EBAZ4205 ZYNQ 7Z010 bare metal program NAND curing JTAG debugging method

EBAZ4205 is ebit's control board, the price is cheap. EBAZ4205 uses XILINX XC7Z010-1CLG400I soc which contains two hard core ARM A9 and ARTIX-7 logic. The board has a 128M x 16 bit DDR3 CLK800Mhz, a 128M byte NAND FLASH, PS 33.333Mhz osc, MII PHY, two LEDs, and three headers that are connected to the PL. Two buttons, SDIO, NAND, UART, etc. are connected to the PS. Note that the PHY is connected to the FPGA. The board power supply uses 12v. By default, the board is connected to power from three headers. There is no soldering diode on the J4 row seat. You can use J4 to supply power after a short circuit. The board also needs to be soldered on the UART JTAG header. Don't use SD card and PHY for now.

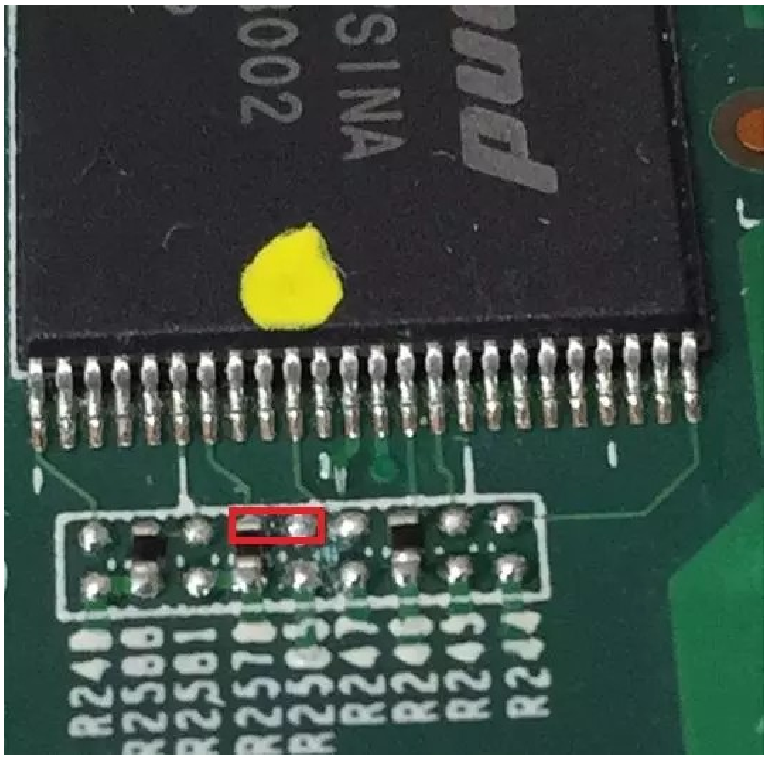
Schematic and PCB: https://github.com/Elrori/EBAZ4205

If you use PHY, refer to: https://hhuysqt.github.io/zynq1/ . In step 4, establish the EMIO of the PHY, and lead the IO, and then constrain it.

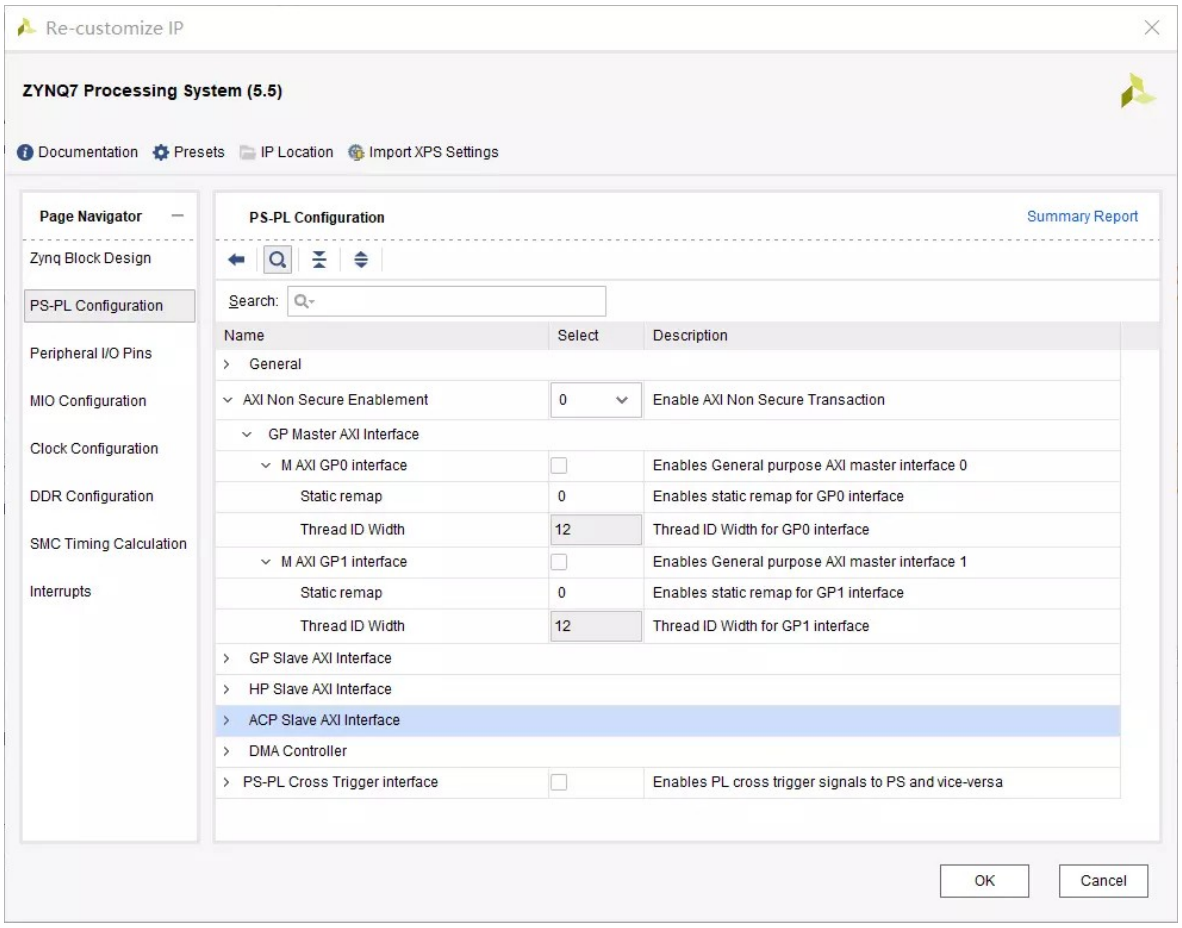
The ZYNQ SOC startup process is based on ARM. After power-on, the hardware reads the PS IO port to determine whether to start from NAND, QSPI-FLASH, SD Card or JTAG. The board is booted from NAND by default. For software debugging and downloading NAND, it must be booted from JTAG.

0. Prepare the board, connect UART, XILINX JTAG

1. Connect the end of resistor R2578 close to NAND to ground before powering up, and then connect the power supply to start from JTAG.



2. Create a new vivado project of the device, create a schematic (IP Integrator, Create Block Design), click "+" to search for zynq7 processing system to add, remove AXI:


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator
- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

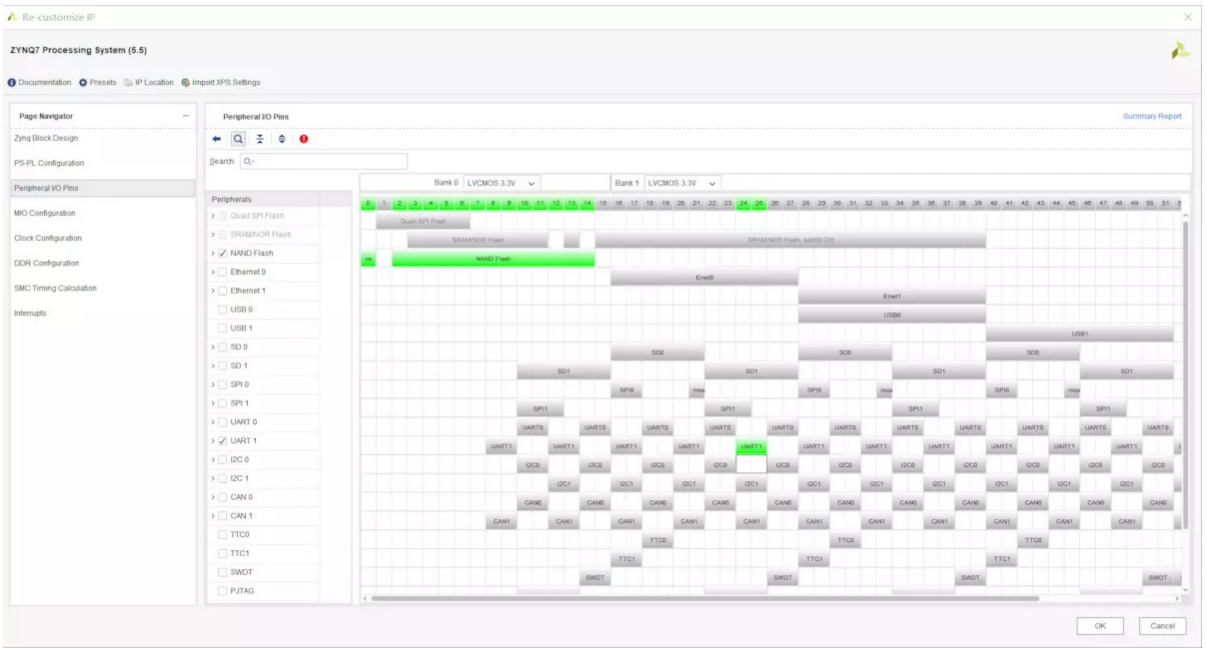
PS-PL Configuration Summary Report

Search:

Name	Select	Description
› General		
⌄ AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
⌄ GP Master AXI Interface		
⌄ M AXI GP0 interface	☐	Enables General purpose AXI master interface 0
Static remap	0	Enables static remap for GP0 interface
Thread ID Width	12	Thread ID Width for GP0 interface
⌄ M AXI GP1 interface	☐	Enables General purpose AXI master interface 1
Static remap	0	Enables static remap for GP1 interface
Thread ID Width	12	Thread ID Width for GP1 interface
› GP Slave AXI Interface		
› HP Slave AXI Interface		
› ACP Slave AXI Interface		
› DMA Controller		
› PS-PL Cross Trigger interface	☐	Enables PL cross trigger signals to PS and vice-versa

OK Cancel

3. Peripheral IO settings use NAND UART1 (default baud rate 115200), according to the following figure:


Re-customize IP

ZYNQ7 Processing System (5.5)

Documentation Presets IP Location Import XPS Settings

Page Navigator
- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

Peripheral I/O Pins Summary Report

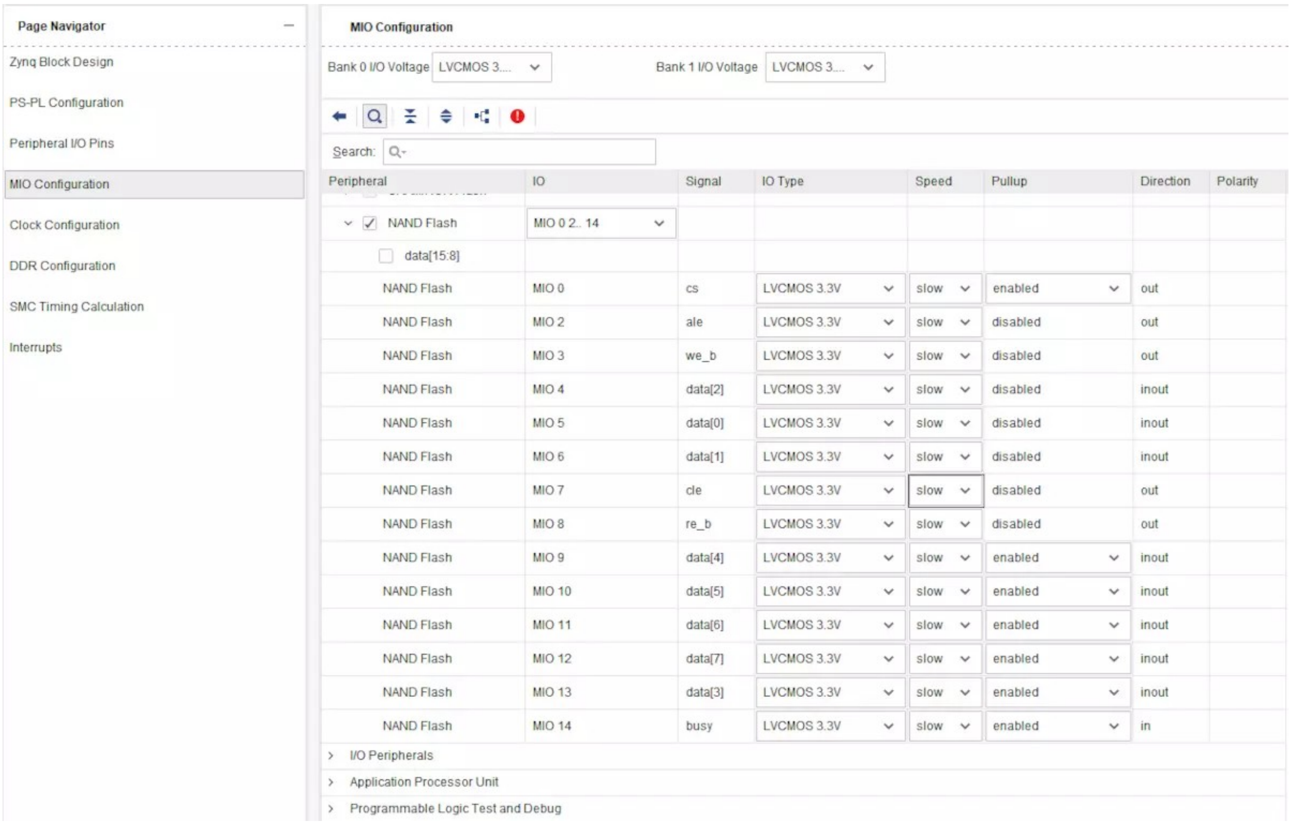
Search:

Bank 0 LVCMOS 3.3V Bank 1 LVCMOS 3.3V

Peripherals
- ☐ Quad SPI Flash
- ☐ SRAM/NOR Flash
- ☑ NAND Flash
- ☐ Ethernet 0
- ☐ Ethernet 1
- ☐ USB 0
- ☐ USB 1
- ☐ SD 0
- ☐ SD 1
- ☐ SPI 0
- ☐ SPI 1
- ☐ UART 0
- ☑ UART 1
- ☐ I2C 0
- ☐ I2C 1
- ☐ CAN 0
- ☐ CAN 1
- ☐ TTC0
- ☐ TTC1
- ☐ SWDT
- ☐ PJTAG

OK Cancel

4. MIO configuration, default: (If you use PHY, you need to check ENET0, and use EMIO to export from PL, see githubio)



MIO Configuration

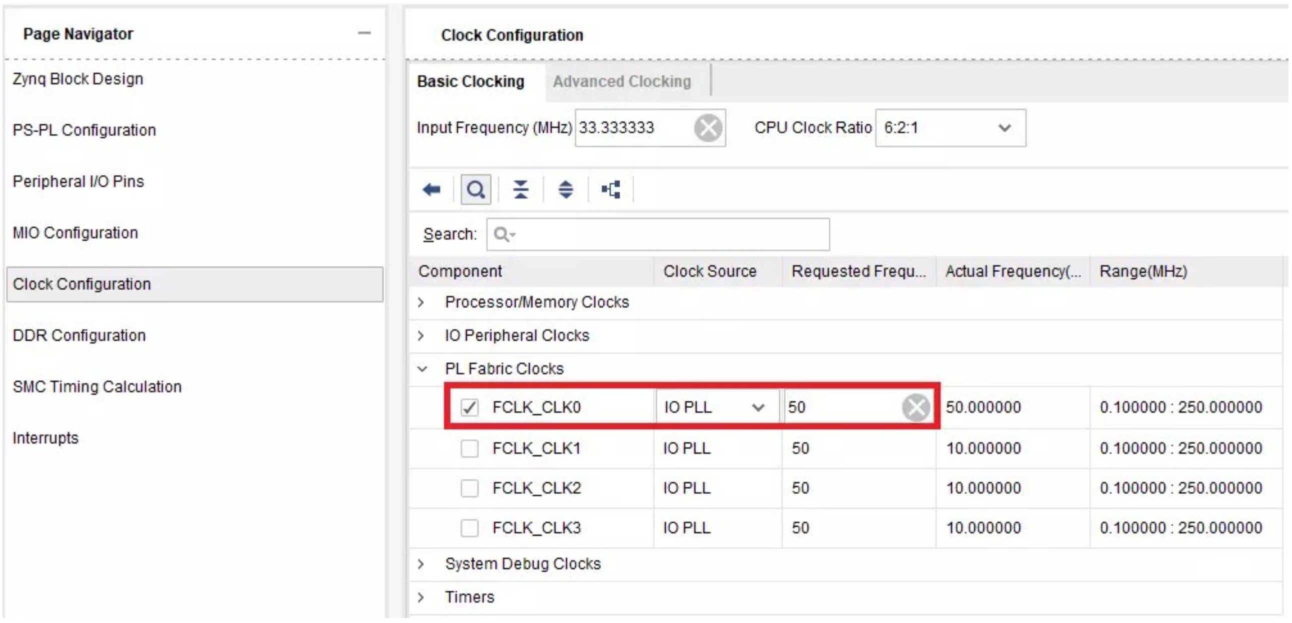
Bank 0 I/O Voltage LVCMOS 3.... Bank 1 I/O Voltage LVCMOS 3....

Search:

Peripheral	IO	Signal	IO Type	Speed	Pullup	Direction	Polarity
✓ NAND Flash	MIO 0 2.. 14						
data[15:8]							
NAND Flash	MIO 0	cs	LVCMOS 3.3V	slow	enabled	out	
NAND Flash	MIO 2	ale	LVCMOS 3.3V	slow	disabled	out	
NAND Flash	MIO 3	we_b	LVCMOS 3.3V	slow	disabled	out	
NAND Flash	MIO 4	data[2]	LVCMOS 3.3V	slow	disabled	inout	
NAND Flash	MIO 5	data[0]	LVCMOS 3.3V	slow	disabled	inout	
NAND Flash	MIO 6	data[1]	LVCMOS 3.3V	slow	disabled	inout	
NAND Flash	MIO 7	cle	LVCMOS 3.3V	slow	disabled	out	
NAND Flash	MIO 8	re_b	LVCMOS 3.3V	slow	disabled	out	
NAND Flash	MIO 9	data[4]	LVCMOS 3.3V	slow	enabled	inout	
NAND Flash	MIO 10	data[5]	LVCMOS 3.3V	slow	enabled	inout	
NAND Flash	MIO 11	data[6]	LVCMOS 3.3V	slow	enabled	inout	
NAND Flash	MIO 12	data[7]	LVCMOS 3.3V	slow	enabled	inout	
NAND Flash	MIO 13	data[3]	LVCMOS 3.3V	slow	enabled	inout	
NAND Flash	MIO 14	busy	LVCMOS 3.3V	slow	enabled	in	

> I/O Peripherals
> Application Processor Unit
> Programmable Logic Test and Debug

5. Clock configuration, use FCLK_CLK0 for PS to PL clock, according to the following figure:



Clock Configuration

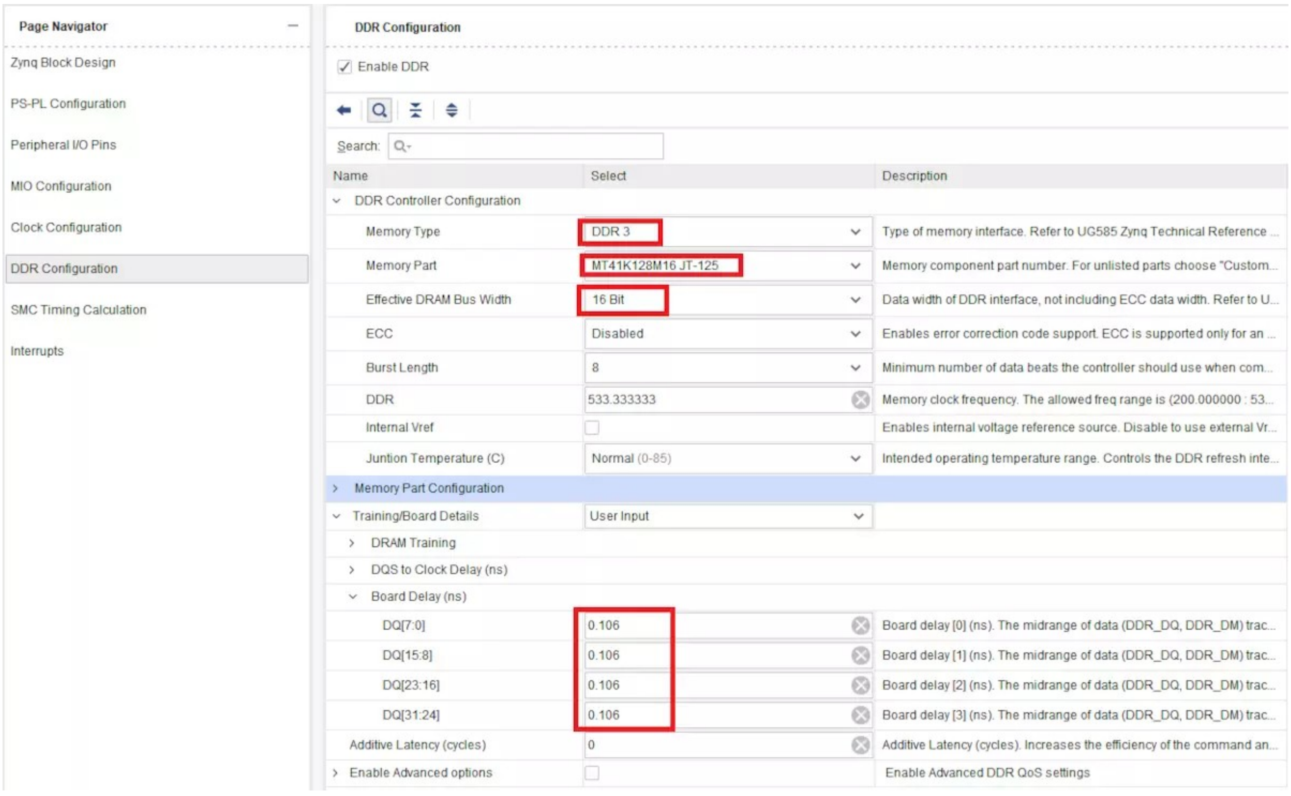
Basic Clocking | Advanced Clocking

Input Frequency (MHz) 33.333333 CPU Clock Ratio 6:2:1

Search:

Component	Clock Source	Requested Frequ...	Actual Frequency(...	Range(MHz)
> Processor/Memory Clocks				
> IO Peripheral Clocks				
PL Fabric Clocks				
✓ FCLK_CLK0	IO PLL	50	50.000000	0.100000 : 250.000000
FCLK_CLK1	IO PLL	50	10.000000	0.100000 : 250.000000
FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000
FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000
> System Debug Clocks				
> Timers				

6. DDR3 configuration, DDR3 uses Micron 128M16 instead, the effective data width is 16bits (single DDR3), the data line PCB delay is set to 0.106, according to the following figure:


Page Navigator

- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

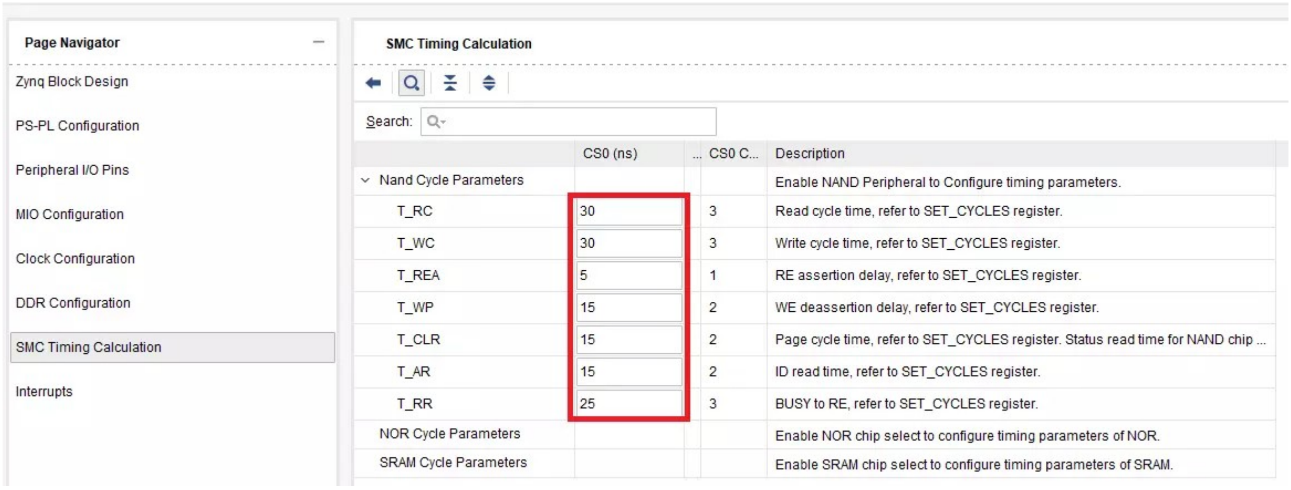
DDR Configuration

☑ Enable DDR

Search:

Name	Select	Description
DDR Controller Configuration		
Memory Type	DDR 3	Type of memory interface. Refer to UG585 Zynq Technical Reference ...
Memory Part	MT41K128M16 JT-125	Memory component part number. For unlisted parts choose "Custom...
Effective DRAM Bus Width	16 Bit	Data width of DDR interface, not including ECC data width. Refer to U...
ECC	Disabled	Enables error correction code support. ECC is supported only for an ...
Burst Length	8	Minimum number of data beats the controller should use when com...
DDR	533.333333	Memory clock frequency. The allowed freq range is (200.000000 : 53...
Internal Vref	☐	Enables internal voltage reference source. Disable to use external Vr...
Juntion Temperature (C)	Normal (0-85)	Intended operating temperature range. Controls the DDR refresh inte...
Memory Part Configuration		
Training/Board Details	User Input	
DRAM Training		
DQS to Clock Delay (ns)		
Board Delay (ns)		
DQ[7:0]	0.106	Board delay [0] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[15:8]	0.106	Board delay [1] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[23:16]	0.106	Board delay [2] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
DQ[31:24]	0.106	Board delay [3] (ns). The midrange of data (DDR_DQ, DDR_DM) trac...
Additive Latency (cycles)	0	Additive Latency (cycles). Increases the efficiency of the command an...
Enable Advanced options	☐	Enable Advanced DDR QoS settings

7. SMC timing, NAND timing according to the following figure:


Page Navigator

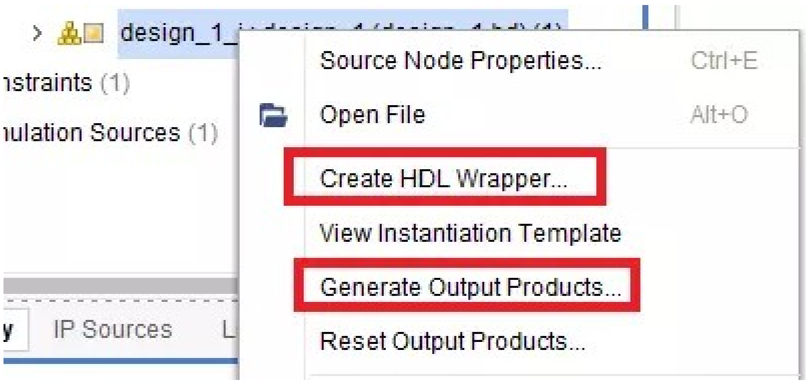
- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

SMC Timing Calculation

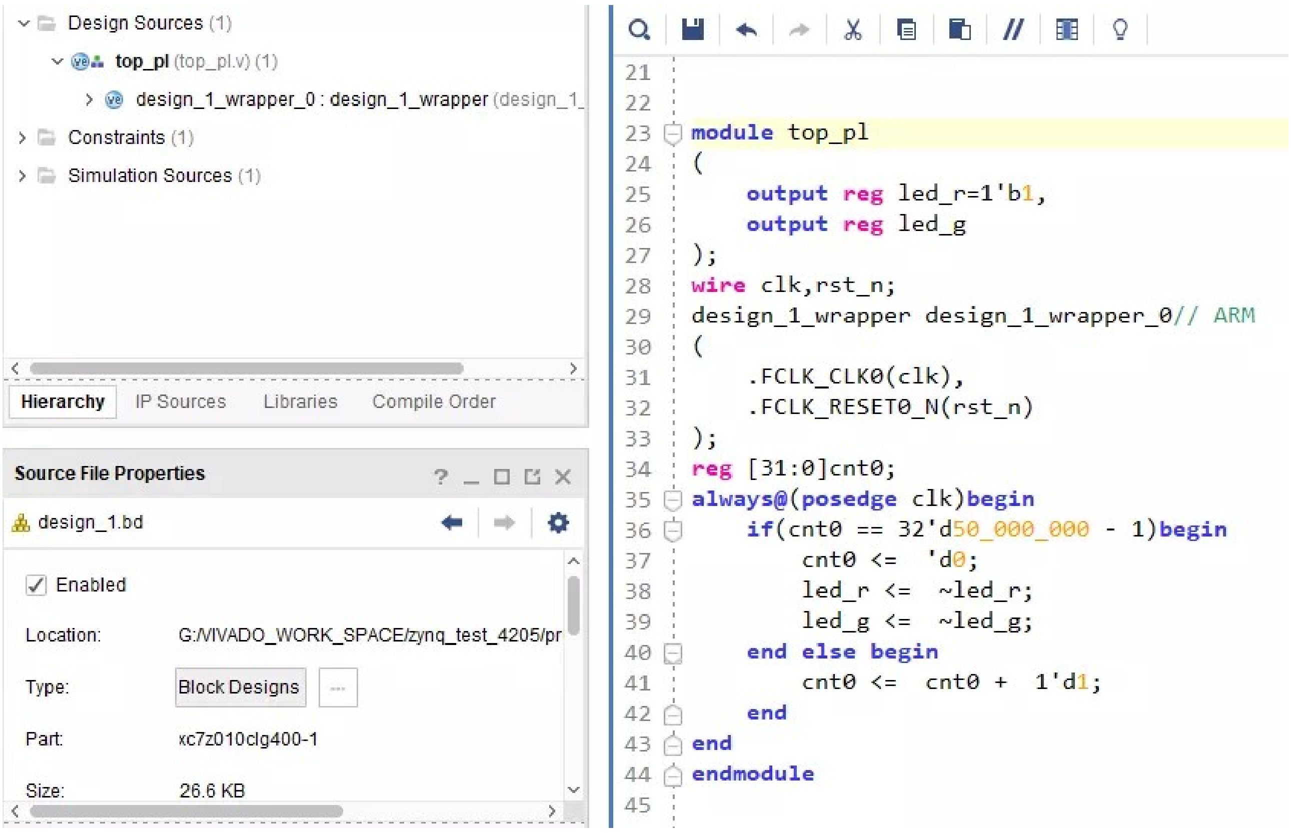
Search:

	CS0 (ns)	... CS0 C...	Description
Nand Cycle Parameters			Enable NAND Peripheral to Configure timing parameters.
T_RC	30	3	Read cycle time, refer to SET_CYCLES register.
T_WC	30	3	Write cycle time, refer to SET_CYCLES register.
T_REA	5	1	RE assertion delay, refer to SET_CYCLES register.
T_WP	15	2	WE deassertion delay, refer to SET_CYCLES register.
T_CLR	15	2	Page cycle time, refer to SET_CYCLES register. Status read time for NAND chip ...
T_AR	15	2	ID read time, refer to SET_CYCLES register.
T_RR	25	3	BUSY to RE, refer to SET_CYCLES register.
NOR Cycle Parameters			Enable NOR chip select to configure timing parameters of NOR.
SRAM Cycle Parameters			Enable SRAM chip select to configure timing parameters of SRAM.

8. Click OK, right-click the schematic zynq7 and click create ports, and then check the availability. Right-click the design in the source to create the package and output, according to the following figure:


design_1_i : design_1 (design_1.bd) (1)
nstraints (1)
nulation Sources (1)
IP Sources
Source Node Properties... Ctrl+E
Open File Alt+O
Create HDL Wrapper...
View Instantiation Template
Generate Output Products...
Reset Output Products...

9. Create top_pl.v at the top of verilog for lighting. The design includes the following:



Enabled: ✓

Location: G:/VIVADO_WORK_SPACE/zynq_test_4205/pr

Type: Block Designs

Part: xc7z010clg400-1

Size: 26.6 KB

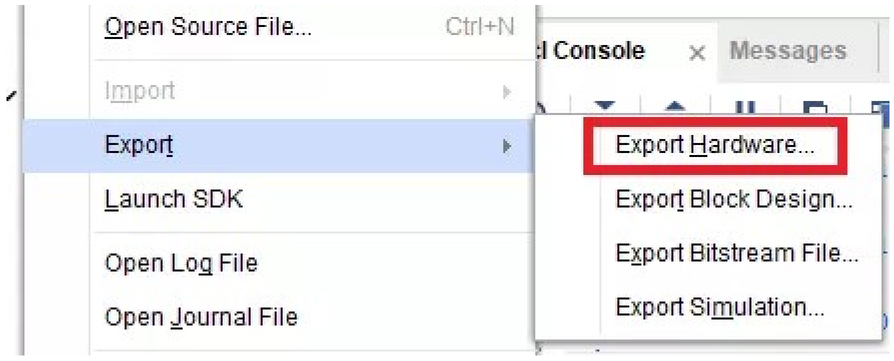
```verilog

module top_pl
(
    output reg led_r=1'b1,
    output reg led_g
);
wire clk,rst_n;
design_1_wrapper design_1_wrapper_0// ARM
(
    .FCLK_CLK0(clk),
    .FCLK_RESET0_N(rst_n)
);
reg [31:0]cnt0;
always@(posedge clk)begin
    if(cnt0 == 32'd50_000_000 - 1)begin
        cnt0 <=  'd0;
        led_r <=  ~led_r;
        led_g <=  ~led_g;
    end else begin
        cnt0 <=  cnt0 +  1'd1;
    end
end
endmodule

```

Pin constraints, xdc file content: (see githubio for PHY pins)

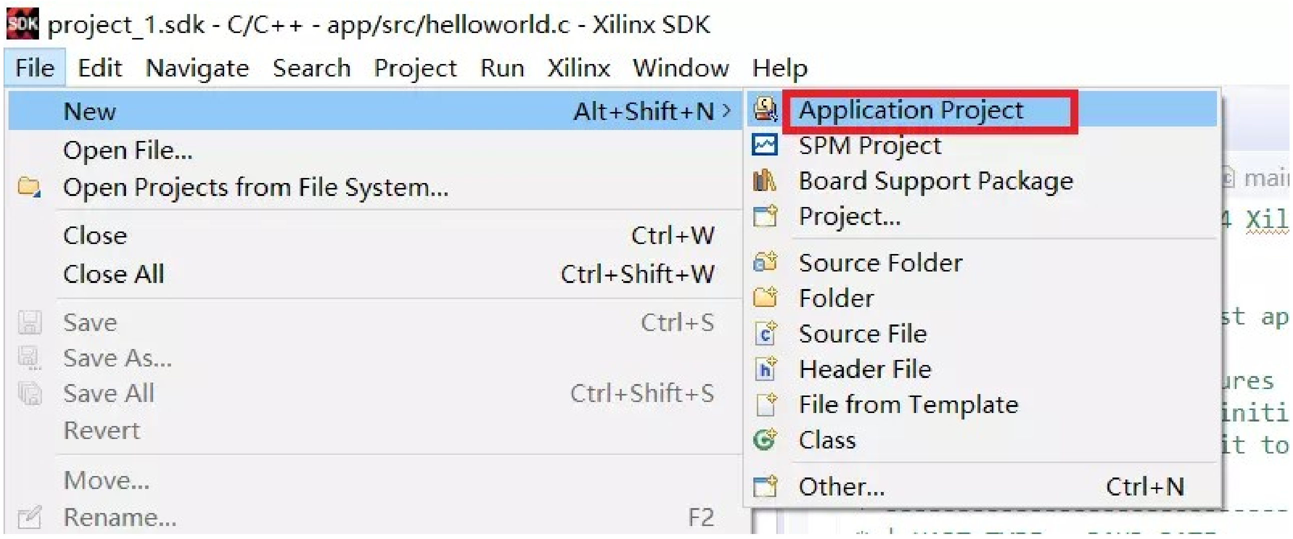
```
set_property PACKAGE_PIN W13 [get_ports {led_r}]

set_property PACKAGE_PIN W14 [get_ports {led_g}]

set_property IOSTANDARD LVCMOS33 [get_ports {led_r}]

set_property IOSTANDARD LVCMOS33 [get_ports {led_g}]
```

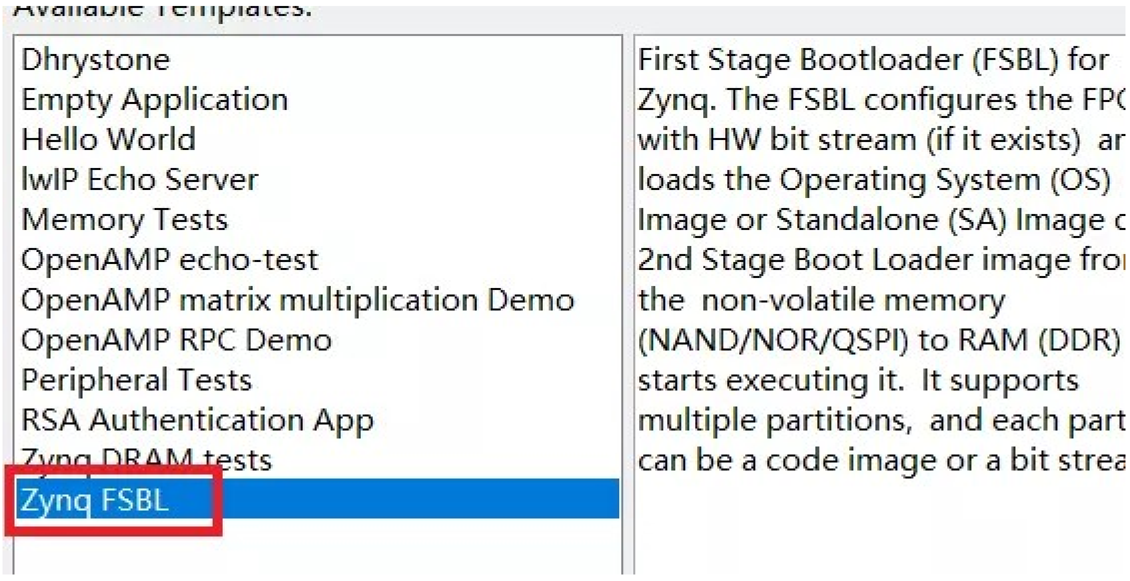
10. generate bitstream to generate .bit files

11. file->export hardware, tick the .bit file:



12. File->launch SDK, create the following in the SDK, then set the project name fsbl, click next to select the FSBL template

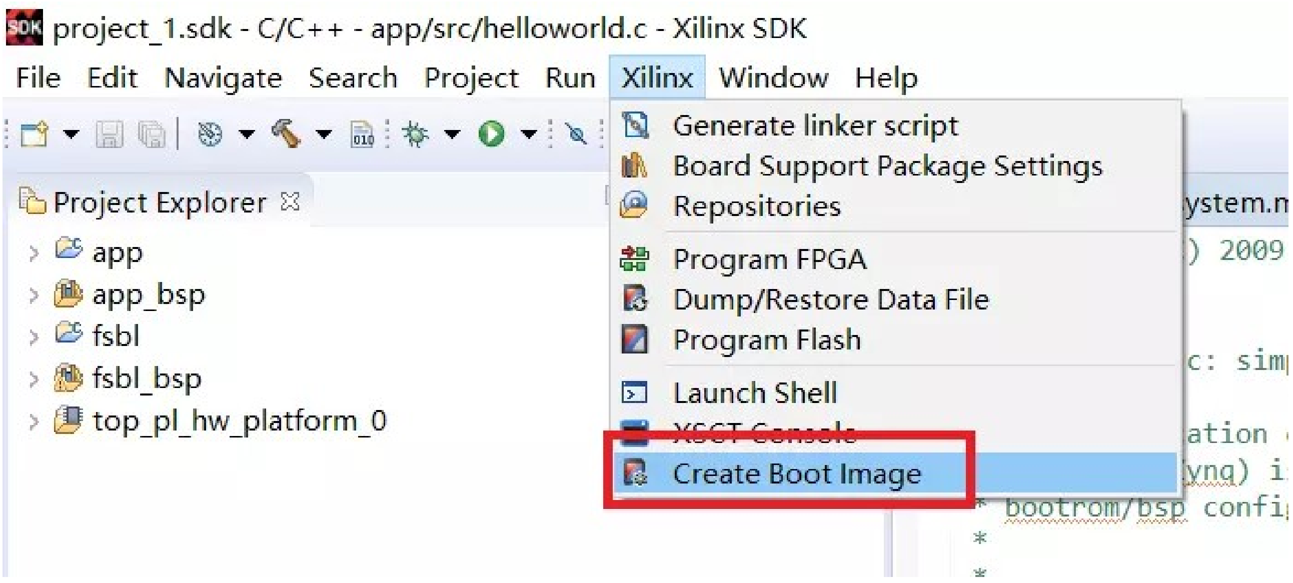


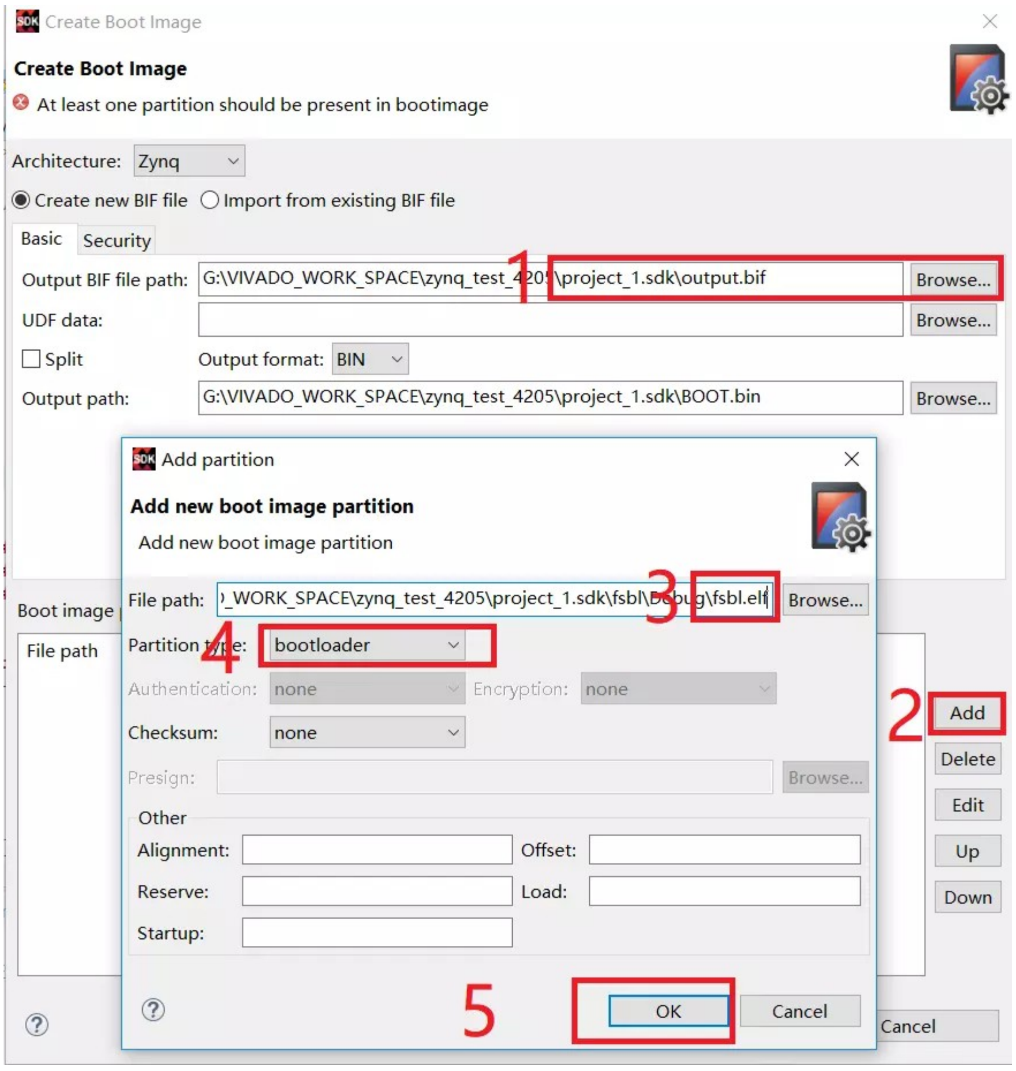


First Stage Bootloader (FSBL) for Zynq. The FSBL configures the FP(with HW bit stream (if it exists) ar loads the Operating System (OS) Image or Standalone (SA) Image c 2nd Stage Boot Loader image fro the non-volatile memory (NAND/NOR/QSPI) to RAM (DDR) starts executing it. It supports multiple partitions, and each part can be a code image or a bit strea

13. In the same way, create a new project named app in the SDK and use the hello world template. The bare metal program is in this project, and the fsbl project is used to initialize ddr and so on. Save and compile two elf, namely app.elf fsbl.elf

14. Make BOOT.bin for curing to NAND. BOOT.bin consists of three files: fsbl.elf top_pl.bit app.elf.


project_1.sdk - C/C++ - app/src/helloworld.c - Xilinx SDK


Create Boot Image

Create Boot Image

At least one partition should be present in bootimage

Architecture: Zynq

(●) Create new BIF file ( ) Import from existing BIF file

Basic | Security

Output BIF file path: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\output.bif

UDF data:

[ ] Split   Output format: BIN

Output path: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\BOOT.bin

Add partition

Add new boot image partition

Add new boot image partition

File path: _WORK_SPACE\zynq_test_4205\project_1.sdk\fsbl\Debug\fsbl.elf

Partition type: bootloader

Authentication: none   Encryption: none

Checksum: none

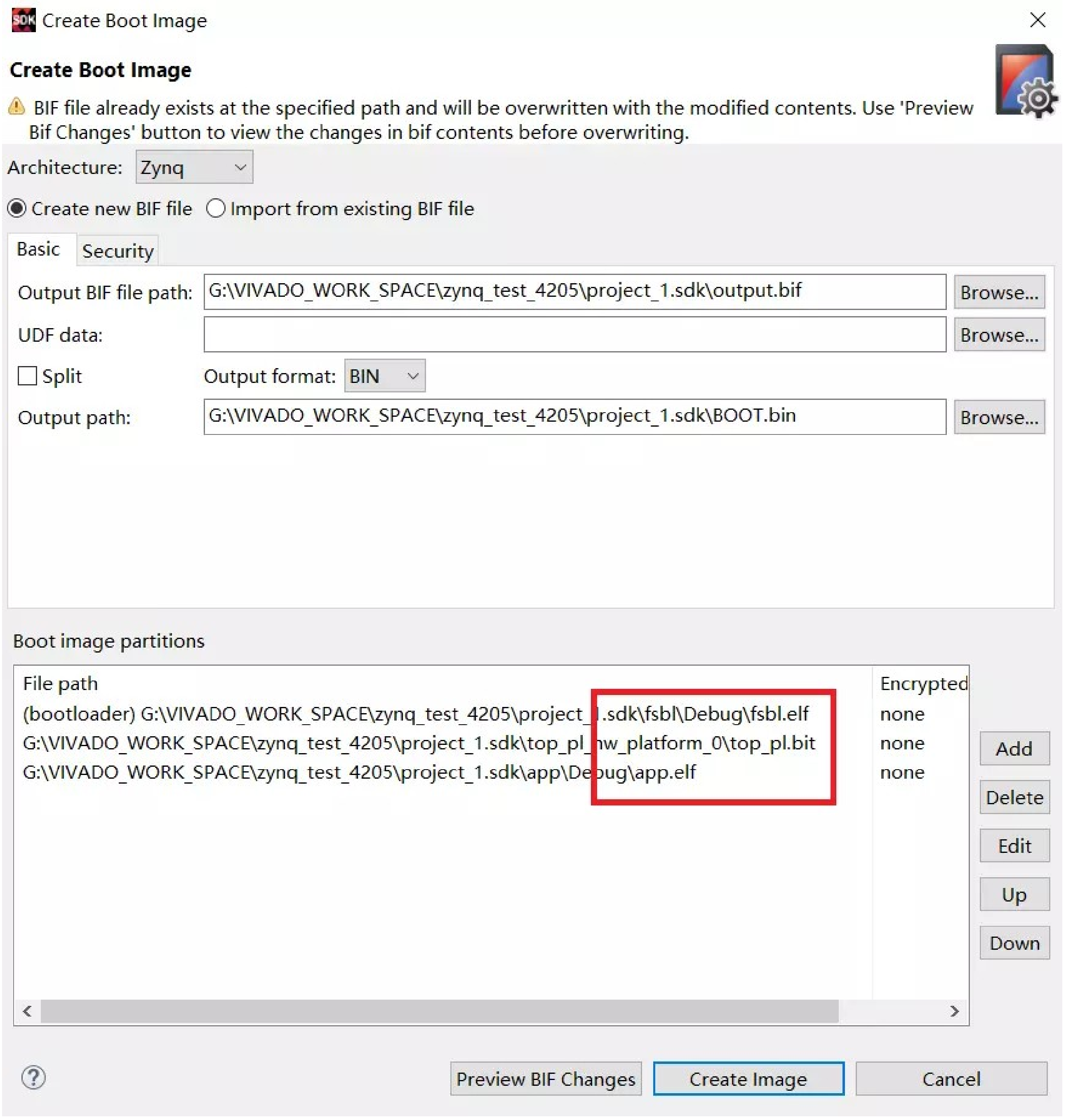
Presign:

Other

Alignment: Offset:

Reserve: Load:

Startup:


Create Boot Image

Create Boot Image

BIF file already exists at the specified path and will be overwritten with the modified contents. Use 'Preview Bif Changes' button to view the changes in bif contents before overwriting.

Architecture: Zynq

(●) Create new BIF file ( ) Import from existing BIF file

Basic | Security

Output BIF file path: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\output.bif

UDF data:

[ ] Split Output format: BIN

Output path: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\BOOT.bin

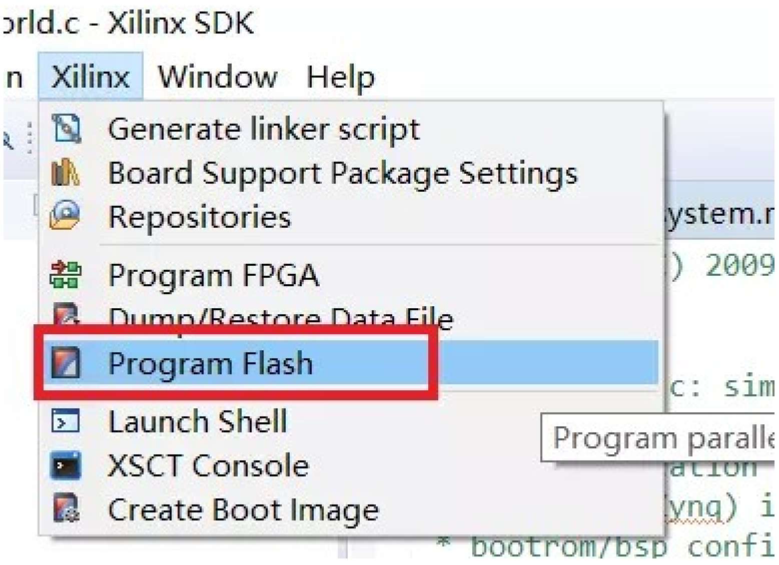
Boot image partitions

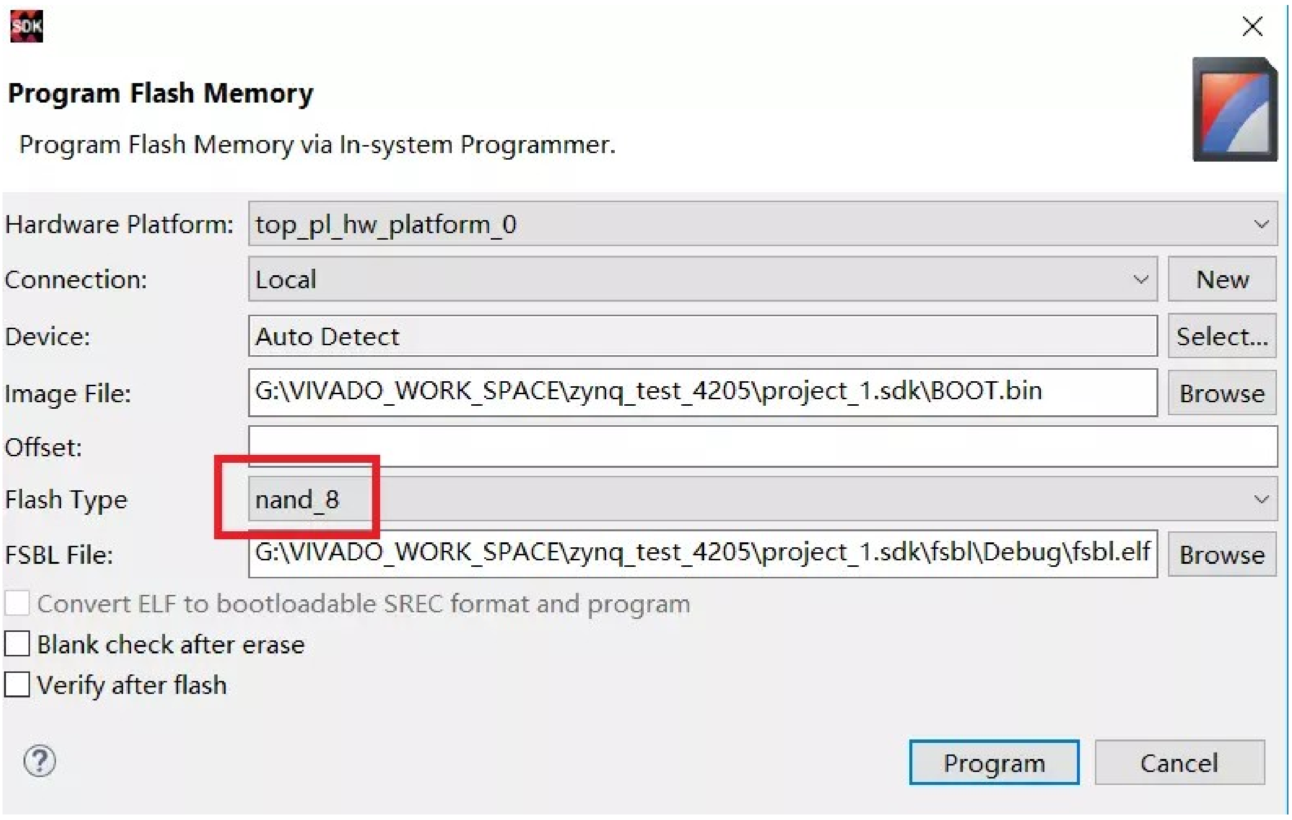
File path	Encrypted
(bootloader) G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\fsbl\Debug\fsbl.elf	none
G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\top_pl_hw_platform_0\top_pl.bit	none
G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\app\Debug\app.elf	none

Add top_pl.bit and app.elf in the same way, select the data file for the partition type, be sure to follow the order.

Click Create Mirror to finish generating BOOT.bin

15. Download BOOT.bin to NAND and set according to the following figure:




Program Flash Memory

Program Flash Memory via In-system Programmer.

Hardware Platform: top_pl_hw_platform_0

Connection: Local

Device: Auto Detect

Image File: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\BOOT.bin

Offset:

Flash Type nand_8

FSBL File: G:\VIVADO_WORK_SPACE\zynq_test_4205\project_1.sdk\fsbl\Debug\fsbl.elf

☐ Convert ELF to bootloadable SREC format and program

☐ Blank check after erase

☐ Verify after flash

16. After power-off, you can see the last line of printout Hello World, indicating that the bare metal program has run successfully, and the LED flashes alternately, indicating that the FPGA configuration is complete. If you need to debug and power on the software, you must do it once in step 1, and then directly debug the app project. Note that ps initialization is selected in the debug option.