

Function Simulation

실습 준비

Function Simulation

실습준비

1. cd SoC1 ↵
2. cd SoC ↵
3. cd smkcow_make_s28 ↵
4. cd TOP ↵
5. cd SIM ↵
6. cd FUNCTION ↵

```
/home/ex_poly1/SoC2/SoC/smkcow_make_S28/TOP/SIM/FUNCTION
```

Function Simulation

파일 확인

1. cd SoC

```
[ex_poly1@npit FUNCTION]$ ll
total 180
drwxr-xr-x 2 ex_poly1 rnd 4096 Jan 3 2024 Verdi-SXLog
-rwxr-xr-x 1 ex_poly1 rnd 296 Oct 1 2022 clean.tcl
-rw-r--r-- 1 ex_poly1 rnd 225 Jul 21 2021 fsdb.tcl
-rwxr-xr-x 1 ex_poly1 rnd 49 Apr 15 2022 fsdb2saif.tcl
-rw-r--r-- 1 ex_poly1 rnd 83516 Nov 8 17:24 func_sim.history
-rw-r--r-- 1 ex_poly1 rnd 3616 Oct 2 2022 hello.hex
drwxr-xr-x 2 ex_poly1 rnd 4096 May 7 2020 logs
-rwxr-xr-x 1 ex_poly1 rnd 4422 Nov 4 20:24 run_function.tcl
-rwxr-xr-x 1 ex_poly1 rnd 4336 Oct 22 00:31 run_function.tcl_ori
-rw-r--r-- 1 ex_poly1 rnd 21340 Jul 25 2022 session.inter.vpd.tcl
-rwxr-xr-x 1 ex_poly1 rnd 450 Jan 2 2024 simv_function.tcl
drwxr-xr-x 2 ex_poly1 rnd 52 Nov 6 23:58 waves.shm
drwxr-xr-x 6 ex_poly1 rnd 214 Nov 8 17:24 xcelium.d
-rw-r--r-- 1 ex_poly1 rnd 23624 Nov 8 17:55 xmprof.out
-rw-r--r-- 1 ex_poly1 rnd 5 Nov 8 17:55 xrun.key
```

FUNCTION SIMULATION

Function Simulation

Function 시뮬레이션의 목적

1. Verilog 소스를 시뮬레이션을 통해 기능 검증하는 단계
2. hex 코드가 rom으로 잘 다운로드 되는지 확인
3. 검증 완료 후 Hello World와 smkcow를 출력시킴

시스템 구조

-
- Crystal Oscillator
- XTAL1
- NRST
- 네거티브 리셋
Negative reset
1->0 될 때 리셋
- ICG
- Cortex M0
- AHB
- GPIO0
- GPIO1
- Pin mux
- AHB to APB
- UART1
- UART2
- Timer
- UART capture
- ROM
- SRAM
- P0, P1은 io셀
0~15bit로 각각
16bit로 총 32bit
- P0 [15:0]
- P1 [15:0]
- UART capture
UART2와 통신

Function Simulation

Hex코드

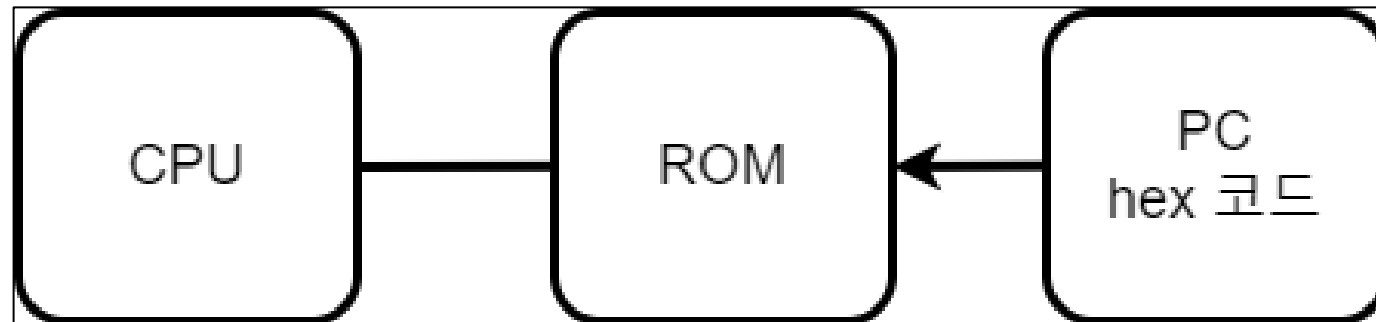
1. C 코드가 CPU로 이동하는 과정

- I. C코드는 컴파일러를 통해 hex로 변환됨(Binary code)
- II. hex 코드를 ROM에 전달

2. CPU의 코드 처리 방식

- **Little Endian:** 모바일 시장 – ARM (hex 코드를 거꾸로 입력하는 방식)
- **Big Endian:** PC 시장(hex 코드를 있는 그대로 입력하는 방식)

```
86 //-----  
87 // AHB_ROM_BEH_MODEL : Simple behavioral model (default)  
88  
89 generate if (MEM_TYPE == `AHB_ROM_BEH_MODEL) begin  
90     // Behavioral memory model  
91     cmsdk_ahb_rom_beh  
92     #(.AW(AW),  
93       .filename(filename),  
94       .WS_N(WS_N),  
95       .WS_S(WS_S)  
96     )
```



Function Simulation

시뮬레이션

1. ./clean.tcl ←

2. ./run_function.tcl ←

```
[ex_poly1@npit FUNCTION]$ ./clean.tcl
[ex_poly1@npit FUNCTION]$ ./run_function.tcl
```

• xrun 실행

```
xrun -64bit \
    +max_err_count+50 \
    +define+function_sim \
    -access +rwc \
    -profile \
    -profthread \
    -gui \
    +libext+.v \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_timer/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_dualtimers/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_uart/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_watchdog/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_slave_mux/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_apb_subsystem/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_ahb_slave_mux/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_ahb_default_slave/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_ahb_gpio/verilog \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_ahb_to_apb/verilog \
    -incdir ../../../../cortexm0_designstart/logical/models/clkgate \
    -incdir ../../../../cortexm0_designstart/logical/cmsdk_iop_gpio/verilog \
    -incdir ../../../../cortexm0_designstart/cores/cortexm0_designstart_rlp0/logical/cortexm0_integration/verilog \
```

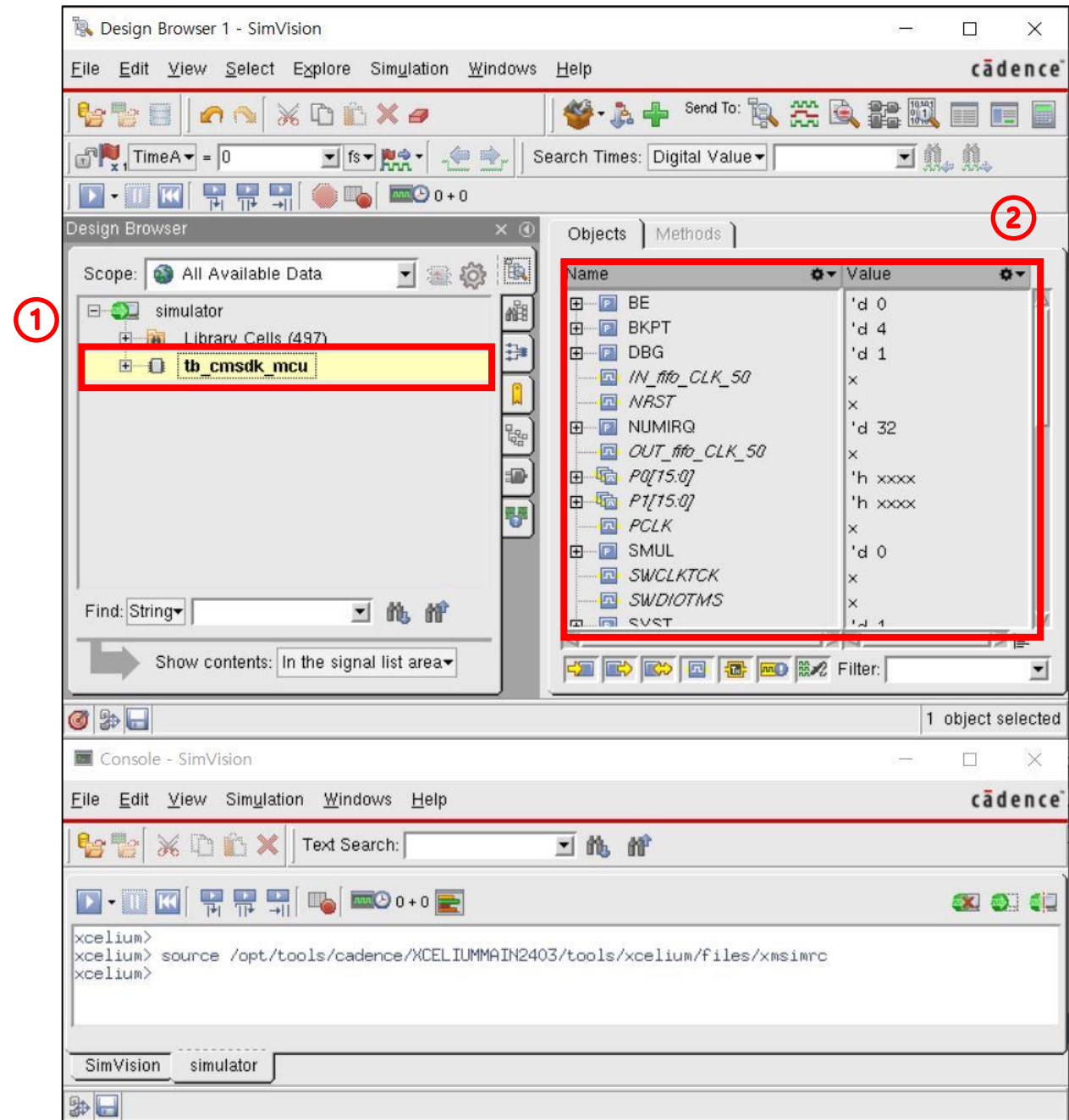
Function Simulation

시뮬레이션

1. xrun 실행 결과

① tb_cmsdk_mcu 클릭

② 포트 확인



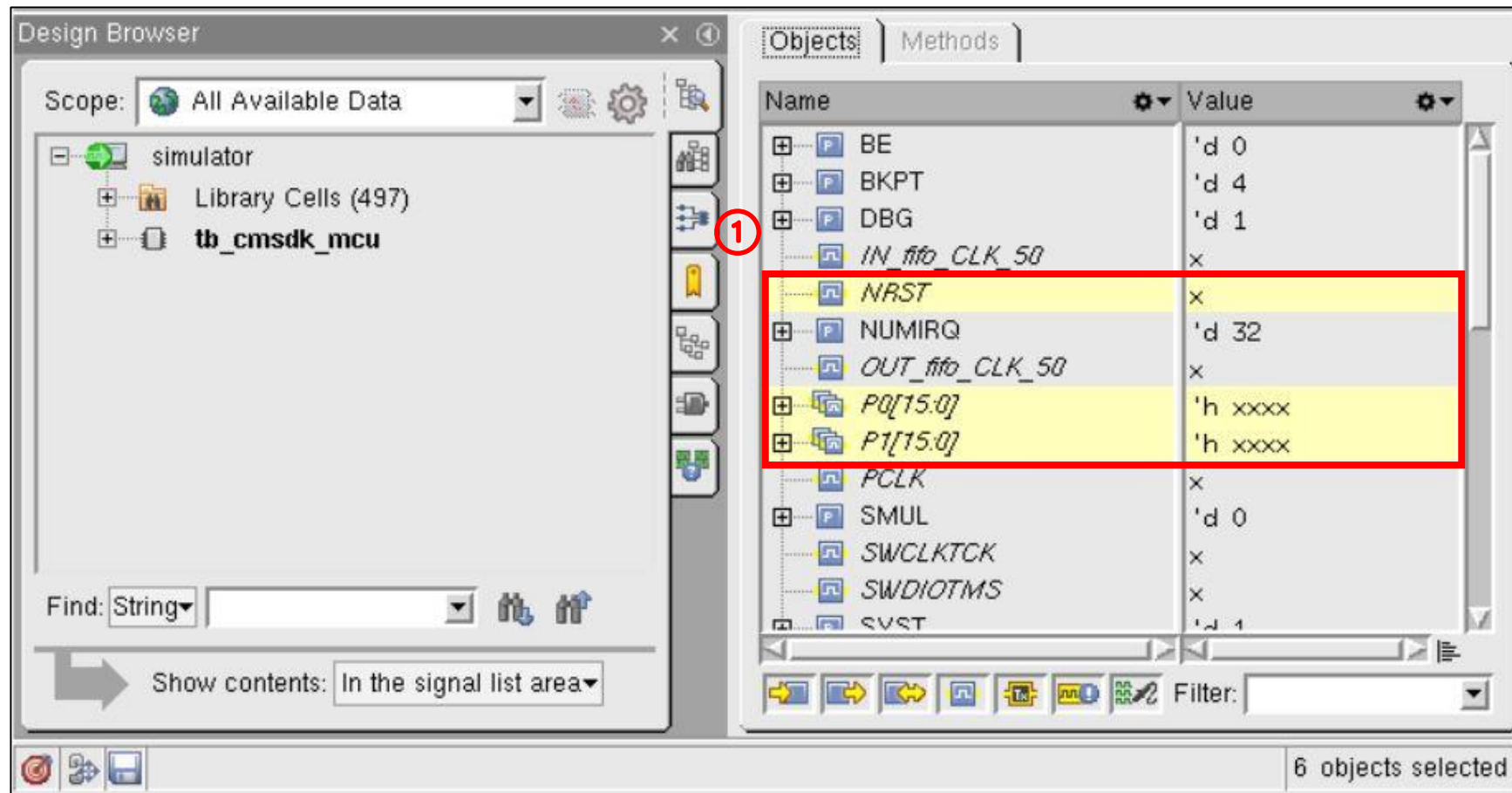
Function Simulation

시뮬레이션

1. xrun 실행 결과

① 시뮬레이션에서 확인할 포트 선택

(XTAL1, NRST, P0, P1, srx, stx)



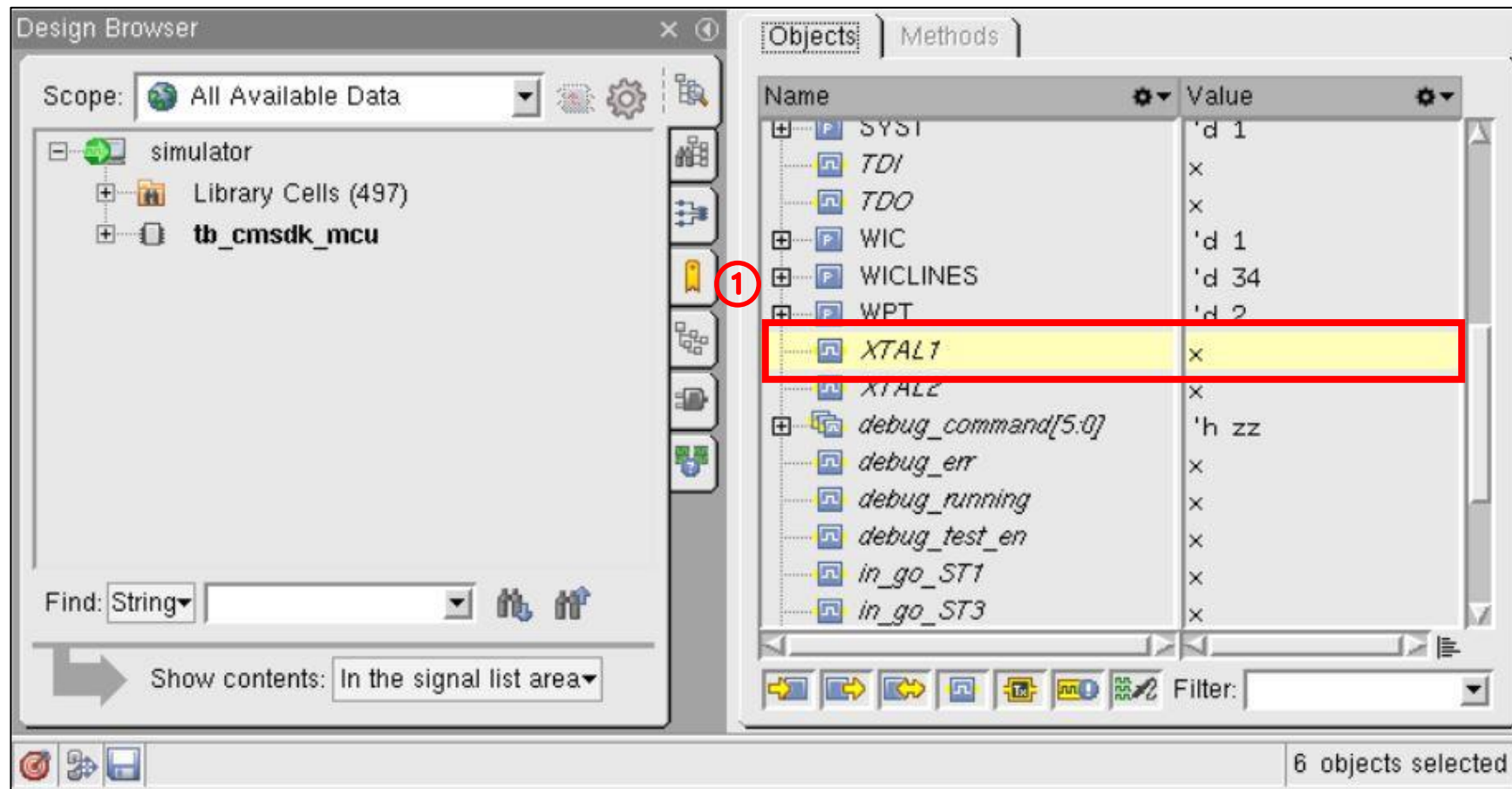
Function Simulation

시뮬레이션

1. xrun 실행 결과

① 시뮬레이션에서 확인할 포트 선택

(XTAL1, NRST, P0, P1, srx, stx)



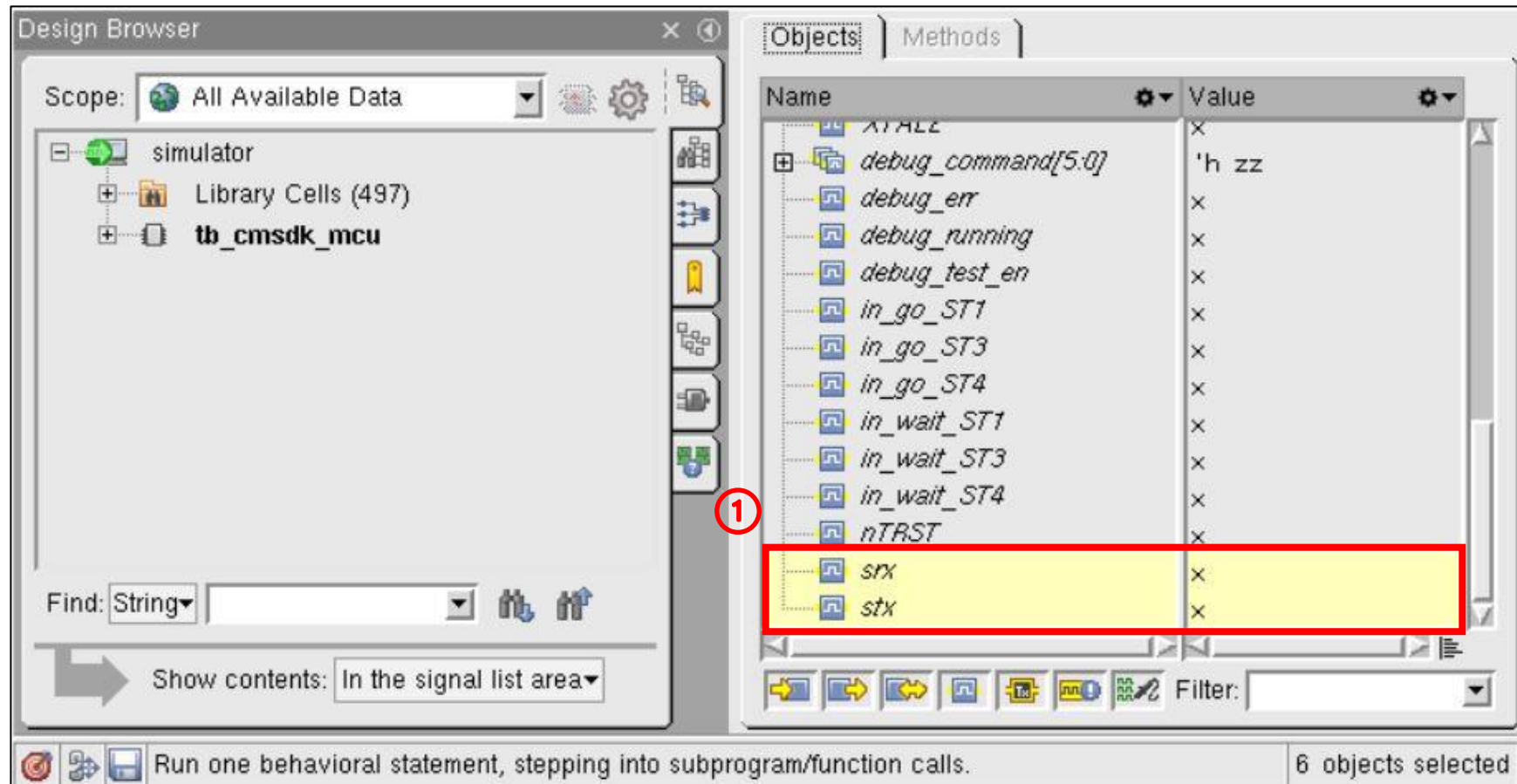
Function Simulation

시뮬레이션

1. xrun 실행 결과

① 시뮬레이션에서 확인할 포트
선택

(XTAL1, NRST, P0, P1, srx, stx)

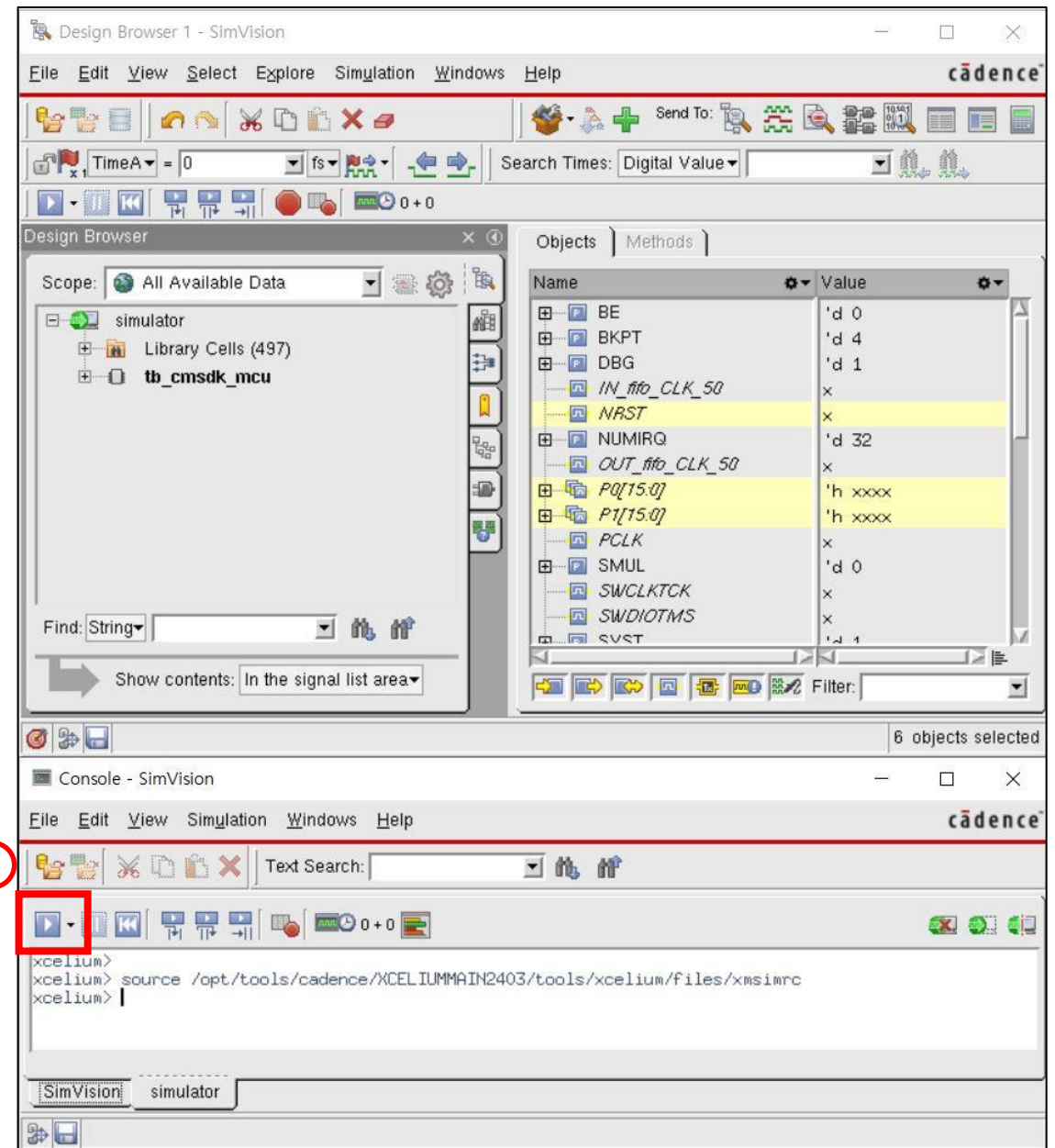


Function Simulation

시뮬레이션

1. xrun 실행 결과

① 모두 선택 후 실행 버튼 클릭



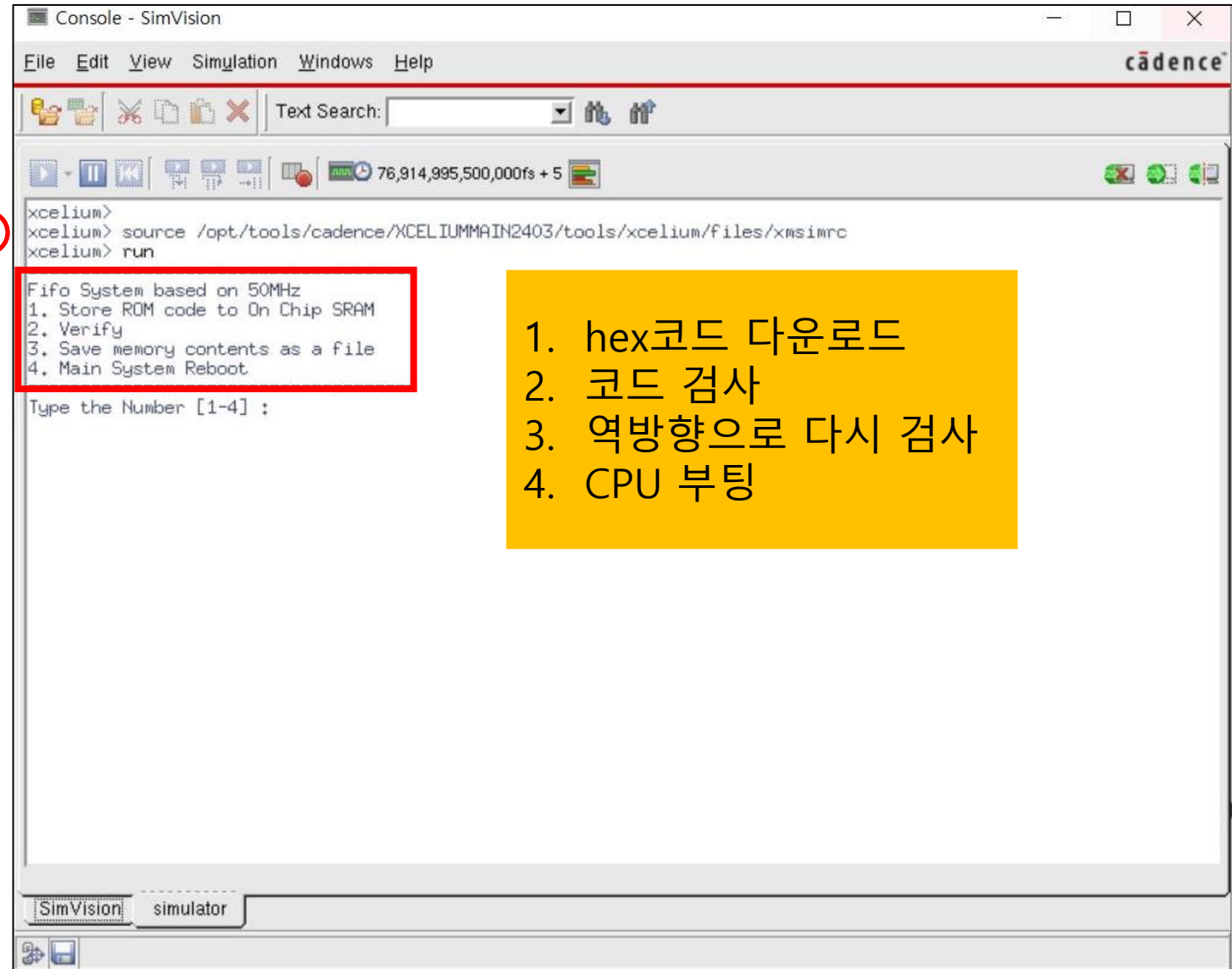
Function Simulation

시뮬레이션

1. xrun 실행 결과

① 1~4까지 숫자가 뜨는데 자동으로 1번이 실행 됨

①



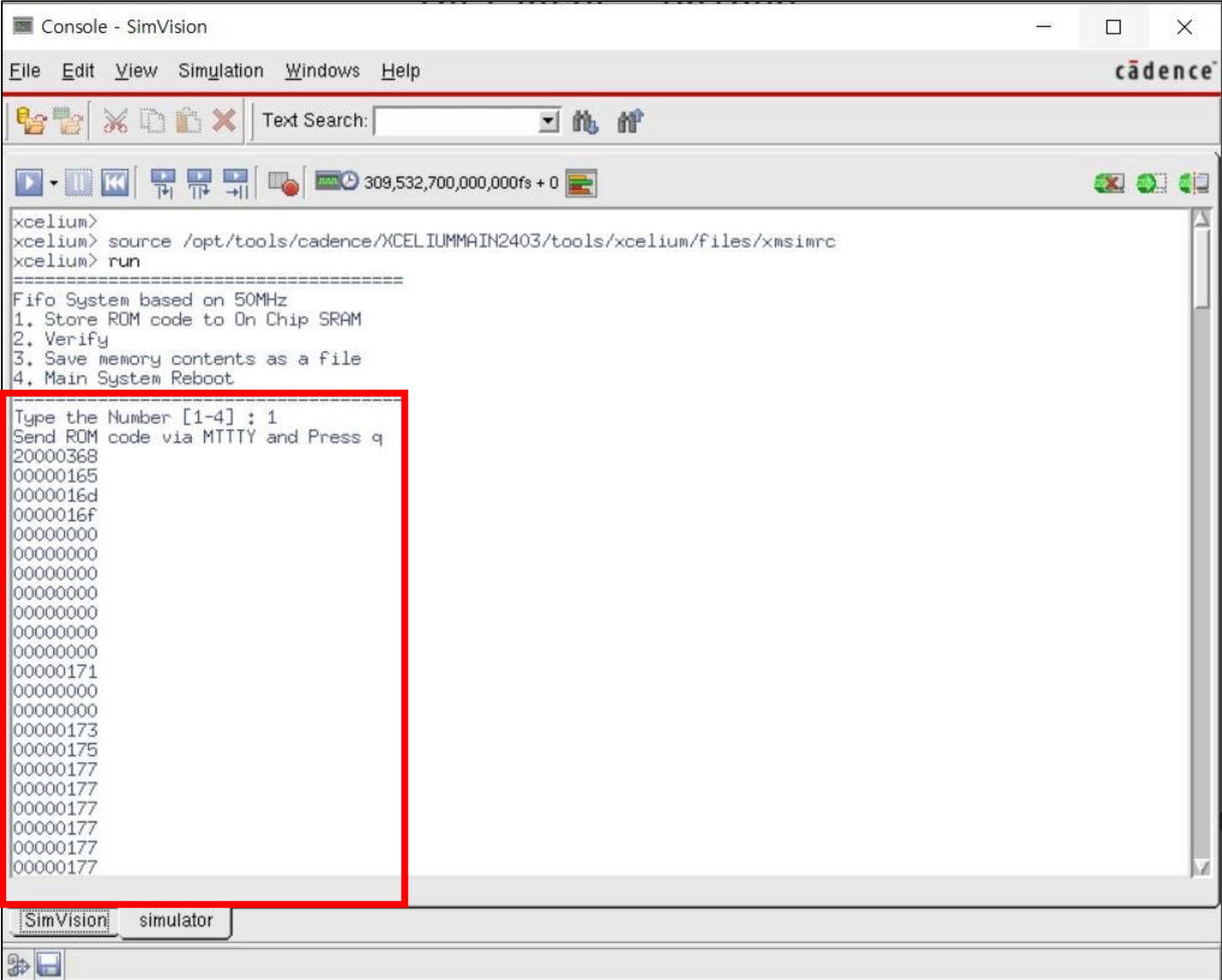
1. hex코드 다운로드
2. 코드 검사
3. 역방향으로 다시 검사
4. CPU 부팅

Function Simulation

시뮬레이션

1. xrun 실행 결과

① 1번이 실행되면서 1번 ~ 4번의 과정이 자동 실행



```
Console - SimVision
File Edit View Simulation Windows Help
Text Search:
309,532,700,000,000fs + 0

xcelium>
xcelium> source /opt/tools/cadence/XCELIUMMAIN2403/tools/xcelium/files/xmsimrc
xcelium> run

=====
Fifo System based on 50MHz
1. Store ROM code to On Chip SRAM
2. Verify
3. Save memory contents as a file
4. Main System Reboot

Type the Number [1-4] : 1
Send ROM code via MTTTY and Press q
20000368
00000165
0000016d
0000016f
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000171
00000000
00000000
00000173
00000175
00000177
00000177
00000177
00000177
00000177
00000177
00000177
```


Function Simulation

시뮬레이션

1. xrun 실행 결과

① 검사과정은 나타나지 않고 부팅이 잘 되었는지 4번으로 결과 확인

```
Console - SimVision
File Edit View Simulation Windows Help
Text Search:
309,532,700,000fs + 0
280047a8
4620d1f8
b510bdf8
46c04604
462046c0
ff01f7ff
0000bd10
47704800
20000008
00000380
20000000
00000008
00000104
00000388
20000008
00000360
00000120
00000000
05f5e100
=====
Fifo System based on 50MHz
1. Store ROM code to On Chip SRAM
2. Verify
3. Save memory contents as a file
4. Main System Reboot
=====
Type the Number [1-4] : 4
START! IDEC Cortex-M0
306920152 ns UART: Hello world
306923743 ns UART: smkcow
306931415 ns UART: ** TEST PASSED **
Simulation stopped via *stop(1) at time 309532700 NS + 0
xcelium> |
```

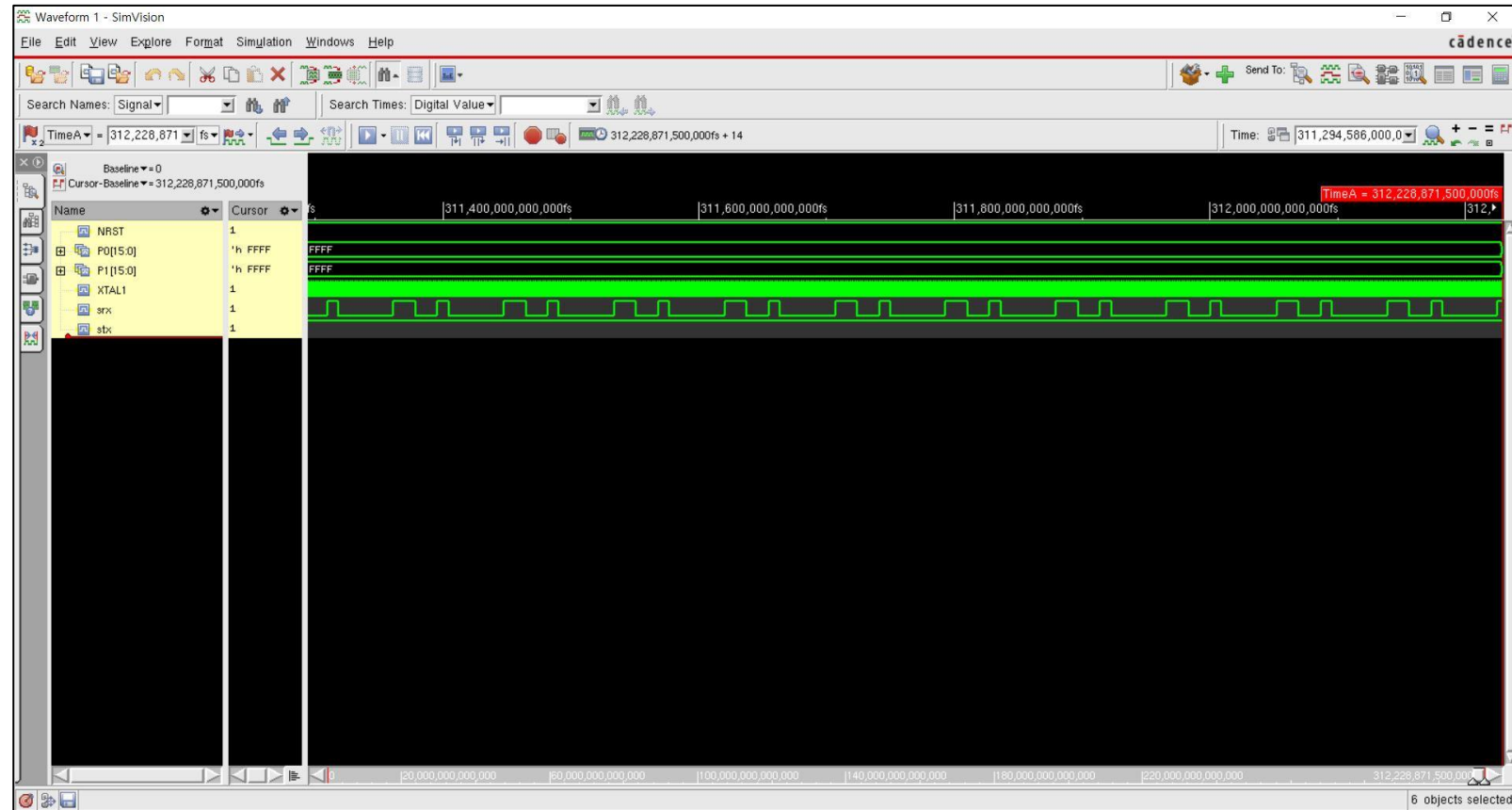
결과가 제대로 나왔다면
Hello world 와
smkcow를 화면에 보여줌

Function Simulation

시뮬레이션

1. xrun 실행 결과

- 파형도 확인은 가능하나 보기 쉽게 텍스트로 확인
- 개인이 보고자 하는 포트나 시그널들을 자유롭게 추가 삭제 가능



Function Simulation

시뮬레이션 분석

1. vi로 hex코드 확인

\$> vi hello.hex

```
[ex_poly1@npit FUNCTION]$ vi hello.hex
```

Function Simulation

시뮬레이션 분석

1. hello.hex

- 시뮬레이션 결과와 비교

```
ex_poly1@npit:FUNCTION
File Edit View Search Terminal Help
68
03
00
20
65
01
00
00
6D
01
00
00
6F
01
00
00
00
00
00
00
00
00
00
00
00
:
```

Function Simulation

시뮬레이션 분석

Little Endian
코드가 역순으로 입력됨

1. hello.hex

- 한 줄의 각 숫자를 2진수로 표현하면 숫자 하나당 4bit이므로 총 32bit임
- Cortex-M0는 32bit 프로세서임을 알 수 있음

```
Type the Number [1-4] : 1
Send ROM code via MTTTY and Press q
20000368
00000165
0000016d
0000016f
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000171
00000000
00000000
00000173
00000175
00000177
00000177
00000177
00000177
00000177
00000177
```

```
ex_poly1@npit:FUNCTION
File Edit View Search Terminal Help
1 68
2 03
3 00
4 20
5 65
6 01
7 00
8 00
9 6D
10 01
11 00
12 00
13 6F
14 01
15 00
16 00
17 00
18 00
19 00
20 00
21 00
22 00
23 00
24 00
:
```