

# Vivado Design Suite Tutorial 908

Programming

12th April 2022

경북대학교 2022222447 박주동

# Ch.6 Programming Configuration Memory Devices

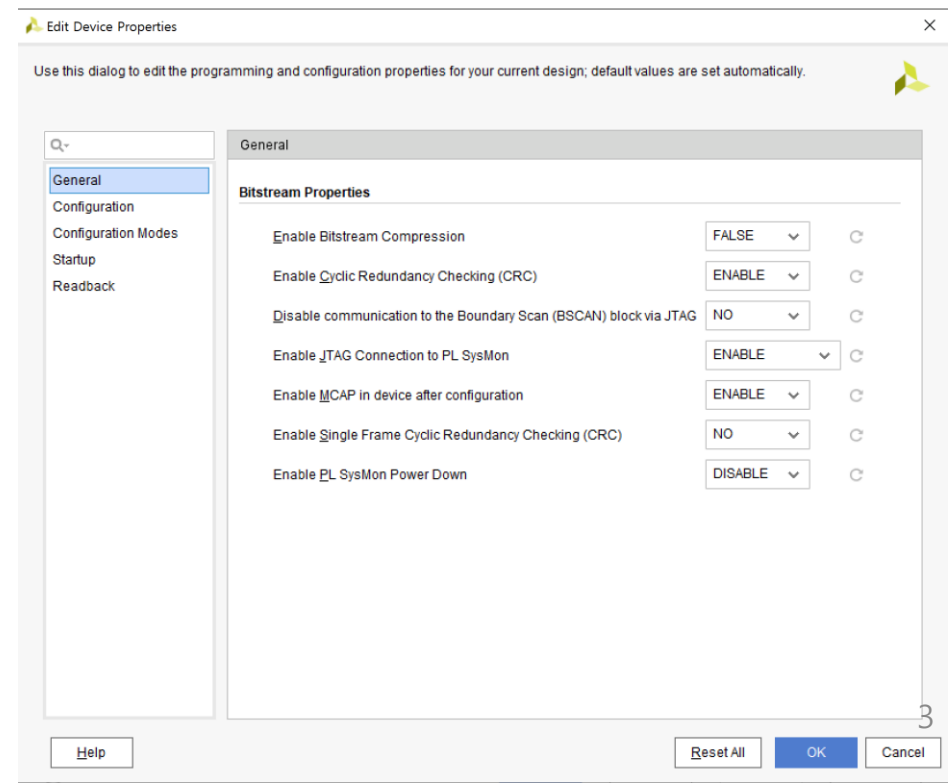
- Fpga에 프로그래밍하는 방법
  - Vivado에서 JTAG를 통해 FPGA를 프로그래밍할 수 있음
    - 전원이 꺼지면 프로그래밍된 데이터 삭제됨
  - 비휘발성 메모리에 파일을 저장시켜서 fpga의 전원이 꺼져도 프로그래밍
  - SVF Target을 통해 fpga에 프로그래밍
- 비휘발성 메모리에 파일을 저장시켜서 fpga의 전원이 꺼져도 프로그래밍
  - Configuration memory 공간에 파일을 저장
- Fpga의 Configuration Memory 공간에서 boot, program 하는 순서
- 비트스트림 생성
- Configuration memory file 생성(.mcs OR .bin)
- Hardware 타겟 연결
- Configuration memory device 추가
- Configuration memory device 프로그래밍
- FPGA 부팅

# Ch.6 Programming Configuration Memory Devices

## 비트스트림 생성

- (Open Implemented Design 이후에)
- Tools – Edit Device Properties
- Settings(from Flow Navigator) – Bitstream – Configure additional bitstream

- CRC
- Boundary Scan(BSCAN)
- PL sysMon
- MCAP



# Ch.6 Programming Configuration Memory Devices

## Configuration Memory File 생성 (for pre-Versal Devices Only)

- Tools - Generate Memory Configuration File
- 메모리 사이즈를 정하거나 메모리 파트를 선택해 해당 비트스트림 파일을 Memory configuration File로 변환
- 메모리 사이즈: 비트스트림 파일보다 큰 2의 제곱승
- 인터페이스: SMAP, SPI, BPI
- checksum
- Bit swapping: MCS file 형식은
- SPIx1|SPIx2|SPIx4|SPIx8 option을 쓴다면 disable
- Hex 파일은 상관없음

Write Memory Configuration File

Create a configuration file to program the device

Format: MCS

☐ Memory Part

☒ Custom Memory Size (MB): 32

Filename:

Options

Interface: SMAPx8

☒ Load bitstream files ☐ Daisy chain configuration file

Start address: 00000000 Direction: up Bitfile:

☐ Lgad data files

Start address: 00000000 Direction: up Datafile:

☐ Write checksum

☐ Disable bit swapping

☐ Overwrite

Command: write\_cfgmem -format mcs -size 32 -interface SMAPx8

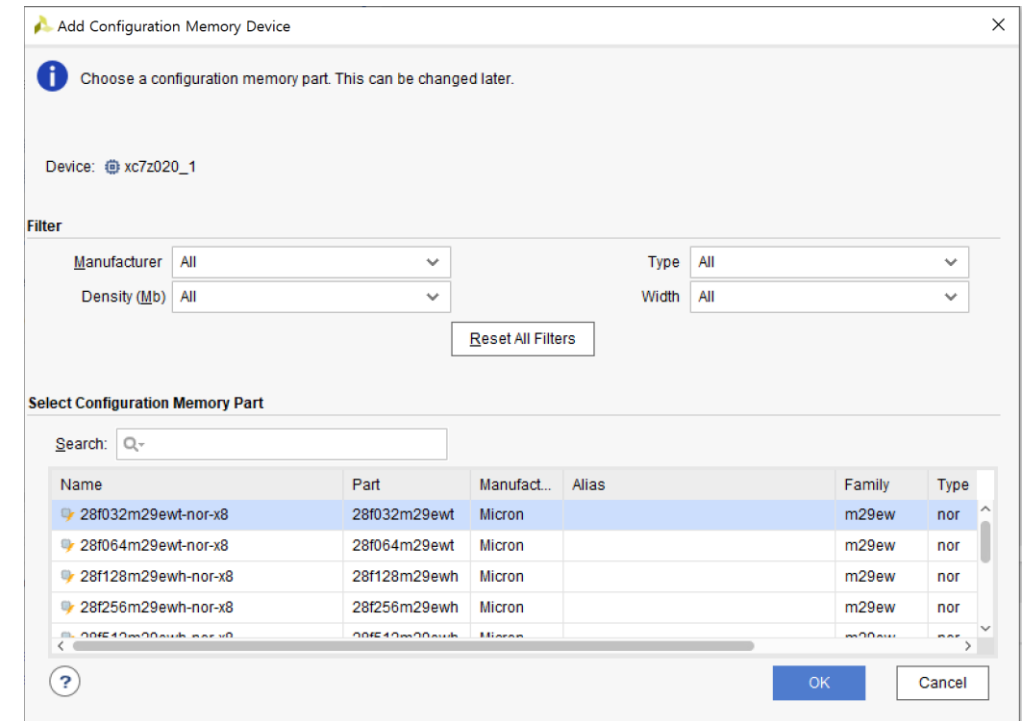
OK Cancel

# Ch.6 Programming Configuration Memory Devices

## Connect to the Hardware Target in Vivado

- Fpga에서의 mode pin들이 알맞게 설정되었는지 확인
- Master SPI
- Master BPI
- ZCU104의 경우 SPI, JTAG 모드 지원

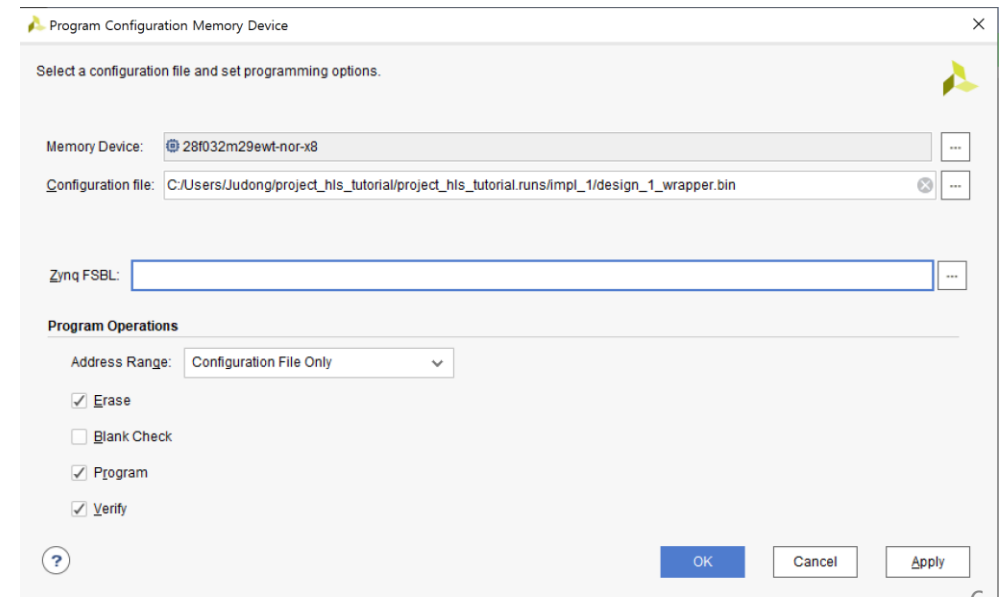
- Add configuration memory device
- 알맞은 memory part 선택



# Ch.6 Programming Configuration Memory Devices

## Programming a Configuration Memory Device

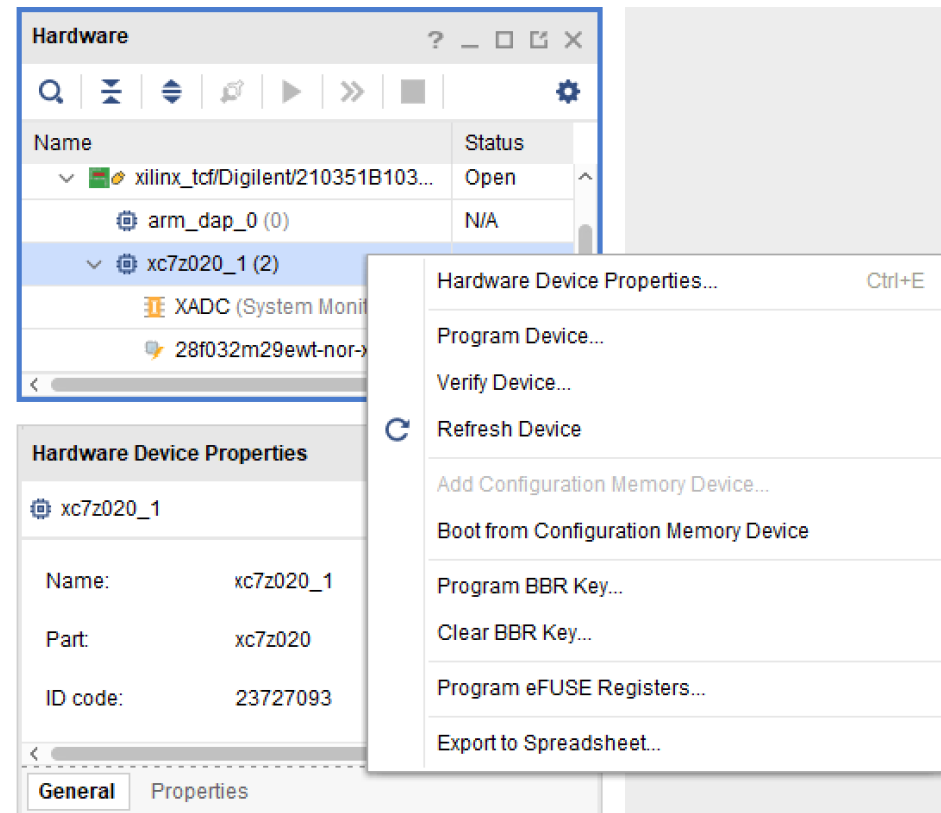
- Configuration file: configuration 메모리를 프로그래밍하는데 사용할 파일(mcs)
- Address range: 프로그래밍할 configuration memory device의 주소영역 범위
- Configuration File Only: 오직 memory configuration file을 프로그래밍할 주소영역만 사용
- Entire Configuration Memory Device: 메모리 전체영역에서 Erase, blank check, program, and verify 수행
- Erase: configuration memory device 내용 삭제
- Blank Check: 프로그래밍 이전에 값이 없는지 확인
- Program: mcs 파일로 configuration 메모리 장치를 프로그래밍
- Verify: 프로그래밍된 내용이 mcs 파일과 같은지 확인



# Ch.6 Programming Configuration Memory Devices

## Booting the Device

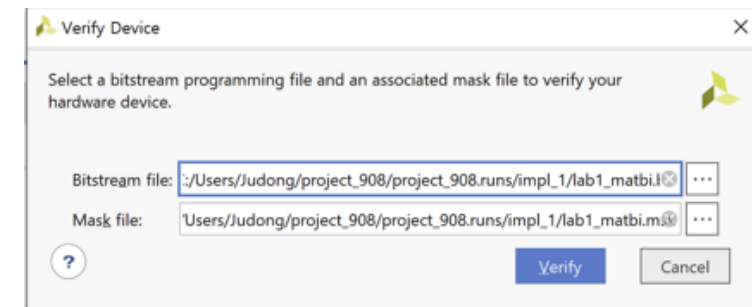
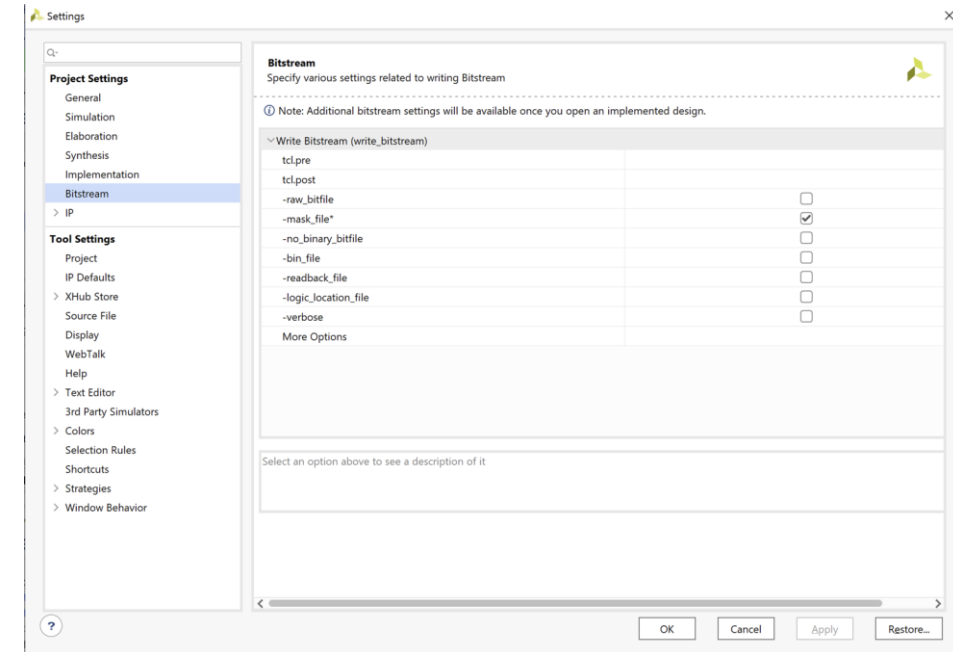
- Boot from Configuration Memory Device
  - Fpga가 켜질 때마다 추가된 configuration memory device로부터 프로그래밍 되었던 내용이 실행



# Ch.7 Advanced Programming Features

## Readback and Verify

- Vivado를 이용하여 FPGA에 있는 비트스트림을 verify, readback할 수 있음
- Verify
  - Fpga에 프로그래밍된 비트스트림 파일이 제대로 전달되었는지 확인
  - 비트스트림 생성 시에 -mask\_file 옵션을 체크하여 bit 파일과 동시에 mask 파일을 생성

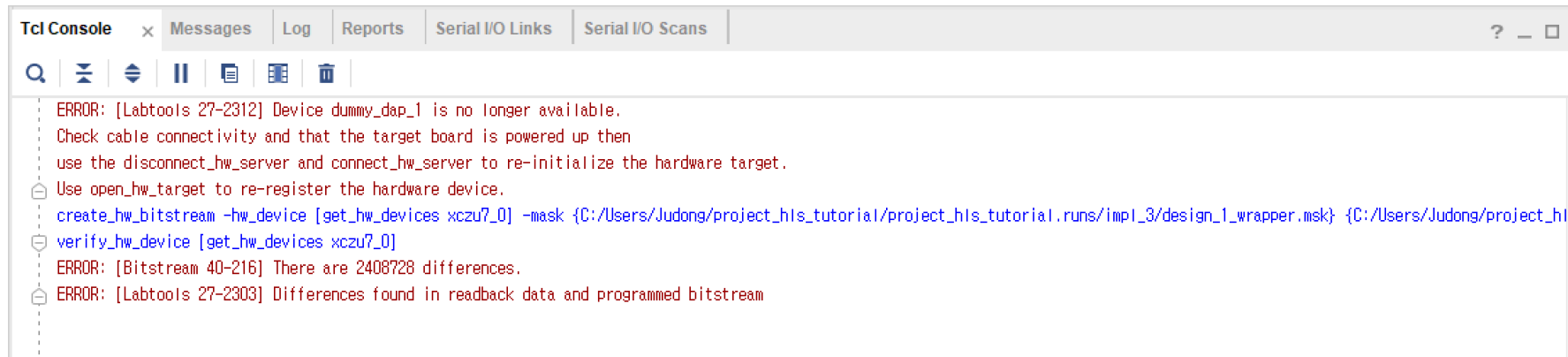




# Ch.7 Advanced Programming Features

## Verify

- Verify
  - 하드웨어 타겟 오른쪽 마우스 클릭 – verify device – bit/msk file 추가 – verify



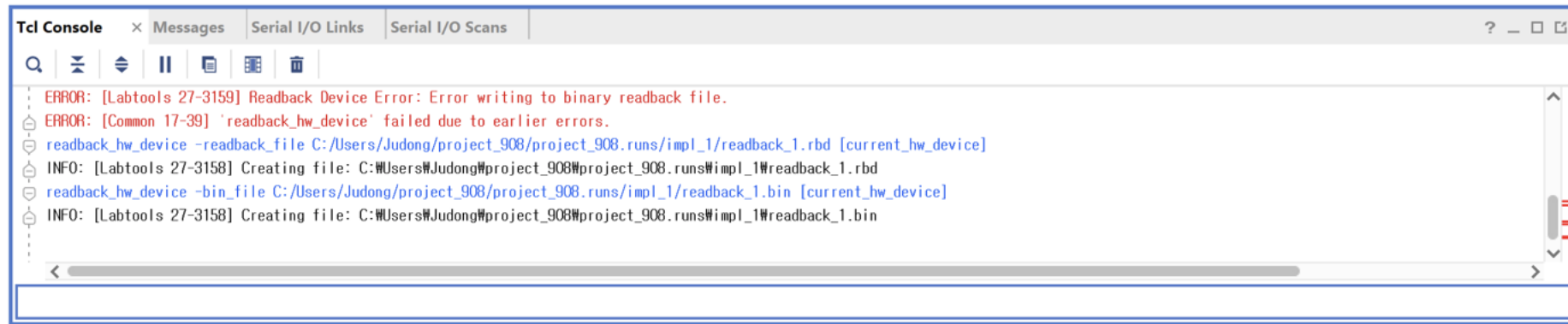
The screenshot shows the Tcl Console window with the following content:

```
Tcl Console x Messages Log Reports Serial I/O Links Serial I/O Scans ? _ □  
Q | Z | A | || | | | |  
ERROR: [Labtools 27-2312] Device dummy_dap_1 is no longer available.  
Check cable connectivity and that the target board is powered up then  
use the disconnect_hw_server and connect_hw_server to re-initialize the hardware target.  
Use open_hw_target to re-register the hardware device.  
create_hw_bitstream -hw_device [get_hw_devices xczu7_0] -mask {C:/Users/Judong/project_hls_tutorial/project_hls_tutorial.runs/impl_3/design_1_wrapper.msk} {C:/Users/Judong/project_hls_tutorial/runs/impl_3/design_1_wrapper.bit}  
verify_hw_device [get_hw_devices xczu7_0]  
ERROR: [Bitstream 40-216] There are 2408728 differences.  
ERROR: [Labtools 27-2303] Differences found in readback data and programmed bitstream
```

# Ch.7 Advanced Programming Features

## Readback

- Readback
  - Configuration memory에서 configuration 데이터(rbd, bin)을 불러 올 수 있음
  - 현재 프로그래밍된 비트스트림과 비교하기위해 readback 명령어 사용
  - Bin, rbd 파일 생성
  - 하드웨어 타겟 오른쪽 마우스 클릭 – verify device – bin/msk 파일 추가 - 결과확인

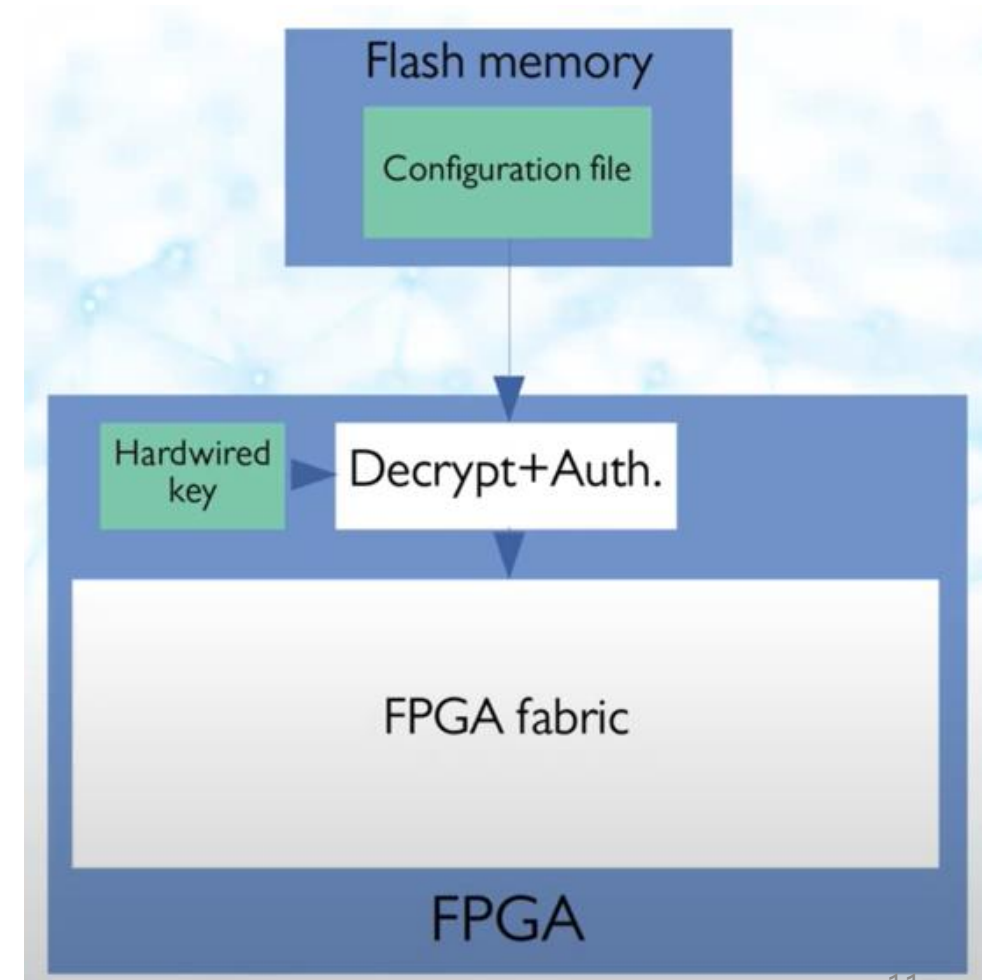


```
Tcl Console x Messages Serial I/O Links Serial I/O Scans
[?] [ ] [x] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
ERROR: [Labtools 27-3159] Readback Device Error: Error writing to binary readback file.
ERROR: [Common 17-39] 'readback_hw_device' failed due to earlier errors.
readback_hw_device -readback_file C:/Users/Judong/project_908/project_908.runs/impl_1/readback_1.rbd [current_hw_device]
INFO: [Labtools 27-3158] Creating file: C:\Users\Judong\project_908\project_908.runs\impl_1\readback_1.rbd
readback_hw_device -bin_file C:/Users/Judong/project_908/project_908.runs/impl_1/readback_1.bin [current_hw_device]
INFO: [Labtools 27-3158] Creating file: C:\Users\Judong\project_908\project_908.runs\impl_1\readback_1.bin
```

# Ch.7 Advanced Programming Features

## Generating Encrypted and Authenticated Files for UltraScale and UltraScale+

- 비트스트림을 암호화
  - 암호화된 파일을 configuration memory에 저장
  - RSA 키를 통해 복호화
  - 해당 파일 실행



# Ch.7 Advanced Programming Features

## Generating Encrypted and Authenticated Files for UltraScale and UltraScale+

- 비트스트림 암호화 설정
  - 비트스트림 생성 전에 설정 변경
  - Flow - Bitstream Settings - Configure Additional Bitstream Settings – Encryption
- Enable Bitstream Encryption: Yes
- Select location of encryption key:  
BBRAM or EFUSE
- Enable obfuscated key:  
ENABLE or DISABLE
- Starting AES encryption key (key0):  
비트스트림을 복호화하기 위한 키(.nky)(~64글자)
- Input encryption file:  
기존에 있던 .nky 파일의 복호화 설정을 가져오기 위해 선택
- Starting AES initial vector (IV0) value:  
키의 초기화 벡터
- Starting obfuscate initial vector (Obfuscate IV0) value  
Obfuscated 키의 초기화 벡터

Use this dialog to edit the programming and configuration properties for your current design; default values are set automatically.

Q:

- General
- Configuration
- Configuration Modes
- Startup
- Encryption**
- Readback
- Authentication

### Encryption

#### Encryption Settings

Enable Bitstream Encryption: NO

Select location of encryption key: BBRAM

Enable obfuscated key load: DISABLE

#### Key Settings

Starting AES encryption key (key0):

Input encryption file:

Starting AES initial vector (IV0) value:

Starting obfuscate initial vector (Obfuscate IV0) value:

#### Key Rolling Settings

Generate a debug file to report all the keys generated in KDF mode: NO

Fixed Input Data for KDF keyrolling:

Seed for KDF keyrolling:

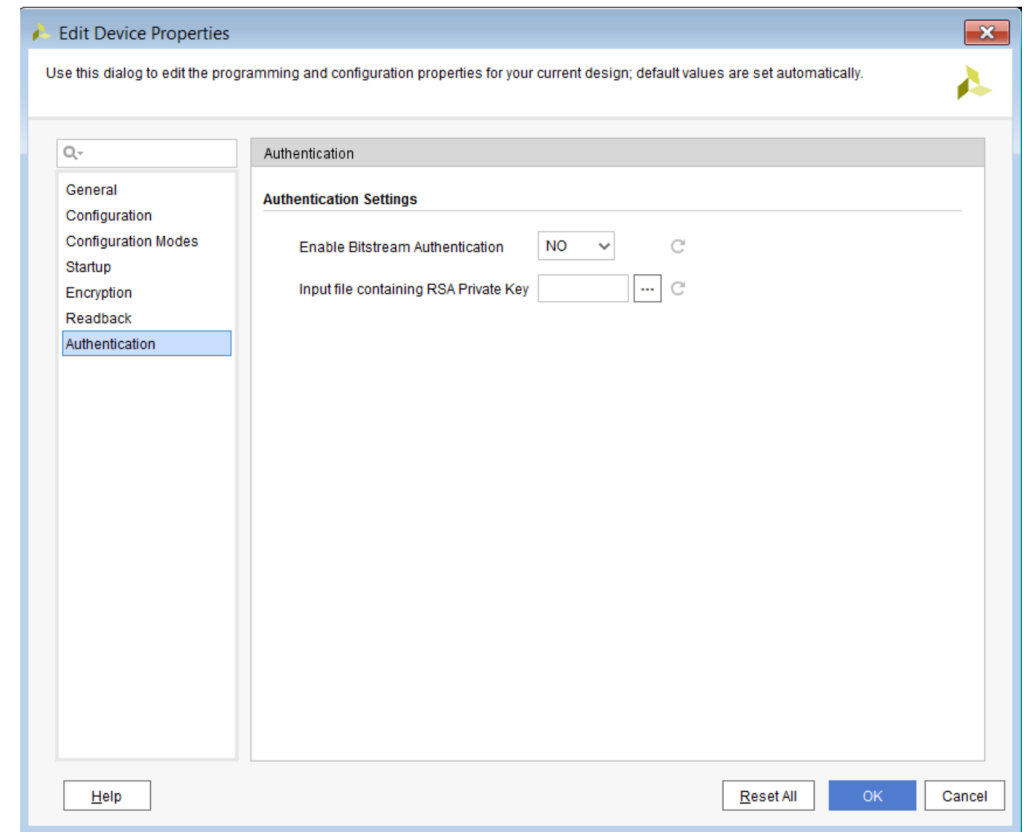
Number of encryption blocks per AES-256 key: 0

Number of frames per AES-256 key: 0

# Ch.7 Advanced Programming Features

## Generating Encrypted and Authenticated Files for UltraScale and UltraScale+

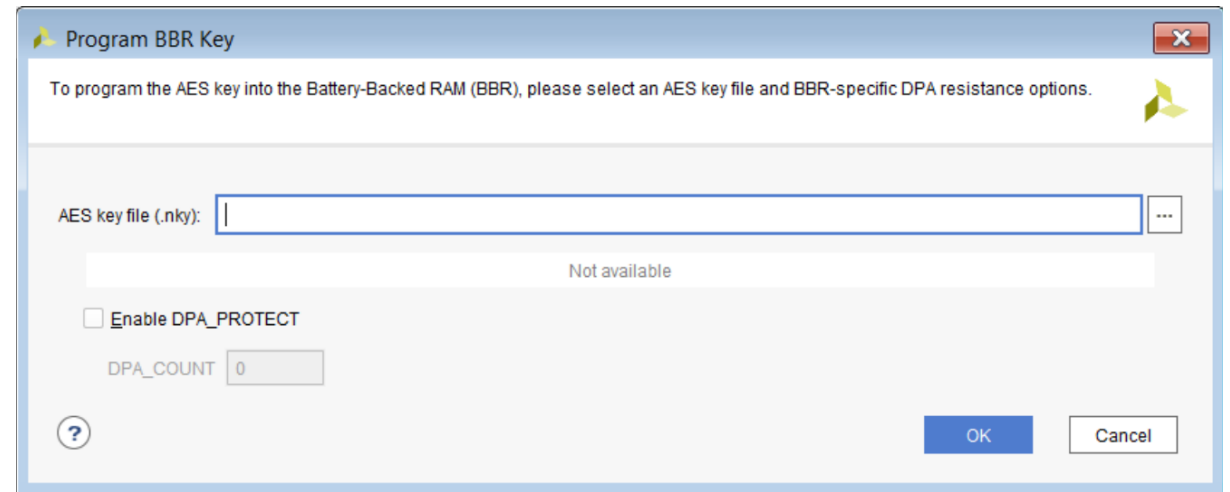
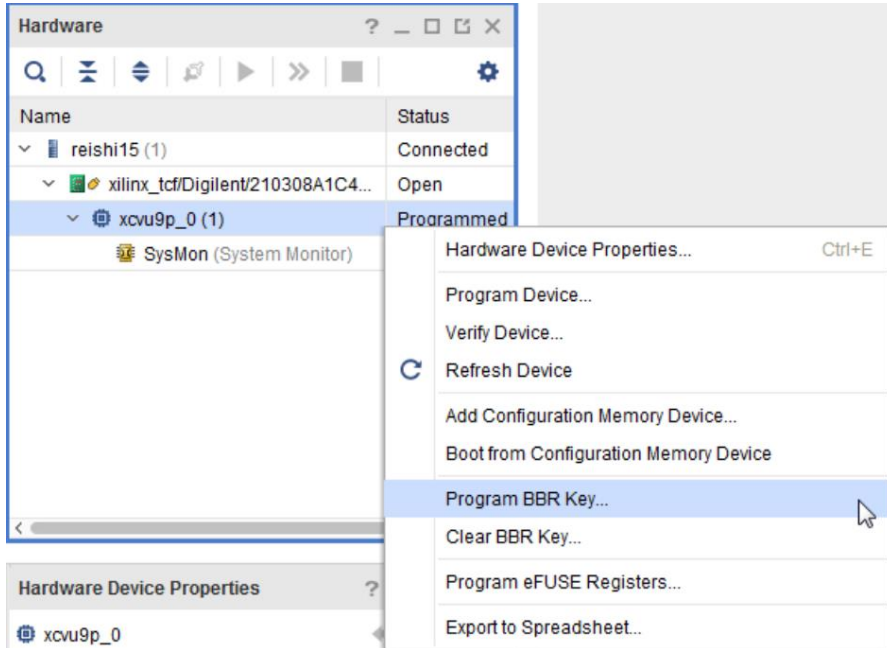
- Enable Bitstream Authentication:  
YES
- Input file containing RSA Private Key:
- OK – 비트스트림 파일과 .nky 암호화 파일 생성



# Ch.7 Advanced Programming Features

## Program the BBR Key

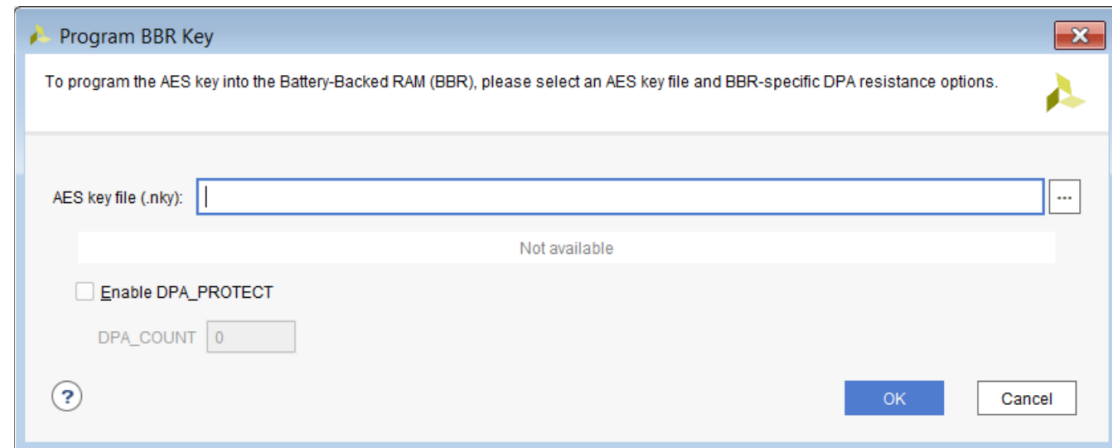
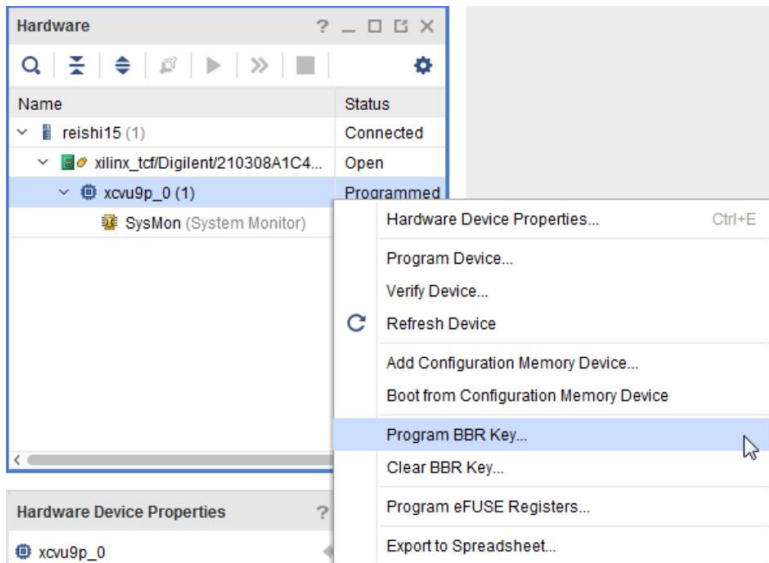
- Target fpga 오른쪽 마우스 클릭 – Program BBR Key – 생성한 키파일 추가 – Enable DPA\_PROTECT 체크 – OK
- Enable DPA\_PROTECT
- 초기값을 가지고 데이터에 접근될 때마다 1씩 감소시켜 그 숫자가 0이 될때 BBRAM을 0으로 초기화



# Ch.7 Advanced Programming Features

## Program the BBR Key

- Target fpga 오른쪽 마우스 클릭 – Program BBR Key – 생성한 키파일 추가 – Enable DPA\_PROTECT 체크 – OK
- Enable DPA\_PROTECT
  - 초기값을 가지고 데이터에 접근될 때마다 1씩 감소시켜 그 숫자가 0이 될때 BBRAM을 0으로 초기화
- Clear BBR Key
  - BBRAM에 저장된 Key를 지움



# Ch.7 Advanced Programming Features

## eFUSE Register Access and Programming for UltraScale and UltraScale+ Devices

- eFUSE bits
  - 7 series, UltraScale, and UltraScale+
  - 1번만 프로그래밍될 수 있는 비트 → 비휘발성
  - eFUSE ↔ BBRAM: 저장된 AES key가 수정되거나 지워질 수 있음
- Device DNA
  - 각 ultrascale device들은 고유의 ID를 가짐

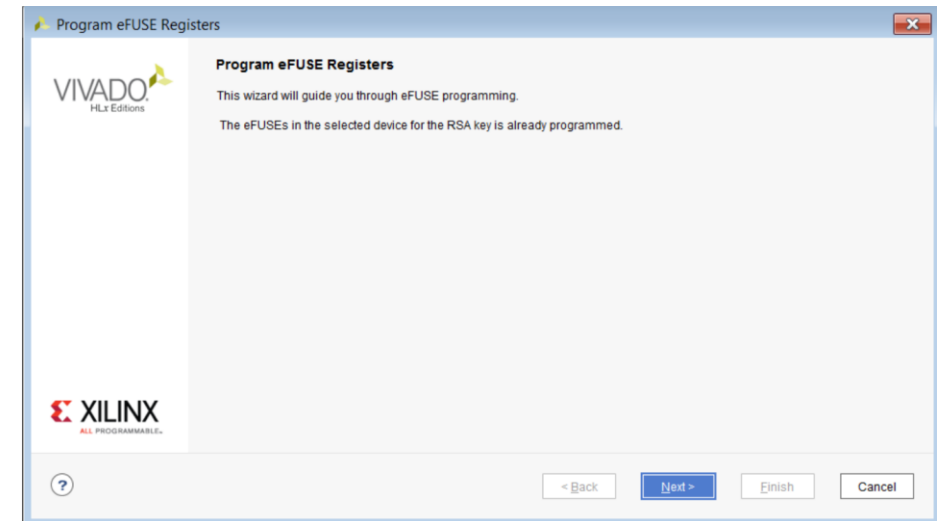
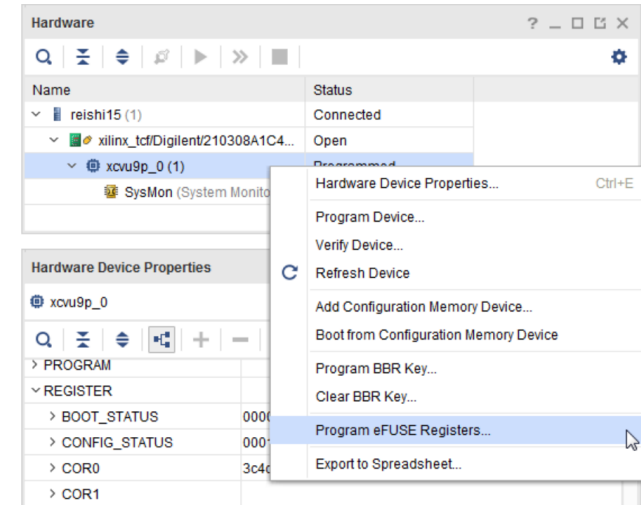
REGISTER	
> BOOT_STATUS	00000000000000000000000000000001
> CONFIG_STATUS	00010010100100000111111111111100
> COR0	38003fe5
> COR1	
EFUSE	
DNA_PORT	40020000012995892471C305
FUSE_DNA	40020000012995892471C305



# Ch.7 Advanced Programming Features

## eFUSE Register Access and Programming

- 타겟 하드웨어 device 오른쪽 마우스 클릭 – Program eFUSE Registers
- Program eFUSE Registers Wizard 실행



# Ch.7 Advanced Programming Features

## Program eFUSE Registers Wizard

- eFUSE Cryptographic Key Setup
  - 비트스트림 생성시 같이 만들어진 .nky 추가
  - Enable AES key programming
    - Device에서 복호화하기 위한 키를 확인
- Next

The screenshot shows the 'Program eFUSE Registers' wizard window. The title bar says 'Program eFUSE Registers'. The main heading is 'Cryptographic Key Setup (optional)'. Below it, a note says 'Provide a cryptographic key file (.nky) to enable AES, RSA, or both encryption keys. Keys are displayed in HEX.' There is a text field for 'Cryptographic key file (.nky):' with the path 'C:/Users/smitha/Documents/lug908\_2017\_1/xcku115\_efuse\_aes.nky'. Below this is a checkbox 'Enable AES key programming' which is checked. Underneath, there are two text boxes for AES keys: 'SLR0 AES Key' with the value '0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_7767' and 'SLR1 AES Key' with the value '0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0000\_0554\_62AA\_ABB8'. Below these is a section for RSA with a checked box 'RSA Key Digest is eFUSE programmed' and a text box 'Current RSA value' containing 'DB0B\_2559\_7A19\_2F6D\_1805\_F964\_C2DF\_F855\_7E74\_BECA\_4FA8\_F065\_7B6B\_3CAA\_5938\_91E'. There is also an unchecked checkbox 'Enable RSA Key Digest (384-bit) programming' with a disabled text box 'Not available'. At the bottom, there are buttons: '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'. A help icon (?) is also present.

# Ch.7 Advanced Programming Features

## Program eFUSE Registers Wizard

- Control Register Setup Pane
  - 프로그래밍할 control register 체크
- Security Register Setup

**Program eFUSE Registers**

**USER register setup (optional)**  
Select a checkbox to specify a 32-bit USER register value, a 128-bit USER128 register value, both, or neither. Specify the values in HEX format.

☒ Specify 32-bit value to program into USER register  
USER bits [31:0]:

☒ Specify 128-bit value to program into USER128 register  
USER bits [127:0]:

? < Back Next > Finish Cancel

**Program eFUSE Registers**

**Control Register Setup (optional)**  
Enable and select the Control Register bits to program. These bits provide security by disabling operations on different eFUSE registers.

☒ Enable control register programming

Register Bit Name	Bit P...	Description
<input type="checkbox"/> R_DIS_KEY	0	Disable reading and programming of FUSE_KEY encryption key
<input type="checkbox"/> R_DIS_USER	1	Disable reading and programming of FUSE_USER user code
<input type="checkbox"/> R_DIS_SEC	2	Disable reading and programming of FUSE_SEC security settings
<input type="checkbox"/> R_DIS_RSA	6	Disable reading and programming of FUSE_RSA authentication key
<input type="checkbox"/> W_DIS_KEY	7	Disable programming of FUSE_KEY encryption key
<input type="checkbox"/> W_DIS_USER	8	Disable programming of FUSE_USER user code
<input type="checkbox"/> W_DIS_SEC	9	Disable programming of FUSE_SEC security settings
<input type="checkbox"/> W_DIS_RSA	15	Disable programming of FUSE_RSA authentication key
<input type="checkbox"/> W_DIS_USER128	16	Disable programming of FUSE_USER128

? < Back Next > Finish 9 Cancel

# Ch.7 Advanced Programming Features

## Program eFUSE Registers Wizard

- Control Register Setup Pane
  - 프로그래밍할 control register 선택
- Security Register Setup
  - 프로그래밍할 Security register 선택

**Program eFUSE Registers**

**USER register setup (optional)**  
Select a checkbox to specify a 32-bit USER register value, a 128-bit USER128 register value, both, or neither. Specify the values in HEX format.

☒ Specify 32-bit value to program into USER register  
USER bits [31:0]:

☒ Specify 128-bit value to program into USER128 register  
USER bits [127:0]:

? < Back Next > Finish Cancel

**Program eFUSE Registers**

**Control Register Setup (optional)**  
Enable and select the Control Register bits to program. These bits provide security by disabling operations on different eFUSE registers.

☒ Enable control register programming

Register Bit Name	Bit P...	Description
<input type="checkbox"/> R_DIS_KEY	0	Disable reading and programming of FUSE_KEY encryption key
<input type="checkbox"/> R_DIS_USER	1	Disable reading and programming of FUSE_USER user code
<input type="checkbox"/> R_DIS_SEC	2	Disable reading and programming of FUSE_SEC security settings
<input type="checkbox"/> R_DIS_RSA	6	Disable reading and programming of FUSE_RSA authentication key
<input type="checkbox"/> W_DIS_KEY	7	Disable programming of FUSE_KEY encryption key
<input type="checkbox"/> W_DIS_USER	8	Disable programming of FUSE_USER user code
<input type="checkbox"/> W_DIS_SEC	9	Disable programming of FUSE_SEC security settings
<input type="checkbox"/> W_DIS_RSA	15	Disable programming of FUSE_RSA authentication key
<input type="checkbox"/> W_DIS_USER128	16	Disable programming of FUSE_USER128

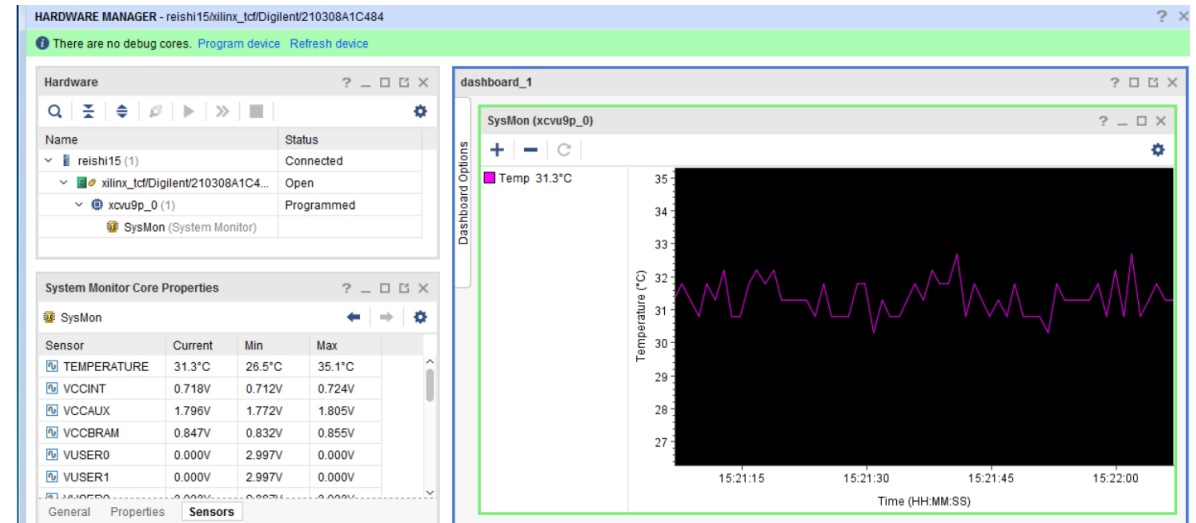
? < Back Next > Finish 20 Cancel



# Ch.7 Advanced Programming Features

## System Monitor

- eFUSE settings을 export
  - 프로그래밍된 eFUSE를 모니터링하기 위해
  - Export된 파일 확장자: .nkz
  - Nkz파일 export를 위한 tcl 명령어: `export_<FUSE_DNA>.nkz`
- System Monitor
  - 하드웨어의 온도 대 전압 그래프
  - 실행 명령어: `get_hw_sysmon_reg`



# Ch.8 Serial Vector Format (SVF) File Programming

- Serial vector format file(SVF)
  - Serial vector format file을 통해 FPGA와 configuration memory devices를 프로그래밍하는 방법
- SVF 파일 실행 순서
  - SVF offline target 만들기
  - SVF target 열기
  - Target에 devices 추가
  - 프로그래밍
  - SVF 쓰기
  - SVF target 닫기
  - SVF 실행

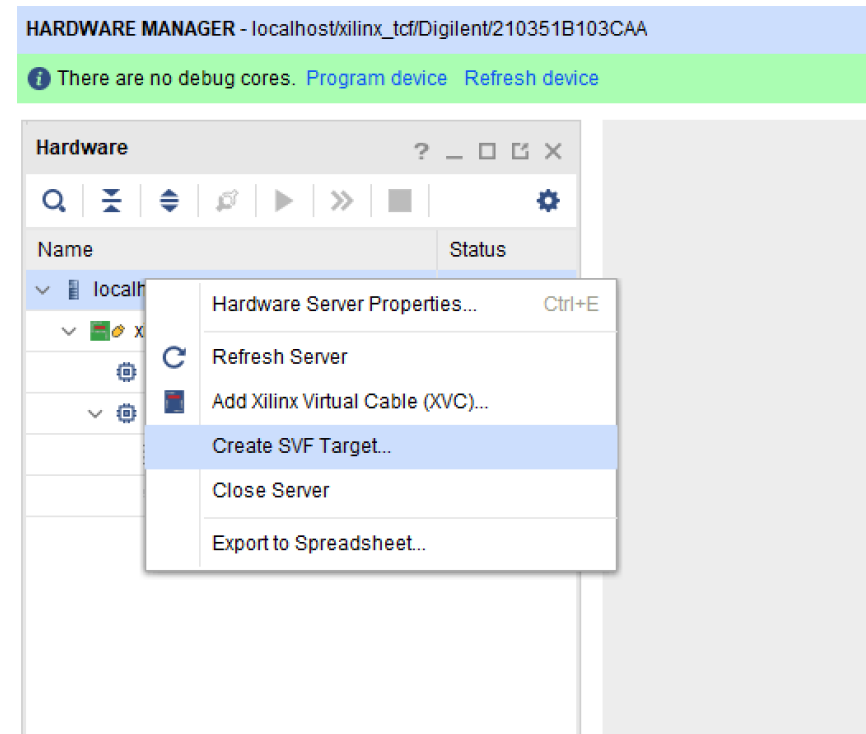
# Ch.8 Serial Vector Format (SVF) File Programming

## Creating an SVF Target

- 비트스트림 파일을 svf파일 변환을 통해 device에 프로그래밍
  - SVF 파일을 실행하여 device를 프로그래밍 가능
  - 차이점: target에 어떤 명령어가 수행되면 SVF는 execute될 때까지 device에 영향을 미치지 않음

### Tools – Create SVF Target

특정한 Fpga의 svf 파일을 만들기 위해 실행

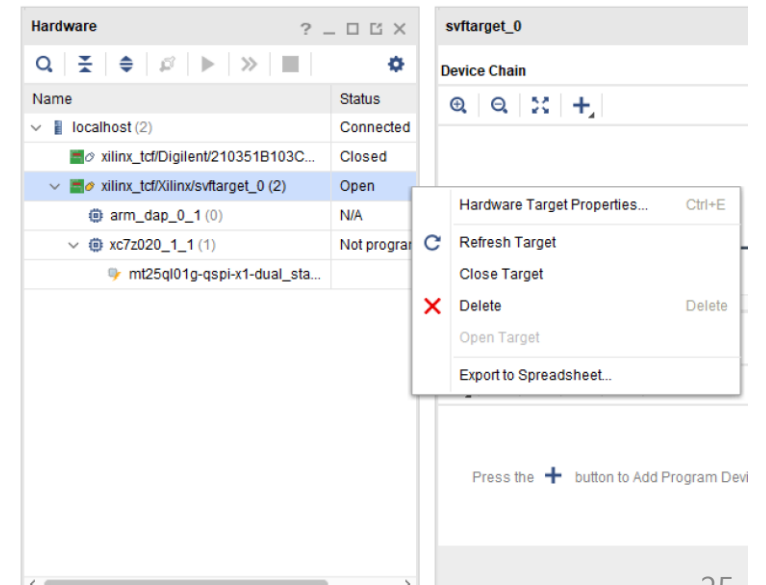
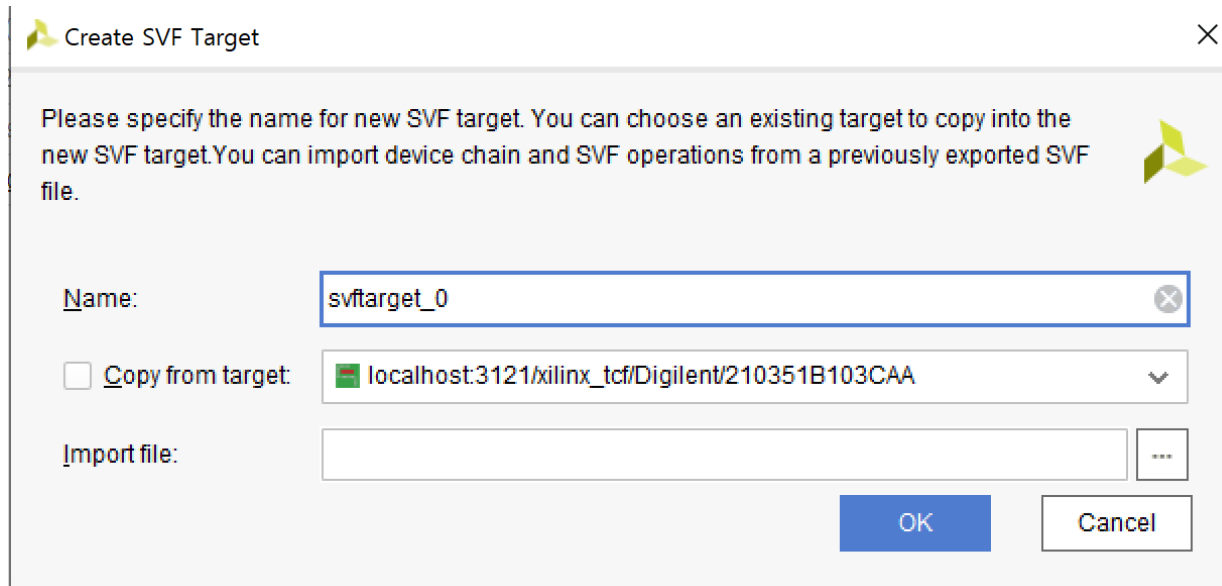




# Ch.8 Serial Vector Format (SVF) File Programming

## Creating an SVF Target

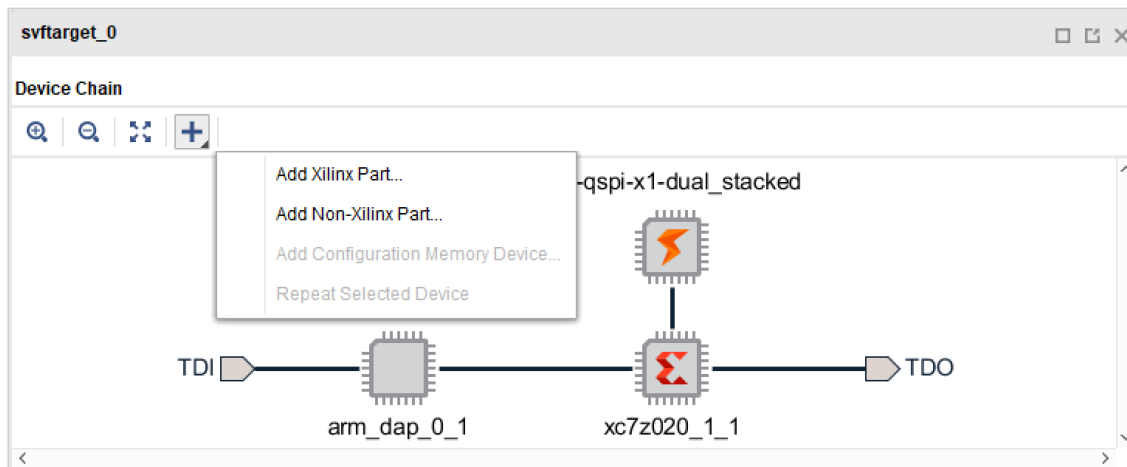
- SVF Target 생성
  - Copy from target 체크: 해당 서버에 있는 하드웨어 타겟의 정보를 복사
  - Import file을 통해 다른 위치에 있는 SVF파일을 가져올 수 있음
- SVF Target 삭제
  - 삭제할 SVF Target 오른쪽 버튼 클릭 – delete



# Ch.8 Serial Vector Format (SVF) File Programming

## Adding Devices to an SVF Target

- Device를 어떻게 configure할 것인가를 정의하기 위해 SVF target – Device chain에서 device들을 추가 가능
  - SVF target – device chain – add Non-Xilinx/Xilinx part
  - SVF JTAG device chain configuration과 target hardware chain은 매치되어야 하기 때문
    - Svf파일은 svf 명령어와 configuration 정보를 담고 있는데 chain configuration 정보가 타겟의 하드웨어 chain정보와 같아야 하기 때문



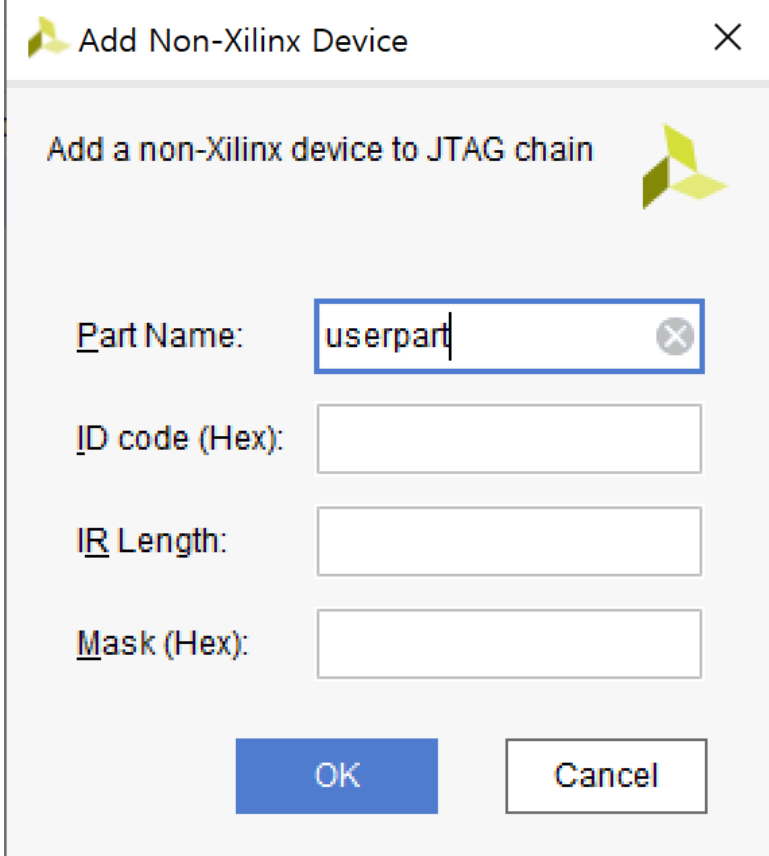
The 'Add Xilinx Part' dialog box is shown, displaying a table of available Xilinx parts. The table has columns for Part, IO Pin C., Available I., LUT Elem., FlipFlo., Block R., and Ultra R.. The first few rows are highlighted in blue.

Part	IO Pin C.	Available I.	LUT Elem.	FlipFlo.	Block R.	Ultra R.
xa7a12kpg238-2i	238	112	8000	16000	20	0
xa7a12kpg238-1i	238	112	8000	16000	20	0
xa7a12kpg238-1Q	238	112	8000	16000	20	0
xa7a12kpg325-2i	325	150	8000	16000	20	0
xa7a12kpg325-1i	325	150	8000	16000	20	0
xa7a12kpg325-1Q	325	150	8000	16000	20	0
xa7a15kpg238-2i	238	106	10400	20800	25	0
xa7a15kpg238-1i	238	106	10400	20800	25	0
xa7a15kpg238-1Q	238	106	10400	20800	25	0
xa7a15kpg324-2i	324	210	10400	20800	25	0
xa7a15kpg324-1i	324	210	10400	20800	25	0
xa7a15kpg324-1Q	324	210	10400	20800	25	0
xa7a15kpg325-2i	325	150	10400	20800	25	0
xa7a15kpg325-1i	325	150	10400	20800	25	0
xa7a15kpg325-1Q	325	150	10400	20800	25	0
xa7a25kpg238-2i	238	112	14800	29200	45	0

# Ch.8 Serial Vector Format (SVF) File Programming

## Adding Devices to an SVF Target

- Part name: part 이름 설정
  - ID code(Hex): 각 보드마다 가지고 있는 ID code
  - IR length: 명령어 레지스터 길이
  - Mask(Hex): Hex 비트, 마스크 값
- 
- ID code, IR Length, Mask 값은 보통 BSDL 파일을 통해 제공됨
  - <https://www.xjtag.com/about-jtag/bsdl-files/#list>



**Add Non-Xilinx Device**

Add a non-Xilinx device to JTAG chain

Part Name:

ID code (Hex):

IR Length:

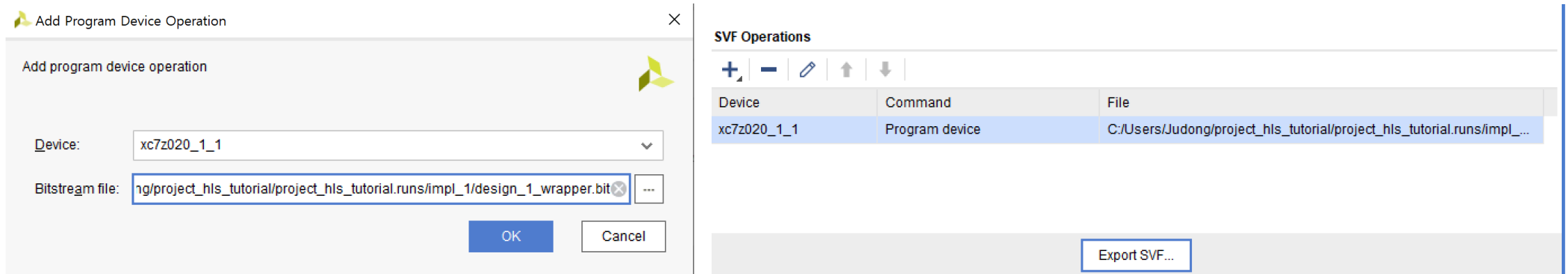
Mask (Hex):

**OK** **Cancel**

# Ch.8 Serial Vector Format (SVF) File Programming

## Operations on the SVF Chain

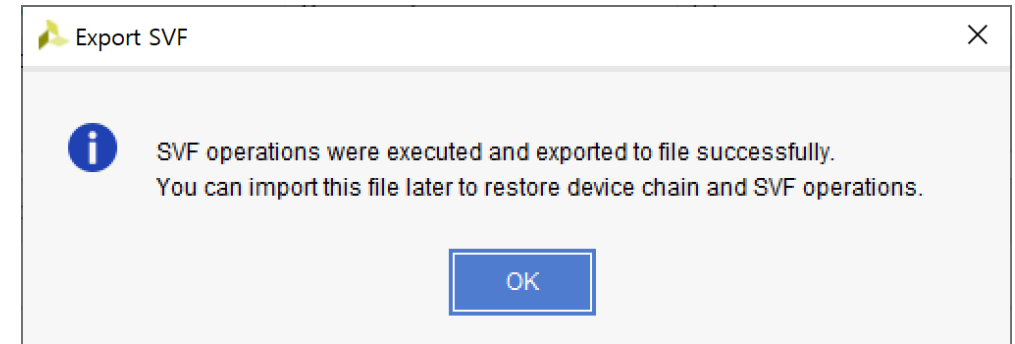
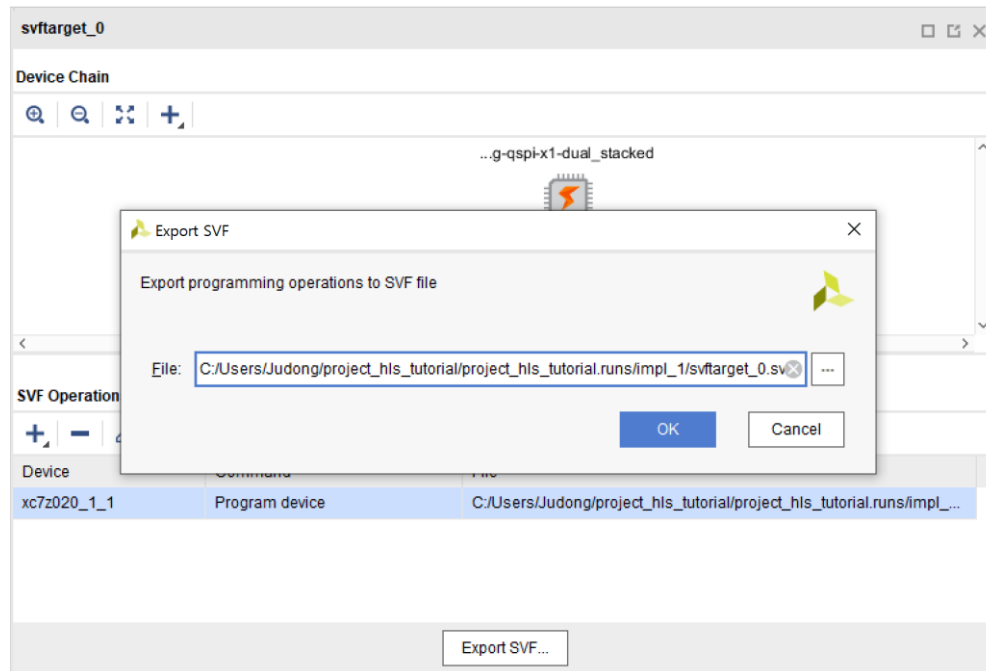
- SVF 파일 출력, 실행하기
  - 해당하는 디바이스 혹은 configuration memory에 대한 device chain이 올바른 순서로 만들어졌다면, SVF를 실행하고 출력할 수 있음
  - Device chain – 화면 오른쪽 마우스 클릭 – Add Program Device Operation – 비트스트림 파일 선택 – ok



# Ch.8 Serial Vector Format (SVF) File Programming

## Writing SVF Files

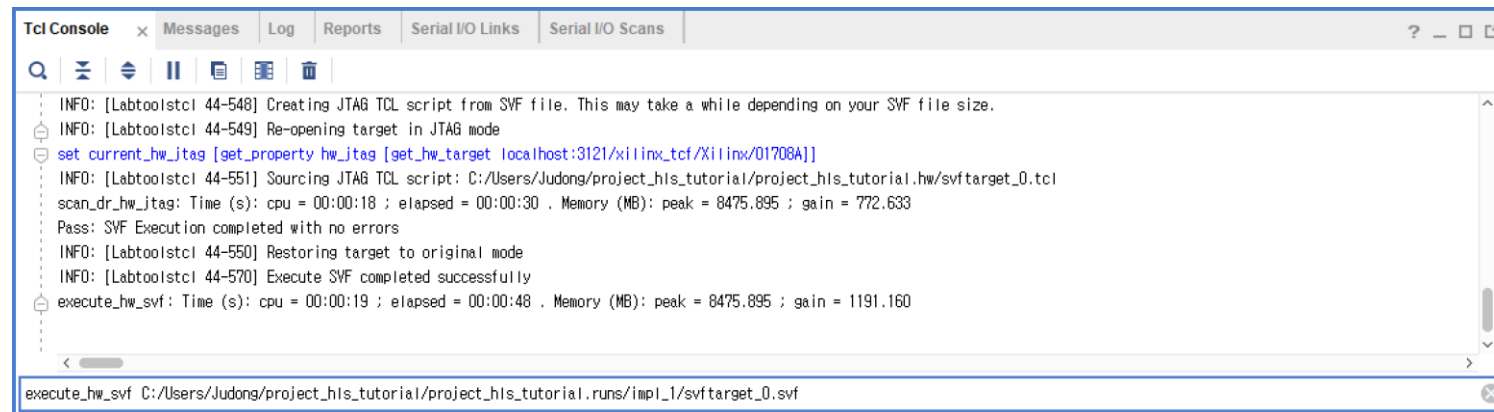
- SVF chain을 svf파일에 저장
  - SVF Operation – Export SVF 클릭
  - Fpga에 첨부한 비트스트림을 프로그래밍할 수 있음
  - Excute&export



# Ch.8 Serial Vector Format (SVF) File Programming

## Exporting the SVF Chain Setup

- Vivado IDE를 통해 SVF file을 실행(execute)
  - Vivado IDE를 통해 SVF가 잘 실행되었는지 에러가 있는지 확인가능
- Excute 명령어로 svf파일이 실행
  - Svf파일이 Tcl 명령어로 변환되어 임시파일에 저장
  - Svf파일에서 변환된 Tcl 명령어를 실행하기 위해 해당 파일을 로드
  - 변환된 Tcl 명령어를 보려면 -verbose 옵션을 사용
  - Pass or Error



```
Tcl Console x Messages Log Reports Serial I/O Links Serial I/O Scans
INFO: [Labtoolstcl 44-548] Creating JTAG TCL script from SVF file. This may take a while depending on your SVF file size.
INFO: [Labtoolstcl 44-549] Re-opening target in JTAG mode
set current_hw_jtag [get_property hw_jtag [get_hw_target localhost:3121/xilinx_tcf/Xilinx/01708A]]
INFO: [Labtoolstcl 44-551] Sourcing JTAG TCL script: C:/Users/Judong/project_hls_tutorial/project_hls_tutorial.hw/svftarget_0.tcl
scan_dr_hw_jtag: Time (s): cpu = 00:00:18 ; elapsed = 00:00:30 , Memory (MB): peak = 8475.895 ; gain = 772.633
Pass: SVF Execution completed with no errors
INFO: [Labtoolstcl 44-550] Restoring target to original mode
INFO: [Labtoolstcl 44-570] Execute SVF completed successfully
execute_hw_svf: Time (s): cpu = 00:00:19 ; elapsed = 00:00:48 , Memory (MB): peak = 8475.895 ; gain = 1191.160
execute_hw_svf C:/Users/Judong/project_hls_tutorial/project_hls_tutorial.runs/impl_1/svftarget_0.svf
```