-friendly IDE features and broad ecosystem integration

Industry leading code optimization technology

Comprehensive debugger

Integrated code analysis tools

ISO/ANSI C/C++ compliance with C18 and C++17





# Support for 12,000+ devices

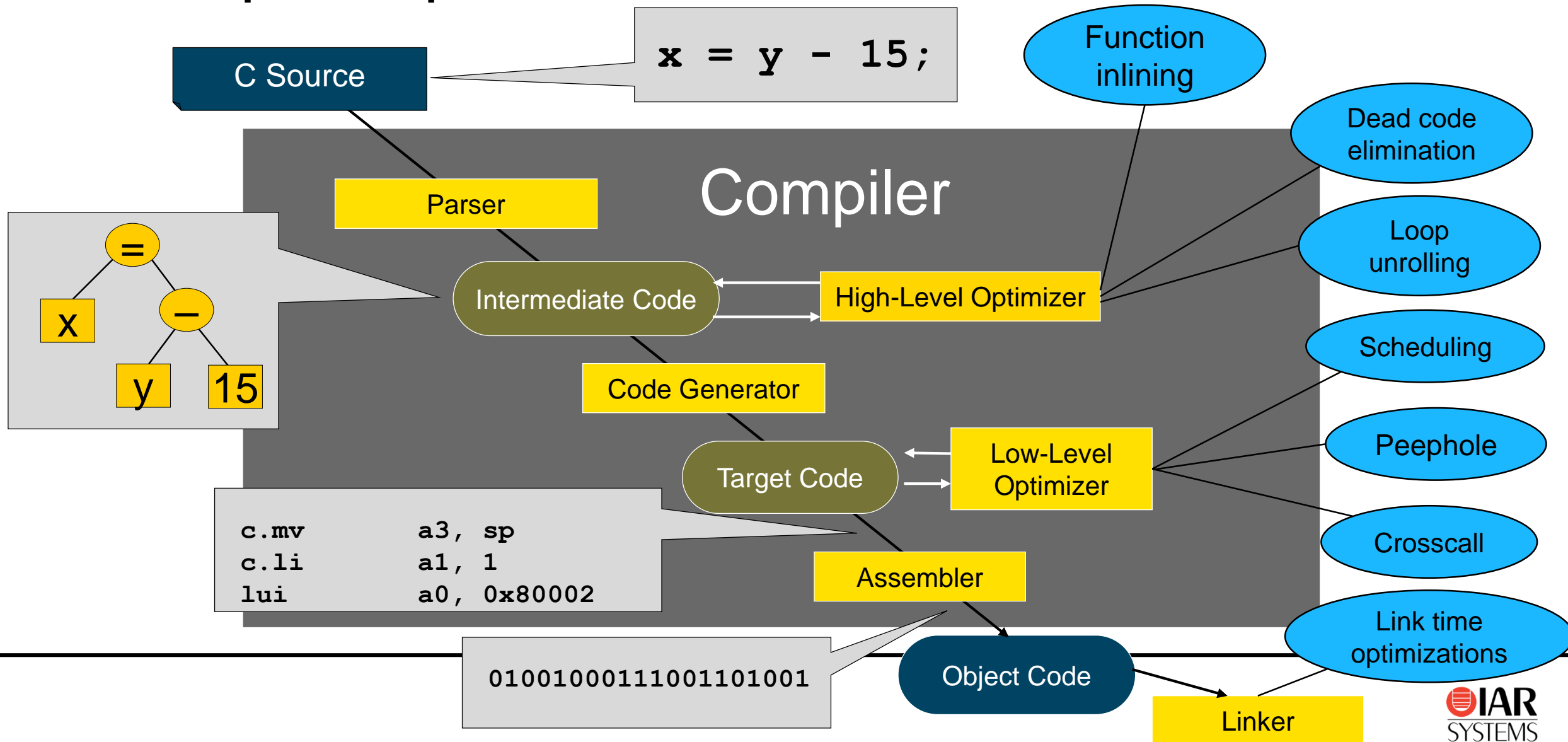
## Different architecture, One solution

All available 8-,16- and 32-bit MCUs



Cortex-M0	Cortex-R8	AVR	H8
Cortex-M0+	Cortex-A5	AVR32	STM8
Cortex-M1	Cortex-A7	RX	ColdFire
Cortex-M3	Cortex-A8	RL78	HCS12
Cortex-M4	Cortex-A9	RH850	S08
Cortex-M7	Cortex-A15	78K	MAXQ
Cortex-M23	ARM11	SuperH	CR16C
Cortex-M33	ARM9	V850	SAM8
Cortex-R4	ARM7	R32C	<b>RISC-V</b>
Cortex-R5	SecurCore	M32C	
Cortex-R52	8051	M16C	
Cortex-R7	MSP430	R8C	

# Compiler optimizations



# Controlling optimizations

Multiple optimization levels for code size and execution speed

The linker can remove unused code

Option to maximize speed with no size constraints

Multi-file compilation allows the optimizer to operate on a larger set of code

## Language standards

- ISO/IEC 14882:2015 (C++14, C++17)
- ISO/IEC 9899:2018 (C18)
- ANSI X3.159-1989 (C89)
- IEEE 754 standard for floating-point arithmetic

Major features of the optimizer can be controlled individually

Balance between size and speed by setting different optimizations for different parts of the code

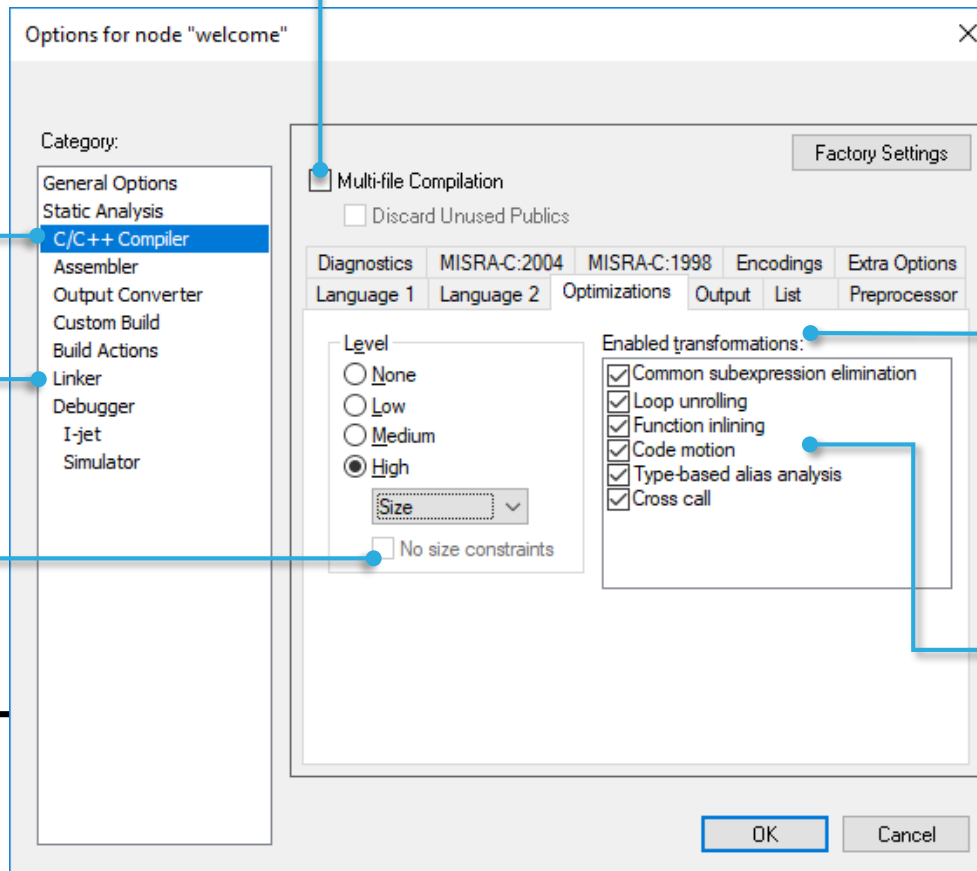
## Well-tested

Commercial test suites

- Plum-Hall Validation test suite
- Perennial EC++VS
- Dinkum C++ Proofer

In-house developed test suite  
>500,000 lines of C/C++ test code run multiple times

- Processor modes
- Memory models
- Optimization levels



# Speed, size or both?

## Optimization

## Effect

Common sub-expressions	Speed ↑	Size ↓
Loop unrolling	Speed ↑	Size ↑
Function inlining	Speed ↑	Size ↑
Code motion	Speed ↑	Size →
Dead code elimination	Speed →	Size ↓
Static clustering	Speed ↑	Size ↓
Instruction scheduling	Speed ↑	Size →
Cross call	Speed ↓	Size ↓

# Challenges on optimization

- **Size**

- Compared to more complex instruction sets, RISC-V have some challenges especially when it comes to code size
- Arithmetic with higher resolution than the natural data size yields larger code
- Absence of carry flags and instructions to save and restore multiple registers are other examples

- **Speed**

- When it comes to speed, RISC-V is relatively competitive
- More speed optimizations come in future releases

Our initial target will be on reduced code size for small embedded systems. Our main focus have always been to supply the best balance of code size and speed on the market.



# GCC attributes

- In extended language mode, the IAR C/C++ compiler supports a selection of commonly used GCC-style attributes
- Use the `__attribute__((attribute-list))` syntax for these attributes
- The following attributes are supported in part or in whole

alias	aligned	always_inline	constructor
deprecated	noinline	noreturn	packed
pcs	section	target	transparent_union
unused	used	volatile	weak

# Custom instructions

- The **.insn** directive generates custom instructions which are not directly supported by the assembler
- The **.insn** directive can be used to inline assembly code in programs written in C and C++
- The **.insn** directive generates instructions on all RISC-V instruction formats

<b>.insn directives</b>		
<b>.insn</b> r	op7, f3, f7, rd, rs1, rs2	
<b>.insn</b> r	op7, f3, f7, rd, rs1, rs2, rs3	
<b>.insn</b> r4	op7, f3, f2, rd, rs1, rs2, rs3	
<b>.insn</b> i	op7, f3, rd, rs1, expr	
<b>.insn</b> i	op7, f3, rd, rs1, expr (rs1)	
<b>.insn</b> s	op7, f3, rd, rs1, expr (rs1)	
<b>.insn</b> sb	op7, f3, rd, rs1, expr	
<b>.insn</b> sb	op7, f3, rd, expr(rs1)	
<b>.insn</b> b	op7, f3, rd, rs1, expr	
<b>.insn</b> u	op7, f3, rd, expr	
<b>.insn</b> uj	op2, rd, expr	
<b>.insn</b> cr	op2, f4, rd, rs1	
<b>.insn</b> ci	op2, f2, rd, expr	
<b>.insn</b> ciw	op2, f3, rd', expr	
<b>.insn</b> ca	op2, f6, f2, rd', rs2'	
<b>.insn</b> cb	op2, f3, rs1', expr	
<b>.insn</b> cj	op2, f3, expr	
<b>.insn</b> cs	op2, f3, rs1', rs2', expr	

op2, op7  
unsigned immediate  
2 or 7-bit opcode

fN  
unsigned immediate for function code  
2-7 bits wide

rd, rsN  
register field  
integer (x0-x31) or FP (f0-f31)

Rd', rsN'  
compact instruction reg. field  
integer (x8-x15) or FP (f8-f15)

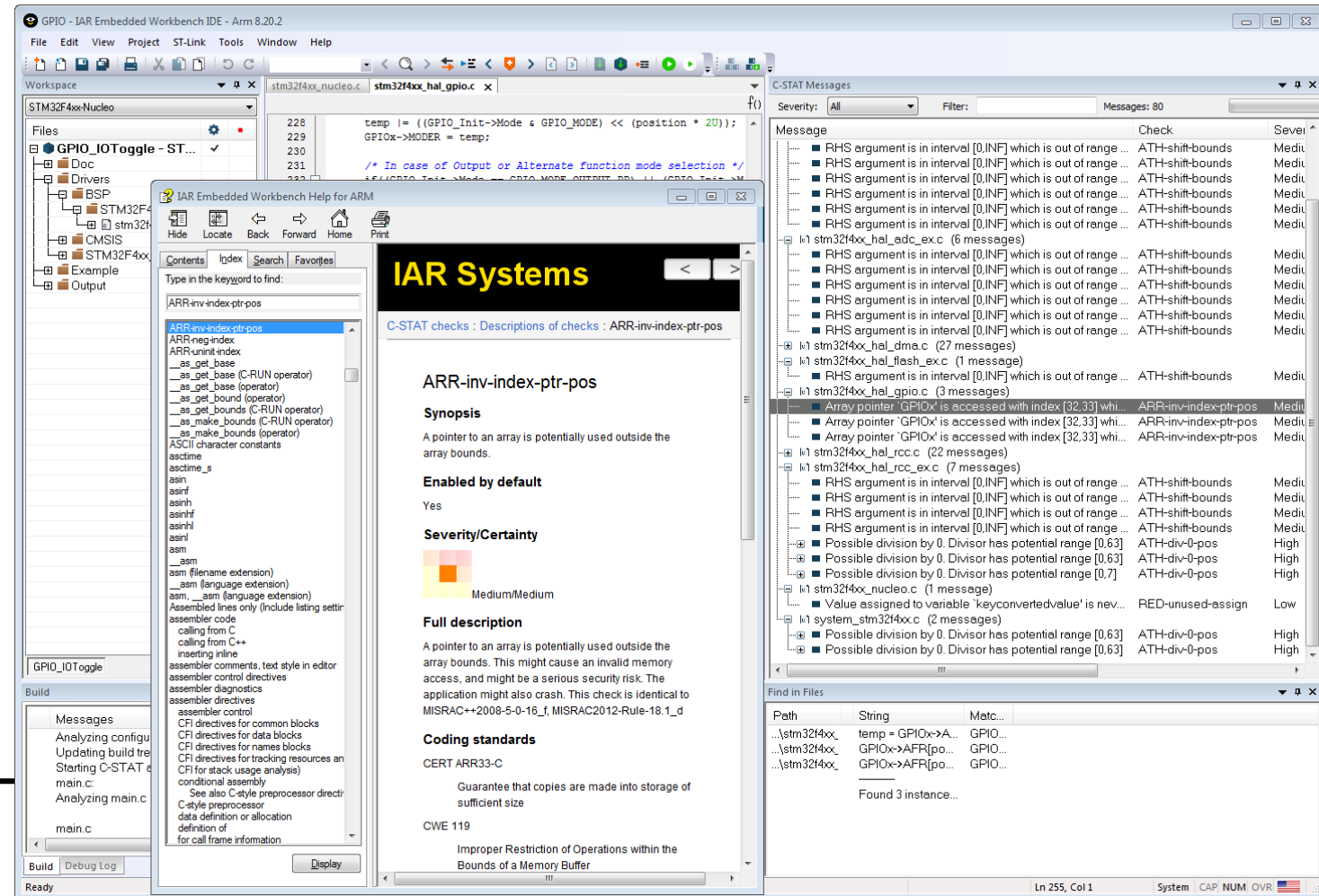
expr: immediate expression

\* Please refer to the RISC-V ISA specification sections 2.3 and 12.2 for details on bit-layout

# Code quality: C-STAT static analysis

- Advanced analysis of C/C++ code
- Fully integrated within IAR Embedded Workbench for RISC-V
- Check compliance with **MISRA C:2004**, **MISRA C++:2008** and **MISRA C:2012**
- Include ~250 checks mapping to hundreds of issues covered by **CWE** and **CERT C/C++**
- Intuitive and easy-to-use settings with flexible rule selection
- Support for command line execution
- Extensive and detailed documentation

CWE (Common Weakness Enumeration): <http://cwe.mitre.org>  
CERT (Computer Emergency Response Team): <http://www.cert.org>



# Debug & Trace probes

	I-jet	I-jet Trace (4-bit MIPI20 model)	I-jet Trace(16-bit Mictor/MIPI20 model)
JTAG/SWD speed	48 MHz	100 MHz	100 MHz
Download speed (RAM)	1.89 MByte/s	3.73MByte/s	3.73 MByte/s
SWO max. bandwidth	~30 Mbps	~60 Mbps	~60 Mbps
Available trace memory	-	64M or 256M bytes	256M or 1G bytes
Trace max. bandwidth	-	1.2 Gbps	11.2 Gbps
Max streaming speed	48MByte/s	~380MByte/s	~380MByte/s
Power sampling resolution	~160 $\mu$ A	~160 $\mu$ A	~160 $\mu$ A
Power sampling rate	200 ksps	200 ksps	200 ksps





# RISC-V debugging

- IAR supports the latest complete RISC-V debug spec, currently v0.13
  - Any additional updates will continuously be supported
- Automated discovery of implemented debug features in a MCU or SoC
  - Implemented debug features like available HW breakpoints, supported extensions etc. is read on connection
- Interrupt and exception catching
  - Configure interrupt and exception catching
  - Distinguish between different priority levels and exception types
- Set different types of breakpoints
  - Watchpoints, set breakpoints on data
  - Set conditional breakpoints
- Single step both on C/C++ and assembler level
- Full low-level access to all registers, memories and resources on RISC-V SoC
  - Low level powerful SoC-oriented debugger on roadmap
- Script/macro execution capabilities

# Debugger

Source and disassembly level debugging

- C-like macro system
- Built-in Simulator
- RTOS awareness
- Trace

Dockable windows and tab groups

Registers

Semihosted Terminal I/O

The screenshot displays the IAR Embedded Workbench IDE interface for RISC-V 1.11.1. The main window shows the source code of `coreplexip_welcome.c` with a breakpoint set at line 139. The disassembly window shows the corresponding assembly instructions. The registers window displays the state of the processor registers. The stack window shows the current stack frame. The watch window monitors expressions. The locals window shows the current function's local variables. The breakpoints window lists the set breakpoints. The terminal window shows the output of the program.

**Source Code (coreplexip\_welcome.c):**

```
119 PWM0_REG(PWM_CFG) = 0;
120 PWM0_REG(PWM_CFG) = (PWM_CFG_ENALWAYS) | (PWM_CFG_ZEROCM
121 PWM0_REG(PWM_COUNT) = 0;
122
123 // The LEDs are intentionally left somewhat dim.
124 PWM0_REG(PWM_CMP0) = 0;
125
126 while(1) {
127     volatile uint64_t * now = (volatile uint64_t*)(CLINT_CT
128     uint64_t then = *now + 100;
129     while (*now < then) { }
130
131     if(r > 0 && b == 0) {
132         r--;
133         g++;
134     }
135     if(g > 0 && r == 0) {
136         g--;
137         b++;
138     }
139     if(b > 0 && g == 0) {
140         b--;
141         r++;
142     }
143
144     PWM0_REG(PWM_CMP1) = 0xFF - (r >> 2);
145     PWM0_REG(PWM_CMP2) = 0xFF - (g >> 2);
146     PWM0_REG(PWM_CMP3) = 0xFF - (b >> 2);
147 }
```

**Disassembly:**

Address	Disassembly	Comment
4040032E	157D	c.addi a0, -1
40400330	0605	c.addi a2, 1
40400332	01061713	slli a4, a2, 0x10
40400336	8341	c.srli a4, 0x10
40400338	C719	c.beqz a4, 0x40400346
4040033A	01051713	slli a4, a0, 0x10
4040033E	8341	c.srli a4, 0x10
40400340	E319	c.bnez a4, 0x40400346
40400342	0585	c.addi a1, 1
40400344	167D	c.addi a2, -1
40400346	0FF00713	lii12 a4, 0xFF
4040034A	01059793	slli a5, a1, 0x10
4040034E	83C1	c.srli a5, 0x10
40400350	8789	c.srai a5, 2
40400352	8F1D	c.sub a4, a5
40400354	20005787	lui a5, 0x20005
40400358	02478793	addi a5, a5, 0x24
4040035C	C398	c.sw a4, 0(a5)
4040035E	0FF00713	lii12 a4, 0xFF
40400362	01051793	slli a5, a0, 0x10
40400366	83C1	c.srli a5, 0x10
40400368	8789	c.srai a5, 2
4040036A	8F1D	c.sub a4, a5
4040036C	20005787	lui a5, 0x20005

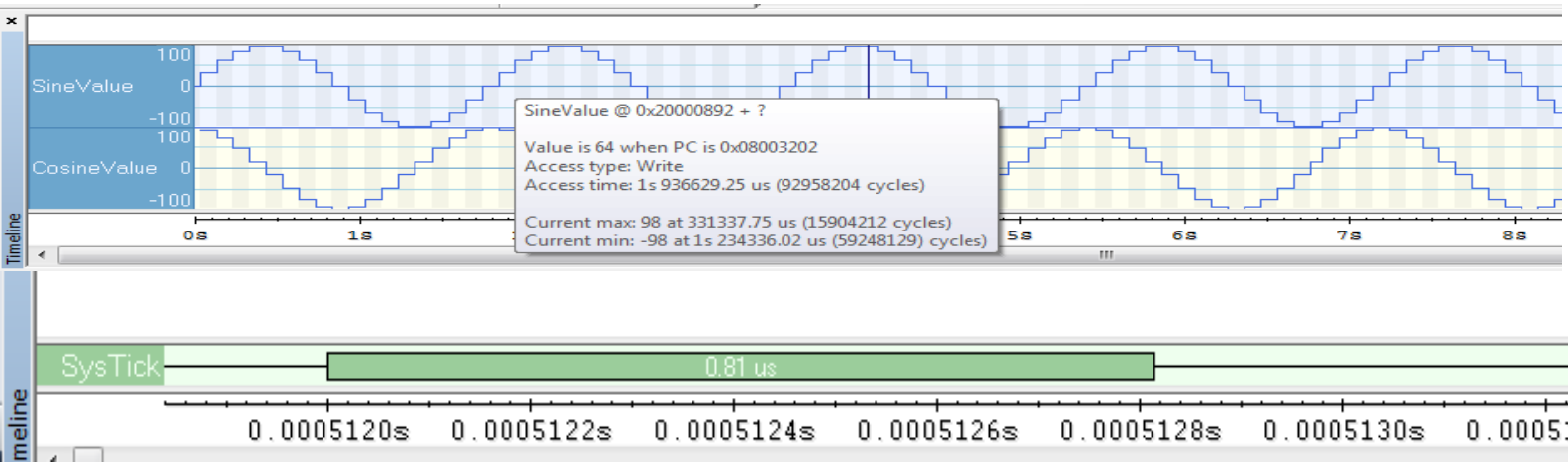
**Registers:**

Name	Value	Access
PWM0CFG	0x00001400	ReadWrite
PWM0COUNT	0x00000077	ReadWrite
PWM0PWS	0x00000000	ReadWrite
PWM0CMP0	0x000000FE	ReadWrite
PWM0CMP1	0x000000F3	ReadWrite
PWM0CMP2	0x000000FC	ReadWrite
PWM0CMP3	0x000000FF	ReadWrite

**Stack:**

Location	Data	Variable	Value	Type	Frame
0x000000FF	0x00002710	i	10000	int	[0] main
0x000000FE	0xCDCDCDCD				
0x000000FD	0xCDCDCDCD				
0x000000FC	0xCDCDCDCD				
0x000000FB	0xCDCDCDCD				
0x000000FA	0xCDCDCDCD				
0x000000F9	0xCDCDCDCD				
0x000000F8	0xCDCDCDCD				
0x000000F7	0xCDCDCDCD				
0x000000F6	0xCDCDCDCD				
0x000000F5	0xCDCDCDCD				
0x000000F4	0xCDCDCDCD				
0x000000F3	0xCDCDCDCD				
0x000000F2	0xCDCDCDCD				
0x000000F1	0xCDCDCDCD				
0x000000F0	0xCDCDCDCD				
0x000000EF	0xCDCDCDCD				
0x000000EE	0xCDCDCDCD				
0x000000ED	0xCDCDCDCD				
0x000000EC	0xCDCDCDCD				
0x000000EB	0xCDCDCDCD				
0x000000EA	0xCDCDCDCD				
0x000000E9	0xCDCDCDCD				
0x000000E8	0xCDCDCDCD				
0x000000E7	0xCDCDCDCD				
0x000000E6	0xCDCDCDCD				
0x000000E5	0xCDCDCDCD				
0x000000E4	0xCDCDCDCD				
0x000000E3	0xCDCDCDCD				
0x000000E2	0xCDCDCDCD				
0x000000E1	0xCDCDCDCD				
0x000000E0	0xCDCDCDCD				
0x000000DF	0xCDCDCDCD				
0x000000DE	0xCDCDCDCD				
0x000000DD	0xCDCDCDCD				
0x000000DC	0xCDCDCDCD				
0x000000DB	0xCDCDCDCD				
0x000000DA	0xCDCDCDCD				
0x000000D9	0xCDCDCDCD				
0x000000D8	0xCDCDCDCD				
0x000000D7	0xCDCDCDCD				
0x000000D6	0xCDCDCDCD				
0x000000D5	0xCDCDCDCD				
0x000000D4	0xCDCDCDCD				
0x000000D3	0xCDCDCDCD				
0x000000D2	0xCDCDCDCD				
0x000000D1	0xCDCDCDCD				
0x000000D0	0xCDCDCDCD				
0x000000CF	0xCDCDCDCD				
0x000000CE	0xCDCDCDCD				
0x000000CD	0xCDCDCDCD				
0x000000CC	0xCDCDCDCD				
0x000000CB	0xCDCDCDCD				
0x000000CA	0xCDCDCDCD				
0x000000C9	0xCDCDCDCD				
0x000000C8	0xCDCDCDCD				
0x000000C7	0xCDCDCDCD				
0x000000C6	0xCDCDCDCD				
0x000000C5	0xCDCDCDCD				
0x000000C4	0xCDCDCDCD				
0x000000C3	0xCDCDCDCD				
0x000000C2	0xCDCDCDCD				
0x000000C1	0xCDCDCDCD				
0x000000C0	0xCDCDCDCD				
0x000000BF	0xCDCDCDCD				
0x000000BE	0xCDCDCDCD				
0x000000BD	0xCDCDCDCD				
0x000000BC	0xCDCDCDCD				
0x000000BB	0xCDCDCDCD				
0x000000BA	0xCDCDCDCD				
0x000000B9	0xCDCDCDCD				
0x000000B8	0xCDCDCDCD				
0x000000B7	0xCDCDCDCD				
0x000000B6	0xCDCDCDCD				
0x000000B5	0xCDCDCDCD				
0x000000B4	0xCDCDCDCD				
0x000000B3	0xCDCDCDCD				
0x000000B2	0xCDCDCDCD				
0x000000B1	0xCDCDCDCD				
0x000000B0	0xCDCDCDCD				
0x000000AF	0xCDCDCDCD				
0x000000AE	0xCDCDCDCD				
0x000000AD	0xCDCDCDCD				
0x000000AC	0xCDCDCDCD				
0x000000AB	0xCDCDCDCD				
0x000000AA	0xCDCDCDCD				
0x000000A9	0xCDCDCDCD				
0x000000A8	0xCDCDCDCD				
0x000000A7	0xCDCDCDCD				
0x000000A6	0xCDCDCDCD				
0x000000A5	0xCDCDCDCD				
0x000000A4	0xCDCDCDCD				
0x000000A3	0xCDCDCDCD				
0x000000A2	0xCDCDCDCD				
0x000000A1	0xCDCDCDCD				
0x000000A0	0xCDCDCDCD				
0x0000009F	0xCDCDCDCD				
0x0000009E	0xCDCDCDCD				
0x0000009D	0xCDCDCDCD				
0x0000009C	0xCDCDCDCD				
0x0000009B	0xCDCDCDCD				
0x0000009A	0xCDCDCDCD				
0x00000099	0xCDCDCDCD				
0x00000098	0xCDCDCDCD				
0x00000097	0xCDCDCDCD				
0x00000096	0xCDCDCDCD				
0x00000095	0xCDCDCDCD				
0x00000094	0xCDCDCDCD				
0x00000093	0xCDCDCDCD				
0x00000092	0xCDCDCDCD				
0x00000091	0xCDCDCDCD				
0x00000090	0xCDCDCDCD				
0x0000008F	0xCDCDCDCD				
0x0000008E	0xCDCDCDCD				
0x0000008D	0xCDCDCDCD				
0x0000008C	0xCDCDCDCD				
0x0000008B	0xCDCDCDCD				
0x0000008A	0xCDCDCDCD				
0x00000089	0xCDCDCDCD				
0x00000088	0xCDCDCDCD				
0x00000087	0xCDCDCDCD				
0x00000086	0xCDCDCDCD				
0x00000085	0xCDCDCDCD				
0x00000084	0xCDCDCDCD				
0x00000083	0xCDCDCDCD				
0x00000082	0xCDCDCDCD				
0x00000081	0xCDCDCDCD				
0x00000080	0xCDCDCDCD				
0x0000007F	0xCDCDCDCD				
0x0000007E	0xCDCDCDCD				
0x0000007D	0xCDCDCDCD				
0x0000007C	0xCDCDCDCD				
0x0000007B	0xCDCDCDCD				
0x0000007A	0xCDCDCDCD				
0x00000079	0xCDCDCDCD				
0x00000078	0xCDCDCDCD				
0x00000077	0xCDCDCDCD				
0x00000076	0xCDCDCDCD				
0x00000075	0xCDCDCDCD				
0x00000074	0xCDCDCDCD				
0x00000073	0xCDCDCDCD				
0x00000072	0xCDCDCDCD				
0x00000071	0xCDCDCDCD				
0x00000070	0xCDCDCDCD				
0x0000006F	0xCDCDCDCD				
0x0000006E	0xCDCDCDCD				
0x0000006D	0xCDCDCDCD				
0x0000006C	0xCDCDCDCD				
0x0000006B	0xCDCDCDCD				
0x0000006A	0xCDCDCDCD				
0x00000069	0xCDCDCDCD				
0x00000068	0xCDCDCDCD				
0x00000067	0xCDCDCDCD				
0x00000066	0xCDCDCDCD				
0x00000065	0xCDCDCDCD				
0x00000064	0xCDCDCDCD				
0x00000063	0xCDCDCDCD				
0x00000062	0xCDCDCDCD				
0x00000061	0xCDCDCDCD				
0x00000060	0xCDCDCDCD				
0x0000005F	0xCDCDCDCD				
0x0000005E	0xCDCDCDCD				
0x0000005D	0xCDCDCDCD				
0x0000005C	0xCDCDCDCD				
0x0000005B	0xCDCDCDCD				
0x0000005A	0xCDCDCDCD				
0x00000059	0xCDCDCDCD				
0x00000058	0xCDCDCDCD				
0x00000057	0xCDCDCDCD				
0x00000056	0xCDCDCDCD				
0x00000055	0xCDCDCDCD				
0x00000054	0xCDCDCDCD				
0x00000053	0xCDCDCDCD				
0x00000052	0xCDCDCDCD				
0x00000051	0xCDCDCDCD				
0x00000050	0xCDCDCDCD				
0x0000004F	0xCDCDCDCD				
0x0000004E	0xCDCDCDCD				
0x0000004D	0xCDCDCDCD				
0x0000004C	0xCDCDCDCD				
0x0000004B	0xCDCDCDCD				
0x0000004A	0xCDCDCDCD				
0x00000049	0xCDCDCDCD				
0x00000048	0xCDCDCDCD				
0x00000047	0xCDCDCDCD				
0x00000046	0xCDCDCDCD				
0x00000045	0xCDCDCDCD				
0x00000044	0xCDCDCDCD				
0x00000043	0xCDCDCDCD				
0x00000042	0xCDCDCDCD				
0x00000041	0xCDCDCDCD				
0x00000040	0xCDCDCDCD				
0x0000003F	0xCDCDCDCD				
0x0000003E	0xCDCDCDCD				
0x0000003D	0xCDCDCDCD				
0x0000003C	0xCDCDCDCD				
0x0000003B	0xCDCDCDCD				
0x0000003A	0xCDCDCDCD				
0x00000039	0xCDCDCDCD				
0x00000038	0xCDCDCDCD				
0x00000037	0xCDCDCDCD				
0x00000036	0xCDCDCDCD				
0x00000035	0xCDCDCDCD				
0x00000034	0xCDCDCDCD				
0x00000033	0xCDCDCDCD				
0x00000032	0xCDCDCDCD				
0x00000031	0xCDCDCDCD				
0x00000030	0xCDCDCDCD				
0x0000002F	0xCDCDCDCD				
0x0000002E	0xCDCDCDCD				
0x0000002D	0xCDCDCDCD				
0x0000002C	0xCDCDCDCD				
0x0000002B	0xCDCDCDCD				
0x0000002A	0xCDCDCDCD				
0x00000029	0xCDCDCDCD				
0x00000028	0xCDCDCDCD				
0x00000027	0xCDCDCDCD				
0x00000026	0xCDCDCDCD				
0x00000025	0xCDCDCDCD				
0x00000024	0xCDCDCDCD				
0x00000023	0xCDCDCDCD				

# More features will come: Data log, Interrupt log, Function profiler, Code coverage, etc.

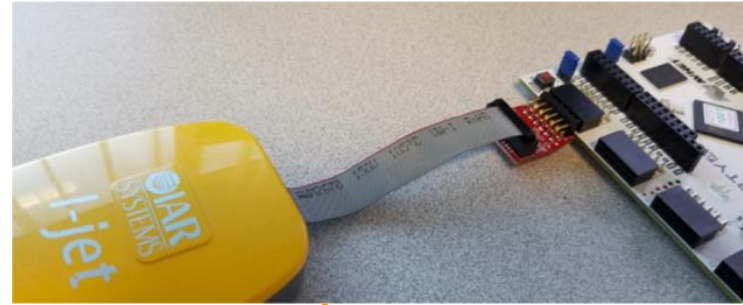
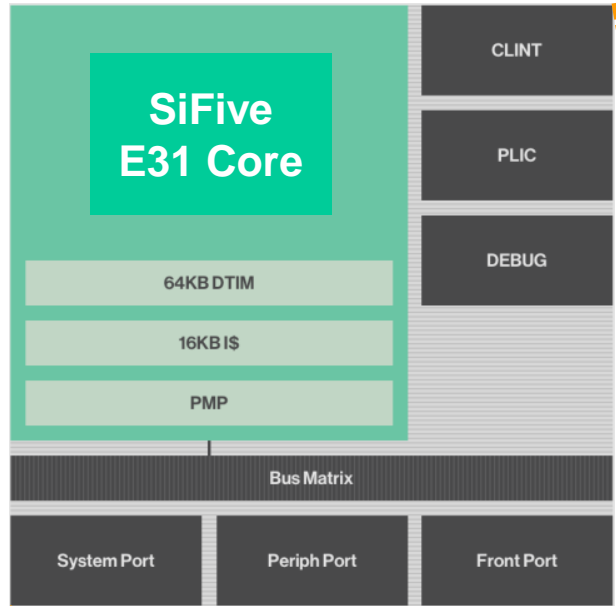


Function	PC Samples	PC Samples (%)
<input checked="" type="checkbox"/> core_state_transition	3401	30.33
<input checked="" type="checkbox"/> matrix_mul_matrix	1342	11.97
<input checked="" type="checkbox"/> matrix_mul_matrix_bitextract	1176	10.49
<input checked="" type="checkbox"/> crcu8	1140	10.17
<input checked="" type="checkbox"/> ee_isdigit	851	7.59
<input checked="" type="checkbox"/> core_list_find	656	5.85
<input checked="" type="checkbox"/> core_list_reverse	588	5.24
<input checked="" type="checkbox"/> matrix_sum	430	3.84
<input checked="" type="checkbox"/> core_bench_state	386	3.44
<input checked="" type="checkbox"/> core_init_state	186	1.66
<input checked="" type="checkbox"/> core_list_mergesort	162	1.44
<input checked="" type="checkbox"/> matrix_add_const	158	1.41
<input checked="" type="checkbox"/> matrix_mul_vect	112	1.00
<input checked="" type="checkbox"/> matrix_mul_const	105	0.94
<input checked="" type="checkbox"/> core_bench_list	76	0.68
<input checked="" type="checkbox"/> crcu16	72	0.64
<input checked="" type="checkbox"/> cmp_idx	48	0.43

Code	Coverage (%)	Code Range	File
Fibonacci(Program)	86.7		
Fibonacci(Module)	100.0		Fibonacci.c
Utilities(Module)	86.4		Utilities.c
GetFib	50.0		
InitFib	100.0		
PutFib	91.7		
__dbg_break(Module)	100.0		__dbg_break.c
__dbg_xxdwrite(M...	100.0		__dbg_xxdwrite.c
__dbg_xxexit(Mod...	33.3		__dbg_xxexit.c
__dbg_xxwrite(Mo...	66.7		__dbg_xxwrite.c

C	Co...	Disassembly
		__iar_cstart_init_gp:
		__iar_program_start:
◆	1	20000000 800001B7 lui gp, 0x80000
◆	1	20000004 01618193 addi gp, gp, 0x16
◆	1	20000008 80001137 lui sp, 0x80001
◆	1	2000000C 03010113 addi sp, sp, 0x30
◆	1	20000010 2ACD c.jal __low_level_init
◆	1	20000012 00050363 beq a0, zero, 0x20000018
◆	1	20000016 22BD c.jal __iar_data_init2
◆	1	20000018 00000513 mv a0, zero
◆	1	2000001C 2225 c.jal main
◆	1	2000001E 22CD c.jal exit
◆	0	20000020 0000006F j 0x20000020
		void PutFib(uint32_t out)
		{

# Demonstration

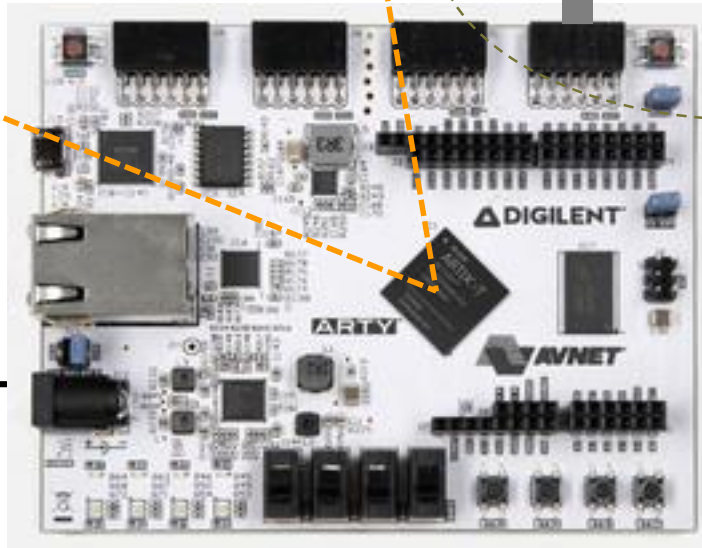


JTAG

I-jet

USB

Xilinx Artix-7  
Arty FPGA Kit



编译/链接/下载/调试

IAR Embedded  
Workbench



IAR  
SYSTEMS





Thanks for your attention!

[www.iar.com/riscv](http://www.iar.com/riscv)