QUICK START GUIDE

FOR INTEL® CYCLONE® V RUNBMC CARD

Author (s) :

Jan Sowinski, Intel

Kasper Wszolek, Intel

Executive Summary

Intel® Cyclone® V RunBMC card is FPGA based reference implementation of Open Compute Project (OCP) RunBMC v1.4.1 specification [1]. RunBMC specification defines interface between Baseboard Management Controller (BMC) and motherboard allowing to move BMC subsystem from solution with BMC soldered down on baseboard into a standardized module. This document instructs how to get OpenBMC FW for Cyclone® V, how to prepare base FPGA projects for Cyclone® V and MAX® 10 devices and finally how to provision card with firmware and FPGA images.

Table of Contents

1 Introduction

Intel® Cyclone® V RunBMC card is FPGA based reference implementation of Open Compute Project (OCP) RunBMC v1.4.1 specification [1]. Module can be used as Baseboard Management Controller (BMC) with compliant motherboards. Core card components are Cyclone® V SoC with Hard Processor System (HPS) and FPGA matrix in one package and MAX® 10 FPGA as separate device. Cyclone® V HPS runs OpenBMC FW [2] to serve manageability features, Cyclone® V FPGA can be used to implement server specific interfaces or custom logic offloading firmware. MAX® 10 role is to configure Cyclone® V FPGA and gives possibility to implement custom or security features.

Specification highlights:

- Intel® Cyclone® V SX
    - 2x ARM Cortex-A9
    - 1GB DDR3 with ECC
    - 1 x PCIe Hard IP
    - 40k ALMs (110k LE)
    - DSP blocks
- Intel® MAX® 10
    - 50K logic elements
    - 18 x 12bit ADC
    - Internal UFM (up to 736 KB)
    - Internal CFM (up to 1344KB)
- Other onboard components
    - 16GB eMMC memory
    - 64MB QSPI Flash memory
    - Fan-speed controller MAX314790
- Form factor Large (50 x 70mm)

Picture 1, Intel® Cyclone® V RunBMC card

In this document we will create base FPGA projects for Cyclone® V FPGA and MAX® 10. After completing this guide, one shall have module up and running OpenBMC FW with following interfaces enabled: RMGII, RMII, eMMC, USBs, I2Cs, RPM/TACH and GPIOs.

To follow this guide following prerequisites are required:

- Linux machine to build OpenBMC FW and FPGA projects (verified on Ubuntu 20.04.05 LTS)
- Intel® Quartus® Prime Design Software Standard or Lite Edition [3] (verified with Quartus Standard 21.1)
- Quartus compatible JTAG programmer, e.g.: USB Blaster II [3]
- RunBMC v1.4.1 compatible host providing access to serial port and ethernet (via RGMII or RMII), e.g.: Hyve Solution's Bring Up Vehicle (HSBUV) [2]

2 OpenBMC FW

Intel® Cyclone® V RunBMC card has been enabled with OpenBMC which is open-source project providing BMC stack. To build OpenBMC for Cyclone® V please follow build instruction on https://github.com/Intel-BMC/meta-runbmc-cyclone5 repository. Linux build environment will be required.

After successful build required artifacts shall be available in **<openbmc-build-dir>/tmp/deploy/images/runbmc-cyclone5** directory:

- **intel-platforms-runbmc-cyclone5.wic** – complete OpenBMC image for eMMC
- **u-boot-spl.bin** – u-boot SPL binary which will be used to initialize Cyclone® V OnChip Memory IP
- **u-boot.img** – u-boot image which will be used during first provisioning process

3 MAX® 10 FPGA

3.1 Design overview

MAX® 10 FPGA is required to perform Cyclone® V FPGA configuration on every RunBMC module boot.
Minimal MAX® 10 FPGA design consists of PLL IP and PFL (Parallel Flash Loader) IP.
PFL IP is used to read Cyclone® V FPGA bitstream from attached QSPI flash and performing Cyclone® V FPGA configuration over Passive Serial interface. PFL IP can be used also for programming QSPI flash over JTAG adapter. PLL is added to improve performance.
Cyclone® V FPGA configuration success is signaled by green LED, failure is indicated by amber LED.

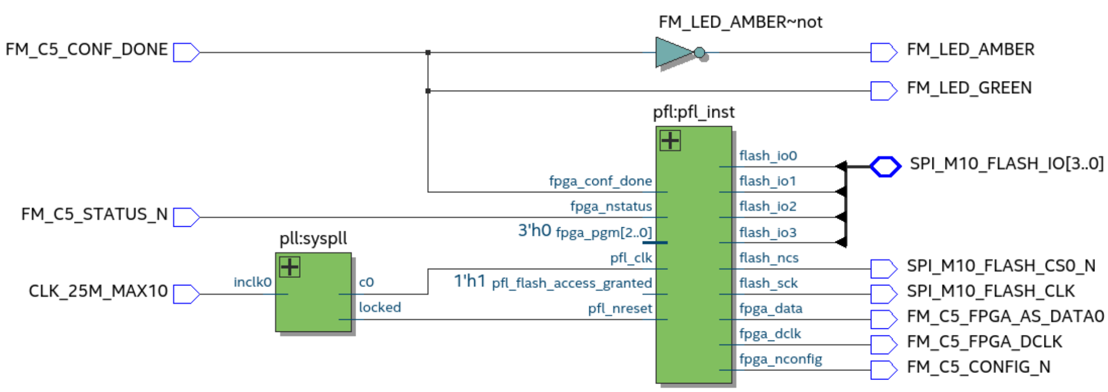
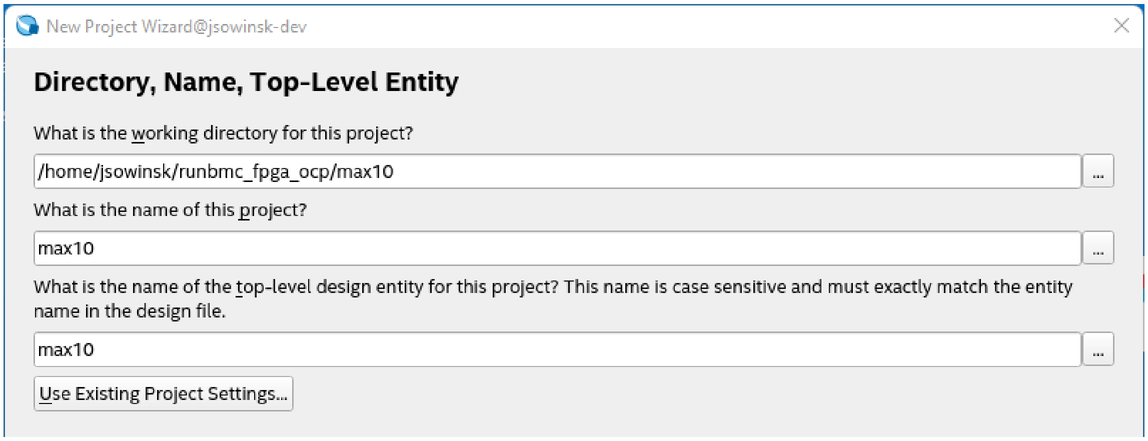


Figure 1, MAX® 10 FPGA configuration diagram

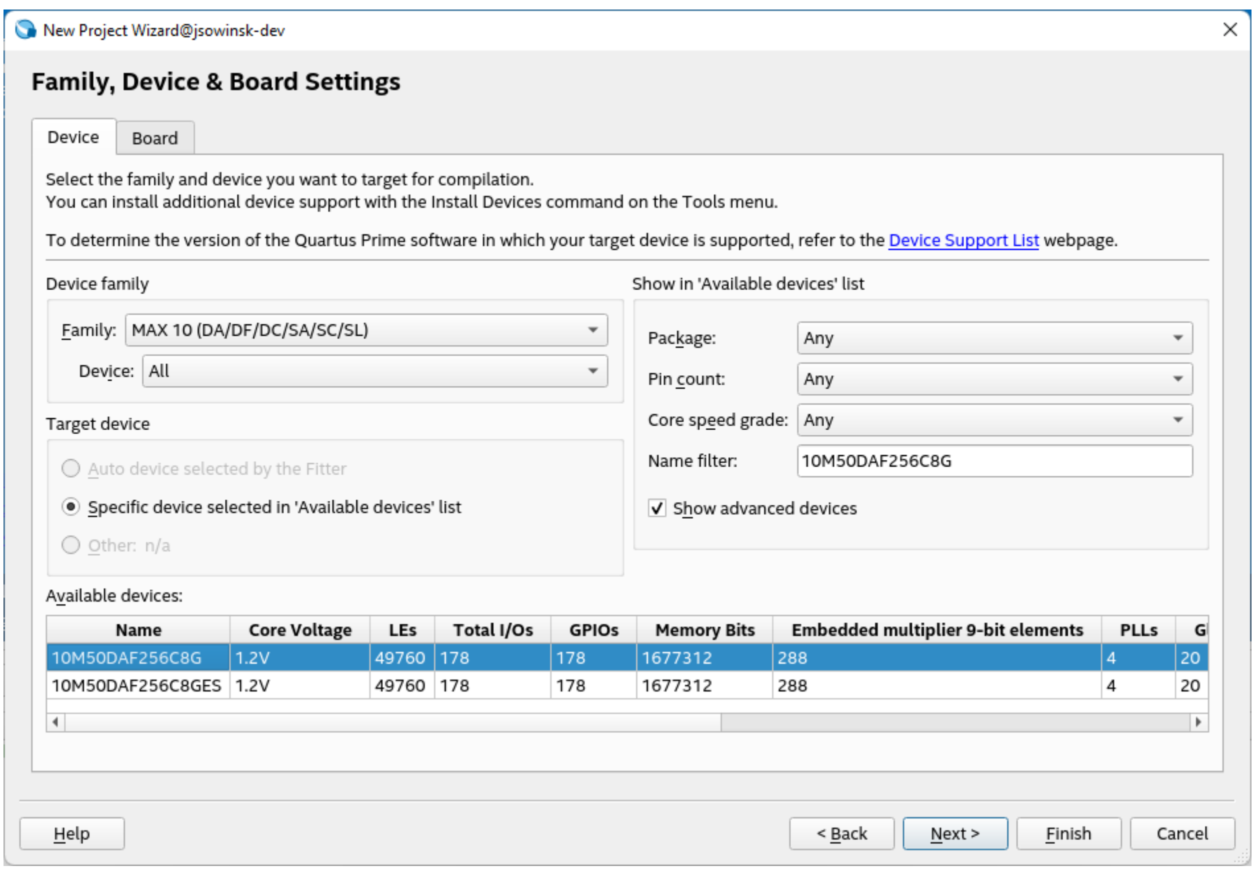
3.2 Creating Quartus project

1. Go **to File -> New Project Wizard** to create new project.

   Name both project and top-level design entity as “**max10**”



2. On “**Project Type**” page select **Empty** project
3. Skip “**Add Files**” page without adding any files to project.
4. Select **10M50DAF256C8G** device in **Family, Device & Board Settings** dialog and click **Finish**

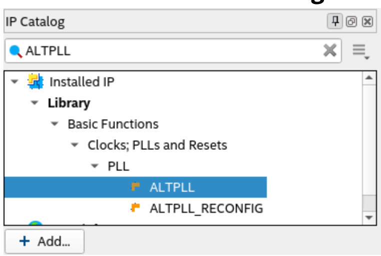


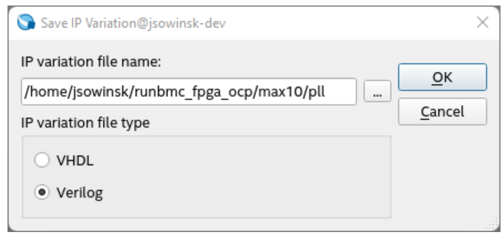
5. Adjust device options according to below table in **Assignments -> Device -> Device and Pin Options**

Category	Option
Unused Pins	Set “Reserve all unused pins” to “As input tri-stated”
Voltage	Set “Default I/O standard” to “3.3-V LVTTL”

3.3 Adding PLL IP

1. Search **ALTPLL** in **IP Catalog** window and add it to project as IP variation named “**pll**”



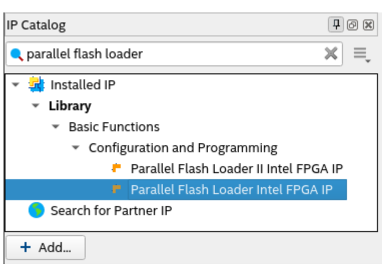


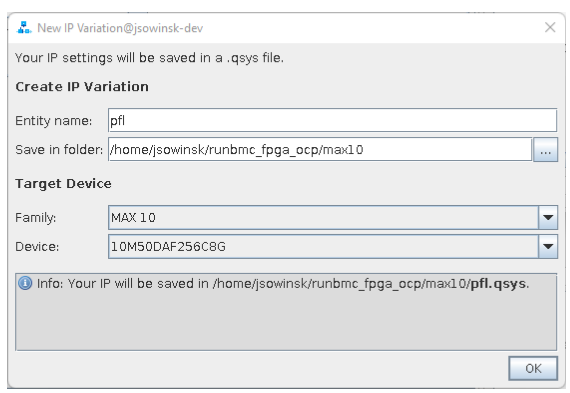
2. In PLL wizard change following options

Page	Tab	Option
Parameter Settings	General/Modes	Set device speed grade to 8
		Set inclk0 input frequency to 25MHz
	Inputs/Lock	Deselect option “create ‘areset’ input to asynchronously reset the PLL”
Output Clocks	clk c0	Set requested output clock frequency to 100MHz

3.4 Adding Parallel Flash Loader IP

1. Search “**Parallel Flash Loader Intel® FPGA IP**” in **IP Catalog** window and add it to project as IP Variation named “**pfl**”





2. In **IP Parameter Editor** change default options according to below table

Tab	Option
General	Set operating mode to **Flash Programming and FPGA Configuration**
	Set **Quad SPI Flash** as target flash

	Select **Set flash bus pins to tri-state when not in use**
Flash Interface Settings	Set "How many flash devices will be used?" to **1**
	Set "Quad SPI flash device manufacturer" to **Micron**
	Set "Quad SPI flash device density" to **QSPI 512 Mbit**
FPGA Configuration	Set "Ratio between input clock and CLK output" to **2**

3. Click **Generate HDL…** button and then **Finish** button.
4. Add generated IP to project by going **Project -> Add/Remove Files** and browsing for **pfl.qsys** file

3.5 Adding top level module

1. Create top-level verilog file (**File -> New -> Verilog HDL File**)
2. Define max10 module with ports required to access QSPI flash, configure Cyclone® V FPGA configuration and drive LEDs. Example top module instantiating PLL and PFL IPs and connecting them together is presented below:

```
module max10 (
    input       CLK_25M_MAX10,
    output      SPI_M10_FLASH_CS0_N,
    output      SPI_M10_FLASH_CLK,
    inout [3:0] SPI_M10_FLASH_IO,
    output      FM_C5_CONFIG_N,
    input       FM_C5_STATUS_N,
    output      FM_C5_FPGA_DCLK,
    output      FM_C5_FPGA_AS_DATA0,
    input       FM_C5_CONF_DONE,
    output      FM_LED_AMBER,
    output      FM_LED_GREEN
);

wire clk;
wire reset_n;
wire conf_done = FM_C5_CONF_DONE;

assign FM_LED_AMBER = ~conf_done;
assign FM_LED_GREEN = conf_done;

pll syspll (
    .inclk0 (CLK_25M_MAX10),
    .c0     (clk),
    .locked (reset_n)
);

pfl pfl_inst (
    .flash_io0                (SPI_M10_FLASH_IO[0]),
    .flash_io1                (SPI_M10_FLASH_IO[1]),
    .flash_io2                (SPI_M10_FLASH_IO[2]),
    .flash_io3                (SPI_M10_FLASH_IO[3]),
    .flash_ncs                (SPI_M10_FLASH_CS0_N),
    .flash_sck                (SPI_M10_FLASH_CLK),
    .fpga_conf_done           (conf_done),
    .fpga_data                (FM_C5_FPGA_AS_DATA0),
    .fpga_dclk                (FM_C5_FPGA_DCLK),
    .fpga_nconfig             (FM_C5_CONFIG_N),
    .fpga_nstatus             (FM_C5_STATUS_N),
    .fpga_pgm                 (3'b000),
    .pfl_clk                  (clk),
    .pfl_flash_access_granted (1'b1),
    .pfl_nreset               (reset_n)
```

```
);

endmodule
```

Example 1, MAX® 10 FPGA project top module (max10.v)

3. Save file as **max10.v** and run **Analysis & Synthesis**

3.6 Pin assignment

Assign pins according to below table by going **Assignments->Pin Planner**

Node Name	Location	I/O Standard
CLK_25M_MAX10	PIN_J11	3.3-V LVTTL
FM_C5_CONFIG_N	PIN_L12	
FM_C5_CONF_DONE	PIN_M15	
FM_C5_FPGA_AS_DATA0	PIN_M14	
FM_C5_FPGA_DCLK	PIN_L16	
FM_C5_STATUS_N	PIN_L15	
FM_LED_AMBER	PIN_K2	
FM_LED_GREEN	PIN_K5	
SPI_M10_FLASH_CLK	PIN_B4	
SPI_M10_FLASH_CS0_N	PIN_B3	
SPI_M10_FLASH_IO[3]	PIN_B9	
SPI_M10_FLASH_IO[2]	PIN_B7	
SPI_M10_FLASH_IO[1]	PIN_B6	
SPI_M10_FLASH_IO[0]	PIN_B5	

Table 1, MAX® 10 pin assignment

3.7 Timing constraints

Create timing constraints by opening Timing Analyzer tool and selecting **File -> New SDC File**, add appropriate timing rules and save file as **max10.sdc** with option “**add to project**” selected. Example timing constrain can be found below.

```
set_time_format -unit ns -decimal_places 3

create_clock -name {CLK_25M_MAX10} -period 40.000 -waveform { 0.000 20.000 } [get_ports
{CLK_25M_MAX10}]
create_clock -name {SPI_M10_FLASH_CLK} -period 20.000 -waveform { 0.000 10.000 } \
    [get_ports {SPI_M10_FLASH_CLK}]
create_clock -name {FM_C5_FPGA_DCLK} -period 20.000 -waveform { 0.000 10.000 } \
    [get_ports {FM_C5_FPGA_DCLK}]

derive_pll_clocks -create_base_clocks
derive_clock_uncertainty

# PFL setup/hold times (max for setup time, min for hold time)
#  QSPI IOs as inputs
set_input_delay -clock {SPI_M10_FLASH_CLK} -max 5 [get_ports {SPI_M10_FLASH_IO[*]}] -clock_fall
set_input_delay -add_delay -clock {SPI_M10_FLASH_CLK} -min 1 [get_ports {SPI_M10_FLASH_IO[*]}] -
clock_fall
#  QSPI IOs as output
set_output_delay -clock {SPI_M10_FLASH_CLK} -max 2 [get_ports {SPI_M10_FLASH_IO[*]}]
set_output_delay -add_delay -clock {SPI_M10_FLASH_CLK} -min 1 [get_ports {SPI_M10_FLASH_IO[*]}]
#  FPGA configurations output IOs
set_output_delay -clock {FM_C5_FPGA_DCLK} -max 5.5 [get_ports {FM_C5_FPGA_AS_DATA0}]
set_output_delay -add_delay -clock {FM_C5_FPGA_DCLK} -min 1 [get_ports {FM_C5_FPGA_AS_DATA0}]
```

```
# Using USB Blaster 2 at 16MHz clock = 62.5 ns period
create_clock -name {altera_reserved_tck} -period 62.5 [get_ports {altera_reserved_tck}]
set_clock_groups -asynchronous -group {altera_reserved_tck}
set_input_delay  -clock { altera_reserved_tck } 5.0 [get_ports {altera_reserved_tdi}]
set_input_delay  -clock { altera_reserved_tck } 5.0 [get_ports {altera_reserved_tms}]
set_output_delay -clock { altera_reserved_tck } 3.5 [get_ports {altera_reserved_tdo}]
set_false_path -from [get_clocks {altera_reserved_tck}] -to [get_clocks {altera_reserved_tck}]
```

Example 2, MAX® 10 timing constraints (max10.sdc)

3.8 Generate MAX® 10 programming file

Compile design by pressing **ctrl + L**.
MAX® 10 programming files shall be available in **<max10-fpga-project-dir>/output_files/max10.pof** after compilation is complete.

4 Cyclone® V FPGA

4.1 Design overview

HPS in Cyclone® V is hard strapped to boot from on-chip memory in FPGA. Additionally, SDIO interface for eMMC is routed to alternative pins which requires routing signals via FPGA. Due to these facts Cyclone® V's FPGA needs to be configured to allow RunBMC card boot. Cyclone® V FPGA project defined in this chapter contains must have on-chip memory initialized with preloader, HPS interfaces configuration and interfaces routing via FPGA to correct pins. Additionally, MII-to-RMII SoftIP is used to convert HPS native MII interface to RMII which is required by RunBMC specification. Finally, GPIO and I2C Soft-IPs are added to cover required number of platform signals and I2C buses.

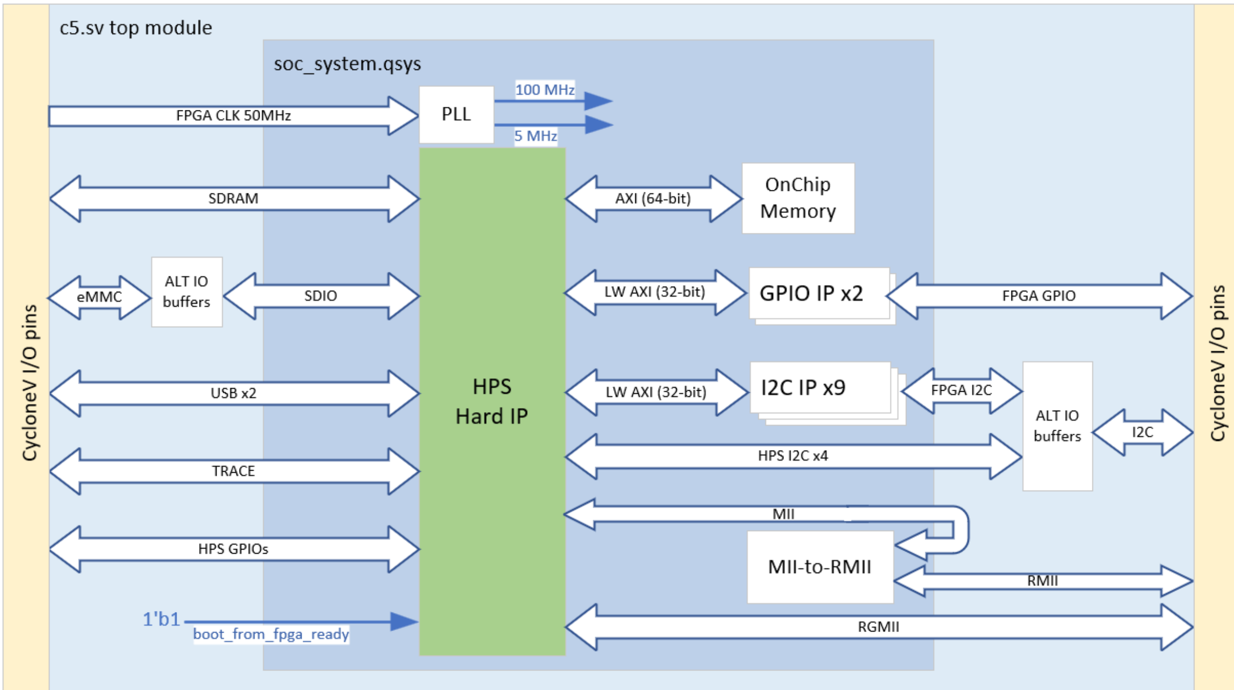
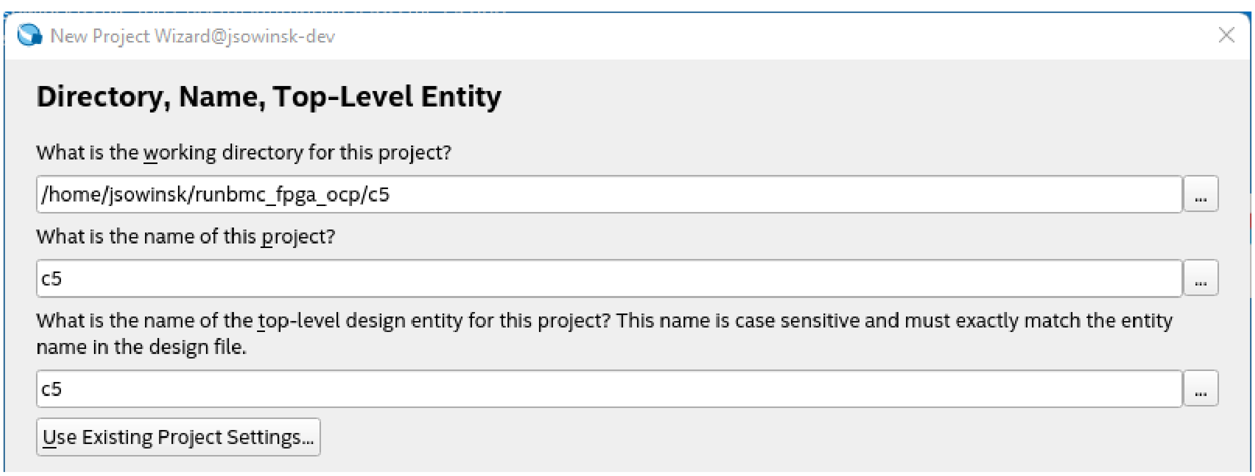


Figure 2, Cyclone® V FPGA configuration diagram

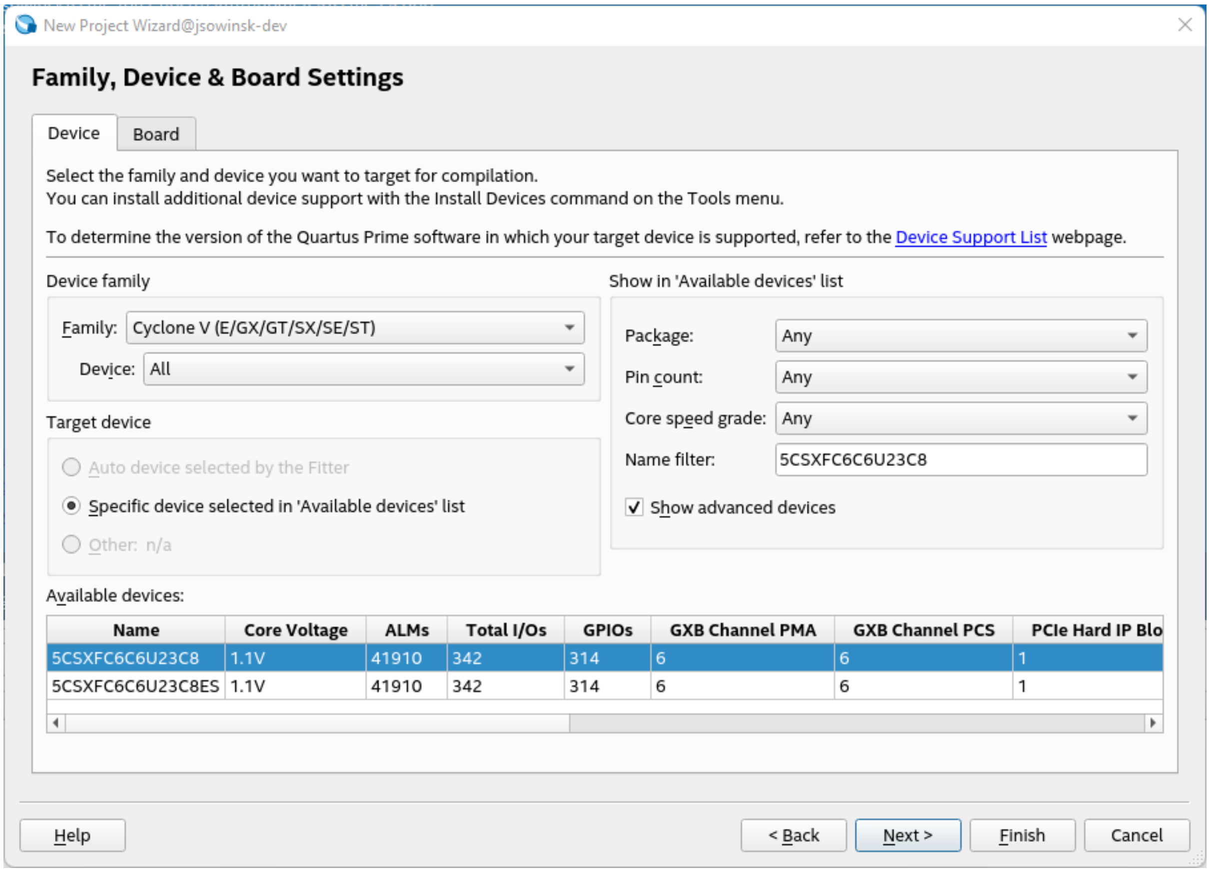
4.2 Creating Quartus project for Cyclone® V

1. Go to **File -> New Project Wizard** to create new project.
   Name both project and top-level design entity as "**c5**"



2. On **Project Type** page select **Empty** project
3. Skip **Add Files** page without adding any files to project.

4. On "**Family, Device & Board Settings**" page use "**Name filter**" textbox to select **5CSXFC6C6U23C8** device and click Finish.



5. Adjust device options according to below table in **Assignments -> Device -> Device and Pin Options**

Category	Option
General	Set "Enable INIT_DONE output"
Unused Pins	Set "Reserve all unused pins" to "As input tri-stated"
Voltage	Set "Default I/O standard" to "3.3-V LVTTL:

4.3 Creating system with Platform Designer

1. Create initialization file for OnChip Memory IP with preloader using below command:

```
srec_cat <openbmc-build-dir>/tmp/deploy/images/runbmc-cyclone5/u-boot-spl.bin \
-binary -fill 0x00 0x0 0x10000 --bs 8 \
-o <cyclone5-fpga-project-dir>/preloader-ocm.mif -mif 64 --obs 8
```

Note: **srec_cat** command is part of srecord package available in Linux distributions through package management systems. For example, to install on Ubuntu run command: **sudo apt install srecord**.

2. Create new system with Platform Designer by selecting **Tools -> Platform Designer**
3. On "**Interconnect Requirements**" tab change "**System-wide Requirements"** as follows:

Option	New value
Limit Interconnect pipeline stages to:	4
Clock crossing adapter type:	Auto

4. Save Platform Designer system (Ctrl + S) as "**soc_system**"

4.3.1 Adding Components

Adding components to the system is done using **IP Catalog** window (**View -> IP Catalog**).
To add component, enter IP name in search box and double click entry or use "**Add…**" button.
Component can be configured using wizard available during adding process manually. or later by selecting component in System Contents window and using **Parameters** window (**View -> Parameters**) .
To rename component select it in **System Contents** window and use right button context menu or pressing F2.

Add components to the system, configure and rename instances using data provided in the table below.

IP library name	Components name(s)	Configuration
Clock Source	clk_0	Use default
PLL Intel FPGA IP	pll_0	Apply configuration from *Table 8, Cyclone® V pll_0 configuration*
Reset bridge	reset_100M, reset_5M	Use default
Aria V/Cyclone V Hard Processor System	hps_0	Apply configuration from *Table 7, Cyclone® V hps_0 configuration*
On-Chip Memory (RAM or ROM) Intel FPGA IP	onchip_memory2_0	Apply configuration from *Table 9, Cyclone® V onchip_memory2_0 configuration*
MII-to-RMII Converter Intel FPGA IP	fpga_mii2rmii_0	Select all options (THROUGHPUT CONTROL, MPBS, MAC_SPEED)
PIO (Parallel I/O) Intel FPGA IP	fpga_gpio_0	Apply configuration from *Table 10, Cyclone® V fpga_gpio_0 configuration*
PIO (Parallel I/O) Intel FPGA IP	fpga_gpio_1	Apply configuration from *Table 11, Cyclone® V fpga_gpio_1 configuration*
Avalon I2C (Master) Intel FPGA IP	fpga_i2c_0 … fpga_i2c_9	Use default settings

Table 2, Cyclone® V Platform Designer components list

After adding all the components to the **soc_system** in Platform Designer **System Content** shall look as follows:

Use	Co...	Name	Description
☑		⊞ clk_0	Clock Source
☑		⊞ pll_0	PLL Intel FPGA IP
☑		⊞ reset_100M	Reset Bridge
☑		⊞ reset_5M	Reset Bridge
☑		⊞ hps_0	Arria V/Cyclone V Hard Processor System
☑		⊞ onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel FPGA IP
☑		⊞ fpga_mii2rmii_0	MII-to-RMII Converter Intel FPGA IP
☑		⊞ fpga_gpio_0	PIO (Parallel I/O) Intel FPGA IP
☑		⊞ fpga_gpio_1	PIO (Parallel I/O) Intel FPGA IP
☑		⊞ fpga_i2c_0	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_1	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_2	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_3	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_4	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_5	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_6	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_7	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_8	Avalon I2C (Master) Intel FPGA IP
☑		⊞ fpga_i2c_9	Avalon I2C (Master) Intel FPGA IP

4.3.2 Connections and exports

There are several ways to make connections between components in Platform Designer.
With many components in system convenient way of connecting them is to right click on component's port in **System Contents** window, go to **Connections** option and choose desired destination port.
Alternative way to make all required connection from port at once is to open "**Connections**" window (**View -> Connections)** and use checkboxes associated with destination ports.
Components ports shall be connected according to below table.

Component	Port	Connect to
clk_0	clk	hps_0.emac_ptp_ref_clock
		hps_0.f2h_axi_clock
		hps_0.h2f_axi_clock
		hps_0.h2f_lw_axi_clock
		onchip_memory2_0.clk1
		pll_0.refclk
	clk_reset	onchip_memory2_0.reset1
		pll_0.reset
		reset_100M.in_reset
		reset_5M.in_reset
pll_0	outclk0	fpga_gpio_0.clk
		fpga_gpio_1.clk
		reset_100M.clk
	outclk1	fpga_i2c_0.clock
		fpga_i2c_1.clock
		fpga_i2c_2.clock
		fpga_i2c_3.clock

		fpga_i2c_4.clock
		fpga_i2c_5.clock
		fpga_i2c_6.clock
		fpga_i2c_7.clock
		fpga_i2c_8.clock
		fpga_i2c_9.clock
		reset_5M.clk
reset_100M	out_reset	fpga_gpio_0.reset
		fpga_gpio_1.reset
reset_5M	out_reset	fpga_i2c_0.reset_sink
		fpga_i2c_1.reset_sink
		fpga_i2c_2.reset_sink
		fpga_i2c_3.reset_sink
		fpga_i2c_4.reset_sink
		fpga_i2c_5.reset_sink
		fpga_i2c_6.reset_sink
		fpga_i2c_7.reset_sink
		fpga_i2c_8.reset_sink
		fpga_i2c_9.reset_sink
hps_0	h2f_axi_master	onchip_memory2_0.s1
	h2f_lw_axi_master	fpga_gpio_0.s1
		fpga_gpio_1.s1
		fpga_i2c_0.csr
		fpga_i2c_1.csr
		fpga_i2c_2.csr
		fpga_i2c_3.csr
		fpga_i2c_4.csr
		fpga_i2c_5.csr
		fpga_i2c_6.csr
		fpga_i2c_7.csr
		fpga_i2c_8.csr
		fpga_i2c_9.csr
	f2h_irq0	fpga_gpio_0.irq
		fpga_gpio_1.irq
		fpga_i2c_0.interrupt_sender
		fpga_i2c_1.interrupt_sender
		fpga_i2c_2.interrupt_sender
		fpga_i2c_3.interrupt_sender
		fpga_i2c_4.interrupt_sender
		fpga_i2c_5.interrupt_sender
		fpga_i2c_6.interrupt_sender
		fpga_i2c_7.interrupt_sender
		fpga_i2c_8.interrupt_sender
		fpga_i2c_9.interrupt_sender
fpga_mii2rmii_0	pcs_mac_rx_clock_connection	hps_0.emac0_rx_clk_in

	pcs_mac_tx_clock_connection	hps_0.emac0_tx_clk_in

Table 3, Cyclone® V Platform Designer components connections

Export following components' ports by double-clicking in Export column.

Component	Port	Export as
clk_0	clk_in	clk
	clk_in_reset	reset
hps_0	f2h_boot_from_fpga	hps_0_f2h_boot_from_fpga
	emac0	hps_0_emac0
	sdio	hps_0_sdio
	sdio_cclk	hps_0_sdio_cclk
	i2c0_scl_in	hps_0_i2c0_scl_in
	i2c0_clk	hps_0_i2c0_clk
	i2c0	hps_0_i2c0
	i2c1_scl_in	hps_0_i2c1_scl_in
	i2c1_clk	hps_0_i2c1_clk
	i2c1	hps_0_i2c1
	i2c2_scl_in	hps_0_i2c2_scl_in
	i2c2_clk	hps_0_i2c2_clk
	i2c2	hps_0_i2c2
	i2c3_scl_in	hps_0_i2c3_scl_in
	i2c3_clk	hps_0_i2c3_clk
	i2c3	hps_0_i2c3
	memory	memory
	hps_io	hps_io
	h2f_reset	hps_0_h2f_reset
fpga_mii2rmii_0	clock_sink	fpga_mii2rmii_0_clock_sink
	reset_sink	fpga_mii2rmii_0_reset_sink
	mii_interface	fpga_mii2rmii_0_mii_interface
	rmii_interface	fpga_mii2rmii_0_rmii_interface
	MACSPEED	fpga_mii2rmii_0_MACSPEED
fpga_gpio_0	external_connection	fpga_gpio_0_external_connection
fpga_gpio_1	external_connection	fpga_gpio_1_external_connection
fpga_i2c_0	i2c_serial	fpga_i2c_0_i2c_serial
fpga_i2c_1	i2c_serial	fpga_i2c_1_i2c_serial
fpga_i2c_2	i2c_serial	fpga_i2c_2_i2c_serial
fpga_i2c_3	i2c_serial	fpga_i2c_3_i2c_serial
fpga_i2c_4	i2c_serial	fpga_i2c_4_i2c_serial
fpga_i2c_5	i2c_serial	fpga_i2c_5_i2c_serial
fpga_i2c_6	i2c_serial	fpga_i2c_6_i2c_serial
fpga_i2c_7	i2c_serial	fpga_i2c_7_i2c_serial
fpga_i2c_8	i2c_serial	fpga_i2c_8_i2c_serial
fpga_i2c_9	i2c_serial	fpga_i2c_9_i2c_serial

Table 4, Cyclone® V Platform Designer exports list

4.3.3 Base addresses and IRQ numbers

In Platform Designer assign following base addresses in **Address Map** tab (**View -> Address Map**)

	hps_0.h2f_axi_master	hps_0.h2f_lw_axi_master
onchip_memory2_0.s1	0x0000_0000 - 0x0000_ffff	
fpga_gpio_0.s1		**0x0001_2000** - 0x0001_201f
fpga_gpio_1.s1		**0x0001_2040** - 0x0001_205f
fpga_i2c_0.csr		**0x0001_3000** - 0x0001_303f
fpga_i2c_1.csr		**0x0001_3040** - 0x0001_307f
fpga_i2c_2.csr		**0x0001_3080** - 0x0001_30bf
fpga_i2c_3.csr		**0x0001_30c0** - 0x0001_30ff
fpga_i2c_4.csr		**0x0001_3100** - 0x0001_313f
fpga_i2c_5.csr		**0x0001_3140** - 0x0001_317f
fpga_i2c_6.csr		**0x0001_3180** - 0x0001_31bf
fpga_i2c_7.csr		**0x0001_31c0** - 0x0001_31ff
fpga_i2c_8.csr		**0x0001_3200** - 0x0001_323f
fpga_i2c_9.csr		**0x0001_3240** - 0x0001_327f

Table 5, Cyclone® V Platform Designer components base addresses

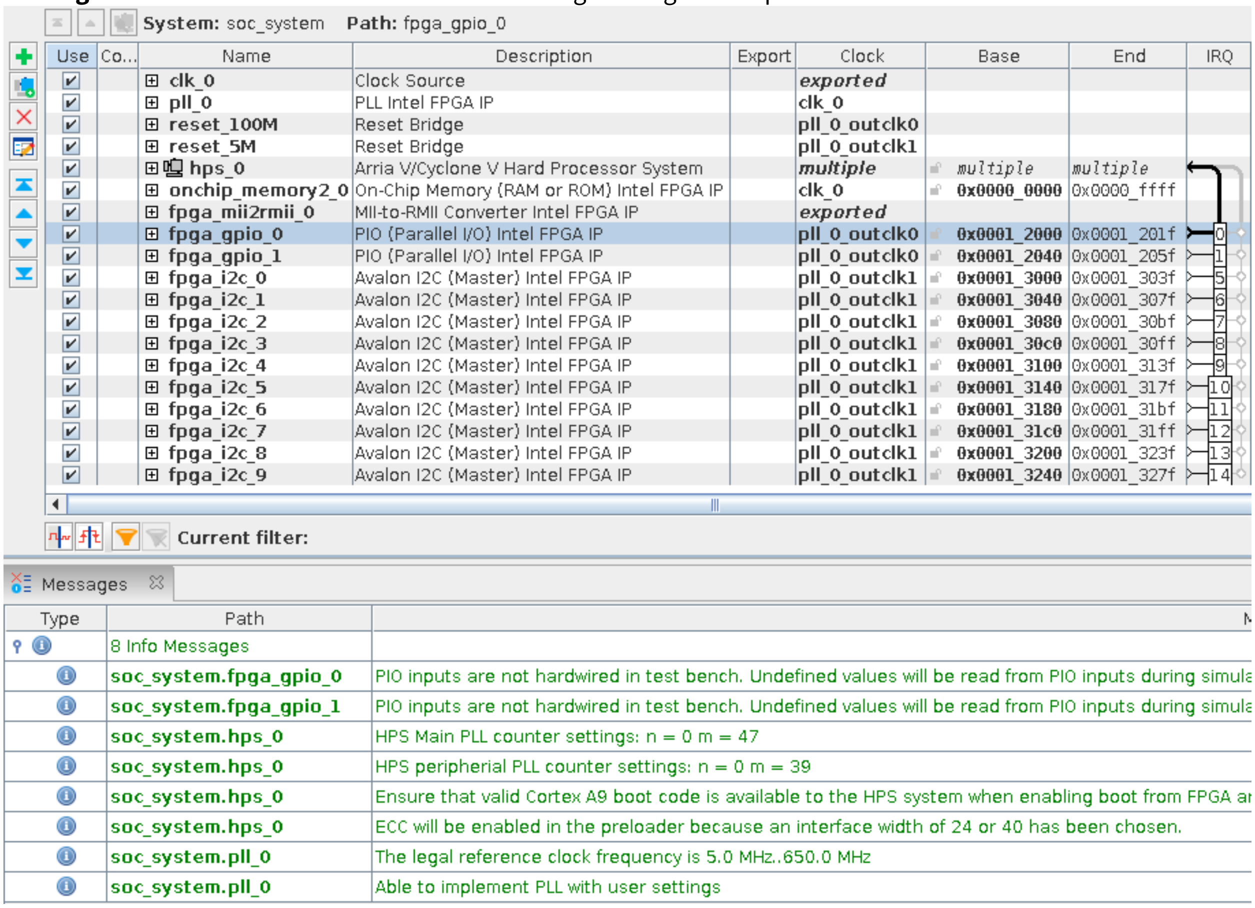
On **System Contents** tab (**View -> System Contents**) right click anywhere and select **Collapse All** option.

Update all interrupts numbers shown in IRQ column according to below table

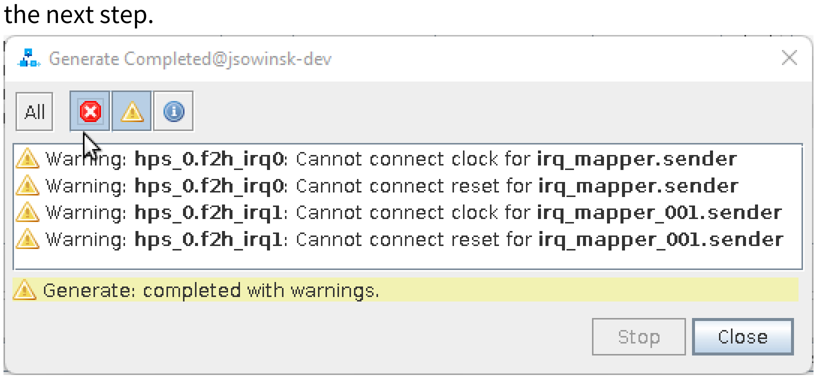
Component name	IRQ number
fpga_gpio_0	0
fpga_gpio_1	1
fpga_i2c_0	5
fpga_i2c_1	6
fpga_i2c_2	7
fpga_i2c_3	8
fpga_i2c_4	9
fpga_i2c_5	10
fpga_i2c_6	11
fpga_i2c_7	12
fpga_i2c_8	13
fpga_i2c_9	14

Table 6, Cyclone® V Platform Designer components IRQ assignments

Base and **IRQ** columns in **System Contents** tab shall be as below.
Messages window shall not contain error or warning messages as depicted below.



Save Platform Designer project (**ctrl+s**) and click "**Generate HDL**" button. Generate with default parameters. Generate process shall complete without errors. Please ignore the following warning messages and proceed to the next step.



Close Platform Designer with "**Finish**" button

Add generated IP to project by going **Project -> Add/Remove Files** and browsing for **soc_system.qsys** file

4.4 Adding top level module

1. Create top-level System Verilog file (**File -> New -> SytemVerilog HDL File**).
   Define **c5** module which shall instantiate previously created **soc_system** and wire its pins to top level ports of c5 module. HPS's SDIO and I2C signals need to be connected to top level input/output ports using **altio_buf** primitives. Additionally, EMAC0 signals shall be wired to MII-to-RMII converter on top of the module. Example **c5 module** is presented in **Example 3, Cyclone® V top module (c5.sv)**.
2. Save file (**ctrl+s**) as **c5.sv**

4.5 Timing constrains

1. Open **Timing Analyzer** tool (**Tools -> Timing Analyzer**).
2. Go to **File -> New SDC File** and add appropriate timing rules.
   Example of timing constrains file can be found in **Example 4, Cyclone® V timing constraints (c5.sdc)**.
3. **Save file** (**ctrl+s**) as **c5.sdc** in <cyclone5-fpga-project-dir> directory with option "**Add file to current project**" selected.

4.6 Pin assignment

1. Double click "**Analysis & Synthesis**" task in **Tasks** window to generate net list.
2. Go to **Tools -> Tcl Scripts**… and run script:

   ./soc_system/synthesis/submodules/hps_sdram_p0_pin_assignments.tcl
3. Assign pins according to **Table 12, Cyclone® V pin assignment** in **Assignments->Pin Planner**

   (ctrl+shift+n)

4.7 Generate Cyclone® V programming file

1. Compile project by going **Processing -> Start Compilation (ctrl + l)**.

   Properly prepared project shall compile **without errors or critical warnings**.

2. Go to **File -> Convert Programming Files** and apply configuration as presented below on Figure 3,

3. Use "**Generate**" button. Programming file shall be generated in

   <cyclone5-fpga-project-dir>/output_files/c5.pof location.

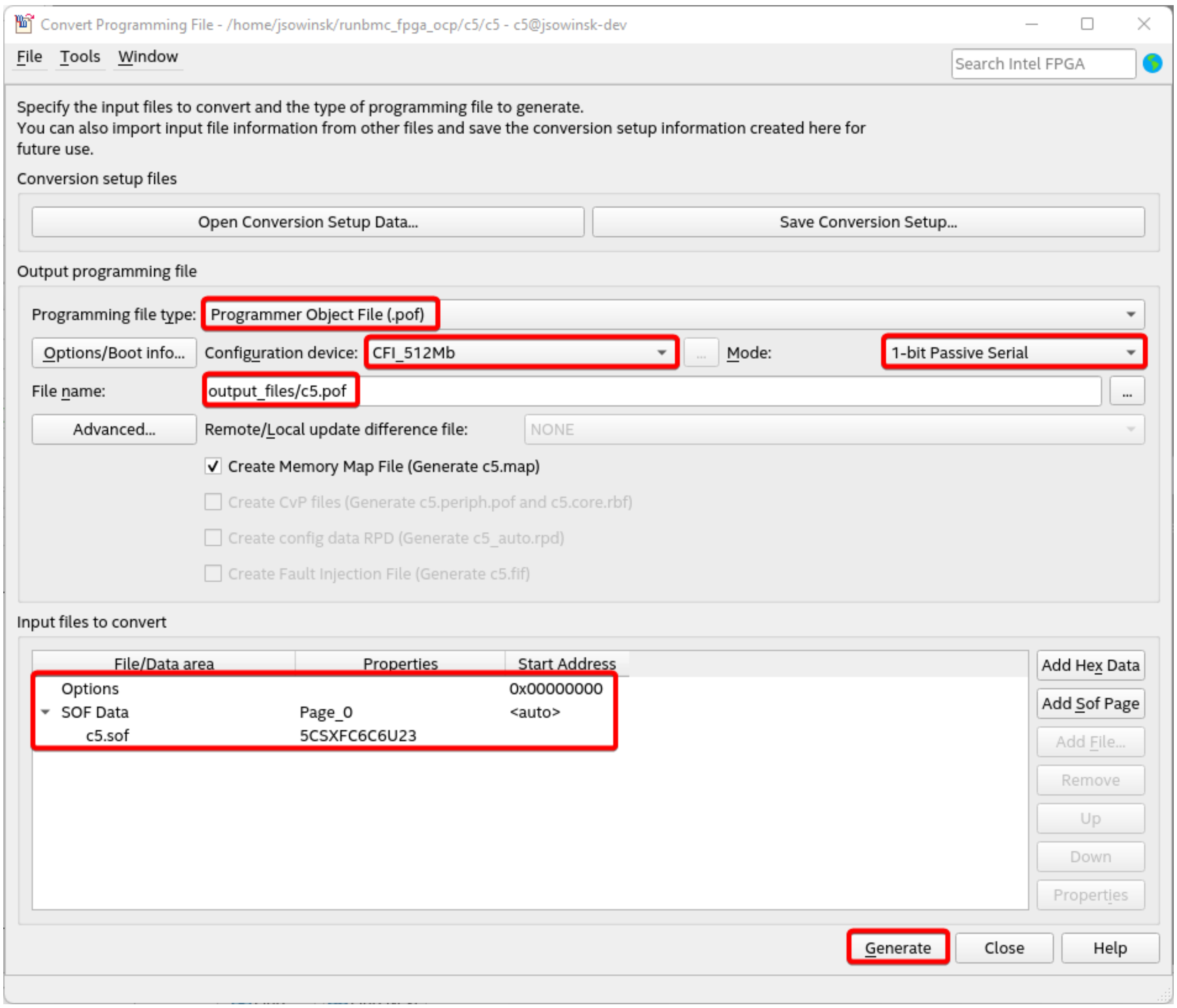
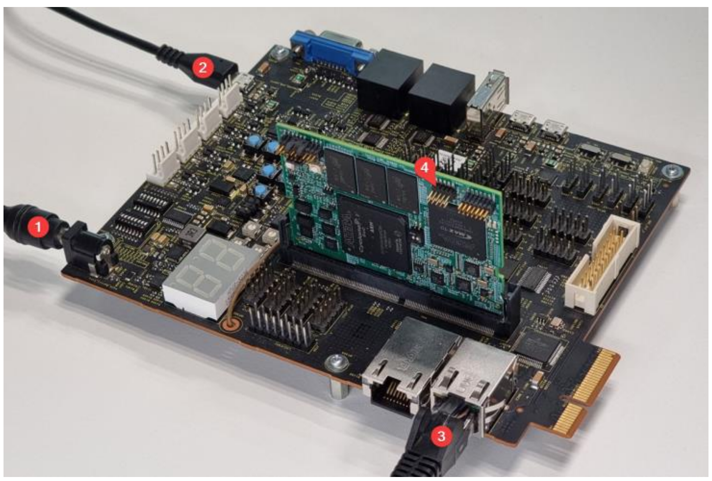


Figure 3, Convert Programming File settings for Cyclone® V

5 Module provisioning

Having built artifacts from OpenBMC FW, MAX® 10 and Cyclone® V FPGA projects Intel® Cyclone® V RunBMC card can be provisioned. Card needs to be inserted into host (platform or bring up board) providing following:

- Power supply 12V / 1A
- Serial connection to BMC console
- Ethernet connection to RunBMC (via RGMII or RMII)
- JTAG programmer, e.g.: USB Blaster II



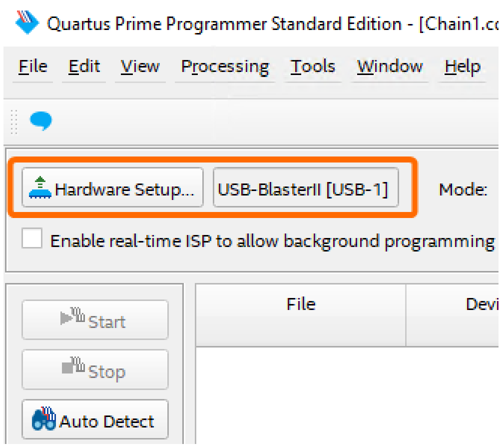
Picture 2, Intel® Cyclone® V RunBMC and Hyve's HSBUV[2]

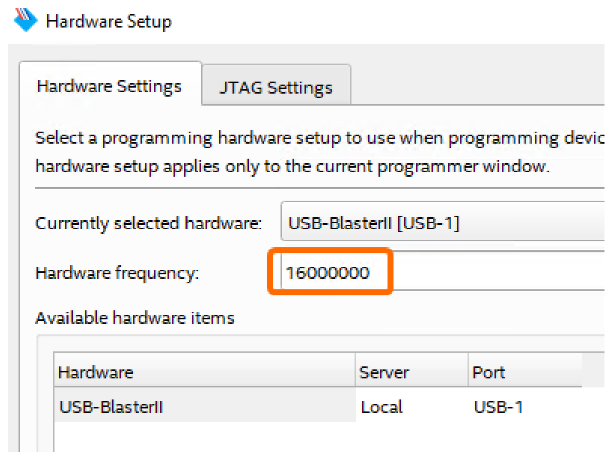
Picture 3, JTAG connection for FPGA programming

5.1 FPGA programming

1. Power on RunBMC module
2. Open Quartus Programmer software which is part of Quartus suite

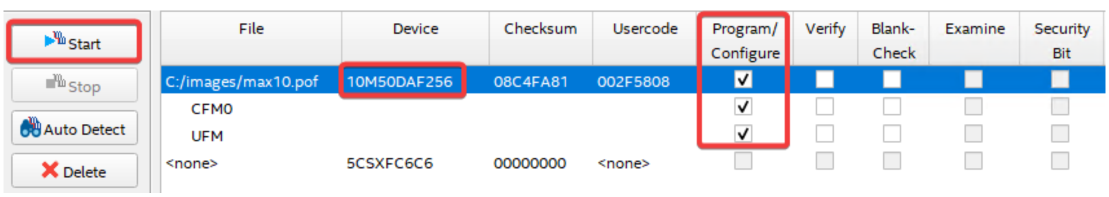
3. Go to **Hardware Setup** and configure speed to **16000000** Hz





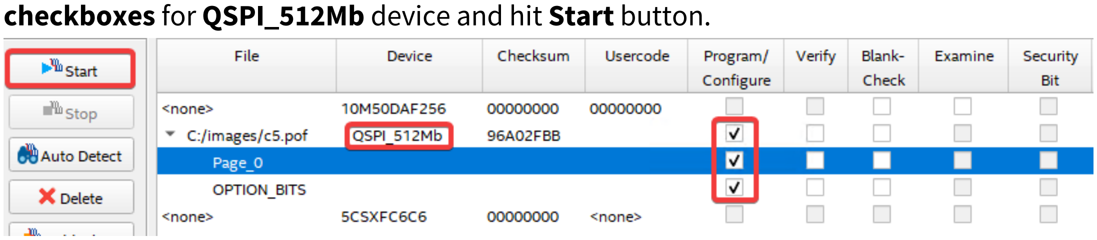
4. Press **Auto Detect** button, **10M50DAF256** device shall be detected and optionally **5CSXFC6C6** depending on DIP switch settings on RunBMC card.

5. **Right click** on **10M50DAF256** device and choosing **Change File** option. Select <max10-fpga-project-dir>/output-files>/**max10.pof** programming file. Check all Program/Configure checkboxes for **10M50DA** device and hit **Start** button.



6. Once MAX® 10 programming is complete press **Auto Detect** button -> **QSPI_512Mb** shall be detected.

7. **Right click QSPI_512Mb** device and choosing **Change File** option. Select **<cyclone5-fpga-project-dir>/output_files/c5.pof** as programming file. Check all **Program/Configure checkboxes** for **QSPI_512Mb** device and hit **Start** button.



Programming will take ~ 10 minutes.

8. FPGA provisioning process is complete.

5.2 OpenBMC FW programming

1. Use serial terminal capable of transferring binaries over **ymodem** protocol, e.g.: minicom.
2. Open serial port connected to RunBMC module using 115200 8n1 parameters.
3. Power cycle RunBMC module, **“Trying to boot from UART”** message shall be seen in terminal.

```
U-Boot SPL 2022.04 (Feb 22 2023 - 11:45:15 +0000)
DDRCAL: Scrubbing ECC RAM (1024 MiB).
DDRCAL: SDRAM-ECC initialized success with 692 ms
Trying to boot from MMC1
spl: partition error
mmc_load_image_raw_sector: mmc block read error
Trying to boot from UART
CCCC
```

4. Transfer **<openbmc-build-dir>/tmp/deploy/images/runbmc-cyclone5/u-boot.img** file using **ymodem** protocol.

5. Upon successful transfer following screen shall be seen

```
U-Boot SPL 2022.04 (Feb 22 2023 - 11:45:15 +0000)
DDRCAL: Scrubbing ECC RAM (1024 MiB).
DDRCAL: SDRAM-ECC initialized success with 692 ms
Trying to boot from MMC1
spl: partition error
mmc_load_image_raw_sector: mmc block read error
Trying to boot from UART
CLoaded 389844 bytes


U-Boot 2022.04 (Feb 22 2023 - 11:45:15 +0000)

CPU:   Altera SoCFPGA Platform
FPGA:  Altera Cyclone V, SE/A6 or SX/C6 or ST/D6, version 0x0
BOOT:  FPGA (HPS2FPGA Bridge)
       Watchdog enabled
DRAM:  1 GiB
Core:  21 devices, 11 uclasses, devicetree: separate
MMC:   dwmmc0@ff704000: 0
Loading Environment from MMC... *** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Model: Intel Cyclone V RunBMC
Net:
Warning: ethernet@ff700000 (eth1) using random MAC address - 12:a3:e7:e3:20:0c
eth1: ethernet@ff700000
Warning: ethernet@ff702000 (eth0) using random MAC address - c2:55:db:0b:a6:8e
, eth0: ethernet@ff702000
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc0(part 0) is current device
** No partition table - mmc 0 **
Couldn't find partition mmc 0:1
=>
```

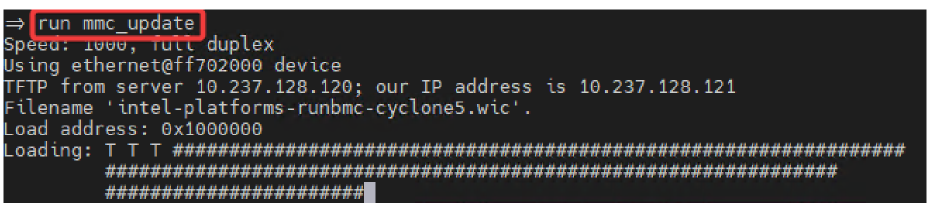
6. Set BMC MAC address. If using RGMII interface for provisioning set **ethaddr** variable, if RMII is used set **eth1addr**:
```
setenv -f ethaddr <BMC_MAC_ADDR>
```

   Get IP from DHCP server run command:
```
setenv autoload no; dhcp
```

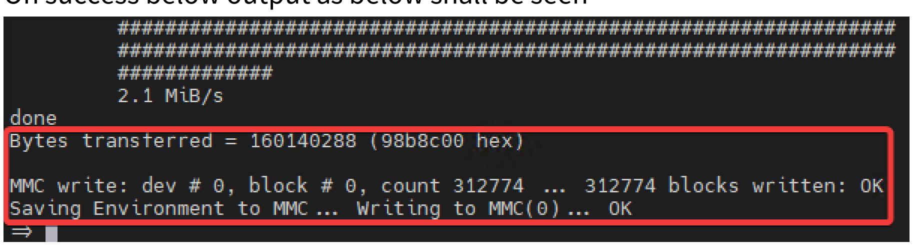
   Alternatively, BMC can be assigned static IP address using below commands:
```
setenv ipaddr <BMC_STATIC_IP_ADDR>; setenv netmask <NETMASK>
```

7. Set TFTP IP address hosting intel-platforms-runbmc-cyclone5.wic which is OpenBMC FW image taken from **<openbmc-build-dir>/tmp/deploy/images/runbmc-cyclone5** directory:
```
setenv serverip <TFTP_IP_ADDR>
```

8. Run below u-boot macro to download OpenBMC FW image and program it to eMMC memory.
```
setenv mmc_update 'tftp $loadaddr intel-platforms-runbmc-cyclone5.wic && setexpr cnt
$filesize + 0x1ff; setexpr cnt $cnt / 0x200; mmc write $loadaddr 0 $cnt && saveenv';
saveenv
```
9. Run below command in BMC console to download image from TFTP and program it to eMMC memory:
```
run mmc_update
```


```
=> run mmc_update
Speed: 1000, full duplex
Using ethernet@ff702000 device
TFTP from server 10.237.128.120; our IP address is 10.237.128.121
Filename 'intel-platforms-runbmc-cyclone5.wic'.
Load address: 0x1000000
Loading: T T T #################################################################
         #################################################################
         #######################
```

   On success below output as below shall be seen


```
         ##################################################################
         ##################################################################
         #############
         2.1 MiB/s
done
Bytes transferred = 160140288 (98b8c00 hex)

MMC write: dev # 0, block # 0, count 312774 ... 312774 blocks written: OK
Saving Environment to MMC ... Writing to MMC(0) ... OK
=>
```

10. Provisioning is complete - AC cycle board to boot to OpenBMC (or type boot or reset)

6 Known limitations

1. RGMII reference voltage is not connected to VDD_RGMII_REF pin. If motherboard requires RGMII reference voltage rework is required connecting VDD_RGMII_REF (pin 2) with VDD_3V_STBY (pin 4).
2. Fan controller supports only six first fans (PWM0..5 and TACH0..6)
3. I2C11SCL_GPIO117 and I2C11SDA_GPIO118 signals can serve only GPIO function.

7 Glossary

Term	Definition
BMC	Baseboard management controller
eMMC	Embedded multimedia card
FPGA	Field-programmable gate array
GPIO	General Purpose I/O
HPS	Hard Processor System
I/O	Input/output
MAC	Media access control
OCP	Open Compute Project
PFL	Parallel Flash Loader
PLL	Phase-locked loop
RGMII	Reduced gigabit media-independent interface
RMII	Reduced media-independent interface
SDIO	Secure digital input output interface
TFTP	Trivial file transfer protocol
USB	Universal serial bus
ymodem	Yet another modem program file transfer protocol

8 References

- [1] OCP_RunBMC_Daughterboard_Card_Design_Specification_v1.4.1

  https://www.opencompute.org/documents/ocp-runbmc-daughterboard-card-design-specification-v1-4-1-pdf

- [2] Hyve RunBMC AST200 Design Package and Quick Start Guide

  https://www.opencompute.org/documents/runbmc-rev1-1-qsg-zip

- [3] Intel® Quartus® Prime Design Software (Standard or Lite Edition)

  https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html

- [4] Intel® FPGA Download Cable II (aka USB Blaster II)

  https://www.intel.com/content/www/us/en/products/sku/215664/intel-fpga-download-cable-ii/

9 License

OCP encourages participants to share their proposals, specifications and designs with the community. This is to promote openness and encourage continuous and open feedback. It is important to remember that by providing feedback for any such documents, whether in written or verbal form, that the contributor or the contributor's organization grants OCP and its members irrevocable right to use this feedback for any purpose without any further obligation.

It is acknowledged that any such documentation and any ancillary materials that are provided to OCP in connection with this document, including without limitation any white papers, articles, photographs, studies, diagrams, contact information (together, "Materials") are made available under the Creative Commons Attribution-ShareAlike 4.0 International License found here: https://creativecommons.org/licenses/by-sa/4.0/, or any later version, and without limiting the foregoing, OCP may make the Materials available under such terms.

As a contributor to this document, all members represent that they have the authority to grant the rights and licenses herein. They further represent and warrant that the Materials do not and will not violate the copyrights or misappropriate the trade secret rights of any third party, including without limitation rights in intellectual property. The contributor(s) also represent that, to the extent the Materials include materials protected by copyright or trade secret rights that are owned or created by any third-party, they have obtained permission for its use consistent with the foregoing. They will provide OCP evidence of such permission upon OCP's request. This document and any "Materials" are published on the respective project's wiki page and are open to the public in accordance with OCP's Bylaws and IP Policy. This can be found at http://www.opencompute.org/participate/legal-documents/. If you have any questions please contact OCP.

10 About Open Compute Foundation

At the core of the Open Compute Project (OCP) is its Community of hyperscale data center operators, joined by telecom and colocation providers and enterprise IT users, working with vendors to develop open innovations that, when embedded in product are deployed from the cloud to the edge. The OCP Foundation is responsible for fostering and serving the OCP Community to meet the market and shape the future, taking hyperscale led innovations to everyone. Meeting the market is accomplished through open designs and best practices, and with data center facility and IT equipment embedding OCP Community-developed innovations for efficiency, at-scale operations and sustainability. Shaping the future includes investing in strategic initiatives that prepare the IT ecosystem for major changes, such as AI & ML, optics, advanced cooling techniques, and composable silicon. Learn more at www.opencompute.org.

11 Appendix A. Cyclone® V soc_system IP configuration

11.1 hps_0 configuration table

Parameters Tab / Section	Option to change
FPGA Interfaces / General	Deselect “Enable MPU standby and event signals”
	Select “**Enable boot from fpga signals**”
FPGA Interfaces / FPGA-to-HPS SDRAM Interface	Delete “f2_sdram0” by selecting entry and using “**-**“ button
FPGA Interfaces / Interrupts	Select “**Enable FPGA-to-HPS Interrupts**”
Peripheral Pins / Ethernet Media Access Controller	EMAC0 pin: **FPGA** EMAC0 mode: **Full**
	EMAC1 pin: **HPS I/O Set 0** EMAC1 mode: **RGMII**
Peripheral Pins / SD/MMC Controller	SDIO pin: **FPGA** SDIO mode: **Full**
Peripheral Pins / USB Controllers	USB0 pin: **HPS I/O Set 0** USB0 PHY interface mode: **SDR with PHY clock output mode**
	USB1 pin: **HPS I/O Set 0** USB1 PHY interface mode: **SDR with PHY clock output mode**
Peripheral Pins / UART Controllers	UART0 pin: **HPS I/O Set 2** UART0 mode: **No Flow Control**
	UART1 pin: **HPS I/O Set0** UART1 mode: **No Flow Control**
Peripheral Pins / I2C Controllers	I2C0 pin: **FPGA** I2C0 mode: **Full**
	I2C1 pin: **FPGA** I2C1 mode: **Full**
	I2C2 pin: **FPGA** I2C2 mode: **Full**
	I2C3 pin: **FPGA** I2C3 mode: **Full**
Peripheral Pins / Trace Port Interface Unit	TRACE pin: **HPS I/O Set 0** TRACE mode: **4-bit Data**
Peripheral Pins / Peripherals Mux Table	Configure HPS pins to work as GPIOs by using “GPIO” buttons: **GPIO00, GPIO09, GPIO29, GPIO30, GPIO31, GPIO32, GPIO34, GPIO35, GPIO57, GPIO58, GPIO59, GPIO61, GPIO62**
HPS Clocks / Peripheral PLL Output clocks – Desired Frequencies	SDMMC clock frequency: 50.0 MHz
SDRAM / PHY Settings / Clocks	Memory clock frequency: 400.0 MHz PLL reference clock frequency: 25.0 MHz
SDRAM / Memory Parameters	Memory device speed grade: 400.0 MHz
	Total Interface width: 40
	Row address width: 15

	Column address width: 10
	Bank-address width: 3
SDRAM / Memory Parameters / Memory Initialization Options	ODT Rtt nominal value: RZQ/6
SDRAM / Memory Timing	tIS (base): 170 ps
	tIH (base): 120 ps
	tDS (base): 10 ps
	tDH (base): 45 ps
	tDQSQ: 100 ps
	tQH 0.38 cycles
	tDQSCK: 225 ps
	tDQSS: 0.27 cycles
	tQSH: 0.4 cycles
	tDSH: 0.18 cycles
	tDSS: 0.18 cycles
	tINIT: 500 us
	tMRD (tMRW): 4 cycles
	tRAS: 35.0 ns
	tRCD: 13.75 ns
	tRP: 13.75 ns
	tREFI (tREFIab): 7.8 us
	tRFC (tRFCab): 260 ns
	tWR: 15.0 ns
	tWTR: 4 cycles
	tFAW: 30.0 ns
	tRRD: 10.0 ns
	tRTP: 10.0 ns

Table 7, Cyclone® V hps_0 configuration

11.2 pll_0 configuration table

Tab / Section name	**Option to change**
General	Reference Clock Frequency: 50.0 MHz
	Deselect "Enable locked output port"
General / Output Clocks	Number Of Clocks: 2
General / Output Clocks /outclk0	Desired Frequency: 100.0 MHz
General / Output Clocks / outclk1	Desired Frequency: 5.0 MHz

Table 8, Cyclone® V pll_0 configuration

11.3 onchip_memory2_0 configuration table

Section	**Option to change**
Size	**Slave S1 Data width: 64**
	Total memory size: 65536
Memory initialization	Select **Enable non-default initialization file**
	User created initialization file: preloader-ocm.mif

Table 9, Cyclone® V onchip_memory2_0 configuration

11.4 fpga_gpio_0 configuration table

Section	**Option to change**
Basic Settings	Width (1-32 bits): **32**
	Direction: **Bidir**
Output Register	Set **Enable individual bit setting/clearing**
Edge capture register	Set **Synchronously capture**
	Edge Type: **ANY**
	Set **Enable bit-clearing for edge capture register**
Interrupt	Set **Generate IRQ**
	IRQ Type: **EDGE**

Table 10, Cyclone® V fpga_gpio_0 configuration

11.5 fpga_gpio_1 configuration table

Section	**Option to change**
Basic Settings	Width (1-32 bits): **8**
	Direction: **Bidir**
Output Register	Set **Enable individual bit setting/clearing**
Edge capture register	Set **Synchronously capture**
	Edge Type: **ANY**
	Set **Enable bit-clearing for edge capture register**
Interrupt	Set **Generate IRQ**
	IRQ Type: **EDGE**

Table 11, Cyclone® V fpga_gpio_1 configuration

12 Appendix B. Cyclone® V top module example (c5.sv)

```
module c5 (
    // FPGA clock and reset
    input            clk_50M_c5_fpga,
```

```
    // HPS IO - memory controller
    output [14:0] memory_mem_a, output [2:0] memory_mem_ba,
    output memory_mem_ck, output memory_mem_ck_n,
    output memory_mem_cke, output memory_mem_cs_n,
    output memory_mem_ras_n, output memory_mem_cas_n,
    output memory_mem_we_n, output memory_mem_reset_n,
    inout [39:0] memory_mem_dq, inout [4:0] memory_mem_dqs,
    inout [4:0] memory_mem_dqs_n, output memory_mem_odt,
    output [4:0] memory_mem_dm, input memory_oct_rzqin,
    // HPS IO - GPIOs
    inout hps_io_hps_io_gpio_inst_GPIO00, inout hps_io_hps_io_gpio_inst_GPIO09,
    inout hps_io_hps_io_gpio_inst_GPIO29, inout hps_io_hps_io_gpio_inst_GPIO30,
    inout hps_io_hps_io_gpio_inst_GPIO31, inout hps_io_hps_io_gpio_inst_GPIO32,
    inout hps_io_hps_io_gpio_inst_GPIO34, inout hps_io_hps_io_gpio_inst_GPIO35,
    inout hps_io_hps_io_gpio_inst_GPIO57, inout hps_io_hps_io_gpio_inst_GPIO58,
    inout hps_io_hps_io_gpio_inst_GPIO59, inout hps_io_hps_io_gpio_inst_GPIO61,
    inout hps_io_hps_io_gpio_inst_GPIO62,
    // RMII (via HPS EMAC0 MII-to-RMII)
    input fpga_mii2rmii_0_clock_sink_clk,
    input fpga_mii2rmii_0_rmii_interface_crs,
    input [1:0] fpga_mii2rmii_0_rmii_interface_rxdata,
    output [1:0] fpga_mii2rmii_0_rmii_interface_txdata,
    output fpga_mii2rmii_0_rmii_interface_txenable,
    //  HPS IO - EMAC1 RGMII
    inout hps_io_hps_io_emac1_inst_MDIO, output hps_io_hps_io_emac1_inst_MDC,
    output hps_io_hps_io_emac1_inst_TX_CLK, output hps_io_hps_io_emac1_inst_TX_CTL,
    output hps_io_hps_io_emac1_inst_TXD0, output hps_io_hps_io_emac1_inst_TXD1,
    output hps_io_hps_io_emac1_inst_TXD2, output hps_io_hps_io_emac1_inst_TXD3,
    input hps_io_hps_io_emac1_inst_RXD0, input hps_io_hps_io_emac1_inst_RXD1,
    input hps_io_hps_io_emac1_inst_RXD2, input hps_io_hps_io_emac1_inst_RXD3,
    input hps_io_hps_io_emac1_inst_RX_CLK, input hps_io_hps_io_emac1_inst_RX_CTL,
    // HPS IO - USB0
    inout hps_io_hps_io_usb0_inst_D0, inout hps_io_hps_io_usb0_inst_D1,
    inout hps_io_hps_io_usb0_inst_D2, inout hps_io_hps_io_usb0_inst_D3,
    inout hps_io_hps_io_usb0_inst_D4, inout hps_io_hps_io_usb0_inst_D5,
    inout hps_io_hps_io_usb0_inst_D6, inout hps_io_hps_io_usb0_inst_D7,
    input hps_io_hps_io_usb0_inst_CLK, output hps_io_hps_io_usb0_inst_STP,
    input hps_io_hps_io_usb0_inst_DIR, input hps_io_hps_io_usb0_inst_NXT,
    // HPS IO - USB1
    inout hps_io_hps_io_usb1_inst_D0, inout hps_io_hps_io_usb1_inst_D1,
    inout hps_io_hps_io_usb1_inst_D2, inout hps_io_hps_io_usb1_inst_D3,
    inout hps_io_hps_io_usb1_inst_D4, inout hps_io_hps_io_usb1_inst_D5,
    inout hps_io_hps_io_usb1_inst_D6, inout hps_io_hps_io_usb1_inst_D7,
    input hps_io_hps_io_usb1_inst_CLK, output hps_io_hps_io_usb1_inst_STP,
    input hps_io_hps_io_usb1_inst_DIR, input hps_io_hps_io_usb1_inst_NXT,
    // HPS IO - UARTs
    input hps_io_hps_io_uart0_inst_RX, output hps_io_hps_io_uart0_inst_TX,
    input hps_io_hps_io_uart1_inst_RX, output hps_io_hps_io_uart1_inst_TX,
    // HPS IO - TRACE
    output hps_io_hps_io_trace_inst_D0, output hps_io_hps_io_trace_inst_D1,
    output hps_io_hps_io_trace_inst_D2, output hps_io_hps_io_trace_inst_D3,
    output hps_io_hps_io_trace_inst_CLK,
    // GPIOs
    inout[31:0] fpga_gpio0, inout [7:0] fpga_gpio1,
    // I2Cs
    inout [11:0] i2c_scl, inout [11:0] i2c_sda,
    // HPS SD/MMC
    inout emmc_clk, inout emmc_cmd, inout [7:0] emmc_dat
);

// HPS SD/MMC interface routed via FPGA
wire sdio_clk, sdio_cmd_i, sdio_cmd_o, sdio_cmd_en;
wire [7:0] sdio_dat_i, sdio_dat_o, sdio_dat_en;
altiobuf sdio_clk_outbuf (.i(sdio_clk), .oe(1'b1), .io(emmc_clk));
```

```
altiobuf sdio_cmd_iobuf (.i(sdio_cmd_o), .oe(sdio_cmd_en), .o(sdio_cmd_i), .io(emmc_cmd));
altiobuf sdio_dat_iobuf [7:0] (.i(sdio_dat_o), .oe(sdio_dat_en), .o(sdio_dat_i), .io(emmc_dat));

// I2C
wire [11:0] i2c_sda_i, i2c_scl_i, i2c_sda_oe, i2c_scl_oe;
wire [1:0] i2c1_sda_oe, i2c1_scl_oe, i2c6_sda_oe, i2c6_scl_oe;
assign {i2c_sda_oe[1], i2c_scl_oe[1]} = {|i2c1_sda_oe, |i2c1_scl_oe};
assign {i2c_sda_oe[6], i2c_scl_oe[6]} = {|i2c6_sda_oe, |i2c6_scl_oe};
altiobuf i2c_scl_iobuf [11:0] (.i(1'b0), .oe(i2c_scl_oe), .o(i2c_scl_i), .io(i2c_scl));
altiobuf i2c_sda_iobuf [11:0] (.i(1'b0), .oe(i2c_sda_oe), .o(i2c_sda_i), .io(i2c_sda));

// EMAC0 with MII-to-RMII connection
wire [7:0] mii_tx_d, mii_rx_d;
wire mii_tx_en, mii_tx_err, mii_rx_dv, mii_rx_err, mii_crs, mii_col;

wire hps_fpga_reset_n;

soc_system soc_inst (
    .clk_clk                 (clk_50M_c5_fpga),
    .reset_reset_n           (hps_fpga_reset_n),
    .hps_0_h2f_reset_reset_n (hps_fpga_reset_n),
    .hps_0_f2h_boot_from_fpga_boot_from_fpga_ready      (1'b1),
    .hps_0_f2h_boot_from_fpga_boot_from_fpga_on_failure (1'b0),
    /* I2C */
    .hps_0_i2c0_out_data(i2c_sda_oe[0]), .hps_0_i2c0_sda(i2c_sda_i[0]),
    .hps_0_i2c0_clk_clk(i2c_scl_oe[0]), .hps_0_i2c0_scl_in_clk(i2c_scl_i[0]),
    .hps_0_i2c1_out_data(i2c_sda_oe[9]), .hps_0_i2c1_sda(i2c_sda_i[9]),
    .hps_0_i2c1_clk_clk(i2c_scl_oe[9]), .hps_0_i2c1_scl_in_clk(i2c_scl_i[9]),
    .hps_0_i2c2_out_data(i2c1_sda_oe[1]), .hps_0_i2c2_sda(i2c_sda_i[1]),
    .hps_0_i2c2_clk_clk(i2c1_scl_oe[1]), .hps_0_i2c2_scl_in_clk(i2c_scl_i[1]),
    .hps_0_i2c3_out_data(i2c6_sda_oe[1]), .hps_0_i2c3_sda(i2c_sda_i[6]),
    .hps_0_i2c3_clk_clk(i2c6_scl_oe[1]), .hps_0_i2c3_scl_in_clk(i2c_scl_i[6]),
    .fpga_i2c_0_i2c_serial_sda_oe(i2c1_sda_oe[0]), .fpga_i2c_0_i2c_serial_sda_in(i2c_sda_i[1]),
    .fpga_i2c_0_i2c_serial_scl_oe(i2c1_scl_oe[0]), .fpga_i2c_0_i2c_serial_scl_in(i2c_scl_i[1]),
    .fpga_i2c_1_i2c_serial_sda_oe(i2c6_sda_oe[0]), .fpga_i2c_1_i2c_serial_sda_in(i2c_sda_i[6]),
    .fpga_i2c_1_i2c_serial_scl_oe(i2c6_scl_oe[0]), .fpga_i2c_1_i2c_serial_scl_in(i2c_scl_i[6]),
    .fpga_i2c_2_i2c_serial_sda_oe(i2c_sda_oe[8]), .fpga_i2c_2_i2c_serial_sda_in(i2c_sda_i[8]),
    .fpga_i2c_2_i2c_serial_scl_oe(i2c_scl_oe[8]), .fpga_i2c_2_i2c_serial_scl_in(i2c_scl_i[8]),
    .fpga_i2c_3_i2c_serial_sda_oe(i2c_sda_oe[7]), .fpga_i2c_3_i2c_serial_sda_in(i2c_sda_i[7]),
    .fpga_i2c_3_i2c_serial_scl_oe(i2c_scl_oe[7]), .fpga_i2c_3_i2c_serial_scl_in(i2c_scl_i[7]),
    .fpga_i2c_4_i2c_serial_sda_oe(i2c_sda_oe[4]), .fpga_i2c_4_i2c_serial_sda_in(i2c_sda_i[4]),
    .fpga_i2c_4_i2c_serial_scl_oe(i2c_scl_oe[4]), .fpga_i2c_4_i2c_serial_scl_in(i2c_scl_i[4]),
    .fpga_i2c_5_i2c_serial_sda_oe(i2c_sda_oe[2]), .fpga_i2c_5_i2c_serial_sda_in(i2c_sda_i[2]),
    .fpga_i2c_5_i2c_serial_scl_oe(i2c_scl_oe[2]), .fpga_i2c_5_i2c_serial_scl_in(i2c_scl_i[2]),
    .fpga_i2c_6_i2c_serial_sda_oe(i2c_sda_oe[11]), .fpga_i2c_6_i2c_serial_sda_in(i2c_sda_i[11]),
    .fpga_i2c_6_i2c_serial_scl_oe(i2c_scl_oe[11]), .fpga_i2c_6_i2c_serial_scl_in(i2c_scl_i[11]),
    .fpga_i2c_7_i2c_serial_sda_oe(i2c_sda_oe[3]), .fpga_i2c_7_i2c_serial_sda_in(i2c_sda_i[3]),
    .fpga_i2c_7_i2c_serial_scl_oe(i2c_scl_oe[3]), .fpga_i2c_7_i2c_serial_scl_in(i2c_scl_i[3]),
    .fpga_i2c_8_i2c_serial_sda_oe(i2c_sda_oe[5]), .fpga_i2c_8_i2c_serial_sda_in(i2c_sda_i[5]),
    .fpga_i2c_8_i2c_serial_scl_oe(i2c_scl_oe[5]), .fpga_i2c_8_i2c_serial_scl_in(i2c_scl_i[5]),
    .fpga_i2c_9_i2c_serial_sda_oe(i2c_sda_oe[10]), .fpga_i2c_9_i2c_serial_sda_in(i2c_sda_i[10]),
    .fpga_i2c_9_i2c_serial_scl_oe(i2c_scl_oe[10]), .fpga_i2c_9_i2c_serial_scl_in(i2c_scl_i[10]),
    /* EMAC0 - MII-to-RMII */
    .hps_0_emac0_phy_txen_o(mii_tx_en), .fpga_mii2rmii_0_mii_interface_mii_tx_en(mii_tx_en),
    .hps_0_emac0_phy_txd_o(mii_tx_d), .fpga_mii2rmii_0_mii_interface_mii_tx_d(mii_tx_d[3:0]),
    .hps_0_emac0_phy_txer_o(mii_tx_err), .fpga_mii2rmii_0_mii_interface_mii_tx_err(mii_tx_err),
    .hps_0_emac0_phy_rxdv_i(mii_rx_dv), .fpga_mii2rmii_0_mii_interface_mii_rx_dv(mii_rx_dv),
    .hps_0_emac0_phy_rxd_i(mii_rx_d), .fpga_mii2rmii_0_mii_interface_mii_rx_d(mii_rx_d[3:0]),
    .hps_0_emac0_phy_rxer_i(mii_rx_err), .fpga_mii2rmii_0_mii_interface_mii_rx_err(mii_rx_err),
    .hps_0_emac0_phy_crs_i(mii_crs), .fpga_mii2rmii_0_mii_interface_mii_crs(mii_crs),
    .hps_0_emac0_phy_col_i(mii_col), .fpga_mii2rmii_0_mii_interface_mii_col(mii_col),
    .hps_0_emac0_ptp_aux_ts_trig_i(1'b0), .hps_0_emac0_gmii_mdo_o(), .hps_0_emac0_gmii_mdo_o_e(),
    .hps_0_emac0_gmii_mdi_i(), .hps_0_emac0_ptp_pps_o(), .fpga_mii2rmii_0_reset_sink_reset_n(1'b1),
    .fpga_mii2rmii_0_rmii_interface_rxerror(1'b0), .fpga_mii2rmii_0_macspeed_ena_10(1'b0),
```

```
    /* SDIO */
    .hps_0_sdio_cmd_i(sdio_cmd_i), .hps_0_sdio_data_i(sdio_dat_i),
    .hps_0_sdio_cmd_o(sdio_cmd_o), .hps_0_sdio_cmd_en(sdio_cmd_en),
    .hps_0_sdio_data_o(sdio_dat_o), .hps_0_sdio_data_en(sdio_dat_en),
    .hps_0_sdio_cclk_clk(sdio_clk), .hps_0_sdio_vs_o(), .hps_0_sdio_pwr_ena_o(),
    .hps_0_sdio_wp_i(), .hps_0_sdio_cdn_i(), .hps_0_sdio_card_intn_i(),
    // FPGA GPIOs
    .fpga_gpio_0_external_connection_export(fpga_gpio0),
    .fpga_gpio_1_external_connection_export(fpga_gpio1),
    // Implicit port connection with top module, directly from HPS to pin. Includes:
    // memory, HPS GPIO, EMAC1 RGMII, USBs, UARTs and TRACE
    .*
);

endmodule

// alt_iobuf wrapper to allow array instantiation
module altiobuf ( input i, oe, output o, inout io);
    alt_iobuf b(.i(i), .oe(oe), .o(o), .io(io));
endmodule
```

Example 3, Cyclone® V top module (c5.sv)

13 Appendix C. Cyclone® V timing constraints example (c5.sdc)

```
create_clock -period 20 [get_ports clk_50M_c5_fpga]
derive_pll_clocks -create_base_clocks
derive_clock_uncertainty

create_clock -period "48 MHz" [get_ports hps_io_hps_io_usb0_inst_CLK]
create_clock -period "48 MHz" [get_ports hps_io_hps_io_usb1_inst_CLK]

# EMAC0 RMII constrains
create_clock -name rmii_ref_clk_in -period "50 MHz" [get_ports {fpga_mii2rmii_0_clock_sink_clk}]
set_input_delay -clock rmii_ref_clk_in -max 14 \
    [get_ports {fpga_mii2rmii_0_rmii_interface_rxdata* fpga_mii2rmii_0_rmii_interface_crs}]

set_input_delay -clock rmii_ref_clk_in -min 2 \
    [get_ports {fpga_mii2rmii_0_rmii_interface_rxdata* fpga_mii2rmii_0_rmii_interface_crs}]

set_output_delay -clock rmii_ref_clk_in -min 2 \
    [get_ports {fpga_mii2rmii_0_rmii_interface_txdata* fpga_mii2rmii_0_rmii_interface_txenable}]

set_output_delay -clock rmii_ref_clk_in -max 4 \
    [get_ports {fpga_mii2rmii_0_rmii_interface_txdata* fpga_mii2rmii_0_rmii_interface_txenable}]

create_generated_clock -name emac0_mii_clk_25M -divide_by 2 \
    -source [get_ports {fpga_mii2rmii_0_clock_sink_clk}] \
    [get_registers {soc_system:soc_inst|intel_fpga_mii2rmii:fpga_mii2rmii_0|clkdiv2:u0_clkdiv|clkby2}]
```

Example 4, Cyclone® V timing constraints (c5.sdc)

14 Appendix D. Cyclone® V pin assignment

Signal Name	Location	I/O Standard	Current Strength	Slew Rate
clk_50M_c5_fpga	PIN_V11	3.3-V LVTTL		
emmc_clk	PIN_AD26	3.3-V LVTTL	4ma	0
emmc_cmd	PIN_AF26	3.3-V LVTTL	4ma	0
emmc_dat[7]	PIN_V16	3.3-V LVTTL	4ma	0
emmc_dat[6]	PIN_AA24	3.3-V LVTTL	4ma	0
emmc_dat[5]	PIN_AB23	3.3-V LVTTL	4ma	0
emmc_dat[4]	PIN_Y18	3.3-V LVTTL	4ma	0
emmc_dat[3]	PIN_Y17	3.3-V LVTTL	4ma	0
emmc_dat[2]	PIN_AE25	3.3-V LVTTL	4ma	0
emmc_dat[1]	PIN_Y19	3.3-V LVTTL	4ma	0
emmc_dat[0]	PIN_AE26	3.3-V LVTTL	4ma	0
fpga_gpio0[31]	PIN_Y24	3.3-V LVTTL		
fpga_gpio0[30]	PIN_AG14	3.3-V LVTTL		
fpga_gpio0[29]	PIN_AH14	3.3-V LVTTL		
fpga_gpio0[28]	PIN_AF21	3.3-V LVTTL		
fpga_gpio0[27]	PIN_AF22	3.3-V LVTTL		

fpga_gpio0[26]	PIN_AE22	3.3-V LVTTL		
fpga_gpio0[25]	PIN_AD23	3.3-V LVTTL		
fpga_gpio0[24]	PIN_AH21	3.3-V LVTTL		
fpga_gpio0[23]	PIN_AF23	3.3-V LVTTL		
fpga_gpio0[22]	PIN_D8	3.3-V LVTTL		
fpga_gpio0[21]	PIN_E8	3.3-V LVTTL		
fpga_gpio0[20]	PIN_C12	3.3-V LVTTL		
fpga_gpio0[19]	PIN_AG24	3.3-V LVTTL		
fpga_gpio0[18]	PIN_D12	3.3-V LVTTL		
fpga_gpio0[17]	PIN_AH4	3.3-V LVTTL		
fpga_gpio0[16]	PIN_AE24	3.3-V LVTTL		
fpga_gpio0[15]	PIN_AH2	3.3-V LVTTL		
fpga_gpio0[14]	PIN_AF6	3.3-V LVTTL		
fpga_gpio0[13]	PIN_AH3	3.3-V LVTTL		
fpga_gpio0[12]	PIN_T13	3.3-V LVTTL		
fpga_gpio0[11]	PIN_AC23	3.3-V LVTTL		
fpga_gpio0[10]	PIN_AE23	3.3-V LVTTL		
fpga_gpio0[9]	PIN_AA26	3.3-V LVTTL		
fpga_gpio0[8]	PIN_AF11	3.3-V LVTTL		
fpga_gpio0[7]	PIN_T12	3.3-V LVTTL		
fpga_gpio0[6]	PIN_AF5	3.3-V LVTTL		
fpga_gpio0[5]	PIN_AF10	3.3-V LVTTL		
fpga_gpio0[4]	PIN_AG6	3.3-V LVTTL		
fpga_gpio0[3]	PIN_AF7	3.3-V LVTTL		
fpga_gpio0[2]	PIN_W20	3.3-V LVTTL		
fpga_gpio0[1]	PIN_AB26	3.3-V LVTTL		
fpga_gpio0[0]	PIN_AH22	3.3-V LVTTL		
fpga_gpio1[7]	PIN_AG15	3.3-V LVTTL		
fpga_gpio1[6]	PIN_AG20	3.3-V LVTTL		
fpga_gpio1[5]	PIN_AD10	3.3-V LVTTL		
fpga_gpio1[4]	PIN_AE4	3.3-V LVTTL		
fpga_gpio1[3]	PIN_U11	3.3-V LVTTL		
fpga_gpio1[2]	PIN_W11	3.3-V LVTTL		
fpga_gpio1[1]	PIN_AF8	3.3-V LVTTL		
fpga_gpio1[0]	PIN_T11	3.3-V LVTTL		
fpga_mii2rmii_0_clock_sink_clk	PIN_V12	3.3-V LVTTL		
fpga_mii2rmii_0_rmii_interface_crs	PIN_AH6	3.3-V LVTTL		
fpga_mii2rmii_0_rmii_interface_rxdata[1]	PIN_AG5	3.3-V LVTTL		
fpga_mii2rmii_0_rmii_interface_rxdata[0]	PIN_AD12	3.3-V LVTTL		

fpga_mii2rmii_0_rmii_interface_txdata[1]	PIN_W12	3.3-V LVTTL		
fpga_mii2rmii_0_rmii_interface_txdata[0]	PIN_AE12	3.3-V LVTTL		
fpga_mii2rmii_0_rmii_interface_txenable	PIN_AH5	3.3-V LVTTL		
i2c_scl[11]	PIN_AH27	3.3-V LVTTL		
i2c_scl[10]	PIN_AH26	3.3-V LVTTL		
i2c_scl[9]	PIN_AG16	3.3-V LVTTL		
i2c_scl[8]	PIN_Y13	3.3-V LVTTL		
i2c_scl[7]	PIN_AA13	3.3-V LVTTL		
i2c_scl[6]	PIN_AG10	3.3-V LVTTL		
i2c_scl[5]	PIN_AH9	3.3-V LVTTL		
i2c_scl[4]	PIN_U14	3.3-V LVTTL		
i2c_scl[3]	PIN_U13	3.3-V LVTTL		
i2c_scl[2]	PIN_AG8	3.3-V LVTTL		
i2c_scl[1]	PIN_AH7	3.3-V LVTTL		
i2c_scl[0]	PIN_AF17	3.3-V LVTTL		
i2c_sda[11]	PIN_AG25	3.3-V LVTTL		
i2c_sda[10]	PIN_AC22	3.3-V LVTTL		
i2c_sda[9]	PIN_AH12	3.3-V LVTTL		
i2c_sda[8]	PIN_AG11	3.3-V LVTTL		
i2c_sda[7]	PIN_AH11	3.3-V LVTTL		
i2c_sda[6]	PIN_AF15	3.3-V LVTTL		
i2c_sda[5]	PIN_AE15	3.3-V LVTTL		
i2c_sda[4]	PIN_AG9	3.3-V LVTTL		
i2c_sda[3]	PIN_AH8	3.3-V LVTTL		
i2c_sda[2]	PIN_AG13	3.3-V LVTTL		
i2c_sda[1]	PIN_AF13	3.3-V LVTTL		
i2c_sda[0]	PIN_V13	3.3-V LVTTL		
memory_mem_a[14]	PIN_G23	SSTL-15 Class I	maximum current	
memory_mem_a[13]	PIN_C24	SSTL-15 Class I	maximum current	
memory_mem_a[12]	PIN_D24	SSTL-15 Class I	maximum current	
memory_mem_a[11]	PIN_B24	SSTL-15 Class I	maximum current	
memory_mem_a[10]	PIN_A24	SSTL-15 Class I	maximum current	
memory_mem_a[9]	PIN_F25	SSTL-15 Class I	maximum current	
memory_mem_a[8]	PIN_F26	SSTL-15 Class I	maximum current	
memory_mem_a[7]	PIN_B26	SSTL-15 Class I	maximum current	
memory_mem_a[6]	PIN_C26	SSTL-15 Class I	maximum current	
memory_mem_a[5]	PIN_J20	SSTL-15 Class I	maximum current	
memory_mem_a[4]	PIN_J21	SSTL-15 Class I	maximum current	
memory_mem_a[3]	PIN_D26	SSTL-15 Class I	maximum current	

memory_mem_a[2]	PIN_E26	SSTL-15 Class I	maximum current	
memory_mem_a[1]	PIN_B28	SSTL-15 Class I	maximum current	
memory_mem_a[0]	PIN_C28	SSTL-15 Class I	maximum current	
memory_mem_ba[2]	PIN_G25	SSTL-15 Class I	maximum current	
memory_mem_ba[1]	PIN_H25	SSTL-15 Class I	maximum current	
memory_mem_ba[0]	PIN_A27	SSTL-15 Class I	maximum current	
memory_mem_cas_n	PIN_A26	SSTL-15 Class I	maximum current	
memory_mem_ck	PIN_N21	Differential 1.5-V SSTL Class I		
memory_mem_ck_n	PIN_N20	Differential 1.5-V SSTL Class I		
memory_mem_cke	PIN_L28	SSTL-15 Class I	maximum current	
memory_mem_cs_n	PIN_L21	SSTL-15 Class I	maximum current	
memory_mem_dm[4]	PIN_AE28	SSTL-15 Class I		
memory_mem_dm[3]	PIN_AB28	SSTL-15 Class I		
memory_mem_dm[2]	PIN_W28	SSTL-15 Class I		
memory_mem_dm[1]	PIN_P28	SSTL-15 Class I		
memory_mem_dm[0]	PIN_G28	SSTL-15 Class I		
memory_mem_dq[39]	PIN_AD28	SSTL-15 Class I		
memory_mem_dq[38]	PIN_AE27	SSTL-15 Class I		
memory_mem_dq[37]	PIN_V20	SSTL-15 Class I		
memory_mem_dq[36]	PIN_V19	SSTL-15 Class I		
memory_mem_dq[35]	PIN_V25	SSTL-15 Class I		
memory_mem_dq[34]	PIN_AC28	SSTL-15 Class I		
memory_mem_dq[33]	PIN_U25	SSTL-15 Class I		
memory_mem_dq[32]	PIN_T26	SSTL-15 Class I		
memory_mem_dq[31]	PIN_AA27	SSTL-15 Class I		
memory_mem_dq[30]	PIN_Y27	SSTL-15 Class I		
memory_mem_dq[29]	PIN_T24	SSTL-15 Class I		
memory_mem_dq[28]	PIN_R24	SSTL-15 Class I		
memory_mem_dq[27]	PIN_W26	SSTL-15 Class I		
memory_mem_dq[26]	PIN_AA28	SSTL-15 Class I		
memory_mem_dq[25]	PIN_R25	SSTL-15 Class I		
memory_mem_dq[24]	PIN_R26	SSTL-15 Class I		
memory_mem_dq[23]	PIN_V27	SSTL-15 Class I		
memory_mem_dq[22]	PIN_R27	SSTL-15 Class I		
memory_mem_dq[21]	PIN_N27	SSTL-15 Class I		
memory_mem_dq[20]	PIN_N26	SSTL-15 Class I		
memory_mem_dq[19]	PIN_U28	SSTL-15 Class I		
memory_mem_dq[18]	PIN_T28	SSTL-15 Class I		
memory_mem_dq[17]	PIN_N25	SSTL-15 Class I		

memory_mem_dq[16]	PIN_N24	SSTL-15 Class I		
memory_mem_dq[15]	PIN_N28	SSTL-15 Class I		
memory_mem_dq[14]	PIN_M28	SSTL-15 Class I		
memory_mem_dq[13]	PIN_M26	SSTL-15 Class I		
memory_mem_dq[12]	PIN_M27	SSTL-15 Class I		
memory_mem_dq[11]	PIN_J28	SSTL-15 Class I		
memory_mem_dq[10]	PIN_J27	SSTL-15 Class I		
memory_mem_dq[9]	PIN_L25	SSTL-15 Class I		
memory_mem_dq[8]	PIN_K25	SSTL-15 Class I		
memory_mem_dq[7]	PIN_F28	SSTL-15 Class I		
memory_mem_dq[6]	PIN_G27	SSTL-15 Class I		
memory_mem_dq[5]	PIN_K26	SSTL-15 Class I		
memory_mem_dq[4]	PIN_J26	SSTL-15 Class I		
memory_mem_dq[3]	PIN_D27	SSTL-15 Class I		
memory_mem_dq[2]	PIN_E28	SSTL-15 Class I		
memory_mem_dq[1]	PIN_J24	SSTL-15 Class I		
memory_mem_dq[0]	PIN_J25	SSTL-15 Class I		
memory_mem_dqs[4]	PIN_V18	Differential 1.5-V SSTL Class I		
memory_mem_dqs[3]	PIN_U19	Differential 1.5-V SSTL Class I		
memory_mem_dqs[2]	PIN_T19	Differential 1.5-V SSTL Class I		
memory_mem_dqs[1]	PIN_R19	Differential 1.5-V SSTL Class I		
memory_mem_dqs[0]	PIN_R17	Differential 1.5-V SSTL Class I		
memory_mem_dqs_n[4]	PIN_V17	Differential 1.5-V SSTL Class I		
memory_mem_dqs_n[3]	PIN_T20	Differential 1.5-V SSTL Class I		
memory_mem_dqs_n[2]	PIN_T18	Differential 1.5-V SSTL Class I		
memory_mem_dqs_n[1]	PIN_R18	Differential 1.5-V SSTL Class I		
memory_mem_dqs_n[0]	PIN_R16	Differential 1.5-V SSTL Class I		
memory_mem_odt	PIN_D28	SSTL-15 Class I	maximum current	
memory_mem_ras_n	PIN_A25	SSTL-15 Class I	maximum current	
memory_mem_reset_n	PIN_V28	SSTL-15 Class I	maximum current	
memory_mem_we_n	PIN_E25	SSTL-15 Class I	maximum current	
memory_oct_rzqin	PIN_D25	SSTL-15 Class I		

Table 12, Cyclone® V pin assignment