ggghdl的解题思路

题目信息

题目名	类型	难度
ggghdl	RE	困难

FLAG

- SUSCTF{40a339d4-f940-4fe0-b382-cabb310d2ead}

知识点

1. 类比推理
2. GHDL

解题步骤

本题使用VHDL硬件设计语言编写了一个简单的异或模块，以及一个顶层仿真模块，实现对Flag的异或，并与答案进行比较。

VHDL文件使用GHDL-LLVM进行编译出Native代码，题目名称给出了ghdl的提示，做题者通过阅读文档可以了解到，可以使用LLVM或GCC的Backend方案编译Native代码，为方便查看，WP为未strip的版本

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  return ghdl_main((unsigned int)argc, argv);
}
```

本题还会给出提示：“试着自己写一个Hello World试试？”，旨在考查做题者的逆向实力，通过工具集编写一个简易模块推理复杂模块行为的能力。此外，逆向就是在于打信息差，VHDL的硬件设计语言天生不属于常规解题者知识范围，还考察解题者学习能力。

本题不套娃，算法极其简单。

本题给人看上去像一个虚拟机

```
__int64 __fastcall ghdl_main(__int64 a1, __int64 *a2)
{
  __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = 0LL;
  grt_init(a1);
  if ( (_DWORD)a1 != 0 || a2 != 0LL )
  {
    if ( !a2 )
      _gnat_last_chance_handler("ghdl_main.adb", 53LL);
    v3 = *a2;
  }
  grt_main_options(v3, (unsigned int)a1, a2);
  grt__main__run();
  return grt__errors__exit_status;
}
```

可以根据Input Flag提示搜索字符串定位一个函数先看看

```
71  __int64 v70; // [rsp+190h] [rbp-38h]
72
73  v2 = *(_DWORD *)(a1 + 276);
74  v3 = "hello.vhdl";
75  v35 = &work__hello_world__ARCH__behaviour__xi__OT__STL[8];
76  v34 = (__int64)&work__hello_world__ARCH__behaviour__P0__ans__OT__STL + 8;
77  while ( 1 )
78  {
79    switch ( v2 )
80    {
81      case 0:
82        *(_DWORD *)(a1 + 272) = 0;
83        v4 = _ghdl_stack2_mark();
84        v57[0] = *(_QWORD *)(a1 + 256);
85        v69[0] = (__int64)"Input Flag";
86        v69[1] = (__int64)&unk_DC548;
87        v57[1] = (__int64)v69;
0002DCE0 work__hello_world__ARCH__behaviour__P0__PROC:44 (2DCE0)
```

可以看出是一个状态机，本质上逻辑电路基本上都是状态机。

```
readline (input, in_line);
if in_line'length < 44 then
  write (out_line, String'("Wrong!"));
  writeline (output, out_line);
  wait;
end if;
for i in 1 to 44 loop
```

首先判断长度

```
case 8:
  if ( *(_DWORD *)(a1 + 376) == 44 )
  {
    v14 = sub_7FF643F4EB56();
    v50[0] = *(_QWORD *)(a1 + 352);
    v70[0] = (__int64)&unk_7FF6440061A8;
    v70[1] = (__int64)&unk_7FF644006198;
    v50[1] = (__int64)v70;
    v51 = 0;
    v52 = 0;
    sub_7FF643F17FD0(v50);
    *(_QWORD *)(a1 + 352) = v50[0];
    sub_7FF643F4EBD4(v14);
    v15 = sub_7FF643F4EB56();
    v48 = dword_7FF644028064;
    v49 = *(_QWORD *)(a1 + 352);
    sub_7FF643F17670(&v48);
    v16 = v49;
  }
  else
  {
    v31 = sub_7FF643F4EB56();
    v45[0] = *(_QWORD *)(a1 + 352);
    v69[0] = (__int64)&unk_7FF6440061C8;
    v69[1] = (__int64)&unk_7FF6440061B8;
    v45[1] = (__int64)v69;
    v46 = 0;
    v47 = 0;
    sub_7FF643F17FD0(v45);
    *(_QWORD *)(a1 + 352) = v45[0];
    sub_7FF643F4EBD4(v31);
    v15 = sub_7FF643F4EB56();
    v43 = dword_7FF644028064;
    v44 = *(_QWORD *)(a1 + 352);
    sub_7FF643F17670(&v43);
    v16 = v44;
  }
  *(_QWORD *)(a1 + 352) = v16;
```

这里是求异或，每次for循环都走一遍状态机

```
for i in 1 to 44 loop
  read (in_line, c);
  c_int := character'pos(c);
  xi1 <= std_logic_vector(to_unsigned(c_int, 8));
  wait for 1 ns;
  xi2 <= std_logic_vector(to_unsigned(a2(i - 1), 8));
  wait for 1 ns;
  if xo = std_logic_vector(to_unsigned(a1(i - 1), 8)) then
    correct := correct + 1;
  end if;
end loop;
```

```
case 1:
  *(_QWORD *)(a1 + 280) = 0x2C00000001LL;
  v2 = 6;
  continue;
```

上面是初始化循环的结束，显然0x2C是44，其余的是程序额外状态

```
case 6:
  v19 = _ghdl_stack2_mark();
  v32 = *(_QWORD *)(a1 + 240);
  v33 = 0;
  std__textio__read08((__int64)&v32);
  *(_QWORD *)(a1 + 240) = v32;
  *(_BYTE *)(a1 + 264) = v33;
  _ghdl_stack2_release(v19);
  *(_DWORD *)(a1 + 268) = *(unsigned __int8 *)(a1 + 264);
  v20 = _ghdl_stack2_mark();
  v21 = *(_DWORD *)(a1 + 268);
  if ( v21 < 0 )
    _ghdl_bound_check_failed("hello.vhdl", 43LL);
  ieee__numeric_std__to_unsigned(&v66, (unsigned int)v21, 8LL);
  v46 = v66;
  v47 = v29;
  v29[0] = *(_DWORD *)v67;
  v29[1] = *(_DWORD *)(v67 + 4);
  v30 = *(_BYTE *)(v67 + 8);
  v31 = *(_DWORD *)(v67 + 12);
  if ( v31 != 8 )
    _ghdl_bound_check_failed("hello.vhdl", 43LL);
  v22 = v46;
  for ( i = 0LL; (unsigned int)i <= 7; ++i )
  {
    v24 = *(_BYTE **)(a1 + 8 * i + 16);
    *(_BYTE *)(a1 + i + 288) = *(_BYTE *)(v22 + i);
    v25 = 1;
    if ( !v24[42] )
      v25 = *v24 != *(_BYTE *)(a1 + i + 288);
    if ( v25 )
      _ghdl_signal_direct_assign((__int64)v24);
  }
  _ghdl_stack2_release(v20);
  result = _ghdl_process_wait_timeout(&unk_F4240, "hello.vhdl", 44LL);
  *(_DWORD *)(a1 + 276) = 7;
  return result;
```

先是direct_assign赋值给xor模块，这里会return，因为要去跑xor的模块，再次进入会去状态7

```
    sub_7FF643F688D2("hello.vhdl", 48i64);
  v23 = v55;
  for ( i = 0i64; (unsigned int)i <= 7; ++i )
  {
    v25 = *(_BYTE **)(a1 + 8 * i + 16);
    *(_BYTE *)(a1 + i + 392) = *(_BYTE *)(v23 + i);
    v26 = 1;
    if ( !v25[42] )
      v26 = *v25 != *(_BYTE *)(a1 + i + 392);
    if ( v26 )
      sub_7FF643F55B00();
  }
  sub_7FF643F4EBD4(v18);
  result = sub_7FF643F4F80C(1000000i64, "hello.vhdl", 49i64);
  *(_DWORD *)(a1 + 380) = 10;
  return result;
```

```
    return result;
  case 10:
    v20 = sub_7FF643F4EB56();
    v21 = *(_DWORD *)(a1 + 384) - 1;
    if ( v21 >= 0x2C )
      sub_7FF643F68AD0("hello.vhdl", 50i64, v21, &unk_7FF644005C38);
    v22 = dword_7FF644005C50[v21];
    if ( v22 < 0 )
      sub_7FF643F688D2("hello.vhdl", 50i64);
    sub_7FF643F40180(&v75, (unsigned int)v22, 8i64);
    v53 = v75;
    v54 = v35;
    v35[0] = *(_DWORD *)v76;
    v35[1] = *(_DWORD *)(v76 + 4);
    v36 = *(_BYTE *)(v76 + 8);
    v37 = *(_DWORD *)(v76 + 12);
    if ( v37 != 8 )
      sub_7FF643F688D2("hello.vhdl", 50i64);
    v27 = v53;
    for ( j = 0i64; (unsigned int)j <= 7; ++j )
    {
      v29 = *(_BYTE **)(a1 + 8 * j + 88);
      *(_BYTE *)(a1 + j + 400) = *(_BYTE *)(v27 + j);
      v30 = 1;
      if ( !v29[42] )
        v30 = *v29 != *(_BYTE *)(a1 + j + 400);
      if ( v30 )
        sub_7FF643F55B00();
    }
```

这里是读取xor的输出信号，和答案比较，如果正确，correct计数+1

```
  case 11:
    v74[0] = v3;
    v74[1] = (__int64)&unk_7FF644005AB8;
    v4 = sub_7FF643F4EB56();
    v5 = *(_DWORD *)(a1 + 384) - 1;
    if ( v5 >= 0x2C )
      sub_7FF643F68AD0("hello.vhdl", 52i64, v5, &unk_7FF644005B68);
    v6 = dword_7FF644005B80[v5];
    if ( v6 < 0 )
      sub_7FF643F688D2("hello.vhdl", 52i64);
    sub_7FF643F40180(&v72, (unsigned int)v6, 8i64);
    v71[0] = v72;
    v71[1] = (__int64)v32;
    v32[0] = *(_DWORD *)v73;
    v32[1] = *(_DWORD *)(v73 + 4);
    v33 = *(_BYTE *)(v73 + 8);
    v34 = *(_DWORD *)(v73 + 12);
    v7 = sub_7FF643F19CC0(v74, v71);
    sub_7FF643F4EBD4(v4);
    v2 = 7;
    if ( (v7 & 1) != 0 )
      ++*(_DWORD *)(a1 + 376);
    break;
```

最后，根据correct是否为44判断正确

```
if correct = 44 then
  write (out_line, String'("C0rReCt!"));
  writeline (output, out_line);
else
  write (out_line, String'("Nah"));
  writeline (output, out_line);
end if;
wait;
```

```
    case 5:
      v12 = sub_7FF643F4EB56();
      v59[0] = *(_QWORD *)(a1 + 352);
      v79[0] = (__int64)"Wrong!";
      v79[1] = (__int64)&unk_7FF644006178;
      v59[1] = (__int64)v79;
      v60 = 0;
      v61 = 0;
      sub_7FF643F17FD0(v59);
      *(_QWORD *)(a1 + 352) = v59[0];
      sub_7FF643F4EBD4(v12);
      v13 = sub_7FF643F4EB56();
      v57 = dword_7FF644028064;
      v58 = *(_QWORD *)(a1 + 352);
      sub_7FF643F17670(&v57);
      *(_QWORD *)(a1 + 352) = v58;
      sub_7FF643F4EBD4(v13);
      result = sub_7FF643F4F782();
      *(_DWORD *)(a1 + 380) = 6;
      return result;
    case 6:
```

做题者参照GHDL文档的官方例程，可以通过自己写一份包含for循环和子模块的小程序，从而分析以上行为。

接下来，需要分析xor的值，以及答案，看异或部分有一个数组

```
    case 10:
      v20 = sub_7FF643F4EB56();
      v21 = *(_DWORD *)(a1 + 384) - 1;
      if ( v21 >= 0x2C )
        sub_7FF643F68AD0("hello.vhdl", 50i64, v21, &unk_7FF644005C38);
      v22 = dword_7FF644005C50[v21];
      if ( v22 < 0 )
        sub_7FF643F688D2("hello.vhdl", 50i64);
```

```
.rdata:00007FF644005C4F                 db    0
.rdata:00007FF644005C50 ; _DWORD dword_7FF644005C50[48]
.rdata:00007FF644005C50 dword_7FF644005C50 dd 56h, 0DAh, 0CDh, 3Ah, 7Eh, 86h, 13h, 0B5h, 1Dh, 9Dh
.rdata:00007FF644005C50                                         ; DATA XREF: sub_7FF643F4CAB0+654↑o
.rdata:00007FF644005C50                                         ; .rdata:00007FF644006050↓o
.rdata:00007FF644005C50                 dd 0FCh, 97h, 8Ch, 31h, 6Bh, 0C9h, 0FBh, 1Ah, 0E2h, 2Dh
.rdata:00007FF644005C50                 dd 0DCh, 0D3h, 0F1h, 0F4h, 36h, 9, 20h, 42h, 4, 6Ah, 71h
.rdata:00007FF644005C50                 dd 53h, 78h, 0A4h, 97h, 8Fh, 7Ah, 72h, 39h, 0E8h, 3Dh
.rdata:00007FF644005C50                 dd 0FAh, 40h, 3Dh, 198h, 3 dup(0)
.rdata:00007FF644005D10 unk_7FF644005D10 db    5                ; DATA XREF: sub_7FF643F4C580+C↑o
.rdata:00007FF644005D10                                         ; sub_7FF643F4C580+67↑o
```

```
.rdata:00007FF644005B7F                 db    0
.rdata:00007FF644005B80 ; _DWORD dword_7FF644005B80[44]
.rdata:00007FF644005B80 dword_7FF644005B80 dd 5, 8Fh, 9Eh, 79h, 2Ah, 0C0h, 68h, 81h, 2Dh, 0FCh, 0CFh
.rdata:00007FF644005B80                                         ; DATA XREF: sub_7FF643F4CAB0+152↑o
.rdata:00007FF644005B80                                         ; .rdata:00007FF644005FF0↓o
.rdata:00007FF644005B80                 dd 0A4h, 0B5h, 55h, 5Fh, 0E4h, 9Dh, 23h, 0D6h, 1Dh, 0F1h
.rdata:00007FF644005B80                 dd 0E7h, 97h, 91h, 6, 24h, 42h, 71h, 3Ch, 58h, 5Ch, 30h
.rdata:00007FF644005B80                 dd 19h, 0C6h, 0F5h, 0BCh, 4Bh, 42h, 5Dh, 0DAh, 58h, 9Bh
.rdata:00007FF644005B80                 dd 24h, 40h
.rdata:00007FF644005C30 unk_7FF644005C30 db 0B0h                ; DATA XREF: .rdata:00007FF644006088↓o
.rdata:00007FF644005C31                 db    0
.rdata:00007FF644005C32                 db    0
.rdata:00007FF644005C33                 db    0
```

找到这两个数组，异或后就是答案，不strip查找太简单，因此题目进行了strip。输入Flag即可

`SUSCTF{40a339d4-f940-4fe0-b382-cabb310d2ead}`