芯动力——硬件加速设计方法

第七章 基于平头哥E902处理器的SoC设计

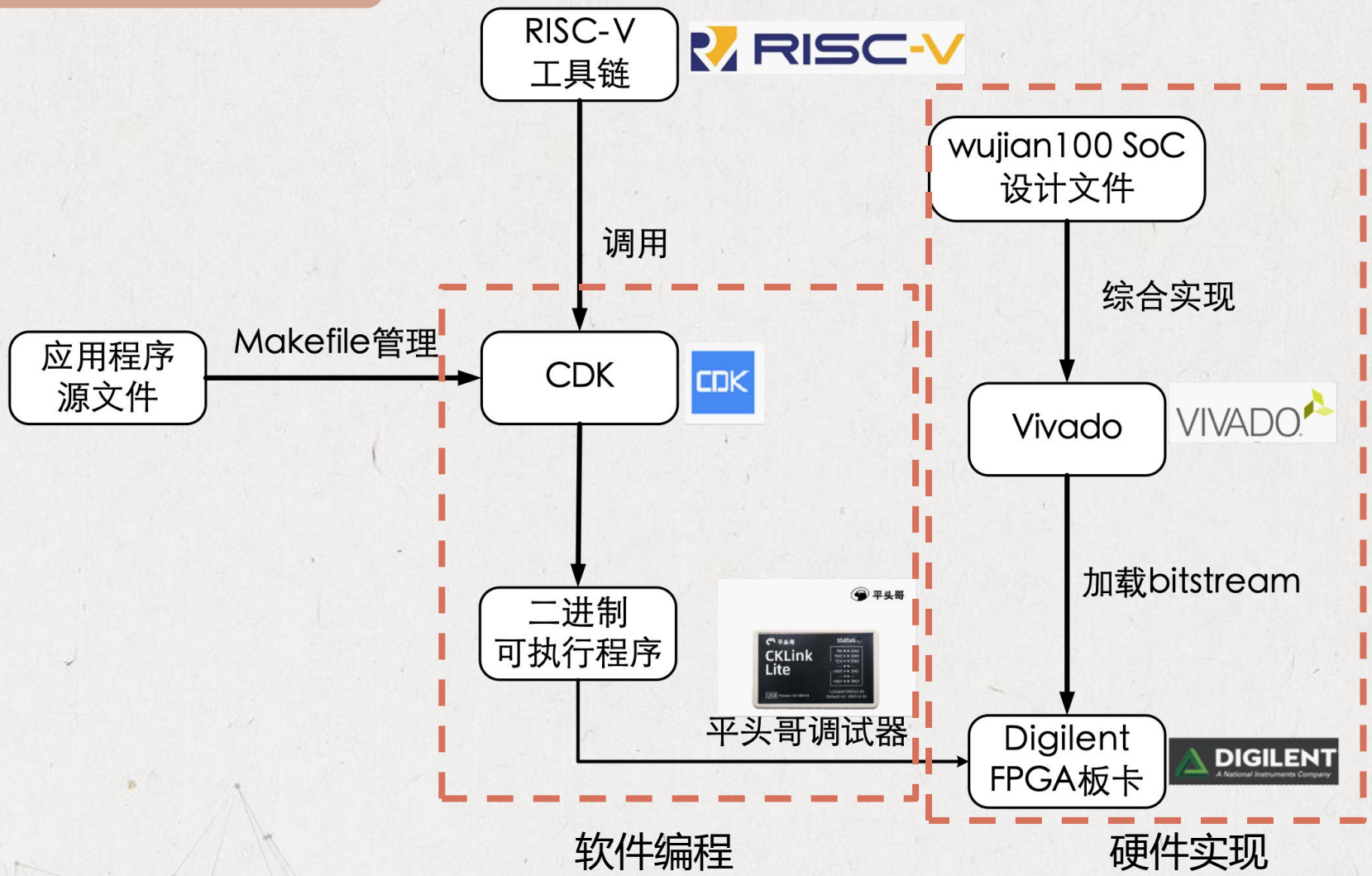
——(4)无剑100 SoC 开发工具介绍

邸志雄@西南交通大学

zxdi@home.swjtu.edu.cn

slides与源代码网址 http://www.dizhixiong.cn/class5/

软硬件环境介绍


RISC-V
工具链
RISC-V
wujian100 SoC
设计文件
调用
综合实现
应用程序
源文件
Makefile管理
CDK
CDK
Vivado
VIVADO
加载bitstream
二进制
可执行程序
CKLink
Lite
平头哥调试器
Digilent
FPGA板卡
DIGILENT
A National Instruments Company
软件编程
硬件实现

平头哥工具链

工具链（toolchain） 辅助完成程序开发、调试等一系列工具的集合

编译器 汇编器 链接器 调试器 模拟器 ...

由于指令集和用户编程模型不同，因此针对每个处理器架构都需要一套完整的、独有的工具链。

平头哥工具链

平头哥官方提供的工具链是以GNU开源工具链为基础，并根据处理器微架构进行了高度优化。

支持CSKY架构的玄铁800系列

支持RISC-V架构的玄铁900系列

根据执行平台和目标程序运行平台的不同又分为不同版本。

Xuantie-900-gcc-elf-newlib-i386-V2.0.3-20210806.tar.gz

Xuantie-900-gcc-elf-newlib-mingw-V2.0.3-20210806.tar.gz

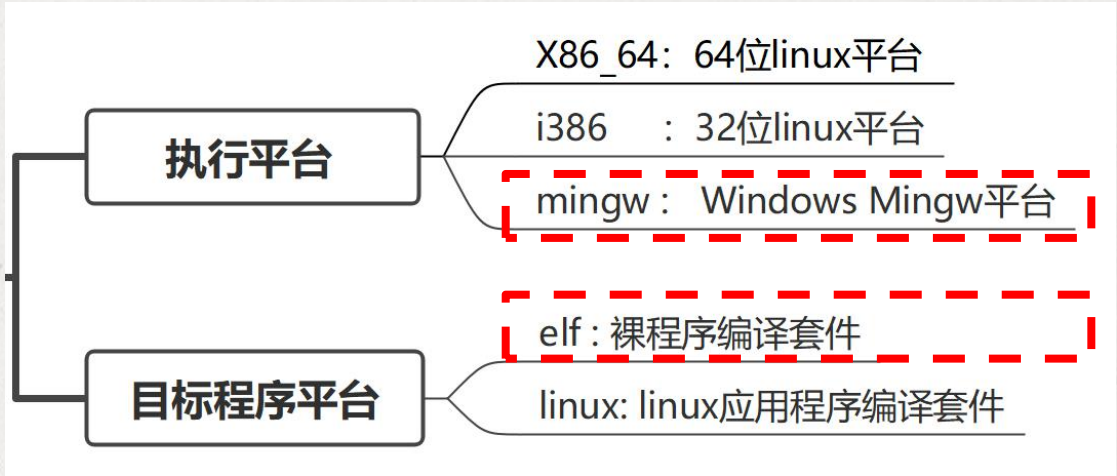
Xuantie-900-gcc-elf-newlib-x86_64-V2.0.3-20210806.tar.gz

Xuantie-900-gcc-linux-5.4.36-glibc-i386-V2.0.3-20210807.tar.gz

Xuantie-900-gcc-linux-5.4.36-glibc-mingw-V2.0.3-20210806.tar.gz

Xuantie-900-gcc-linux-5.4.36-glibc-x86_64-V2.0.3-20210806.tar.gz

平头哥工具链



不同版本的划分

平头哥GCC编译工具链

GCC编译工具链是指以GCC编译器为核心的一整套工具，用于把源代码转化成可执行应用程序。

GCC（GNU Compiler Collection） 由 GNU 开发的编程语言编译器。

GCC 最初代表 “GNU C Compiler”即当时只支持C语言。

后来又扩展能够支持更多编程语言，因此，GCC 也被重新定义为“GNU Compiler Collection”。

GCC编译过程

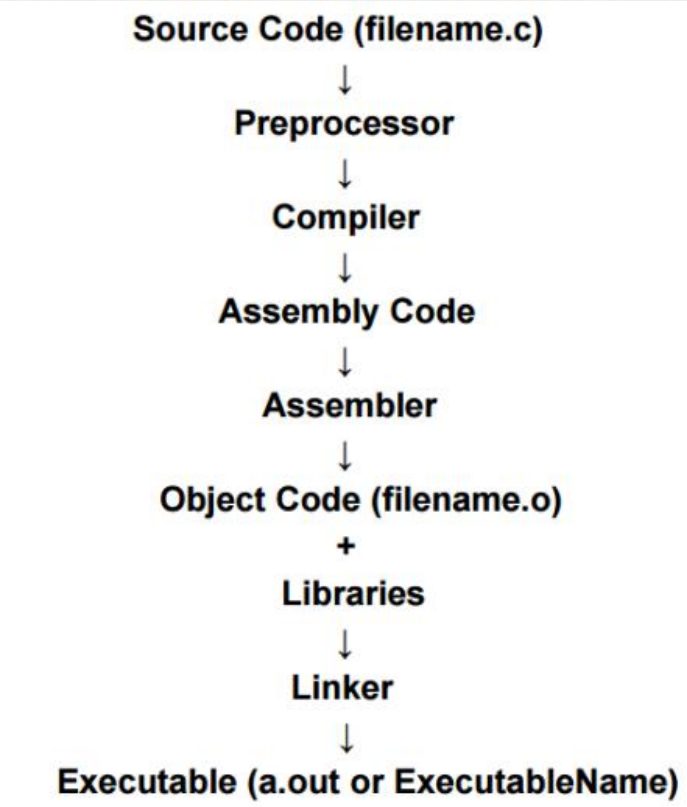
C源文件到可执行程序过程

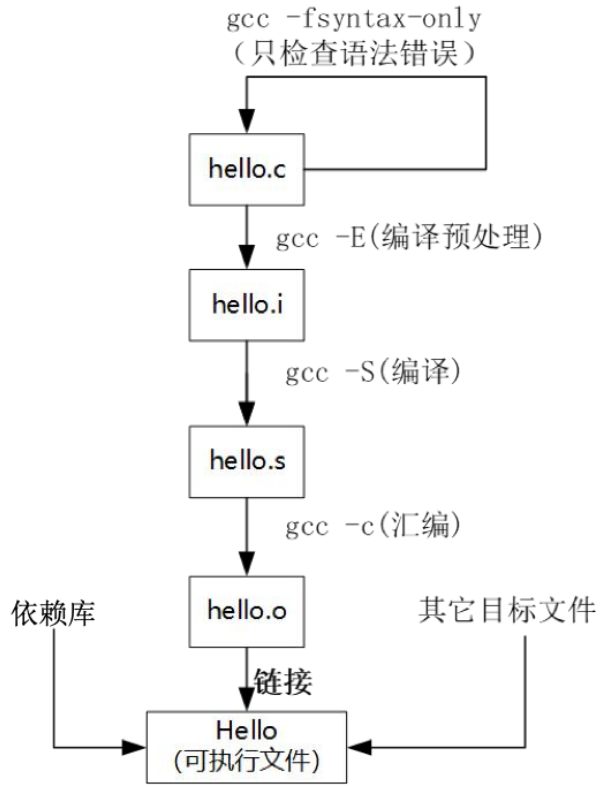
① 预处理 将宏定义、头文件包含进行展开，生成.i文件。

② 编译 将预处理后的.i文件通过编译成为汇编语言，生成.s文件。

③ 汇编 将汇编语言文件经过汇编，生成.o目标文件，由as汇编器将汇编语言代码转换为机器码。

④ 链接 由链接器ld完成将各个.o目标文件链接起来生成一个可执行程序文件。





目标文件

概念

编译器编译源代码后生成的能被CPU直接识别的二进制代码文件。

三种类型

- 可重定位文件(Relocatable File)
- 共享目标文件(shared Object File)
- 可执行文件(Executable File)

基础概念

可重定位文件

经过编译生成但未链接的二进制文件，包含代码和数据但不能执行。

共享目标文件

共享目标文件理解为依赖库。

可重定位文件

与其它可重定位文件或共享目标文件在链接器的作用下生成可执行文件。

基础概念

ELF头（ELF Header）

- 用来描述整个文件的组织。

Section节和Segment段

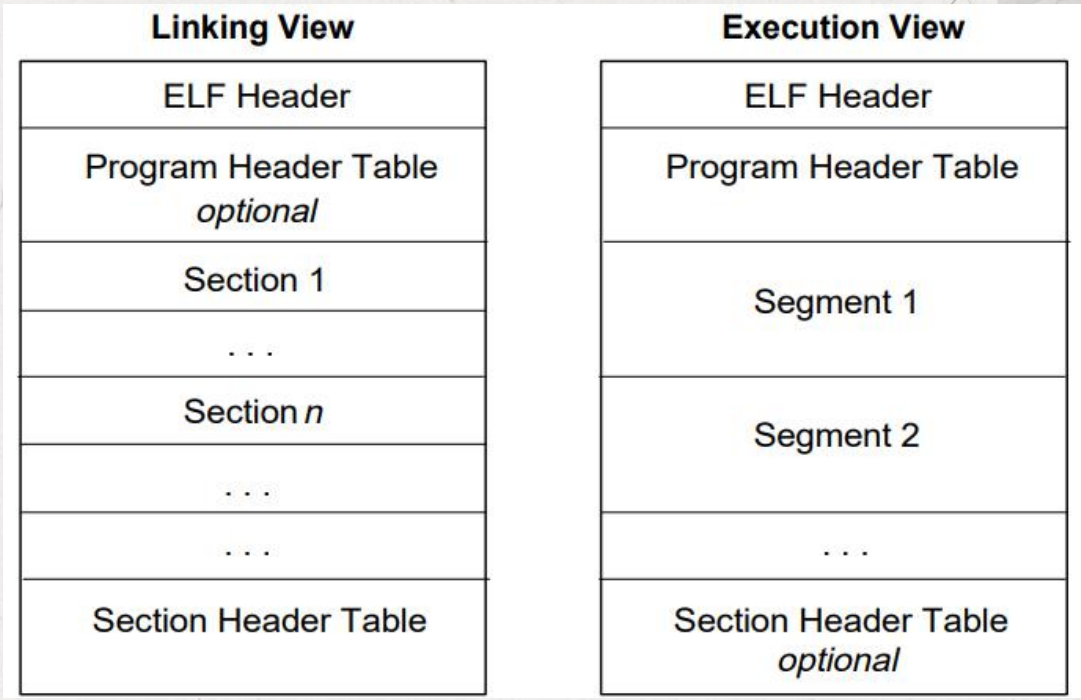
- 容纳相同属性数据的最小容器。

Section Header Table和Program Header Table

- 分别用来描述文件中的各个Section和Segment

链接器

- 链接器会根据链接脚本的映射关系将多个目标文件的Section按一定规则合并为Segment。



可重定位文件
和共享目标文件的文件格式

可执行文件的文件格式

基础概念

何为链接

链接是一个“打包”的过程，它将所有二进制形式的目标文件和依赖库组合成一个可执行文件。

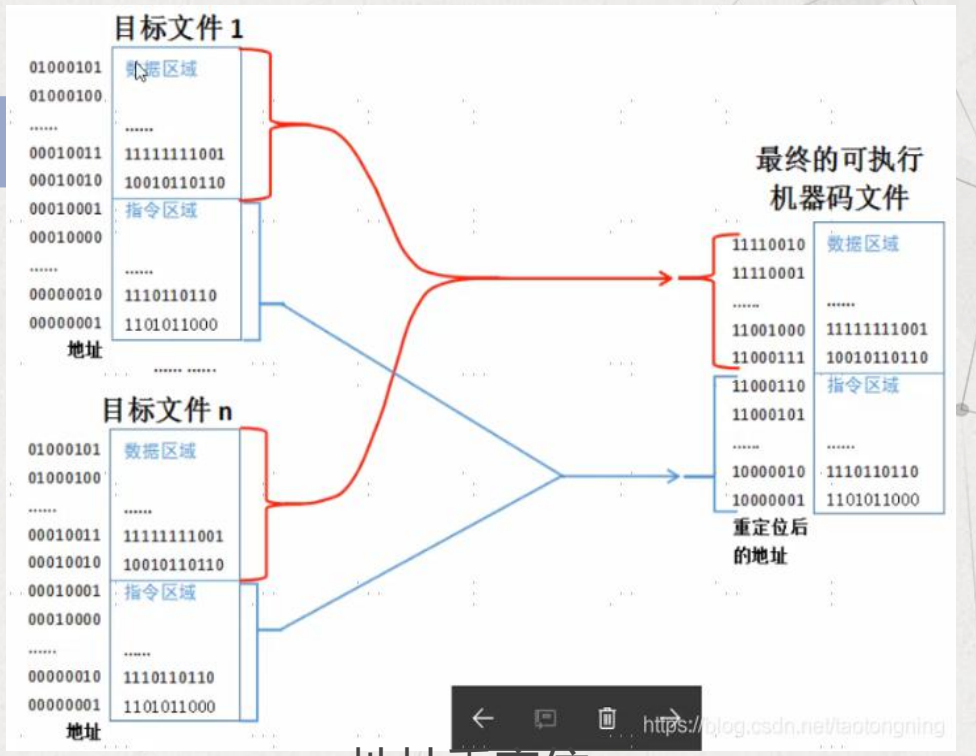
主要完成

①解析未定义的符号引用，将目标文件中的占位符替换为符号的地址

假如目标文件a调用目标文件b的函数，则在链接时会将目标文件b中函数的地址与a中调用位置建立关系。

②地址重定位

目标文件被整合的时候，每个目标文件的数据区被整合到一起，每个目标文件的指令区被整合到一起。



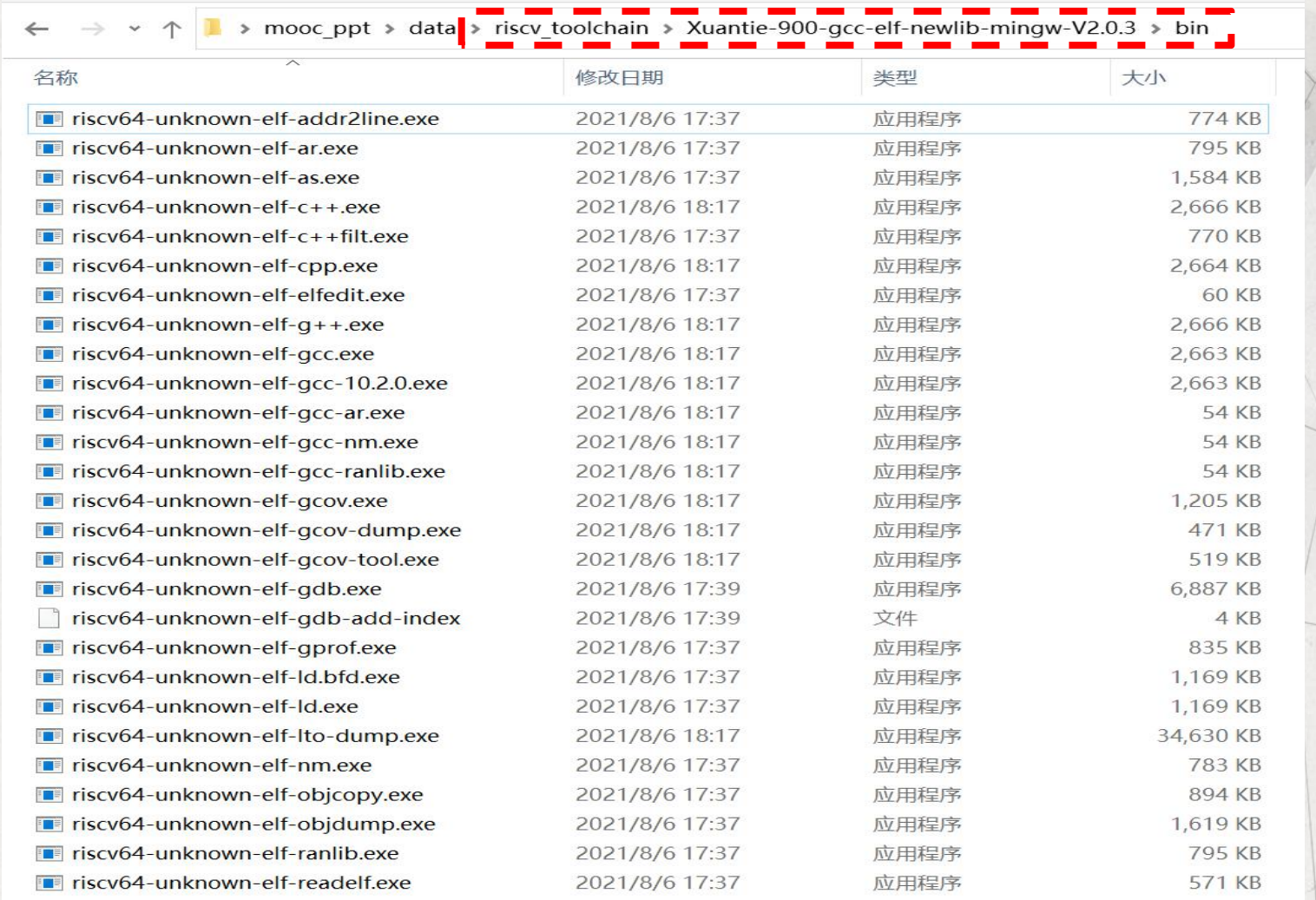
地址重定位

工具链准备

需要预先安装平头哥玄铁900系列的工具链

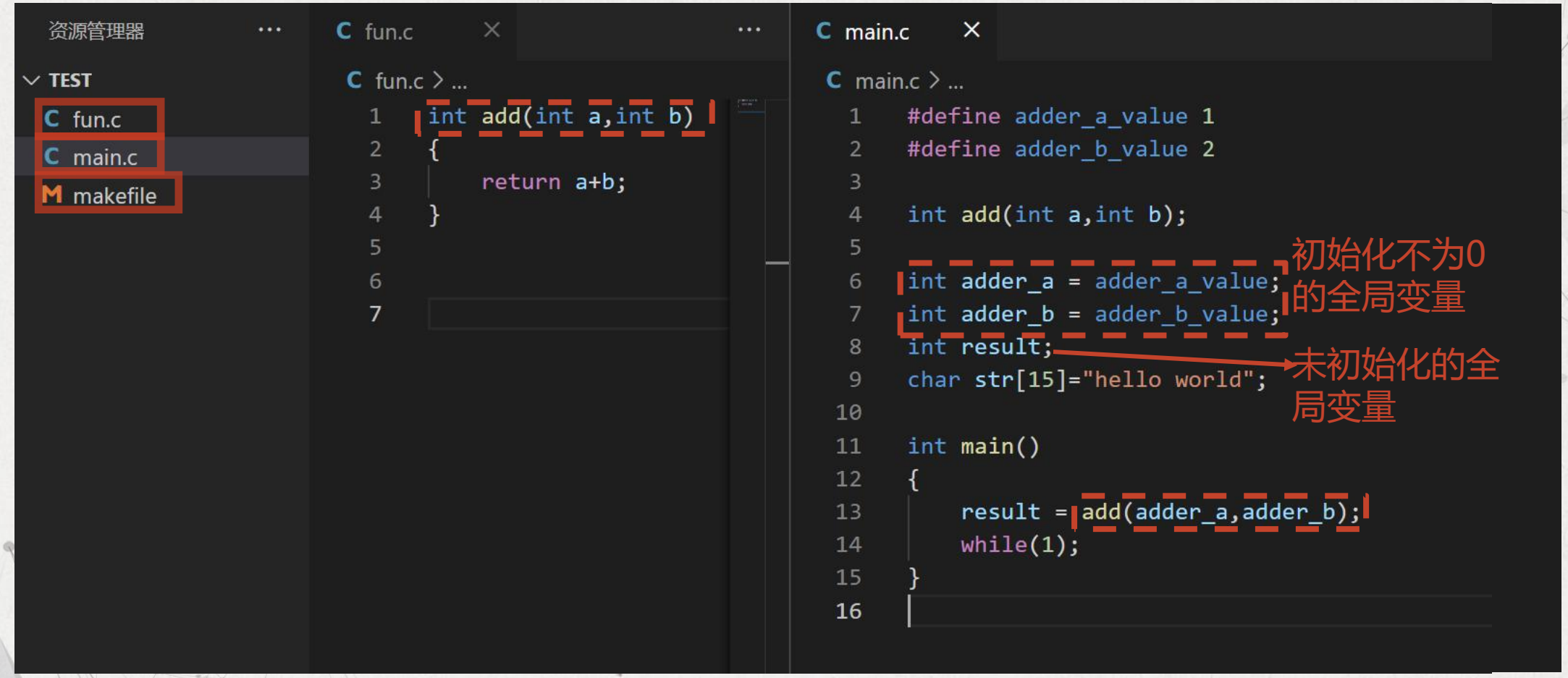
并将工具对应可执行程序的路径添加进环境变量以便于操作。

路径添加至环境变量



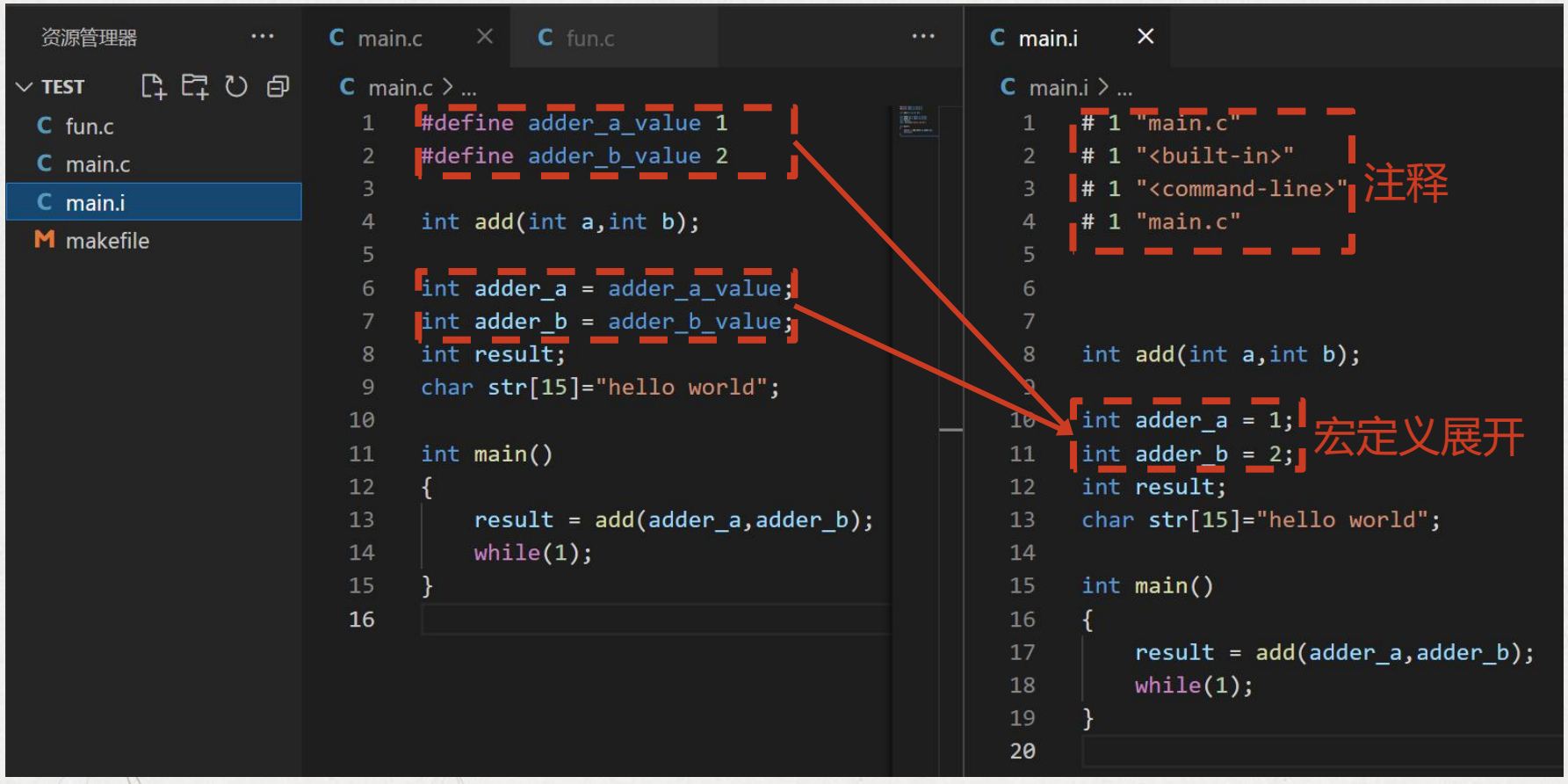
名称	修改日期	类型	大小
riscv64-unknown-elf-addr2line.exe	2021/8/6 17:37	应用程序	774 KB
riscv64-unknown-elf-ar.exe	2021/8/6 17:37	应用程序	795 KB
riscv64-unknown-elf-as.exe	2021/8/6 17:37	应用程序	1,584 KB
riscv64-unknown-elf-c++.exe	2021/8/6 18:17	应用程序	2,666 KB
riscv64-unknown-elf-c++filt.exe	2021/8/6 17:37	应用程序	770 KB
riscv64-unknown-elf-cpp.exe	2021/8/6 18:17	应用程序	2,664 KB
riscv64-unknown-elf-elfedit.exe	2021/8/6 17:37	应用程序	60 KB
riscv64-unknown-elf-g++.exe	2021/8/6 18:17	应用程序	2,666 KB
riscv64-unknown-elf-gcc.exe	2021/8/6 18:17	应用程序	2,663 KB
riscv64-unknown-elf-gcc-10.2.0.exe	2021/8/6 18:17	应用程序	2,663 KB
riscv64-unknown-elf-gcc-ar.exe	2021/8/6 18:17	应用程序	54 KB
riscv64-unknown-elf-gcc-nm.exe	2021/8/6 18:17	应用程序	54 KB
riscv64-unknown-elf-gcc-ranlib.exe	2021/8/6 18:17	应用程序	54 KB
riscv64-unknown-elf-gcov.exe	2021/8/6 18:17	应用程序	1,205 KB
riscv64-unknown-elf-gcov-dump.exe	2021/8/6 18:17	应用程序	471 KB
riscv64-unknown-elf-gcov-tool.exe	2021/8/6 18:17	应用程序	519 KB
riscv64-unknown-elf-gdb.exe	2021/8/6 17:39	应用程序	6,887 KB
riscv64-unknown-elf-gdb-add-index	2021/8/6 17:39	文件	4 KB
riscv64-unknown-elf-gprof.exe	2021/8/6 17:37	应用程序	835 KB
riscv64-unknown-elf-ld.bfd.exe	2021/8/6 17:37	应用程序	1,169 KB
riscv64-unknown-elf-ld.exe	2021/8/6 17:37	应用程序	1,169 KB
riscv64-unknown-elf-lto-dump.exe	2021/8/6 18:17	应用程序	34,630 KB
riscv64-unknown-elf-nm.exe	2021/8/6 17:37	应用程序	783 KB
riscv64-unknown-elf-objcopy.exe	2021/8/6 17:37	应用程序	894 KB
riscv64-unknown-elf-objdump.exe	2021/8/6 17:37	应用程序	1,619 KB
riscv64-unknown-elf-ranlib.exe	2021/8/6 17:37	应用程序	795 KB
riscv64-unknown-elf-readelf.exe	2021/8/6 17:37	应用程序	571 KB

案例源代码分析


资源管理器
TEST
fun.c
main.c
makefile
fun.c
fun.c > ...
1 int add(int a,int b)
2 {
3 return a+b;
4 }
5
6
7
main.c
main.c > ...
1 #define adder_a_value 1
2 #define adder_b_value 2
3
4 int add(int a,int b);
5
6 int adder_a = adder_a_value;
7 int adder_b = adder_b_value;
8 int result;
9 char str[15]="hello world";
10
11 int main()
12 {
13 result = add(adder_a,adder_b);
14 while(1);
15 }
16
初始化不为0的全局变量
未初始化的全局变量

预处理操作

输入命令：riscv64-unknown-elf-gcc.exe -E main.c -o main.i



main.i文件中以“#”开头的是注释。

预处理将源文件main.c中的宏定义展开成具体的内容，如果源文件中包含头文件，则在预处理过程会将头文件中的内容汇总

编译汇编

输入命令： riscv64-unknown-elf-gcc.exe -S main.i -O2

main.s

```
1   .file   "main.c"
2   .option nopic
3   .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0_xtheadc2p0"
4   .attribute unaligned_access, 1
5   .attribute stack_align, 16
6   .text
7   .section    .text.startup,"ax",@progbits
8   .align  1
9   .align  4
10  .globl  main
11  .type   main, @function
12 main:
13  lui a5,%hi(.LANCHOR0)
14  addi    a5,a5,%lo(.LANCHOR0)
15  lwd a1,a0,(a5),0,3
16  addi    sp,sp,-16
17  sd  ra,8(sp)
18  call    add
19  lui a5,%hi(.LANCHOR1)
20  sw  a0,%lo(.LANCHOR1)(a5)
21 .L2:
22  j   .L2
23  .size   main, .-main
```

main.s

```
24  .globl  str
25  .globl  result
26  .globl  adder_b
27  .globl  adder_a
28  .data
29  .align  3
30  .set    .LANCHOR0,. + 0
31  .type   adder_b, @object
32  .size   adder_b, 4
33 adder_b:
34  .word   2
35  .type   adder_a, @object
36  .size   adder_a, 4
37 adder_a:
38  .word   1
39  .type   str, @object
40  .size   str, 15
41 str:
42  .string "hello world"
43  .zero   3
44  .bss
45  .align  2
46  .set    .LANCHOR1,. + 0
47  .type   result, @object
48  .size   result, 4
49 result:
50  .zero   4
51  .ident  "GCC: (Xuantie Xuantie-900 elf newlib gcc Toolchain V2.0.3 B-20210806) 10.2.0"
```

定义四个符号（Symbol），使得链接器能够识别

Data数据段:
存放初始化且不为0的全局或静态变量。

bss数据段:
存放位初始化或初始化为0的全局或静态变量。

adder_a、adder_b变量存放在.data数据段，大小为4字节。Str数组也存放在data段，大小为15个字节。result全局变量未经过初始化，所以存放在.bss数据段。

输入命令： riscv64-unknown-elf-gcc.exe -c main.s
执行完该命令后可以看到目录下生成了main.o目标文件，由于该文件内容为二进制，因此需要借助readelf、objdump工具来查看内容。

使用readelf工具解析目标文件

输入命令： riscv64-unknown-elf-readelf.exe -a .\main.o –W

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -a .\main.o
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           RISC-V
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          1112 (bytes into file)
  Flags:                             0x5, RVC, double-float ABI
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         10
  Section header string table index: 9
```

文件类型为可重定位文件
机器类型为RISC-V

ELF Header信息

描述了整个文件的基本属性，可以看到该目标文件为可重定位文件且为RISC-V机器类型。

Section Headers

反映了目标文件中所有Section的基本属性。如段名、段的大小、在文件中的偏移和读写权限等。

```
Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        0000000000000000 000040 000032 00  AX  0   0  2
  [ 2] .rela.text        RELA            0000000000000000 0002a0 000168 18   I  7   1  8
  [ 3] .data             PROGBITS        0000000000000000 000078 000017 00  WA  0   0  8
  [ 4] .bss              NOBITS          0000000000000000 000090 000004 00  WA  0   0  4
  [ 5] .comment          PROGBITS        0000000000000000 000090 00004e 01  MS  0   0  1
  [ 6] .riscv.attributes RISCV_ATTRIBUTES 0000000000000000 0000de 000042 00      0   0  1
  [ 7] .symtab           SYMTAB          0000000000000000 000120 000150 18      8   8  8
  [ 8] .strtab           STRTAB          0000000000000000 000270 000030 00      0   0  1
  [ 9] .shstrtab         STRTAB          0000000000000000 000408 00004c 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  p (processor specific)
```

使用readelf工具解析目标文件

输入命令： riscv64-unknown-elf-readelf.exe -x .data .\main.o
执行完该命令后可以以字符串的形式显示该.data Section。

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -x .data .\main.o

Hex dump of section '.data':
  0x00000000 01000000 02000000 68656c6c 6f20776f ........hello wo
  0x00000010 726c6400 000000                     rld....
```

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -x .bss .\main.o
Section '.bss' has no data to dump.
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test>
```

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -x .comment .\main.o

Hex dump of section '.comment':
  0x00000000 00474343 3a202858 75616e74 69652058 .GCC: (Xuantie X
  0x00000010 75616e74 69652d39 30302065 6c66206e uantie-900 elf n
  0x00000020 65776c69 62206763 6320546f 6f6c6368 ewlib gcc Toolch
  0x00000030 61696e20 56322e30 2e332042 2d323032 ain V2.0.3 B-202
  0x00000040 31303830 36292031 302e322e 3000     10806) 10.2.0.
```

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -x .riscv.attributes .\main.o

Hex dump of section '.riscv.attributes':
  0x00000000 41410000 00726973 63760001 37000000 AA...riscv..7...
  0x00000010 04100572 76363469 3270305f 6d327030 ...rv64i2p0_m2p0
  0x00000020 5f613270 305f6632 70305f64 3270305f _a2p0_f2p0_d2p0_
  0x00000030 63327030 5f787468 65616463 32703000 c2p0_xtheadc2p0.
  0x00000040 0601                                ..
```

使用readelf工具解析目标文件

输入命令： riscv64-unknown-elf-readelf.exe -s .\main.o –W
执行完该命令后可以查看.symtab段的内容

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-readelf.exe -s  .\main.o -W

Symbol table '.symtab' contains 14 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS main.c
     2: 0000000000000000     0 SECTION LOCAL  DEFAULT    1
     3: 0000000000000000     0 SECTION LOCAL  DEFAULT    3
     4: 0000000000000000     0 SECTION LOCAL  DEFAULT    4
     5: 0000000000000030     0 NOTYPE  LOCAL  DEFAULT    1 .L2
     6: 0000000000000000     0 SECTION LOCAL  DEFAULT    5
     7: 0000000000000000     0 SECTION LOCAL  DEFAULT    6
     8: 0000000000000000     4 OBJECT  GLOBAL DEFAULT    3 adder_a
     9: 0000000000000004     4 OBJECT  GLOBAL DEFAULT    3 adder_b
    10: 0000000000000000     4 OBJECT  GLOBAL DEFAULT    4 result
    11: 0000000000000008    15 OBJECT  GLOBAL DEFAULT    3 str
    12: 0000000000000000    50 FUNC    GLOBAL DEFAULT    1 main
    13: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT  UND add
```

.symtab段内容为符号表，记录了目标文件中使用的所有符号，比如变量和函数名。对于变量和函数而言，符号对应的值为它们所在的地址，用于链接器链接时找到对应地址。
可以看到该目标文件的符号表记录了三个变量、数组和main函数符号的地址。

使用objdump工具解析目标文件

输入命令： riscv64-unknown-elf-objdump.exe -d .\main.o

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-objdump.exe  -d .\main.o

.\main.o:     file format elf64-littleriscv


Disassembly of section .text:

0000000000000000 <main>:
   0:   1141                    addi    sp,sp,-16
   2:   e406                    sd      ra,8(sp)
   4:   e022                    sd      s0,0(sp)
   6:   0800                    addi    s0,sp,16
   8:   000007b7                lui     a5,0x0
   c:   0007a703                lw      a4,0(a5) # 0 <main>
  10:   000007b7                lui     a5,0x0
  14:   0007a783                lw      a5,0(a5) # 0 <main>
  18:   85be                    mv      a1,a5
  1a:   853a                    mv      a0,a4
  1c:   00000097                auipc   ra,0x0
  20:   000080e7                jalr    ra # 1c <main+0x1c>
  24:   87aa                    mv      a5,a0
  26:   873e                    mv      a4,a5
  28:   000007b7                lui     a5,0x0
  2c:   00e7a023                sw      a4,0(a5) # 0 <main>

0000000000000030 <.L2>:
  30:   a001                    j       30 <.L2>
```

使用objdump工具解析目标文件

输入命令： riscv64-unknown-elf-objdump.exe -s .\main.o

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> riscv64-unknown-elf-objdump.exe  -s .\main.o

.\main.o:     file format elf64-littleriscv

Contents of section .text:
 0000 411106e4 22e00008 b7070000 03a70700  A..."...........
 0010 b7070000 83a70700 be853a85 97000000  ..........:.....
 0020 e7800000 aa873e87 b7070000 23a0e700  ......>.....#...
 0030 01a0                                 ..
Contents of section .data:
 0000 01000000 02000000 68656c6c 6f20776f  ........hello wo
 0010 726c6400 000000                      rld....
Contents of section .comment:
 0000 00474343 3a202858 75616e74 69652058  .GCC: (Xuantie X
 0010 75616e74 69652d39 30302065 6c66206e  uantie-900 elf n
 0020 65776c69 62206763 6320546f 6f6c6368  ewlib gcc Toolch
 0030 61696e20 56322e30 2e332042 2d323032  ain V2.0.3 B-202
 0040 31303830 36292031 302e322e 3000      10806) 10.2.0.
Contents of section .riscv.attributes:
 0000 41410000 00726973 63760001 37000000  AA...riscv..7...
 0010 04100572 76363469 3270305f 6d327030  ...rv64i2p0_m2p0
 0020 5f613270 305f6632 70305f64 3270305f  _a2p0_f2p0_d2p0_
 0030 63327030 5f787468 65616463 32703000  c2p0_xtheadc2p0.
 0040 0601                                 ..
```

链接

输入命令： riscv64-unknown-elf-gcc.exe .\fun.c .\main.c -o main
执行完命令后可以看到目录下的main可执行文件。

由于我们使用的编译工具链的目标平台是RISC-V架构的，即它生成的应用程序只能运行于RISC-V平台的开发板，而不适用于x86平台，因此无法在电脑上运行生成的main可执行程序。

输入命令： riscv64-unknown-elf-readelf.exe -a .\main –W
执行完该命令后可以查看main可执行程序进行解析。

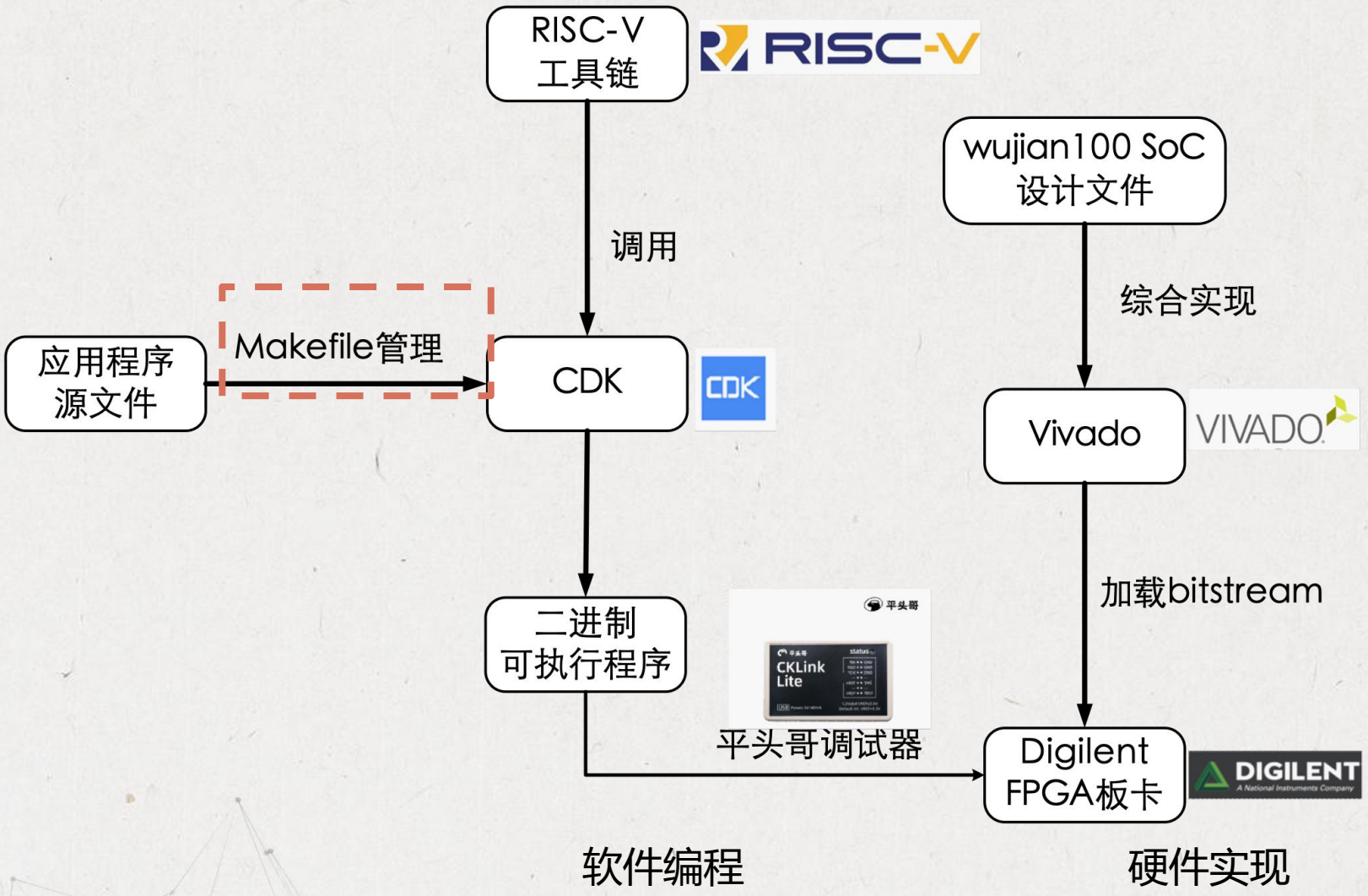
```
Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        00000000000100b0 0000b0 00042e 00  AX  0   0 16
  [ 2] .rodata           PROGBITS        00000000000104e0 0004e0 000008 00   A  0   0  8
  [ 3] .eh_frame         PROGBITS        00000000000114e8 0004e8 000004 00  WA  0   0  4
  [ 4] .init_array       INIT_ARRAY      00000000000114f0 0004f0 000010 08  WA  0   0  8
  [ 5] .fini_array       FINI_ARRAY      0000000000011500 000500 000008 08  WA  0   0  8
  [ 6] .data             PROGBITS        0000000000011508 000508 000770 00  WA  0   0  8
  [ 7] .bss              NOBITS          0000000000011c78 000c78 000040 00  WA  0   0  8
  [ 8] .comment          PROGBITS        0000000000000000 000c78 00004d 01  MS  0   0  1
  [ 9] .riscv.attributes RISCV_ATTRIBUTES 0000000000000000 000cc5 000042 00      0   0  1
  [10] .symtab           SYMTAB          0000000000000000 000d08 000630 18     11  40  8
  [11] .strtab           STRTAB          0000000000000000 001338 00026e 00      0   0  1
  [12] .shstrtab         STRTAB          0000000000000000 0015a6 000071 00      0   0  1

Program Headers:
  Type           Offset   VirtAddr           PhysAddr           FileSiz  MemSiz   Flg Align
  LOAD           0x000000 0x0000000000010000 0x0000000000010000 0x0004e8 0x0004e8 R E 0x1000
  LOAD           0x0004e8 0x00000000000114e8 0x00000000000114e8 0x000790 0x0007d0 RW  0x1000

 Section to Segment mapping:
  Segment Sections...
   00     .text .rodata
   01     .eh_frame .init_array .fini_array .data .bss
```

VMA运行地址

LMA加载地址


RISC-V
工具链
RISC-V
调用
wujian100 SoC
设计文件
综合实现
应用程序
源文件
Makefile管理
CDK
CDK
Vivado
VIVADO
加载bitstream
二进制
可执行程序
CKLink
Lite
平头哥调试器
Digilent
FPGA板卡
DIGILENT
A National Instruments Company
软件编程
硬件实现

Makefile介绍

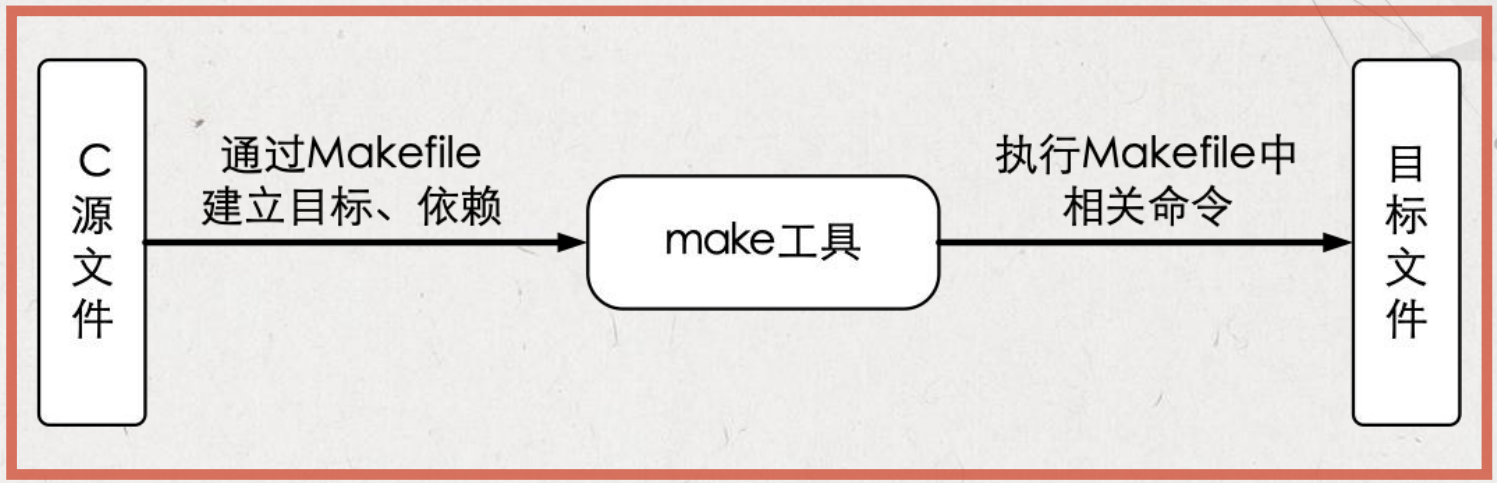
随着工程变得复杂，需要越来越多的C文件。

出现的问题

- 在每次编译生成最终可执行程序时都需要输入所有文件名。
- 如果有一个文件发生变化，则需要重新编译所有文件。

解决方法

搭配使用make工具和makefile脚本文件



Makefile介绍

makefile

```
1   CC:=riscv64-unknown-elf-gcc.exe
2   DUMP:=riscv64-unknown-elf-readelf.exe
3
4   .PHONY:clean
5   .PHONY:dump
6
7   main:fun.c main.c
8       $(CC) fun.c main.c  -o main
9
10  clean:
11      del .\*.o main
12  dump:
13      $(DUMP) -a main -W
```

脚本的前两行用CC和DUMP变量代替gcc和readelf工具

Makefile介绍

makefile

```
1   CC:=riscv64-unknown-elf-gcc.exe
2   DUMP:=riscv64-unknown-elf-readelf.exe
3
4   .PHONY:clean
5   .PHONY:dump
6
7   main:fun.c main.c
8       $(CC) fun.c main.c  -o main
9
10  clean:
11      del .\*.o main
12  dump:
13      $(DUMP) -a main -W
```

建立目标依赖关系，其中main可执行文件依赖于fun.c和main.c两个文件。当执行make main操作时，make工具会对比目标文件和依赖文件最后一次修改的时间。如果main目标文件最后一次更新时间要早于依赖文件fun.c、main.c，则会执行下一行的命令重新进行编译。

Makefile文件格式
#目标：依赖的文件或其它目标
#Tab　命令1
#Tab　命令n
顶格书写目标名称
下一行是需要先输入Tab，再编写对应的命令

Makefile介绍

makefile

```
1   CC:=riscv64-unknown-elf-gcc.exe
2   DUMP:=riscv64-unknown-elf-readelf.exe
3
4   .PHONY:clean
5   .PHONY:dump
6
7   main:fun.c main.c
8       $(CC) fun.c main.c  -o main
9
10  clean:
11      del .\*.o main
12  dump:
13      $(DUMP) -a main -W
```

同样建立目标依赖关系，只是目标不依赖于其他内容。脚本中4、5行规定了clean、dump是伪目标，即只要执行make clean/make dump操作就一定会执行对应的命令。

Makefile介绍

makefile

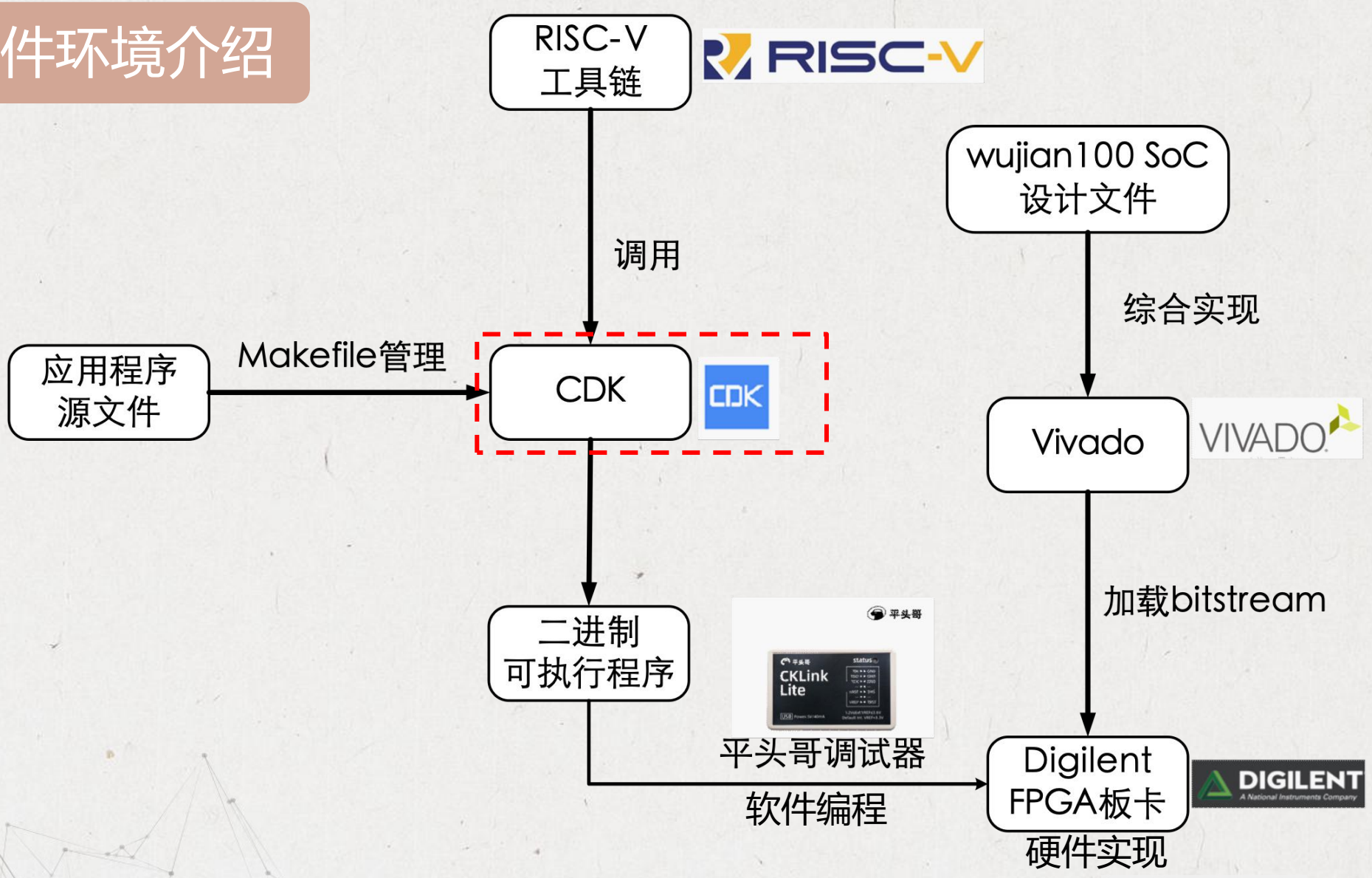
```
1   CC:=riscv64-unknown-elf-gcc.exe
2   DUMP:=riscv64-unknown-elf-readelf.exe
3
4   .PHONY:clean
5   .PHONY:dump
6
7   main:fun.c main.c
8       $(CC) fun.c main.c  -o main
9
10  clean:
11      del .\*.o main
12  dump:
13      $(DUMP) -a main -W
```

```
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> make main
riscv64-unknown-elf-gcc.exe  fun.c main.c  -o main
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> make dump
riscv64-unknown-elf-readelf.exe -a main -W
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           RISC-V
  Version:                           0x1
  Entry point address:               0x100cc
  Start of program headers:          64 (bytes into file)
  Start of section headers:          5656 (bytes into file)
  Flags:                             0x5, RVC, double-float ABI
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         2
  Size of section headers:           64 (bytes)
  Number of section headers:         13
  Section header string table index: 12
PS C:\Users\hnaym\Desktop\mooc_ppt\data\riscv_toolchain\test> make clean
del .\*.o main
```

由于当前目录下不含main可执行文件，满足目标文件最后一次修改时间早于依赖文件。因此执行make main命令时会自动调用makefile中对应的命令编译生成可执行文件。

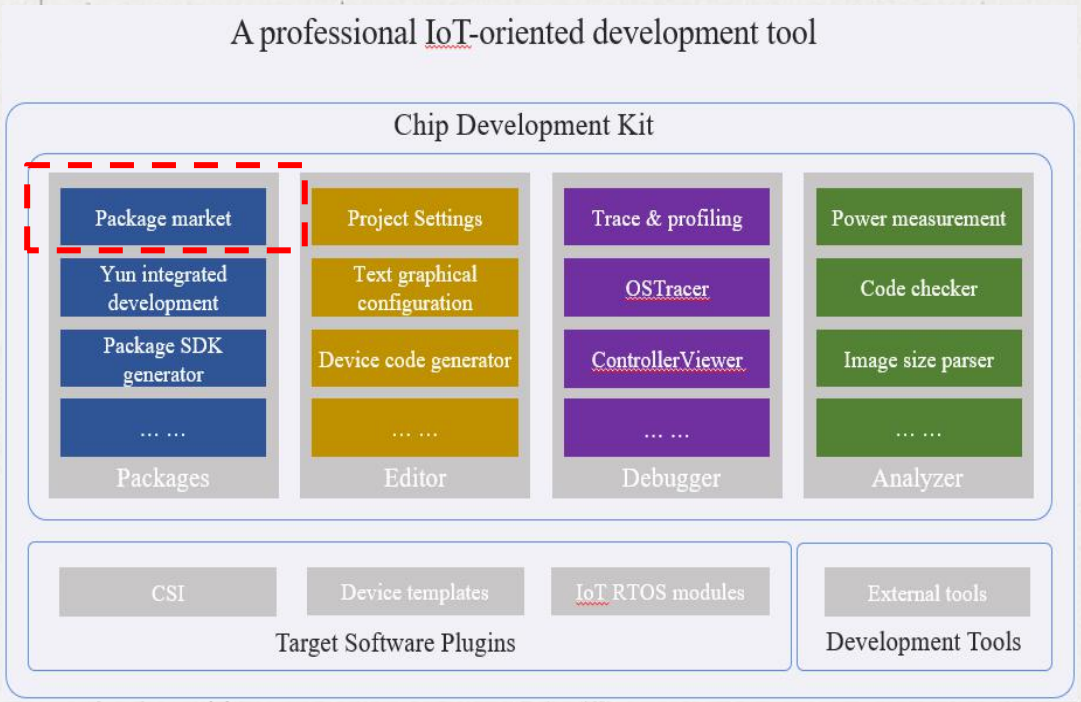
执行make dump、make clean命令时同样会调用makefile中规定的命令使用readelf工具读取文件信息、删除指定文件。

软硬件环境介绍


RISC-V
工具链
RISC-V
调用
wujian100 SoC
设计文件
综合实现
应用程序
源文件
Makefile管理
CDK
CDK
Vivado
VIVADO
加载bitstream
二进制
可执行程序
平头哥
CKLink
Lite
平头哥调试器
软件编程
Digilent
FPGA板卡
DIGILENT
A National Instruments Company
硬件实现

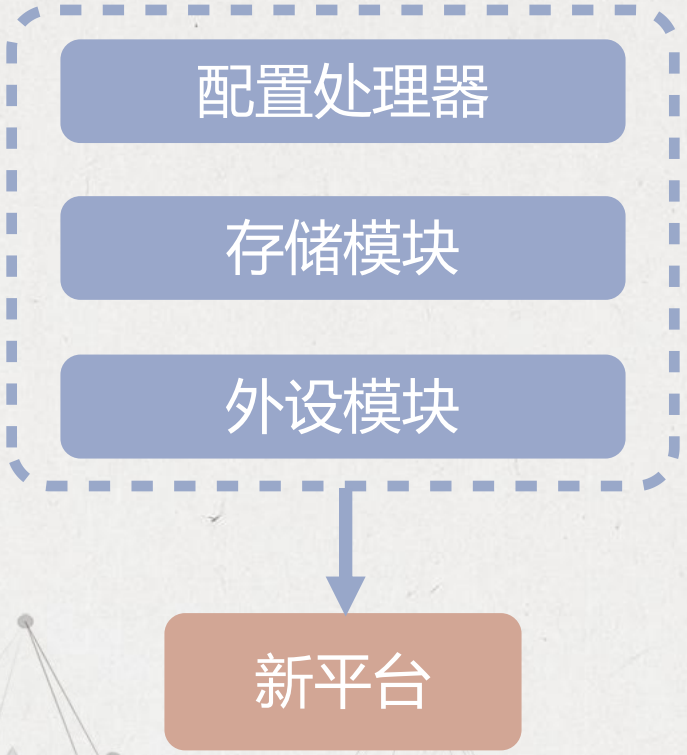
CDK软件介绍

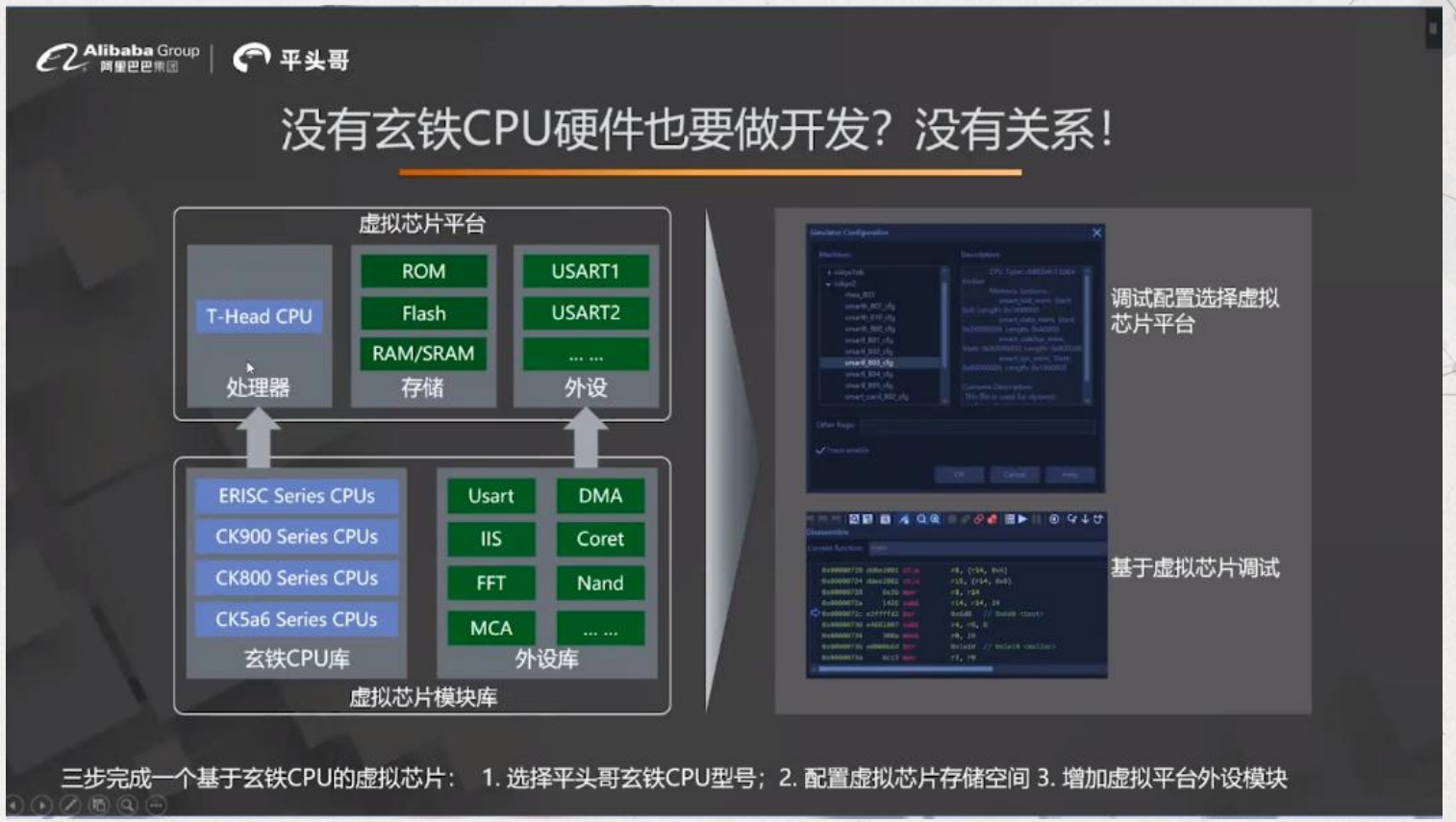
剑池CDK是平头哥公司推出的一款专业面向IoT开发领域的集成开发环境，该集成开发环境为开发者提供简洁统一的图形开发界面，帮助开发者进行应用开发。该开发环境目前已支持平头哥自研指令集C-SKY架构和RISC-V架构的芯片的开发。



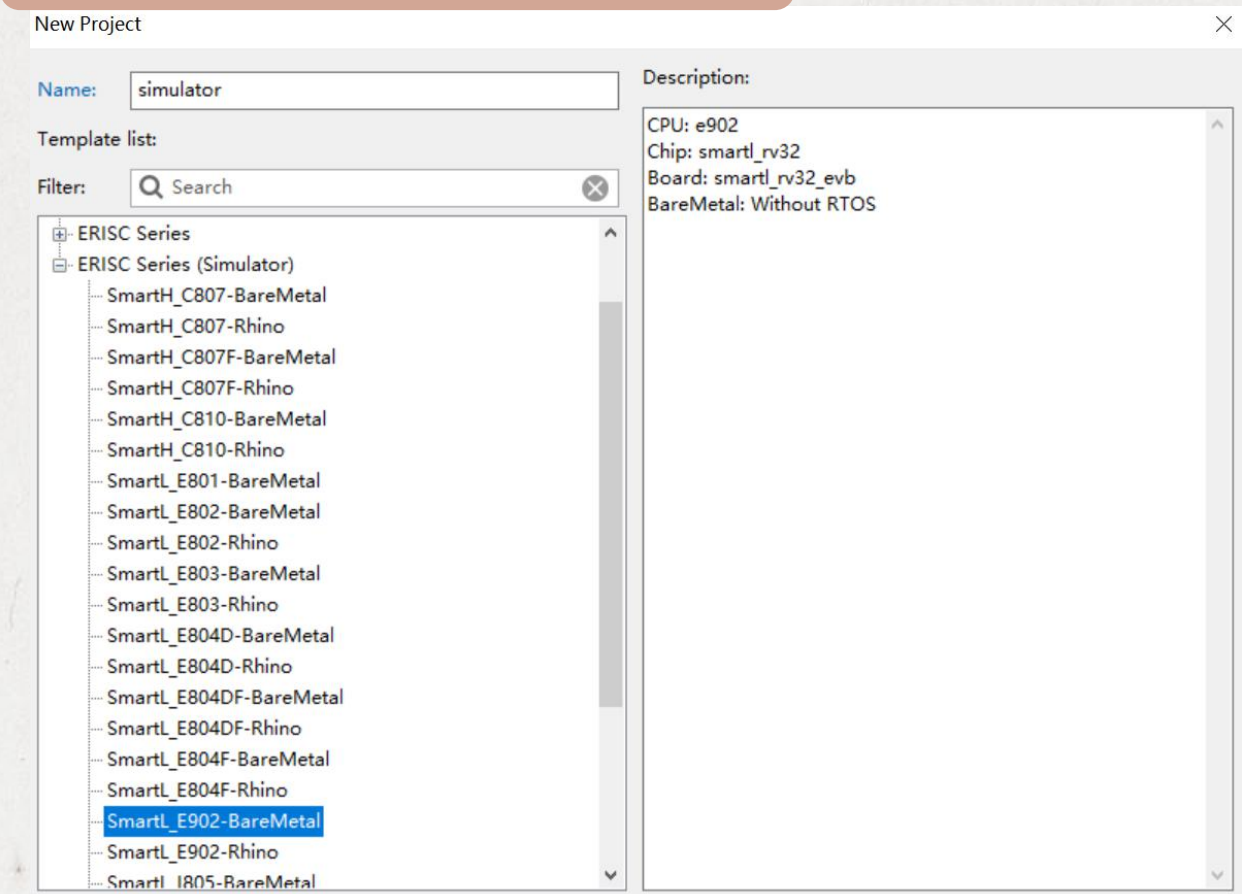
CDK内置软件模拟器

在进行SDK开发过程中，在没有完成硬件CPU的情况下，可以使用CDK软件内置的模拟器进行虚拟芯片平台的模拟，从而在调试过程中将程序运行在此虚拟平台上。



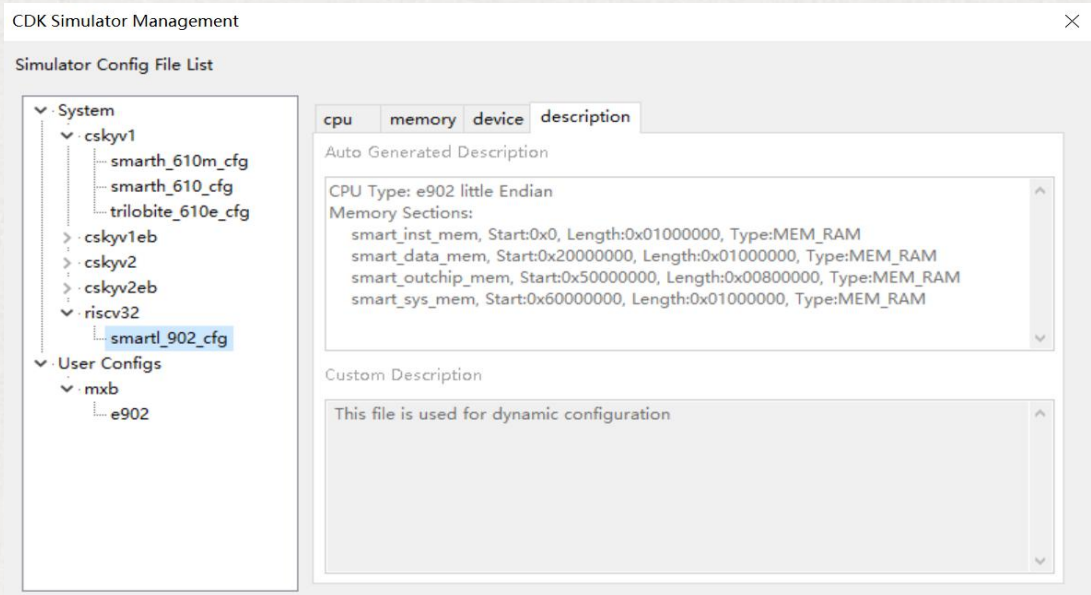


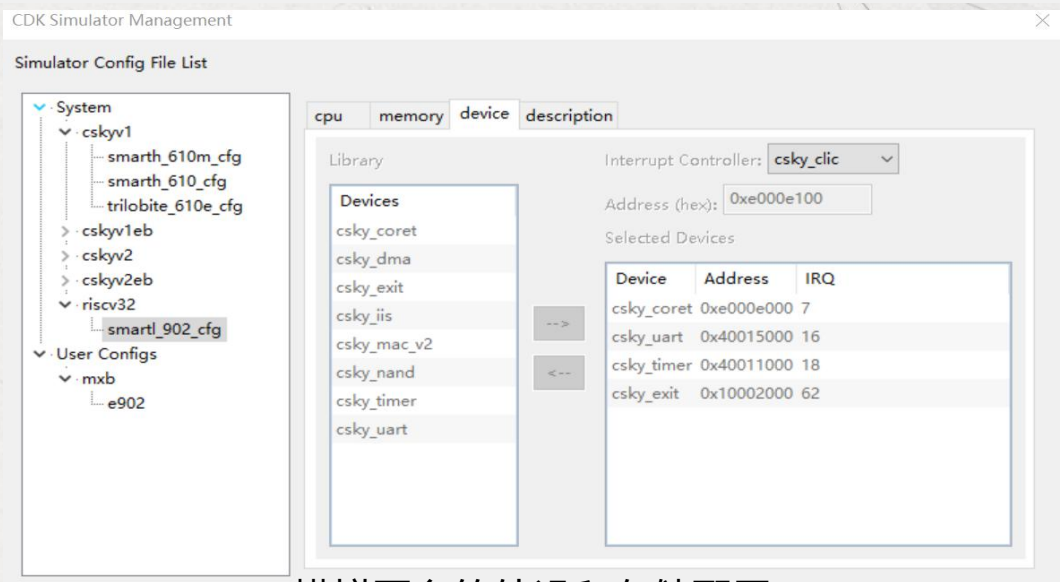
CDK内置软件模拟器



基于E902内核处理器的模拟平台

可以基于此工程完成对E902处理器的运行模拟、进行SDK开发使用等。

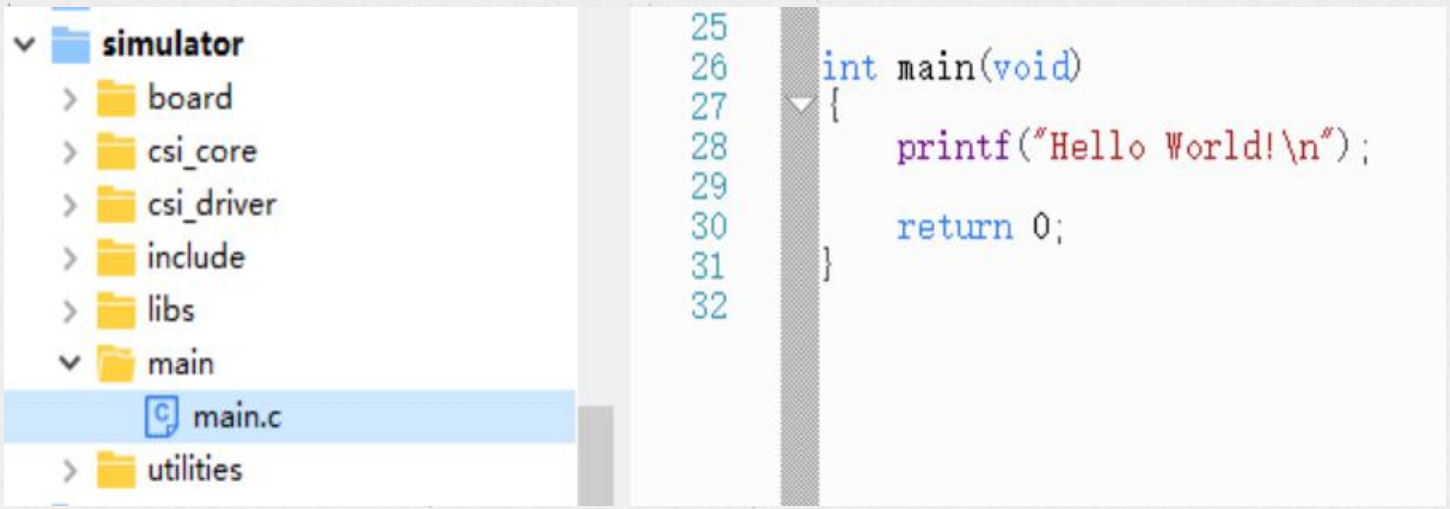




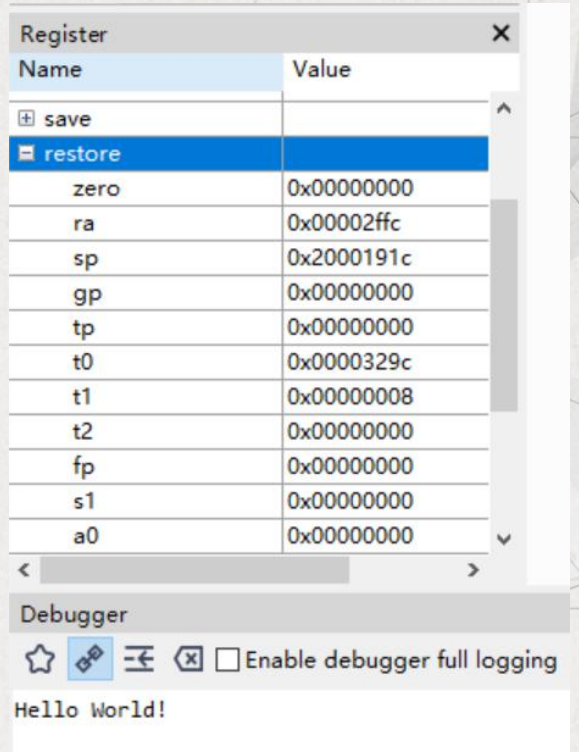
模拟平台的外设和存储配置

CDK内置软件模拟器

当创建好默认的虚拟平台后，CDK软件工程会附带配套的外设驱动库和链接脚本等完整的SDK。如果是基于自定义虚拟平台的工程，CDK软件也会根据外设驱动和基地址等的不同对SDK进行调整，用户只需考虑SDK开发即可，无需关心底层的硬件。



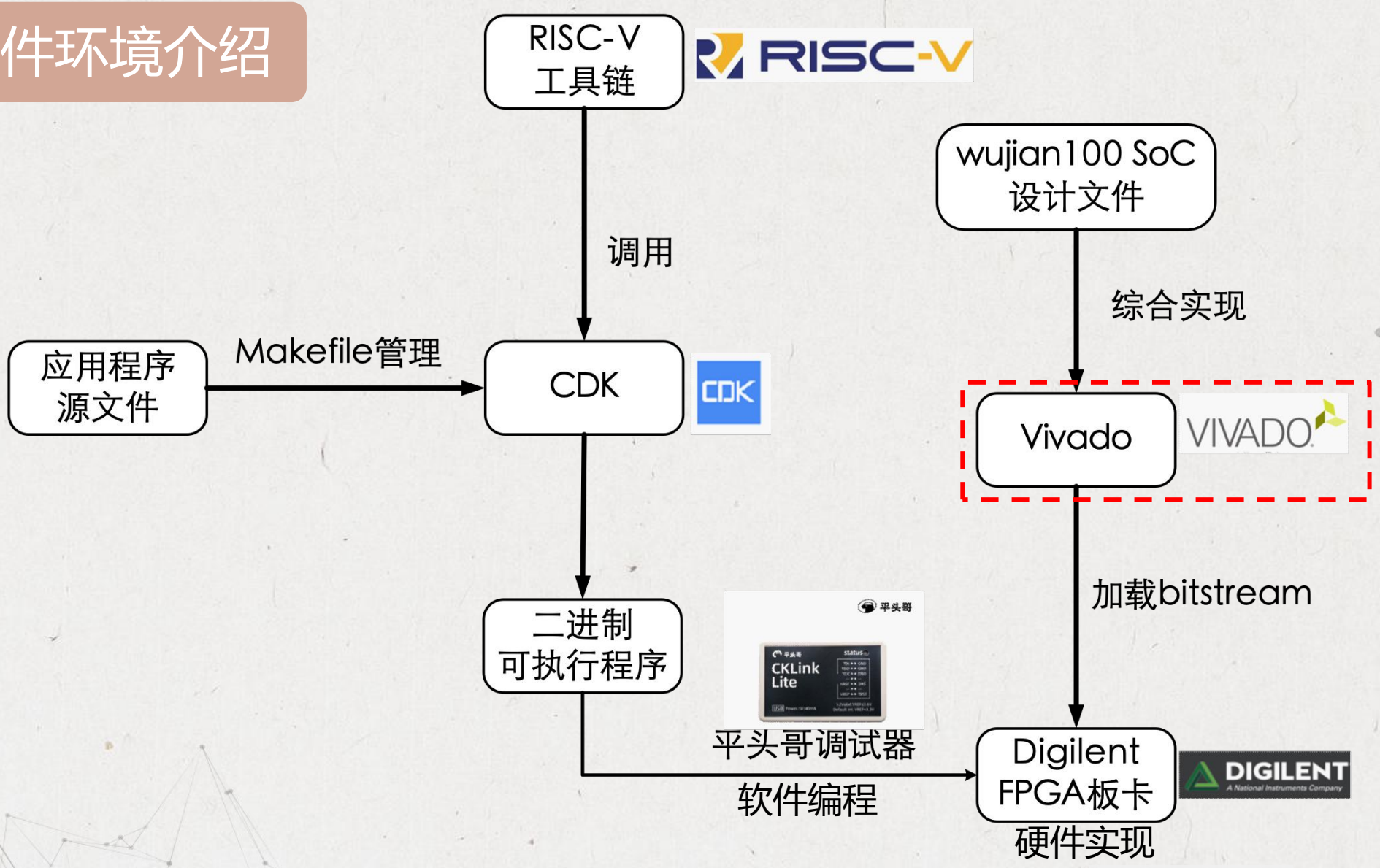
配套SDK工程和应用程序



仿真调试

完成对工程的编译后，点击调试按钮即可以对虚拟平台进行模拟。此处是对串口输出“Hello World\n”进行模拟，可以看到控制台接收到串口输出信息。除此之外，CDK软件模拟器还提供了通用寄存器、控制状态基础器和内存值的查看和修改，实现了和硬件调试相同的效果。

软硬件环境介绍

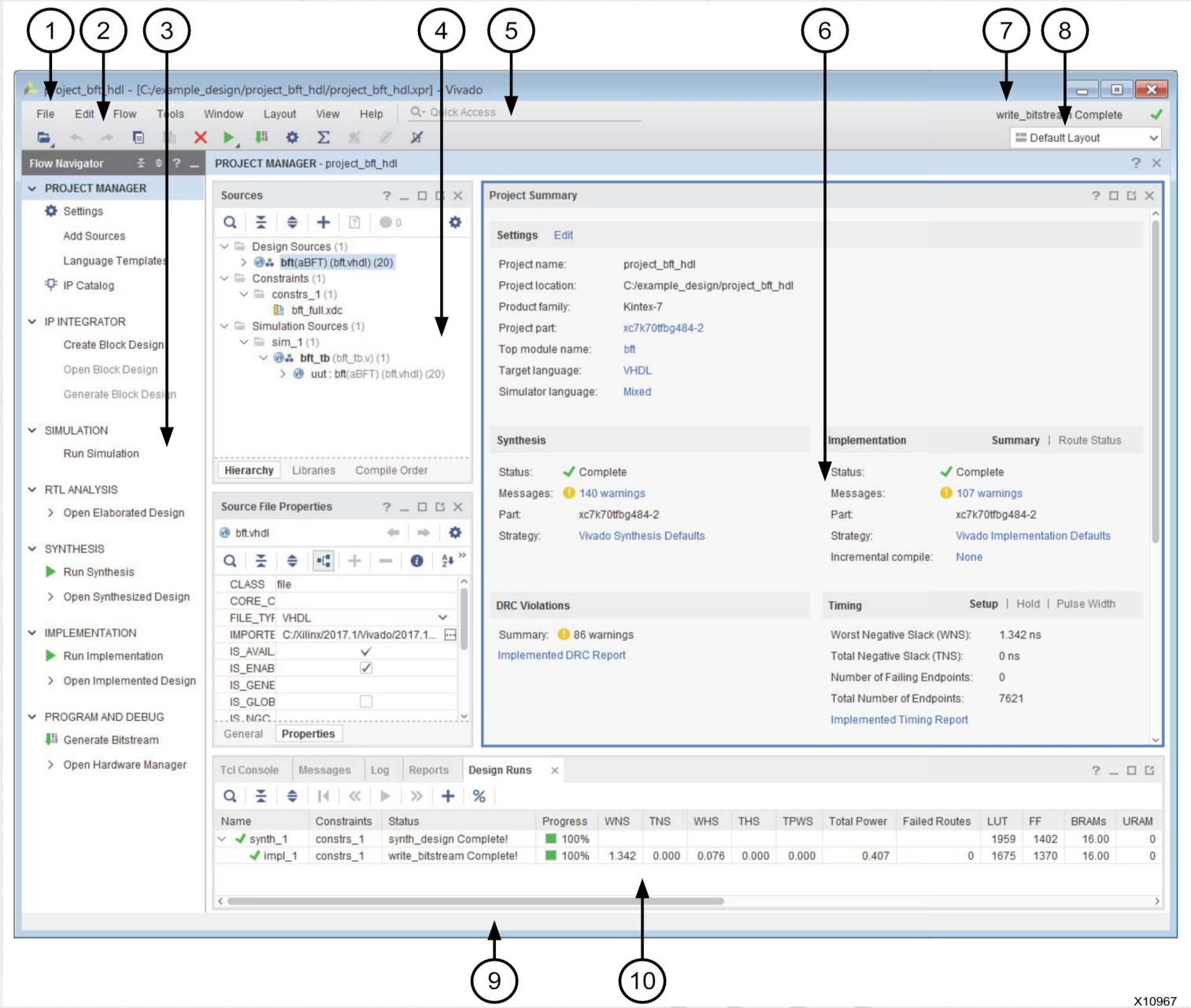

RISC-V
工具链
RISC-V
调用
wujian100 SoC
设计文件
综合实现
应用程序
源文件
Makefile管理
CDK
CDK
Vivado
VIVADO
加载bitstream
二进制
可执行程序
平头哥
CKLink
Lite
平头哥调试器
软件编程
Digilent
FPGA板卡
DIGILENT
A National Instruments Company
硬件实现

Vivado FPGA开发平台

Vivado是FPGA厂商赛灵思公司2012年发布的集成设计环境，包括功能仿真、设计综合、布局布线和板级调试等。开发者可以根据需求自行编写RTL代码、调用套件提供的IP核等。Vivado提供windows版本和Linux版本，开发者可以根据自己的需求来选择不同的工具版本。

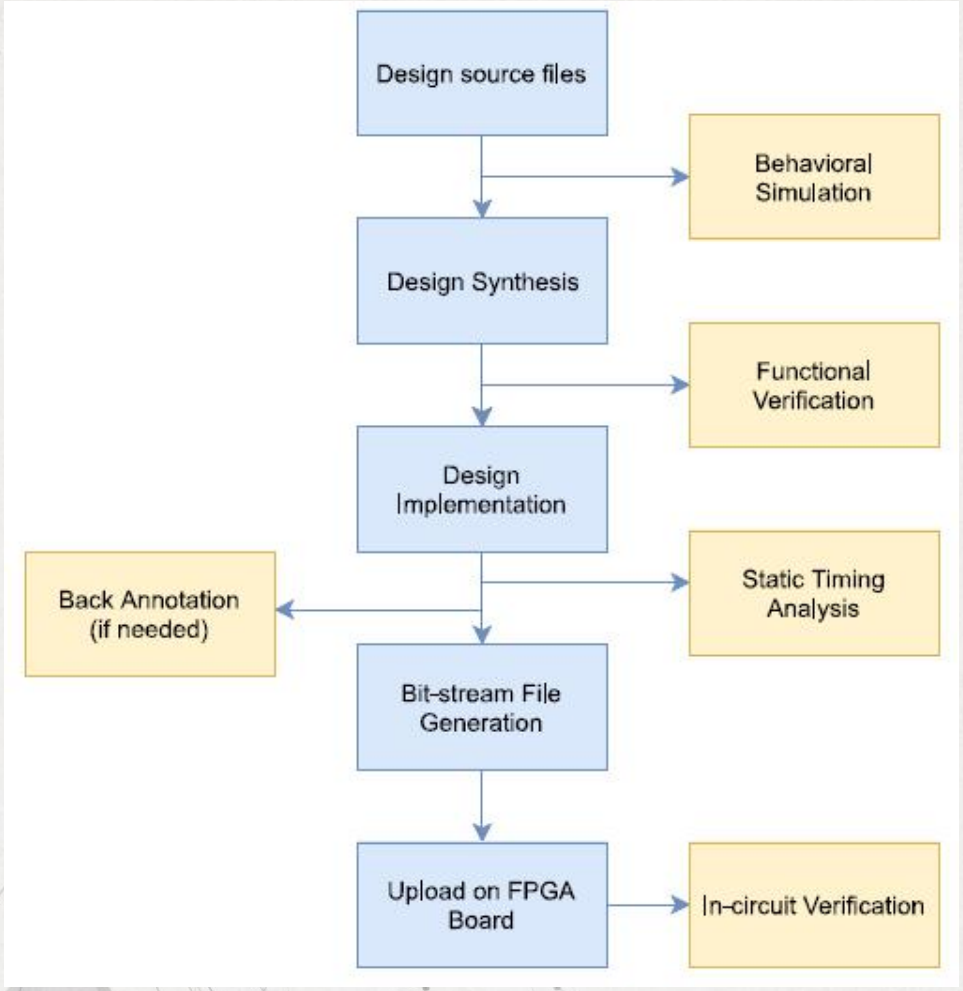
Vivado IDE主要部件：

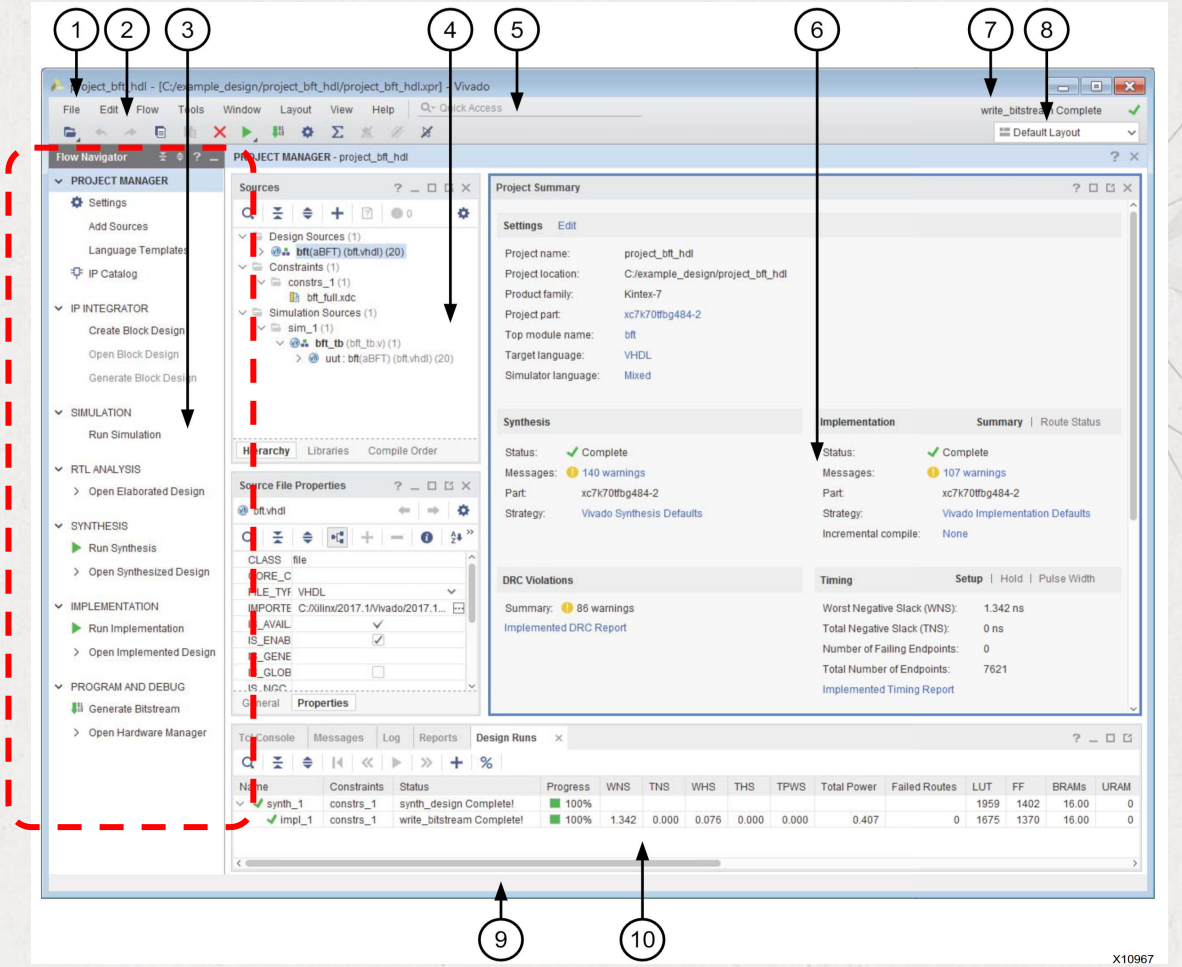
1. Menu Bar
2. Main Toolbar
3. Flow Navigator
4. Data Windows Area
5. Menu Command Quick Access Search Field
6. Workspace
7. Project Status Bar
8. Layout Selector
9. Status Bar
10. Results Windows Area



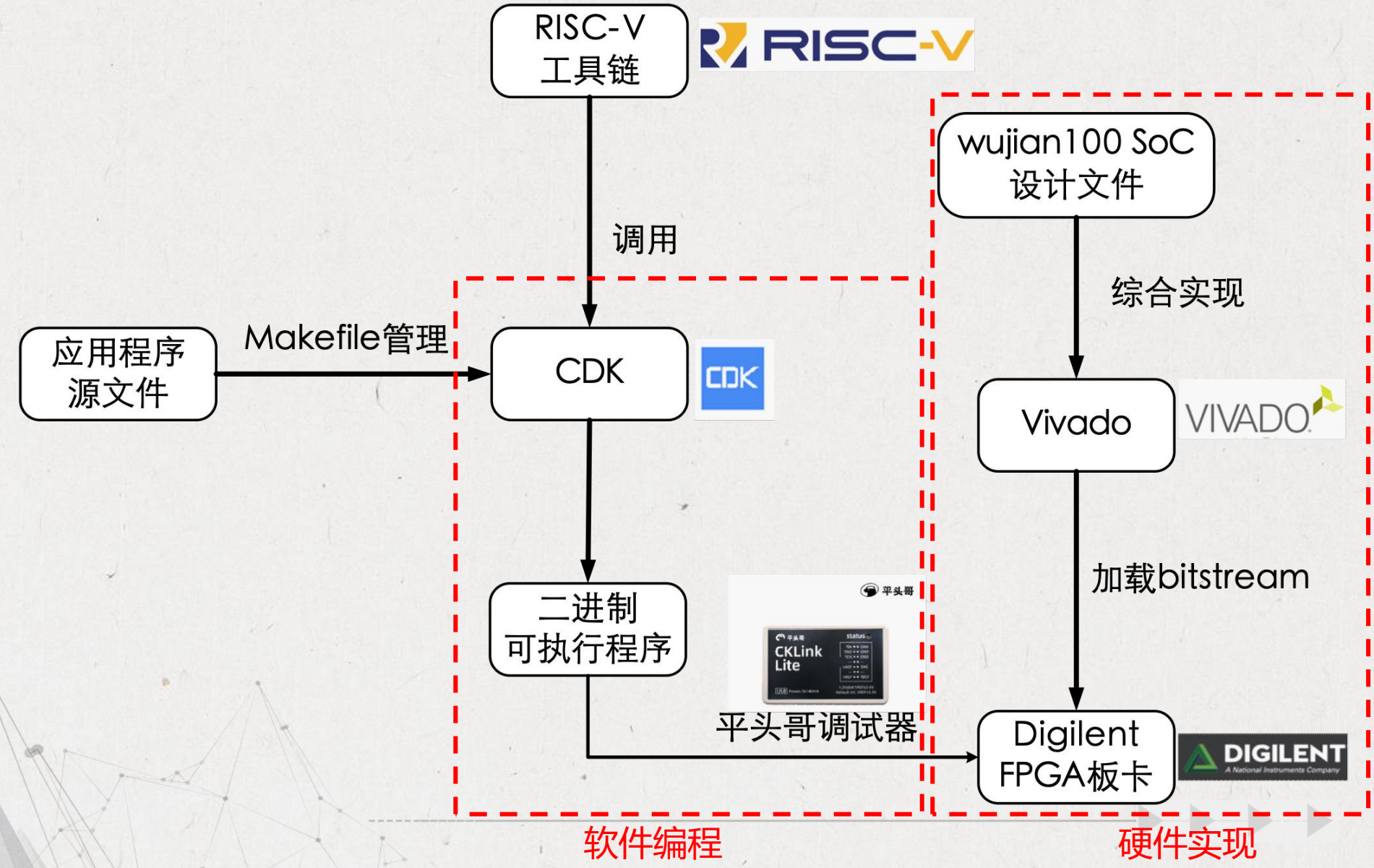
Vivado FPGA开发平台

包括代码编写、逻辑综合、设计实现、bit流文件生成、FPGA烧写等几个主要的步骤。Vivado IDE的“使用流程导航区”清晰地给出了各个环节及其先后执行顺序。





小节


RISC-V
工具链
RISC-V
调用
wujian100 SoC
设计文件
综合实现
应用程序
源文件
Makefile管理
CDK
CDK
Vivado
VIVADO
加载bitstream
二进制
可执行程序
CKLink
Lite
平头哥调试器
Digilent
FPGA板卡
DIGILENT
A National Instruments Company
软件编程
硬件实现