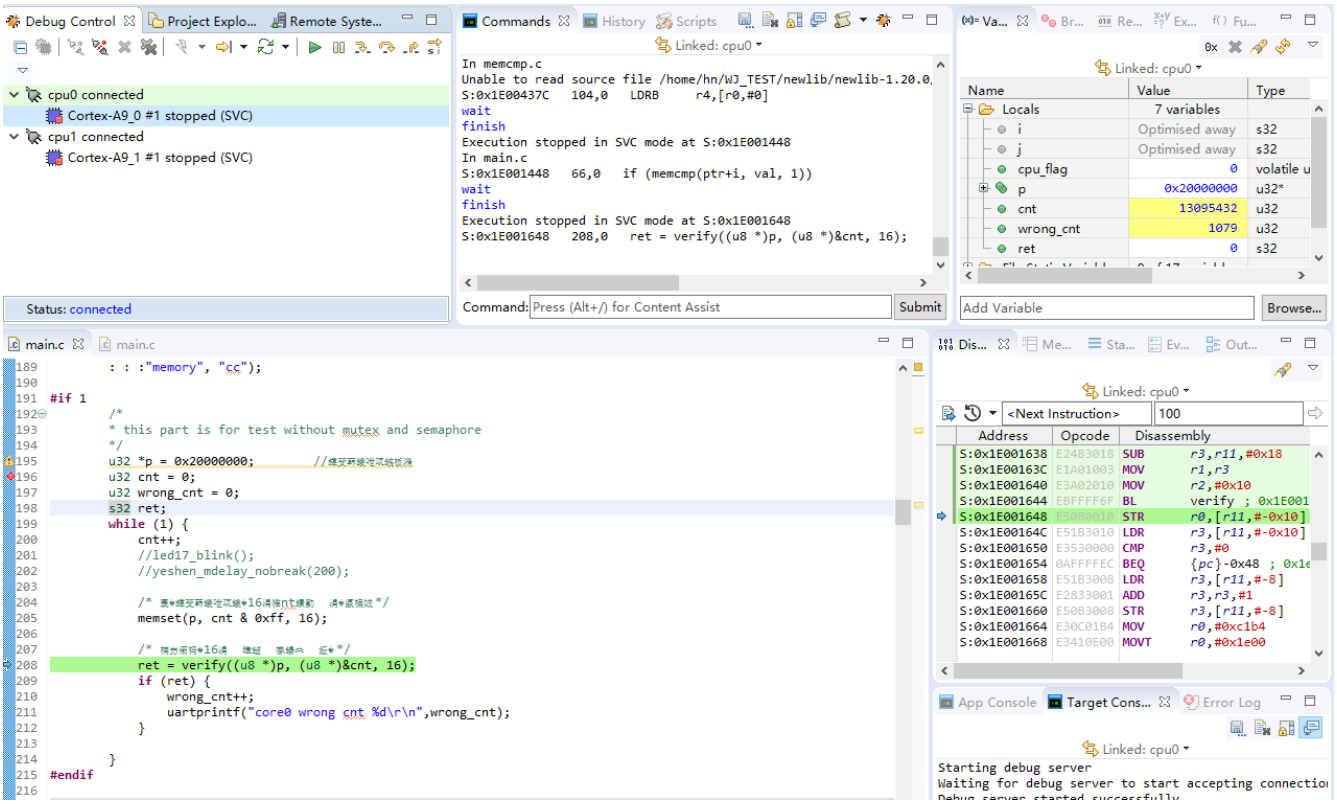
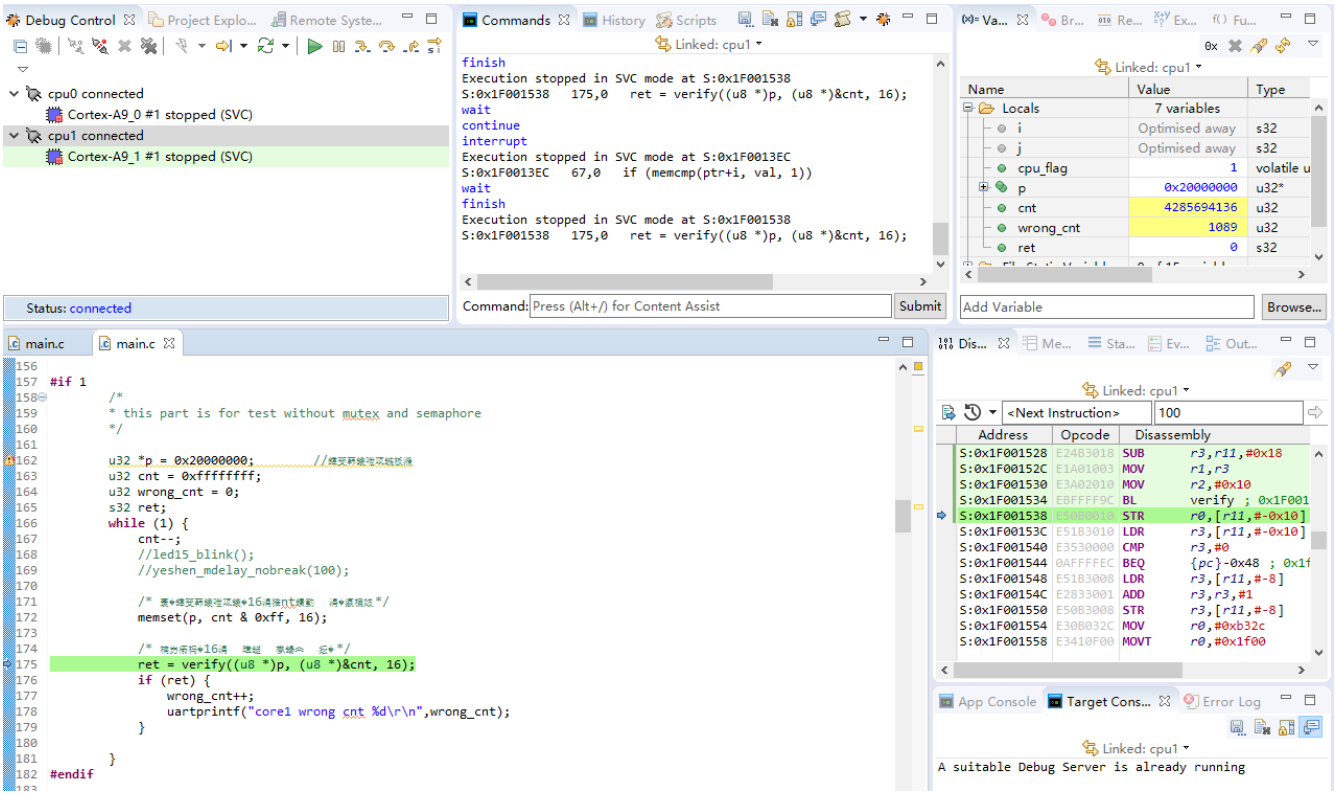
信号量测试

除了互斥锁外，信号量是另外一种对共享资源进行保护的机制。互斥锁其实是特殊的信号量，只有0和1，而一般信号量的值可以超过1。下面展示有无信号量的测试对比。

无信号量:

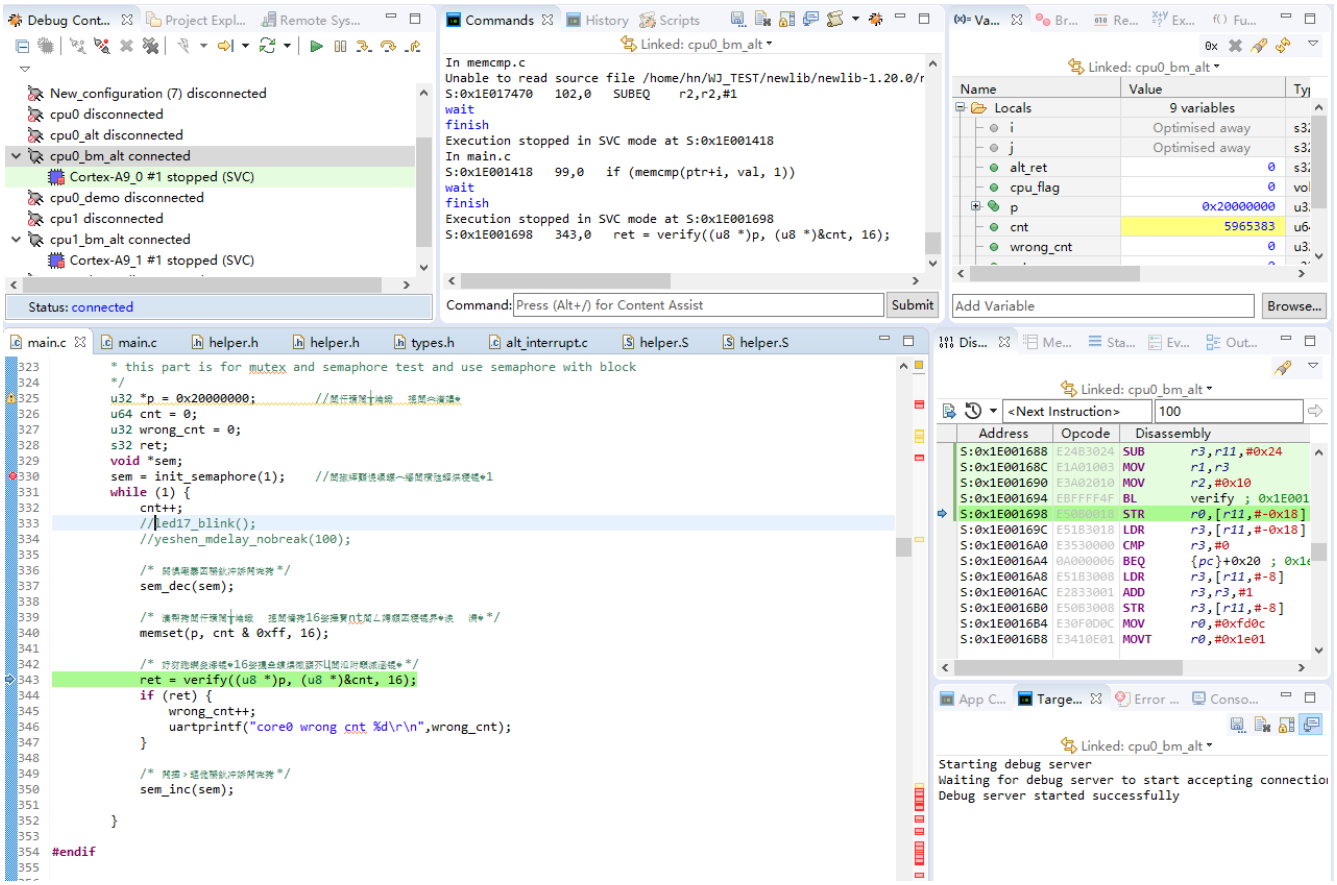


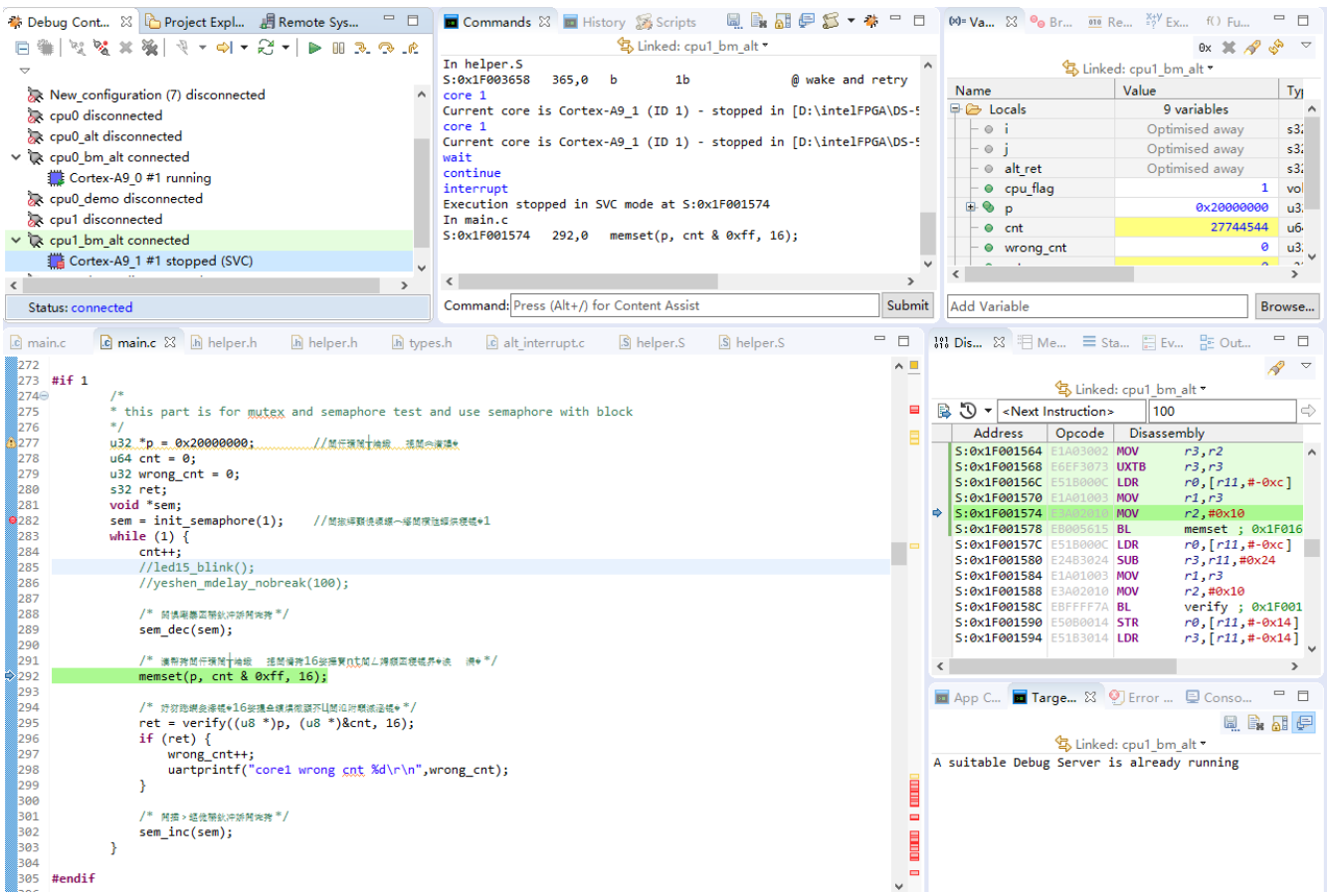


以上展示的是两个核在没有任何同步保护机制的情况下对同一块共享内存进行写入然后读取验证。wrong_cnt记录验证错误的次数，也就是读出来的值不是之前写进去的值的次数，从图中看到两个核在进行一定次数的测试后分别发生了1079次和1089次的验证数据错误，因此没有信号量等同步机制的情况下访问共享资源是不安全的。

有信号量:

阻塞版：

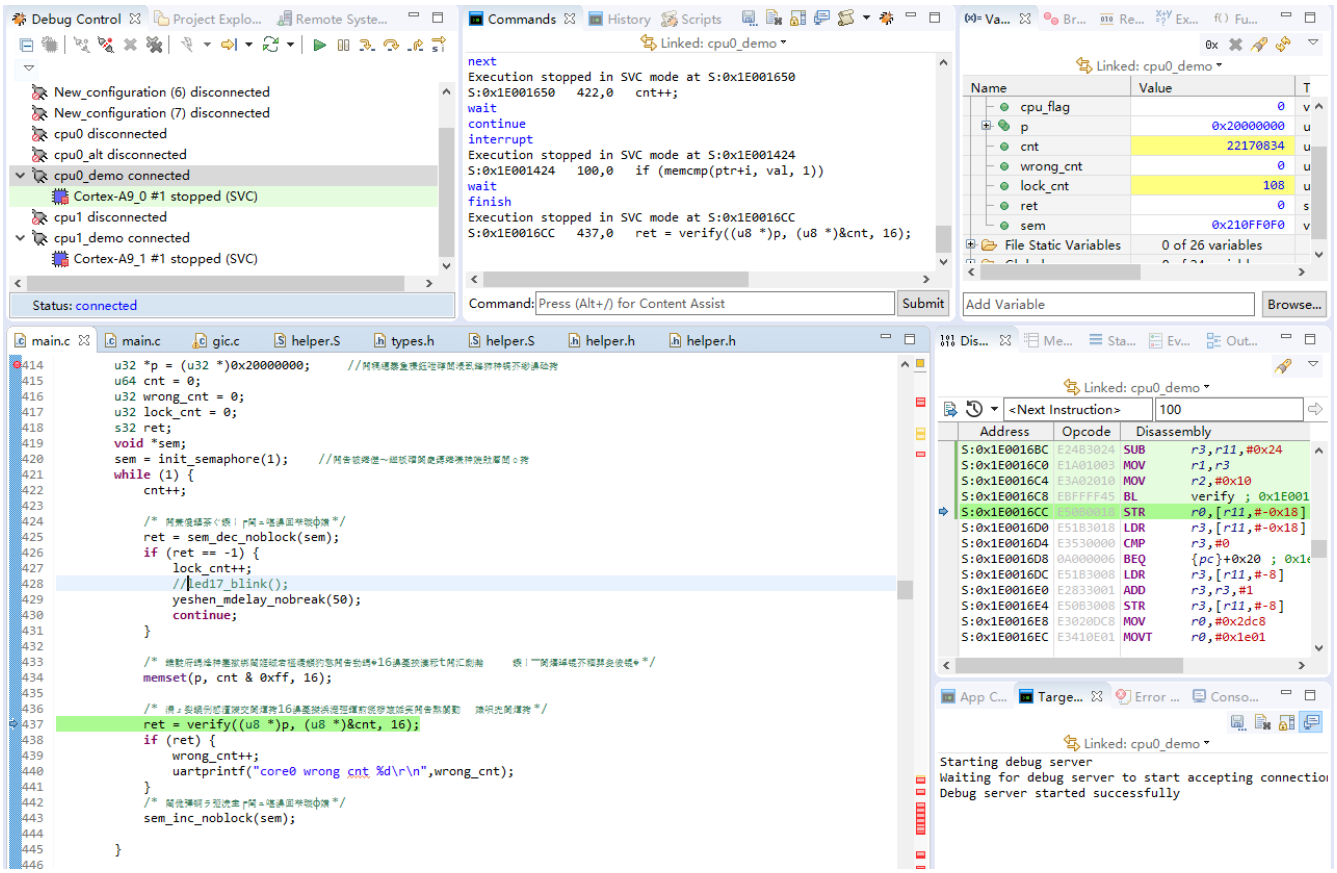

Debug Cont...
Project Expl...
Remote Sys...
New_configuration (7) disconnected
cpu0 disconnected
cpu0_alt disconnected
cpu0_bm_alt connected
Cortex-A9_0 #1 stopped (SVC)
cpu0_demo disconnected
cpu1 disconnected
cpu1_bm_alt connected
Cortex-A9_1 #1 stopped (SVC)
Status: connected
Commands
History
Scripts
Linked: cpu0_bm_alt
In memcmp.c
Unable to read source file /home/hn/WJ_TEST/newlib/newlib-1.20.0/r
S:0x1E017470 102,0 SUBEQ r2,r2,#1
wait
finish
Execution stopped in SVC mode at S:0x1E001418
In main.c
S:0x1E001418 99,0 if (memcmp(ptr+i, val, 1))
wait
finish
Execution stopped in SVC mode at S:0x1E001698
S:0x1E001698 343,0 ret = verify((u8 *)p, (u8 *)&cnt, 16);
Command:
Press (Alt+/) for Content Assist
Submit
Va...
Br...
Re...
Ex...
Fu...
Linked: cpu0_bm_alt
Name
Value
Locals
9 variables
i
Optimised away
j
Optimised away
alt_ret
0
cpu_flag
0
p
0x20000000
cnt
5965383
wrong_cnt
0
Add Variable
Browse...
main.c
helper.h
helper.h
types.h
alt_interrupt.c
helper.S
helper.S
* this part is for mutex and semaphore test and use semaphore with block
u32 *p = 0x20000000;
u64 cnt = 0;
u32 wrong_cnt = 0;
s32 ret;
void *sem;
sem = init_semaphore(1);
while (1) {
cnt++;
//led17_blink();
//yeshen_mdelay_nobreak(100);
sem_dec(sem);
memset(p, cnt & 0xff, 16);
ret = verify((u8 *)p, (u8 *)&cnt, 16);
if (ret) {
wrong_cnt++;
uartprintf("core0 wrong cnt %d\r\n",wrong_cnt);
}
sem_inc(sem);
}
#endif
Dis...
Me...
Sta...
Ev...
Out...
Linked: cpu0_bm_alt
<Next Instruction>
100
Address
Opcode
Disassembly
S:0x1E001688 E24B3024 SUB r3,r11,#0x24
S:0x1E00168C E1A01003 MOV r1,r3
S:0x1E001690 E3A02010 MOV r2,#0x10
S:0x1E001694 EBFFFF4F BL verify ; 0x1E001
S:0x1E001698 E50B0018 STR r0,[r11,#-0x18]
S:0x1E00169C E51B3018 LDR r3,[r11,#-0x18]
S:0x1E0016A0 E3530000 CMP r3,#0
S:0x1E0016A4 0A000006 BEQ {pc}+0x20 ; 0x1
S:0x1E0016A8 E51B3008 LDR r3,[r11,#-8]
S:0x1E0016AC E2833001 ADD r3,r3,#1
S:0x1E0016B0 E50B3008 STR r3,[r11,#-8]
S:0x1E0016B4 E30F0D0C MOV r0,#0xfd0c
S:0x1E0016B8 E3410E01 MOVT r0,#0x1e01
App C...
Targe...
Error ...
Conso...
Linked: cpu0_bm_alt
Starting debug server
Waiting for debug server to start accepting connectio
Debug server started successfully

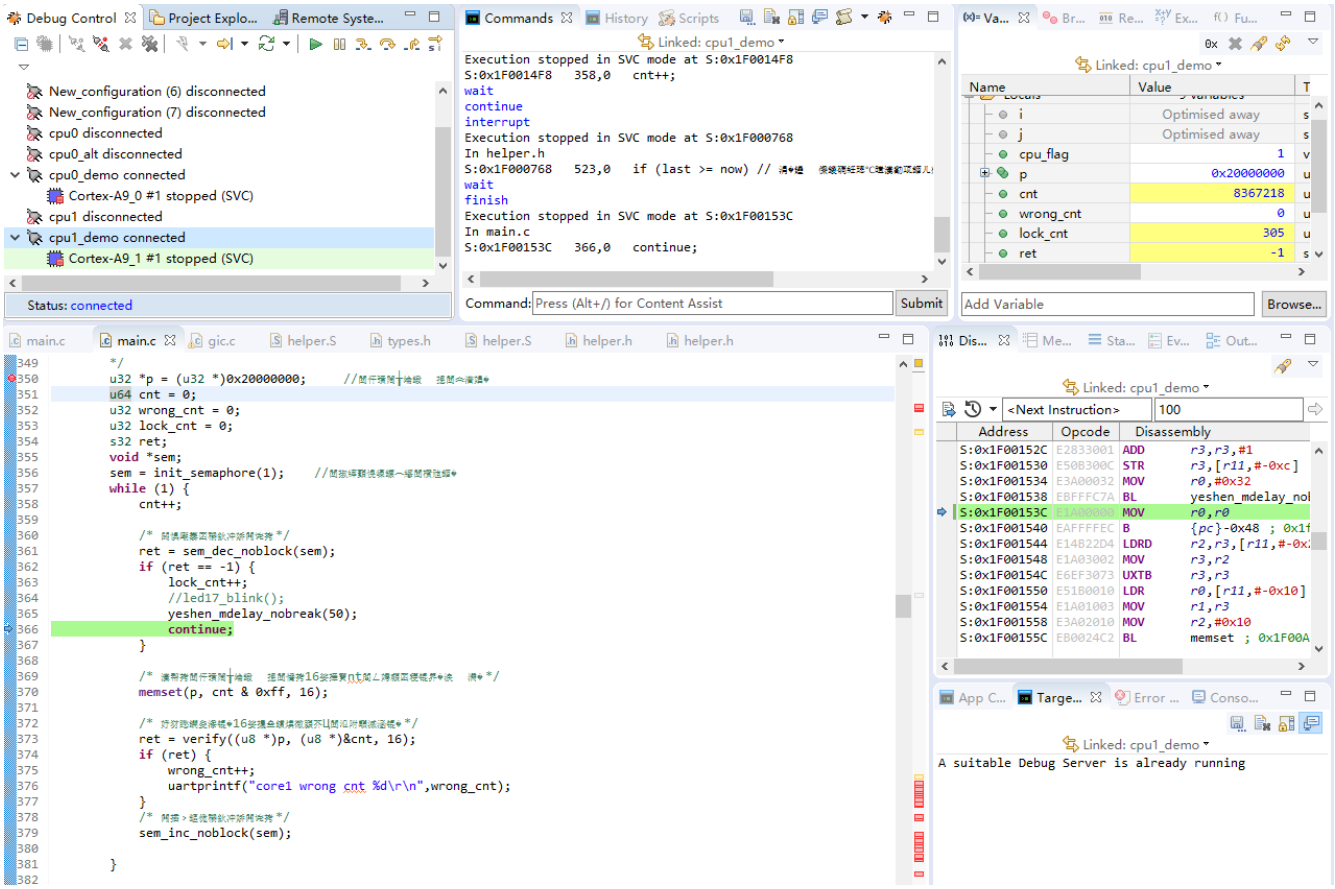

Debug Cont...
Project Expl...
Remote Sys...
New_configuration (7) disconnected
cpu0 disconnected
cpu0_alt disconnected
cpu0_bm_alt connected
Cortex-A9_0 #1 running
cpu0_demo disconnected
cpu1 disconnected
cpu1_bm_alt connected
Cortex-A9_1 #1 stopped (SVC)
Status: connected
Commands
History
Scripts
Linked: cpu1_bm_alt
In helper.S
S:0x1F003658 365,0 b 1b @ wake and retry
core 1
Current core is Cortex-A9_1 (ID 1) - stopped in [D:\intelFPGA\DS-5
core 1
Current core is Cortex-A9_1 (ID 1) - stopped in [D:\intelFPGA\DS-5
wait
continue
interrupt
Execution stopped in SVC mode at S:0x1F001574
In main.c
S:0x1F001574 292,0 memset(p, cnt & 0xff, 16);
Command:
Press (Alt+/) for Content Assist
Submit
Va...
Br...
Re...
Ex...
Fu...
Linked: cpu1_bm_alt
Name
Value
Locals
9 variables
i
Optimised away
j
Optimised away
alt_ret
Optimised away
cpu_flag
1
p
0x20000000
cnt
27744544
wrong_cnt
0
Add Variable
Browse...
main.c
helper.h
helper.h
types.h
alt_interrupt.c
helper.S
helper.S
#if 1
* this part is for mutex and semaphore test and use semaphore with block
u32 *p = 0x20000000;
u64 cnt = 0;
u32 wrong_cnt = 0;
s32 ret;
void *sem;
sem = init_semaphore(1);
while (1) {
cnt++;
//led15_blink();
//yeshen_mdelay_nobreak(100);
sem_dec(sem);
memset(p, cnt & 0xff, 16);
ret = verify((u8 *)p, (u8 *)&cnt, 16);
if (ret) {
wrong_cnt++;
uartprintf("core1 wrong cnt %d\r\n",wrong_cnt);
}
sem_inc(sem);
}
#endif
Dis...
Me...
Sta...
Ev...
Out...
Linked: cpu1_bm_alt
<Next Instruction>
100
Address
Opcode
Disassembly
S:0x1F001564 E1A03002 MOV r3,r2
S:0x1F001568 E6EF3073 UXTB r3,r3
S:0x1F00156C E51B000C LDR r0,[r11,#-0xc]
S:0x1F001570 E1A01003 MOV r1,r3
S:0x1F001574 E3A02010 MOV r2,#0x10
S:0x1F001578 EB005615 BL memset ; 0x1F016
S:0x1F00157C E51B000C LDR r0,[r11,#-0xc]
S:0x1F001580 E24B3024 SUB r3,r11,#0x24
S:0x1F001584 E1A01003 MOV r1,r3
S:0x1F001588 E3A02010 MOV r2,#0x10
S:0x1F00158C EBFFFF7A BL verify ; 0x1F001
S:0x1F001590 E50B0014 STR r0,[r11,#-0x14]
S:0x1F001594 E51B3014 LDR r3,[r11,#-0x14]
App C...
Targe...
Error ...
Conso...
Linked: cpu1_bm_alt
A suitable Debug Server is already running

上图是两个核利用信号量对共享内存进行读写的测试，每个核在操作共享内存时都要通过init_semaphore初始化信号量，传入的参数为初始化的信号量的值，第一个调用init_semaphore的线程成功的初始化信号量，第二个调用init_semaphore的线程无法初始化信号量为传入的参数值，只是单纯返回信号量地址，信号量值由第一个线程是否已经释放信号量来决定。每个核在操作共享内存时候通过sem_dec将信号量减1，操作完后通过sem_inc将信号量加1，当信号量大于0时，sem_dec才能成功返回并将信号量减1，否则一直阻塞直到信号量大于0。从图中可以发现两个核的测试中的wrong_cnt一直为0，即一直没有出错，因此本文实现的信号量能很好的保护双核通信中共享资源的访问。

非阻塞版：

上诉版本的信号量是阻塞版的，就是当本线程调用sem_dec获取信号量时发现信号量等于0的时候会阻塞在临界区，直到信号量被其他线程通过sem_inc加1后sem_dec获取了信号量并减1后才能返回访问共享资源。在这阻塞期间本线程什么都不能做，本文提供了另外一种非阻塞方案，可以让用户选择在信号量不大于0被占用的情况下是否做其他工作。





非阻塞版和阻塞版其中行为不同的只是获取并将信号量减1的操作，初始化和释放信号量的行为一致。而获取信号量并减1在不成功的时候直接返回-1,表示信号量已经为0，返回0表示获取信号量并减1成功，用户可以根据返回值作出相应的动作，不必一直阻塞直到信号量值大于0。从图中看到两个核的测试中wrong_cnt也是一直为0，说明共享资源访问没有冲突，而另外用一个lock_cnt的变量记录获取信号量时不大于0的次数，两个核分别有108和305次申请获取信号量时信号量不大于0。

总结

在没有同步机制保护的情况下双核同时访问共享资源会导致数据错乱冲突，在有信号量保护的情况下双核可以同时访问共享资源而不发生冲突，且根据应用的需求提供了阻塞版和非阻塞版两种信号量保护机制。