SATA-IP User Guide on KC705

Rev1.1 06-Mar-13

Referring to 1-ch SATA host demo on KC705 Evaluation board, this document describes how to update hardware by using Vivado2012.4/EDK14.4 and how to update firmware in the demo by using SDK14.4.

1. DDR3 IP Update (AR#53420)

This demo uses MIG v1.8 IP core for controlling DDR3 memory. Following AR#53420 of Xilinx website, default IP core in Xilinx installation directory has the problem and it needs to install the patch to fix this bug. User is recommended to download "mig_v1_8_calibration_patch.zip" from http://www.xilinx.com/support/answers/53420.htm and install it to user platform before re-implement the demo project.

2. Vivado Tool

2.1 Overview

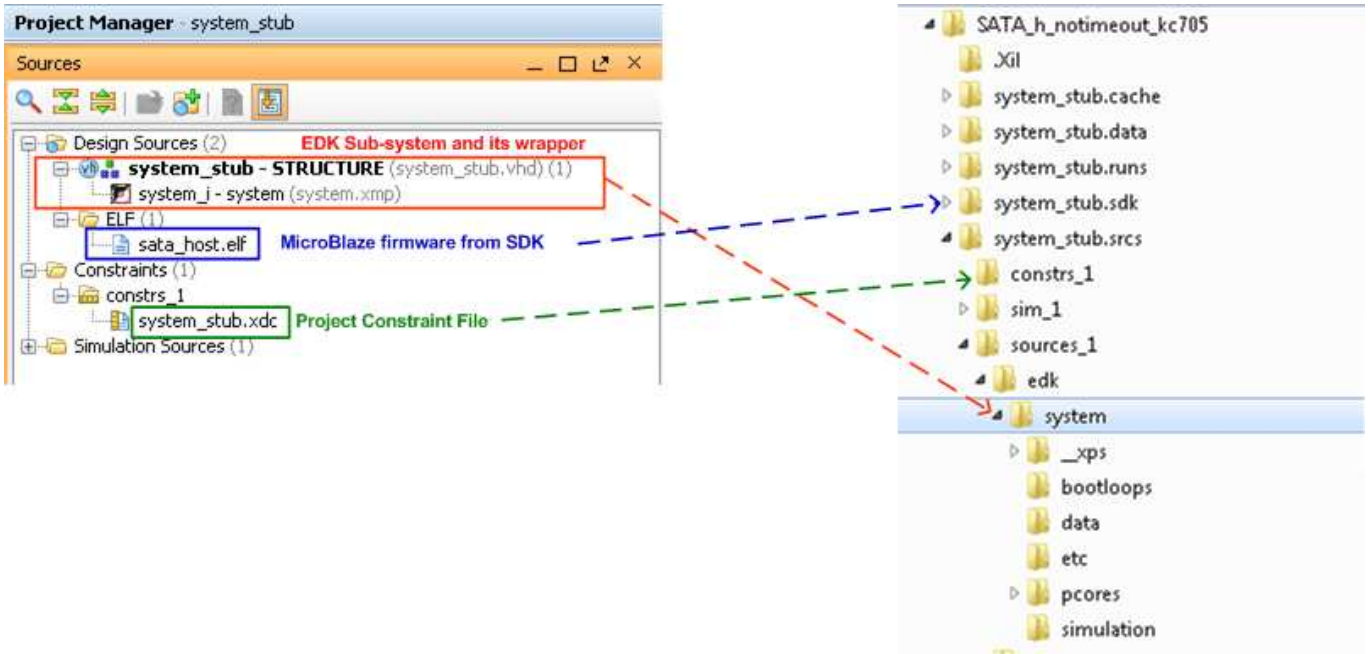

Figure 2-1 Demo system on Vivado Tool

The demo is designed based on EDK sub-system. "system_stub.vhd" is generated to be the wrapper file of EDK sub-system, so this file needs to update if port of EDK sub-system is modified. User can check details of EDK sub-system by double-click from icon in Vivado tool or opening directly from "system_stub.srcs/sources_1/edk/system" folder. More details of EDK sub-system is described in next topic.

The demo includes MicroBlaze to control SATA-IP operation and its firmware (sata_host.c) will be compiled to "sata_host.elf" file by SDK tool. By including "sata_host.elf" into Vivado project, the final bitstream "system_stub.bit", stored in "system_stub.runs/impl_1" folder, will built-in the firmware and FPGA can boot the demo after configuration complete. SDK project is stored into "system_stub.sdk" folder and the details of SDK tools will be described later.

2.2 How to update design in Vivado

- If EDK sub-system is modified, user should right-click at EDK icon and select “Reset Output Products…” to clean up sub-system netlist, as shown in Figure 2-2.

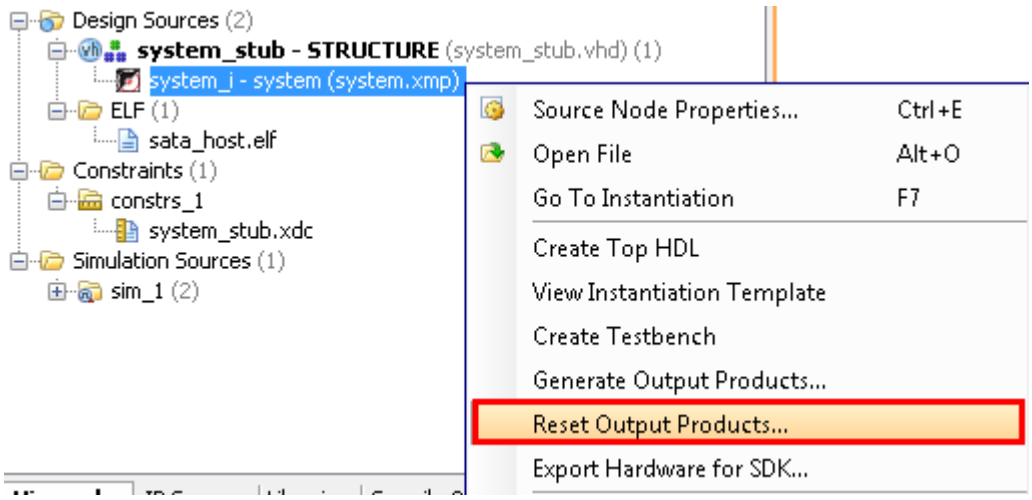

Figure 2-2 Reset EDK sub-system

- Select “Generate Bitstream” menu to start re-implementation the project, as shown in Figure 2-3.

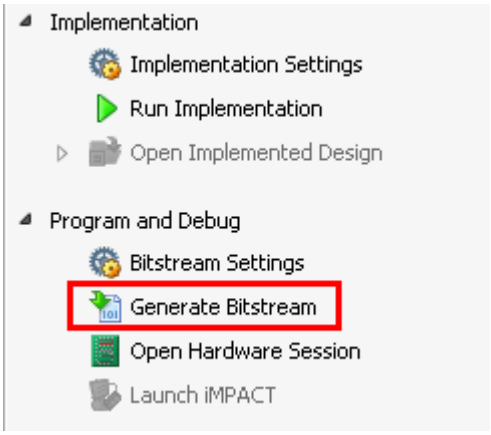

Figure 2-3 Re-generate bitstream

3. EDK Tool

3.1 Overview

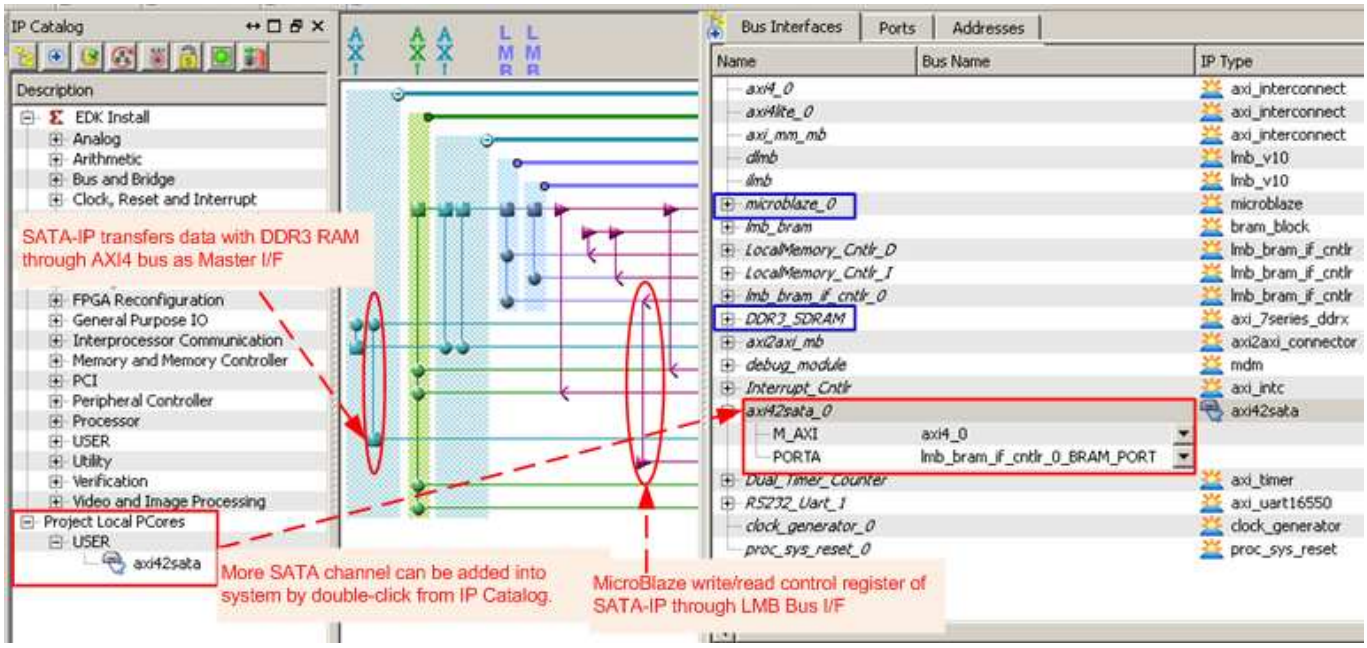

Figure 3-1 SATA-IP demo system on EDK tool

In EDK sub-system, SATA-IP with additional logic to connect the IP to system by using standard bus is called “axi42sata” module. “axi42sata” module is imported to EDK system and displayed in IP Catalog in the left window, as shown in Figure 3-1. To interface with “axi42sata”, two standard bus interfaces are designed, i.e. AXI4 bus for data transferring with DDR3 and LMB bus for control register with MicroBlaze. In this system, clock frequency of AXI4 bus is equal to 200 MHz for synchronous with DDR3 frequency while LMB bus clock frequency is equal to 150 MHz for synchronous with MicroBlaze frequency.

3.2 AXI2SATA Folder

"axi42sata" stuff is stored into pcores\axi2sata_v1_00_a folder. The details of each file are follows.

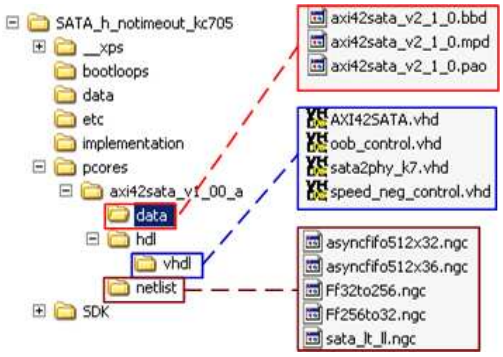

Figure 3-2 Folder/File List within axi42sata module

"data" folder

- axi42sata_v2_1_0.bbd: List the netlist files which is used within "axi42sata" module

```
1:  FILES
2:  asyncfifo512x32.ngc, asyncfifo512x36.ngc, Ff256to32.ngc, Ff32to256.ngc, sata_lt_ll.ngc
```

Figure 3-3 BBD file of axi42sata module

- axi42sata_v2_1_0.pao: Contain a list of HDL files

```
10:
11:  lib axi42sata_v1_00_a oob_control vhdl
12:  lib axi42sata_v1_00_a speed_neg_control vhdl
13:  lib axi42sata_v1_00_a sata2phy_k7 vhdl
14:  lib axi42sata_v1_00_a axi42sata vhdl
15:
```

Figure 3-4 PAO file of axi4sata module

- axi42sata_v2_1_0.mpd: Define the interface of peripheral such as port list, bus interface, parameter

```
BEGIN axi42sata

## Peripheral Options
OPTION IPTYPE = PERIPHERAL
OPTION IMP_NETLIST = TRUE
OPTION IP_GROUP = MICROBLAZE:USER
OPTION ARCH_SUPPORT_MAP = (OTHERS=DEVELOPMENT)
OPTION STYLE = MIX

## Bus Interfaces
BUS_INTERFACE BUS = M_AXI, BUS_STD = AXI, BUS_TYPE = MASTER
BUS_INTERFACE BUS = PORTA, BUS_STD = XIL_BRAM, BUS_TYPE = TARGET
#PARAMETER C_M_AXI_ADDR_WIDTH = 32, DT = INTEGER, BUS = M_AXI, ASSIGNMENT = CONSTANT
PARAMETER C_M_AXI_DATA_WIDTH = 256, DT = INTEGER, BUS = M_AXI, ASSIGNMENT = CONSTANT
#PARAMETER C_M_AXI_PROTOCOL = AXI4, TYPE = NON_HDL, ASSIGNMENT = CONSTANT, DT = STRING, BUS = M_AXI
#PARAMETER C_M_AXI_SUPPORTS_NARROW_BURST = 0, DT = INTEGER, BUS = M_AXI, ASSIGNMENT = CONSTANT

## Generics for VHDL or Parameters for Verilog

## Ports
PORT Clk            = "", DIR = I, SIGIS = CLK, BUS = M_AXI
PORT RstB           = ARESETN, DIR = I, BUS = M_AXI
PORT AXIArReady     = ARREADY, DIR = I, BUS = M_AXI
PORT AXIArValid     = ARVALID, DIR = O, BUS = M_AXI
 |
PORT AXIAwReady     = AWREADY, DIR = I, BUS = M_AXI
PORT AXIAwValid     = AWVALID, DIR = O, BUS = M_AXI
PORT AXIAwAddr      = AWADDR, DIR = O, VEC = [31:0], ENDIAN = LITTLE, BUS = M_AXI
 |
PORT BRAM_Rst_A     = BRAM_Rst, DIR = I, BUS = PORTA
PORT BRAM_Clk_A     = BRAM_Clk, DIR = I, BUS = PORTA, SIGIS = CLK
PORT BRAM_EN_A      = BRAM_EN, DIR = I, BUS = PORTA
 |
PORT MGTCLK_N       = "", DIR = I
PORT MGTCLK_P       = "", DIR = I
PORT MGT_RXP        = "", DIR = I
PORT MGT_RXN        = "", DIR = I
PORT MGT_TXP        = "", DIR = O
PORT MGT_TXN        = "", DIR = O
```

Figure 3-5 MPD file of axi4sata module

"hdl/vhdl" folder

- AXI42SATA.vhd: Top module of axi42sata which includes SATA-IP, SATA-PHY, and additional logic for bus interface
- sata2phy_k7.vhd: SATA-PHY hdl code which includes speed negotiation and OOB control
- speed_neg_control.vhd: HDL code for auto speed negotiation function
- oob_control.vhd: HDL code for OOB timing control

"netlist" folder

- asyncfifo512x32/512x36.ngc: netlist within SATA-IP
- sata_lt_ll.ngc: SATA-IP netlist
- Ff256to32/32to256.ngc: FIFO netlist which uses within AXI42SATA for converting bus size

3.3 How to update AXI2SATA

If demo project is designed by using Vivado, Step 6) – 8) can be skipped and change to start re-implement on Vivado tool instead.

To update “axi42sata” module, user can follow the below steps.

1) Close EDK tools if the project is opened.
2) Update AXI42SATA.vhd code
3) Add/Remove port list in “axi42sata_v2_1_0.mpd” when port list is changed. Then, open “system.mhs” file to add/remove connection of “axi42sata” with system.
4) If any hdl code is added into sub module of “axi42sata”, please store new hdl code to “hdl/vhdl” folder and store new netlist to “netlist” folder. Then, update file list within “axi42sata_v2_1_0.bbd” and “axi42sata_v2_1_0.pao”.
5) Open “system.xmp” by EDK and check all system connections.
6) Select Hardware->Clean Hardware menu to clean up previous backup file

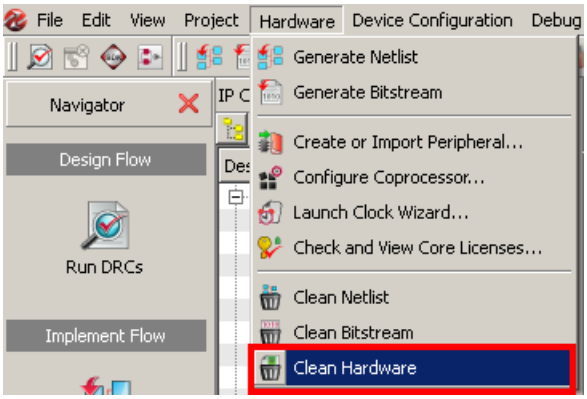

Figure 3-6 Clean Hardware menu

7) Select Device Configuration->Update Bitstream to start re-implementation

Figure 3-7 Update bitstream

8) Wait until implementation complete. “download.bit” configuration file will be updated and stored in “implementation” folder.

4. SDK Tool

4.1 Export from EDK Tool

This topic describes about how to generate new SDK project from EDK tool. If user only updates "sata_host.c", step 3) – 10) can be skipped.

1) Export hardware from EDK to SDK workspace by selecting "Export Design" icon and "Export & Launch SDK".

Figure 4-1 Export Hardware to SDK

2) Select "workspace" folder. In demo project, "workspace" is subfolder within "SDK" folder. Then, hardware platform will be exported to project explorer of SDK, as shown in Figure 4-3.

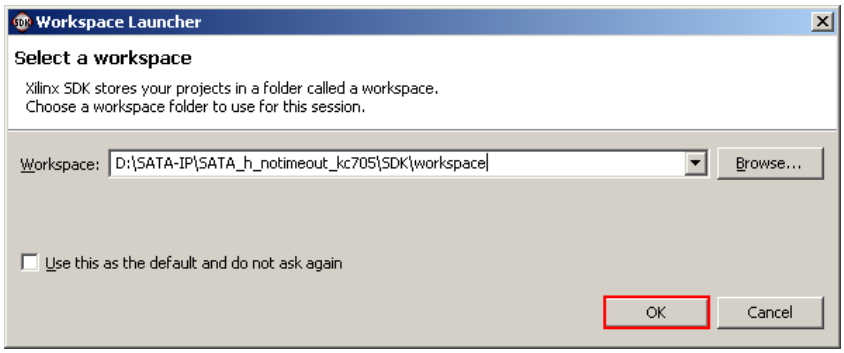

Figure 4-2 Select SDK workspace

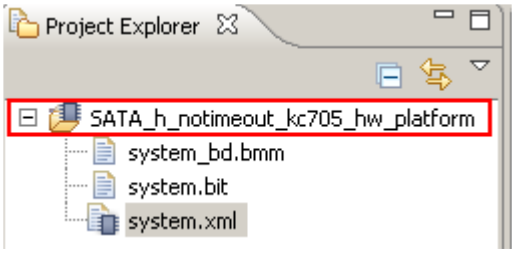

Figure 4-3 Hardware platform in SDK

3) Generate “Board Support Package” by selecting File->New->Xilinx Board Support Package. Select “standalone” and then click “Finish”.

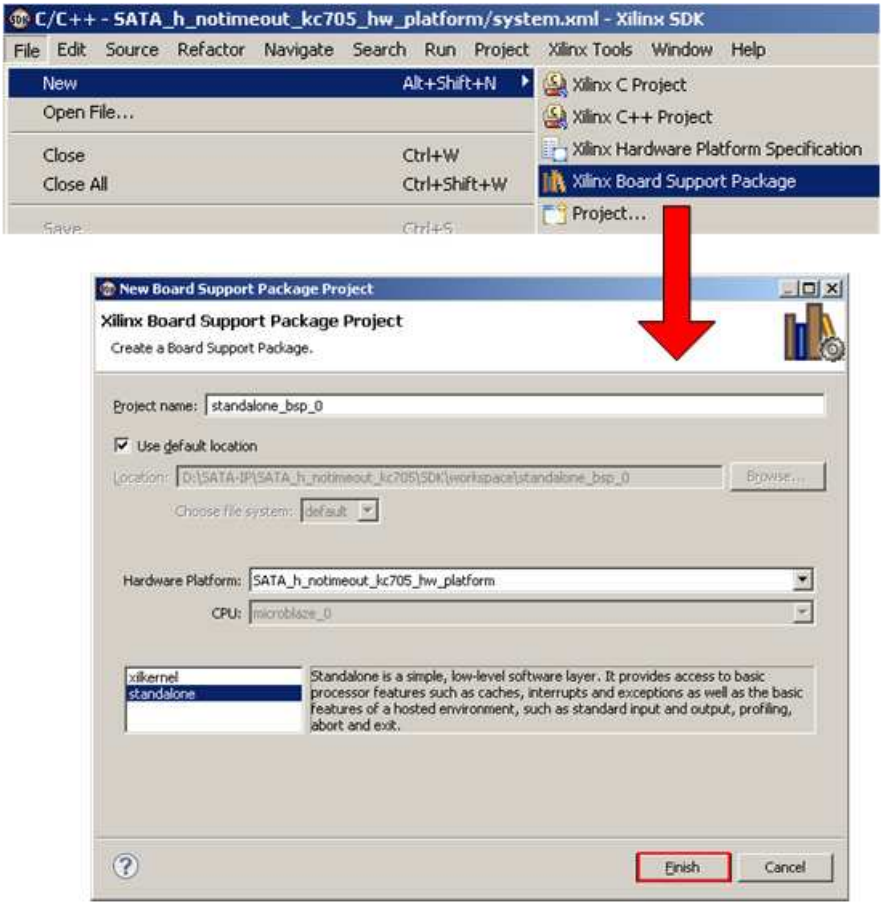

Figure 4-4 BSP Generating

4) In Library option, click OK after setting complete. After that, bsp folder will be generated and displayed on project explorer of SDK, as shown in Figure 4-5.

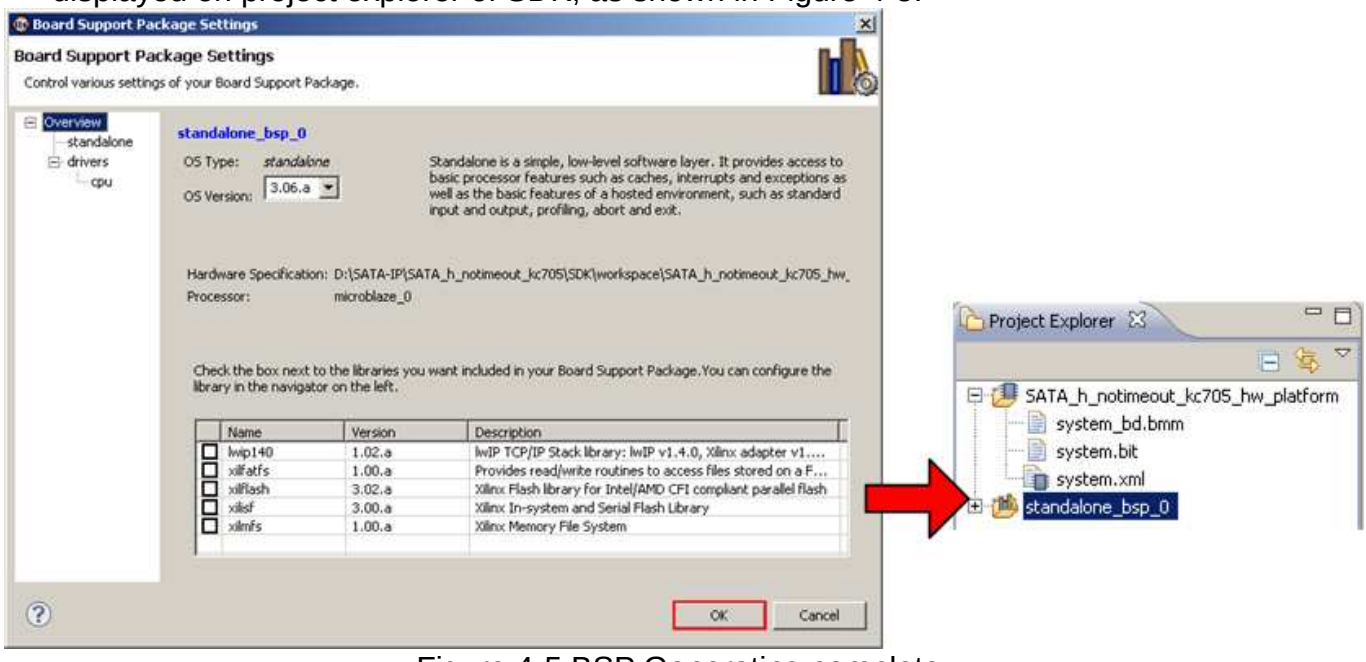

Figure 4-5 BSP Generating complete

5) Create new C Project by selecting File->New->Xilinx C Project, as shown in Figure 4-6.

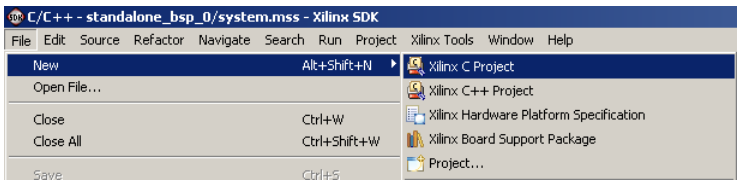

Figure 4-6 New C Project

6) Select “Empty Application” from Project Template, and type project name such as “sata_host” to Project name box. Click “Next” button to continue next step.

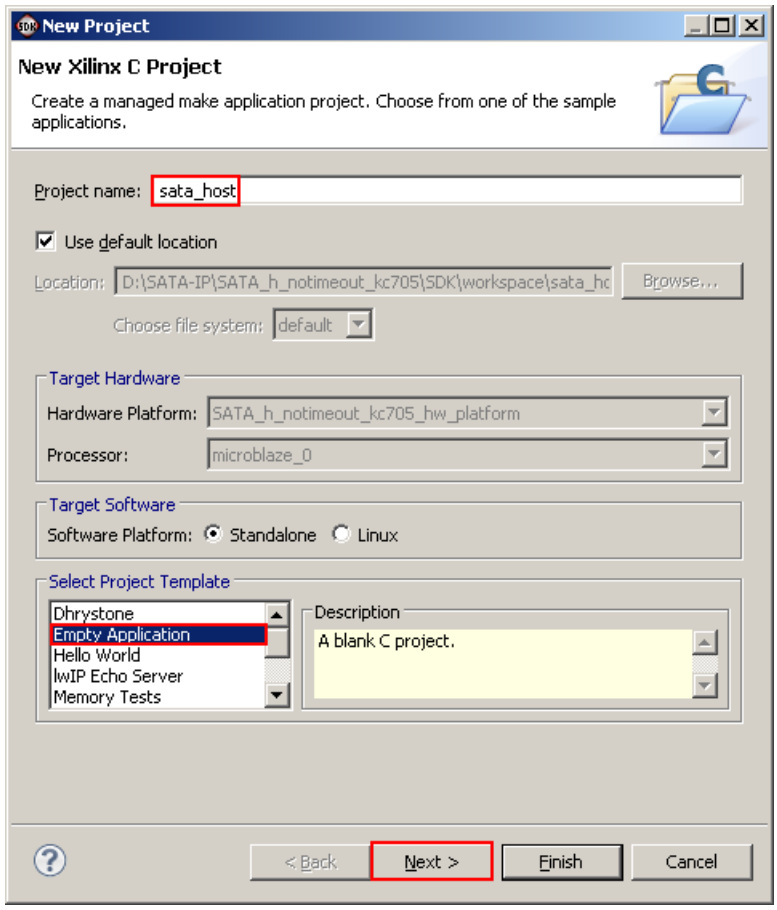

Figure 4-7 Define Project name

7) Click to select “Target an existing Board Support Package” and select “standalone_bsp_0”. After that, click “Finish” to complete setup. New project (“sata_host”) will be displayed on project explorer, as shown in Figure 4-8.

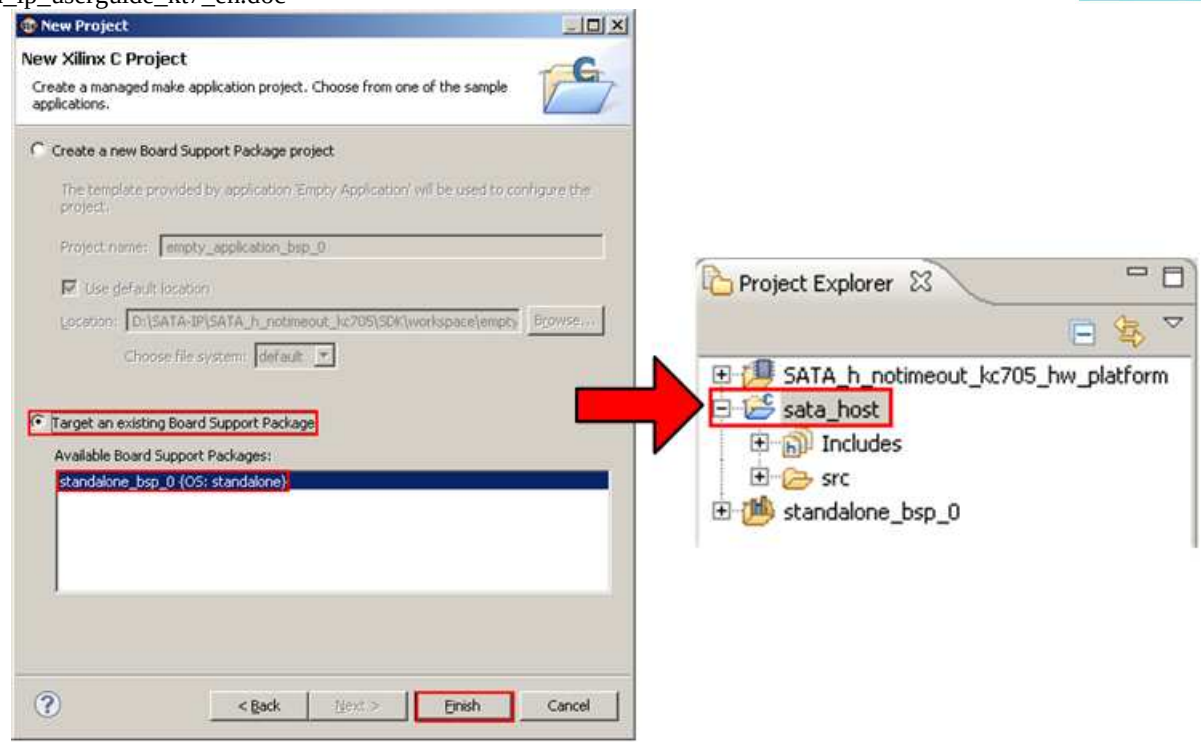

Figure 4-8 Selecting BSP for C Project

8) Create new C source code by right click at project name folder ("sata_host") in project explorer->New->File.

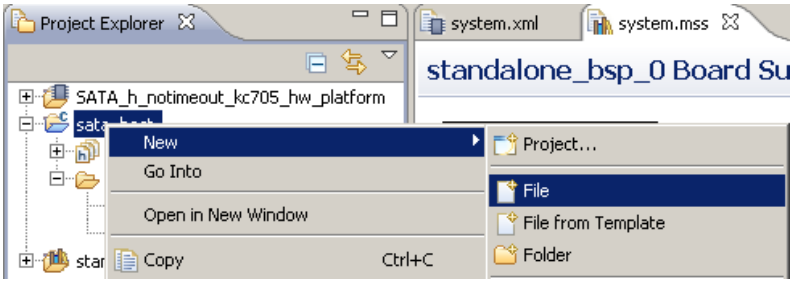

Figure 4-9 Create new C file

9) Select "src" folder and type file name such as "sata_host.c". Click "Finish" to complete, and new C code (sata_host.c) will be displayed on project explorer in "src" folder, as shown in Figure 4-10.

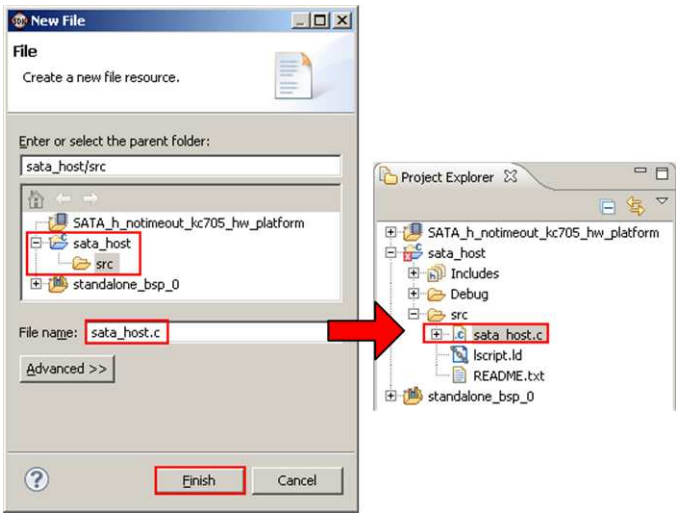

Figure 4-10 Add C source code

10) Edit new C source code ("sata_host.c"). For 1st time usage, user needs to generate a linker script by right click at project name ("sata_host") and select "Geneate Linker Script". Then, click "Generate" button in new window to complete.

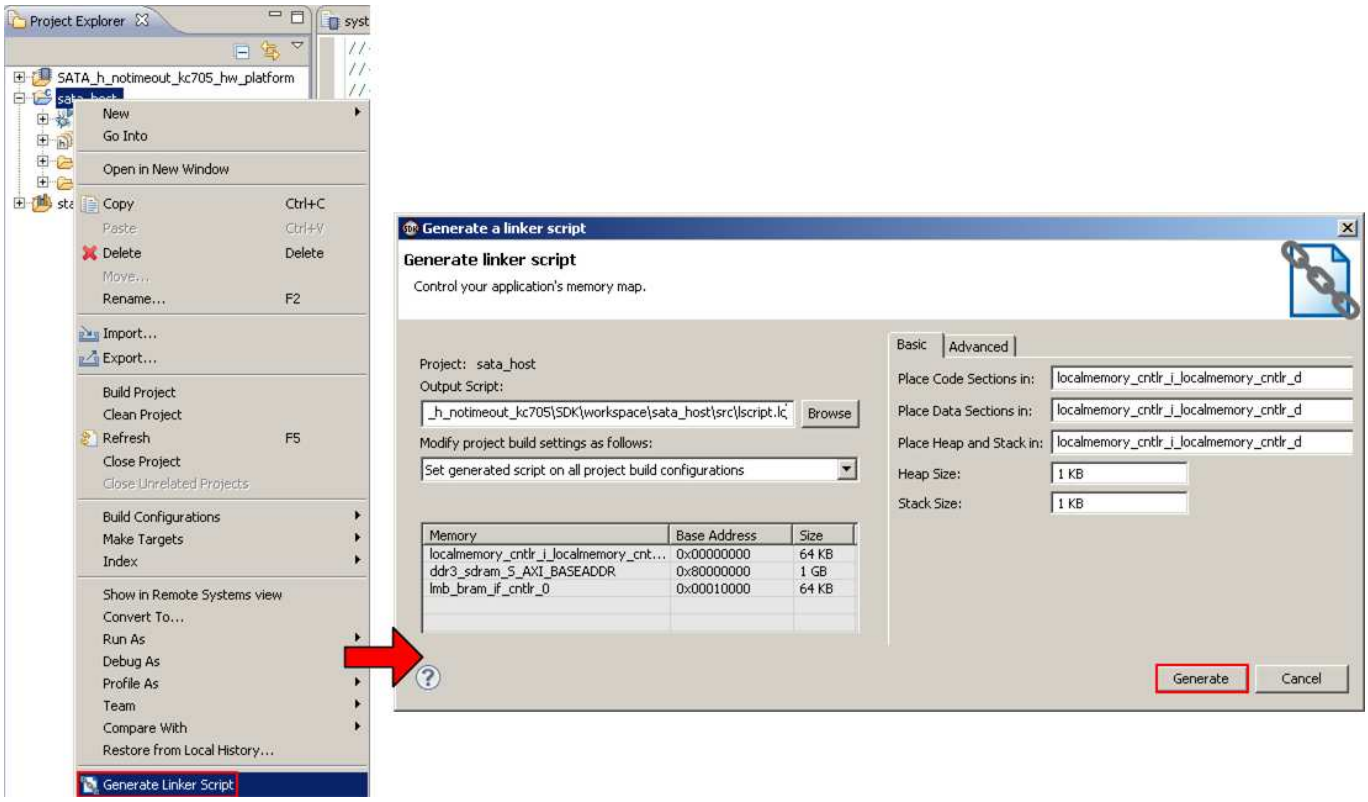

Figure 4-11 Generate Linker Script

11) After user updates source code ("sata_host.c"), user can re-compile project by selecting Refresh to update code and start re-compile operation.

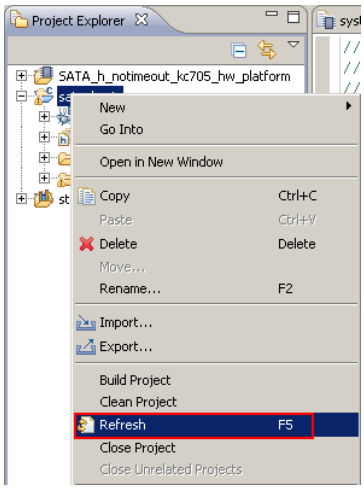

Figure 4-12 Refresh Project

12) To run on real board, select Xilinx Tools->Program FPGA. Then, press browse to select "system.bit" and "system_bd.bmm" file from "hw_platform" folder and change elf file from bootloop to new file such as "sata_host.elf". Click Program to start FPGA programming.

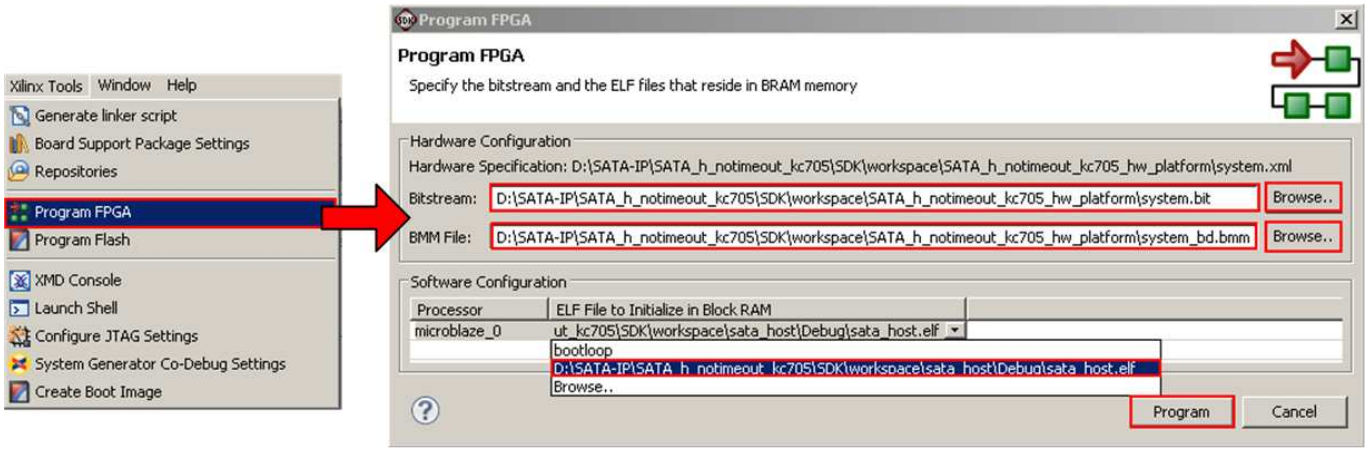

Figure 4-13 FPGA Programming on SDK

13) Wait programming complete and new firmware will be updated to FPGA.

4.2 Export from Vivado Tool

This topic describes about how to generate new SDK project from Vivado tool.

1) Before exporting hardware, select “Implemented Design” menu to open design details in Vivado.
2) Export hardware from Vivado to SDK workspace by selecting File->Export->Export Harware for SDK… and then check box for “Launch SDK” following before click “OK” button. After this step, “system_stub.sdk” folder will be created in project directory and SDK workspace is defined to “../SDK/SDK_Export” path.

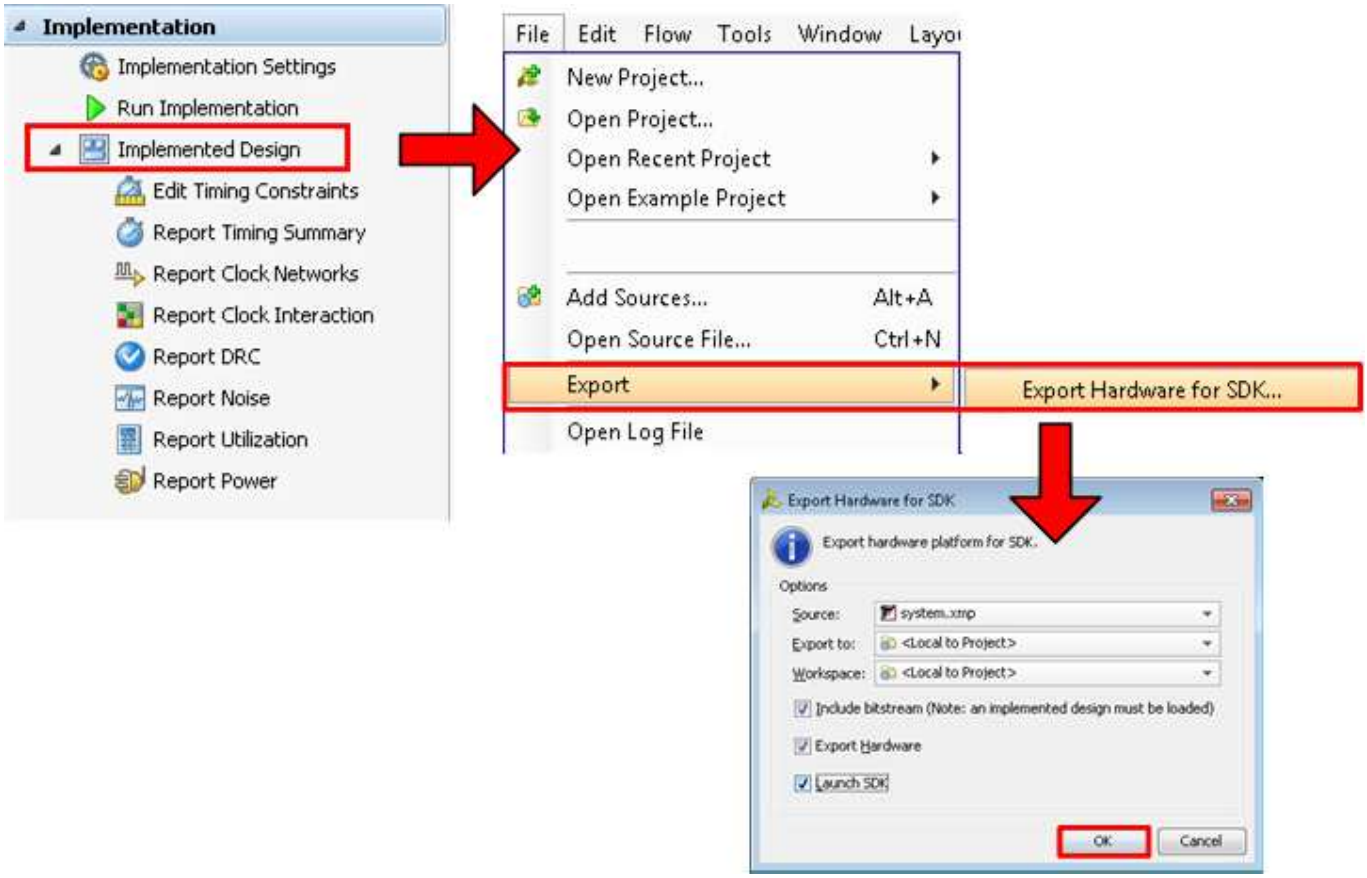

Figure 4-14 Export for SDK by Vivado Tool

3) Continue next step by following step 3) in Topic 4.1 “Export from EDK Tool”. The step for both EDK and Vivado is similar, but SDK workspace path is different.

5. Revision History

Revision	Date	Description
1.0	18-Feb-13	Initial release
1.1	06-Mar-13	Support Vivado tool