Лабораторна робота № 4

Ядро ARM та Intel FPGA Monitor Program
Частина2 : Налаштування ядра ARM

```c
#include <stdio.h>
#include <time.h>

/* This program demonstrates use of functions in C Standard Library through semihosting of Arm A9.
 * It performs the following:
 *  1. prints characters to the terminal of the host computer
 *  2. reads characters from host computer to target system
*/
int main(void){

    char str[64];
    int age = 0;

    // Get the initial timestamp
    clock_t start_time = clock();

    // Printf, scanf, and some time functions
    while(1){
        printf("What is your name?\n");
        scanf("%s",str);
        printf("What is your age?\n");
        scanf("%d",&age);

        printf("%s, you are %d years old.\n",str,age);
        printf("It's been %d seconds since we started\n",(int)(clock()-start_time)/CLOCKS_PER_SEC);
        printf("It's been %d seconds since 1970 Jan 1\n",(int)time(NULL));
    }

    return 0;
}
```

Джерела, за якими підготовлено інструкцію:

Monitor Program Tutorial for the ARM Processor

ftp://ftp.intel.com/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Intel_FPGA_Monitor_Program_ARM.pdf

Intel FPGA Academic Program Tutorials

https://software.intel.com/en-us/fpga-academic/learn/tutorials

Вступ

Мета цієї лабораторної роботи – навчитися самостійно налаштовувати процесорне ядро ARM. Познайомитись з архітектурою ядра можна за наступним посиланням: https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/ARM_A9_intro_intelfpga.pdf .

Крім того, до процесорного ядра будуть додані компоненти, що синтезуються на логічних комірках FPGA, та поєднуються з ARM за допомогою спеціальних каналів зв'язку (LWH2F – Lightweight HPS-to-FPGA bridge).

На завершення, для синтезованої системи на кристалі, буде розроблено та перевірено відповідне програмне забезпечення.

1. **Порядок виконання роботи.**

1. Створіть у середовищі Quartus Prime новий проект. Для цього:
   - встановіть робочу директорію проекту;
   - вкажіть назву проекту – ***comp_arm;***
   - оберіть тип мікросхеми - **5CSEMA5F31C6.**
2. Додайте до проекту файли ***comp_arm.v*** та ***hex7seg.v***. Визначте файл ***comp_arm.v*** файлом верхнього рівня ієрархії.
3. Відкрийте середовище QSys.
4. Перед початком створення нової системи видаліть компонент ***Clock Source***.
5. У вікні **IP Catalog** оберіть з папки **University Program → Clock** модуль ***System and SDRAM Clocks for DE-series Boards*** та додайте його до проекту. Цей модуль дозволяє відразу зробити необідні підключення блоку PLL, що використовується для формування тактових сигналів на платі DE1-SoC.

6. Зробіть для нього налаштування у відповідності до рисунку 1

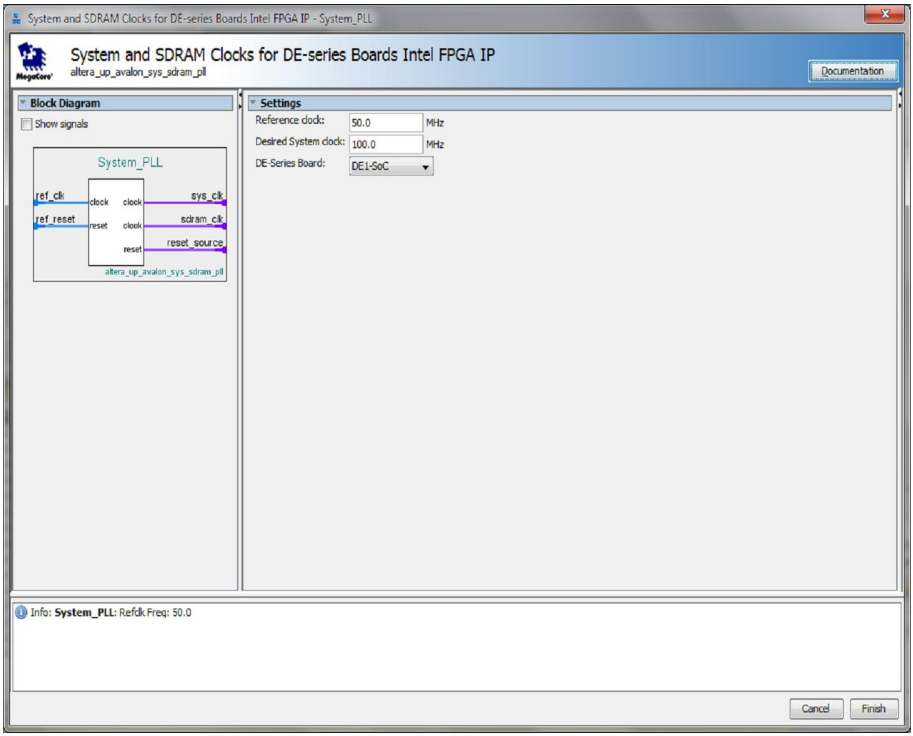


Рисунок 1. Налаштування блоку формування тактових сигналів.

7. Додайте до системи процесорне ядро. Для цього, у вкладці ***IP Catalog*** (вкладка знаходиться зліва у вікні ***Platform Designer Tool***) оберіть ***Library → Processors and Peripherals → Hard Processor Systems → ArriaV/CycloneV Hard Processor System*** та натисніть кнопку ***Add.*** Відкриється вікно зображене на рис. 2.

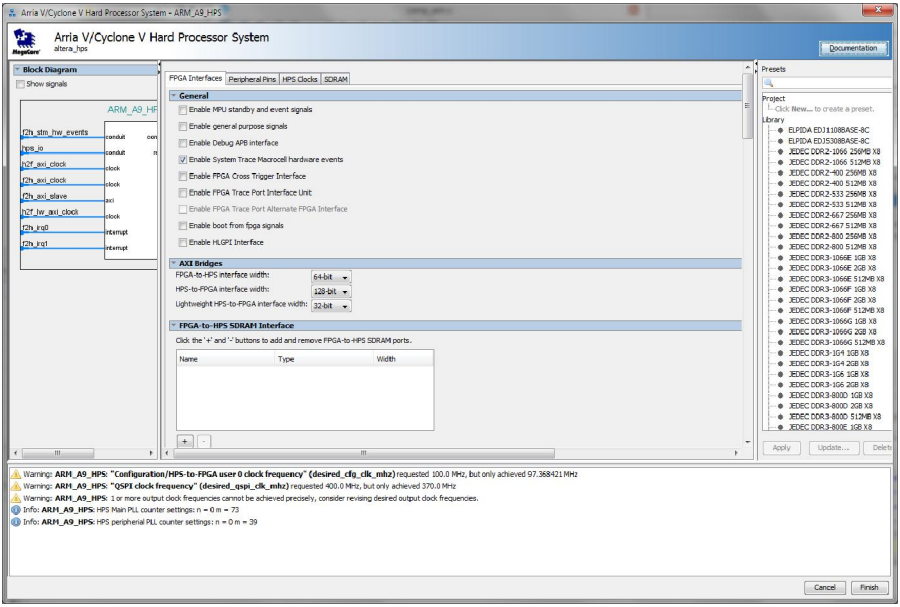


Рисунок 2. Налаштування процесорного ядра ARM.

8. На сторінці ***FPGA Interfaces*** виконайте налаштування у відповідності до рисунків 2 – 4.

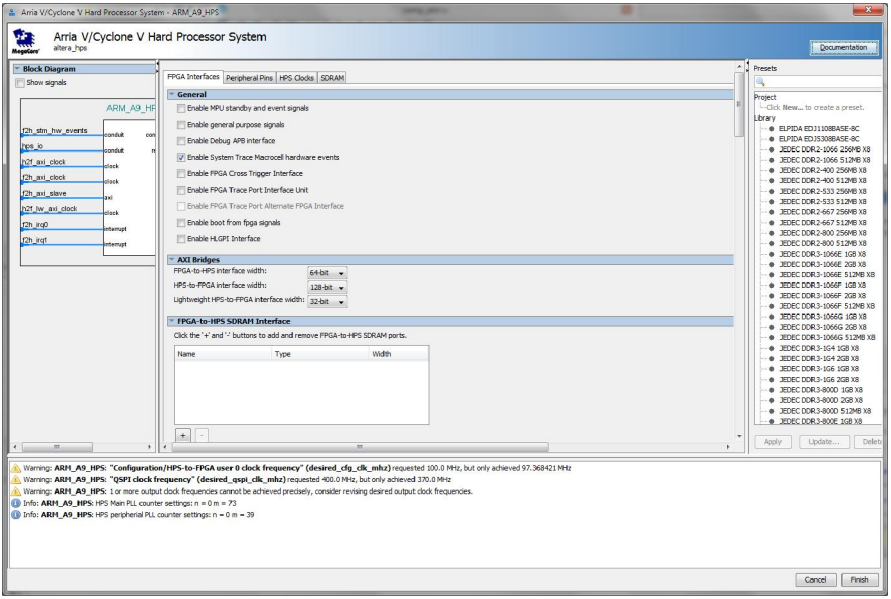


Рисунок 3. Налаштування процесорного ядра ARM.

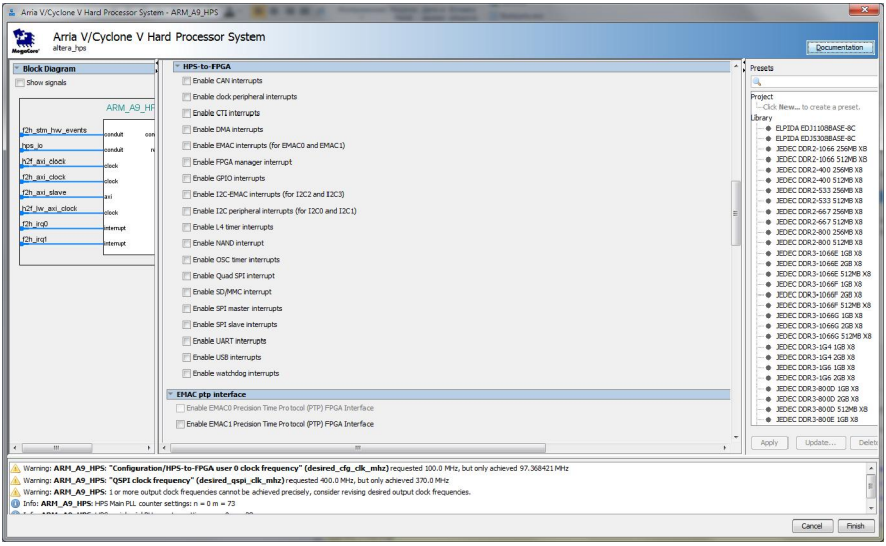


Рисунок 4. Налаштування процесорного ядра ARM.

9. Перейдіть до сторінки ***Peripheral Pins*** та виконайте налаштування периферійних компонентів процесорного ядра у відповідності до рисунків 5 – 7.

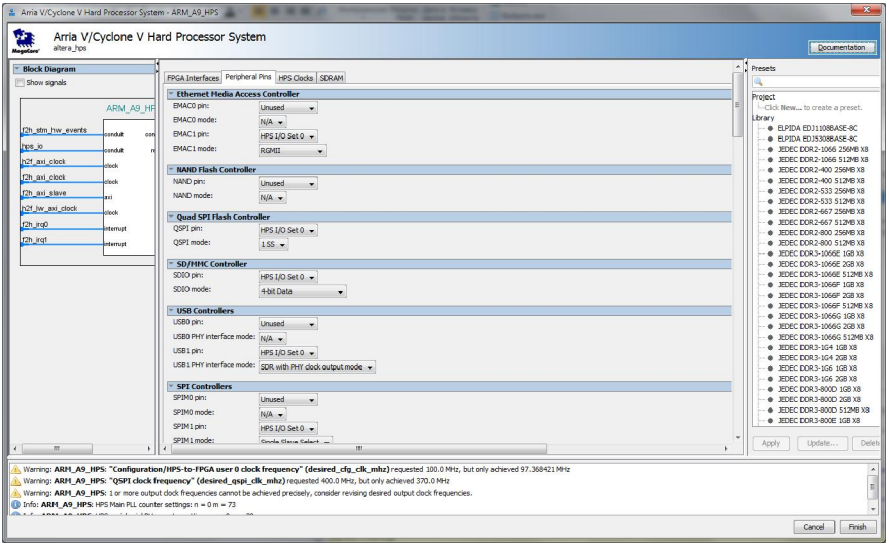


Рисунок 5. Налаштування периферійних пристроїв.

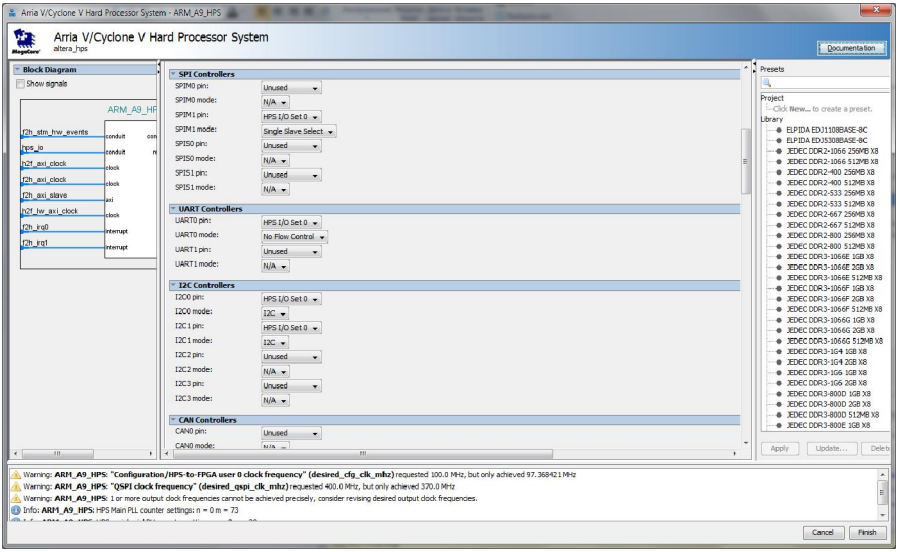


Рисунок 6. Налаштування периферійних пристроїв.

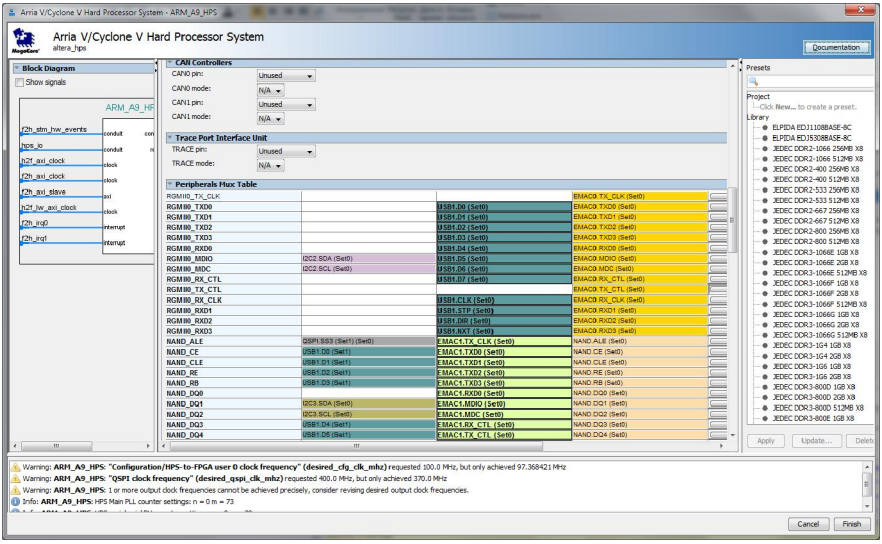


Рисунок 7. Налаштування периферійних пристроїв.

Крім того вкажіть, що процесором будуть використовуватись контакти загального призначення ***GPIO09, GPIO35, GPIO40, GPIO41, GPIO48, GPIO53, GPIO54, GPIO61***. За їх допомогою відбувається взаємодія з зовнішніми периферійними пристроями.

10. Перейдіть до сторінки ***HPS Clocks*** та виконайте налаштування синхронізуючих сигналів процесорного ядра у відповідності до рисунків 8 – 10.

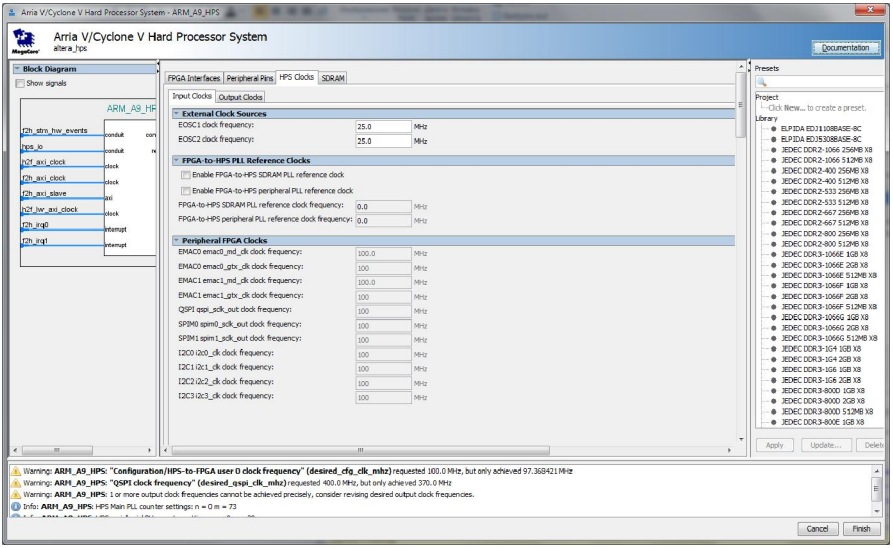


Рисунок 8. Налаштування синхронізуючих сигналів.

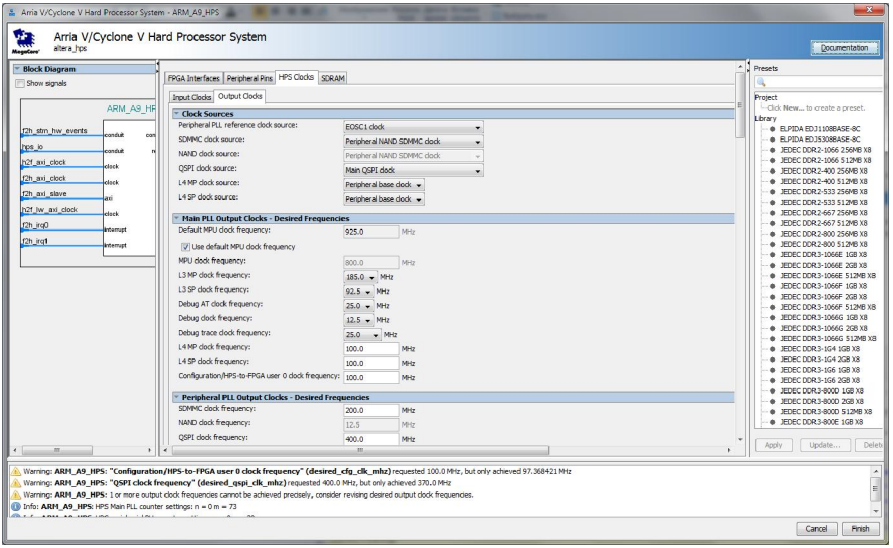


Рисунок 9. Налаштування синхронізуючих сигналів.

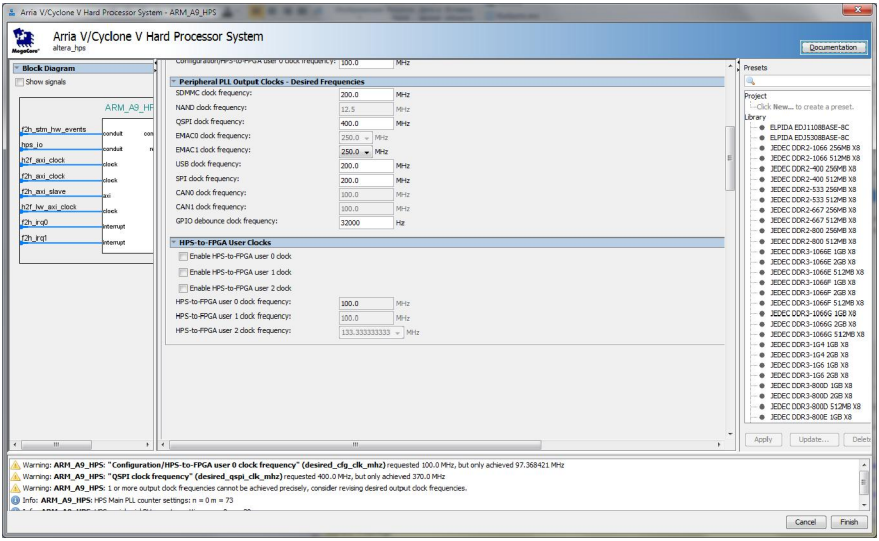


Рисунок 10. Налаштування синхронізуючих сигналів.

11. Перейдіть до сторінки ***SDRAM*** та виконайте налаштування інтерфейсу зовнішньої оперативної пам'яті у відповідності до рисунків 11 – 15.

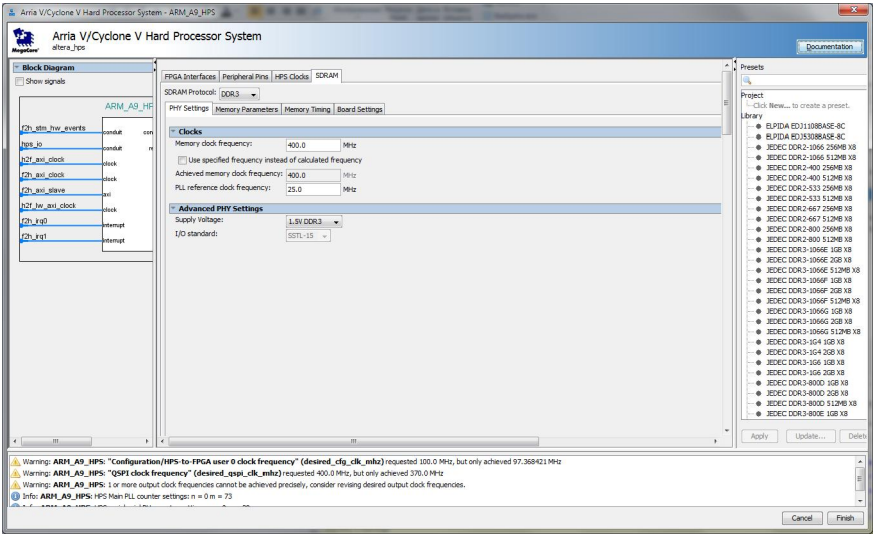


Рисунок 11. Налаштування інтерфейсу SDRAM.

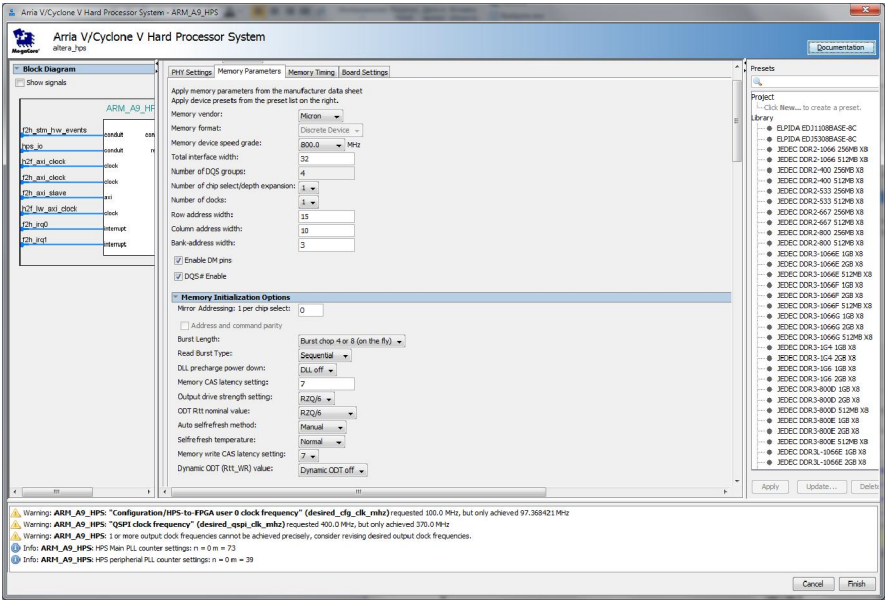


Рисунок 12. Налаштування інтерфейсу SDRAM.

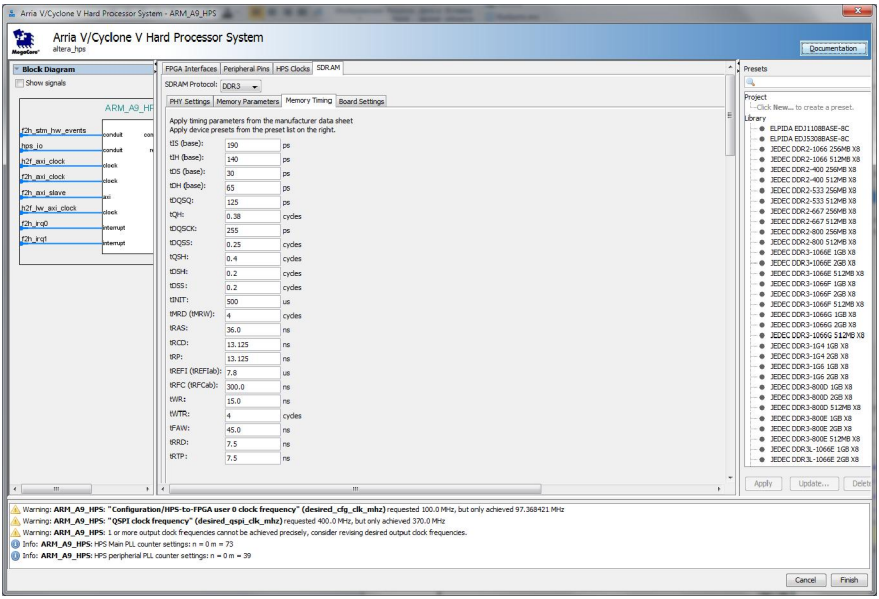


Рисунок 13. Налаштування інтерфейсу SDRAM.

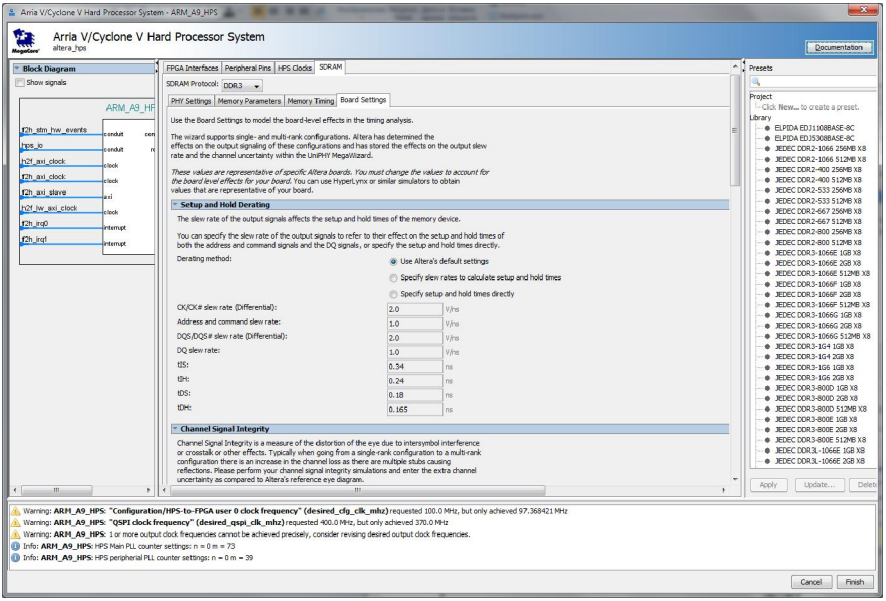


Рисунок 14. Налаштування інтерфейсу SDRAM.

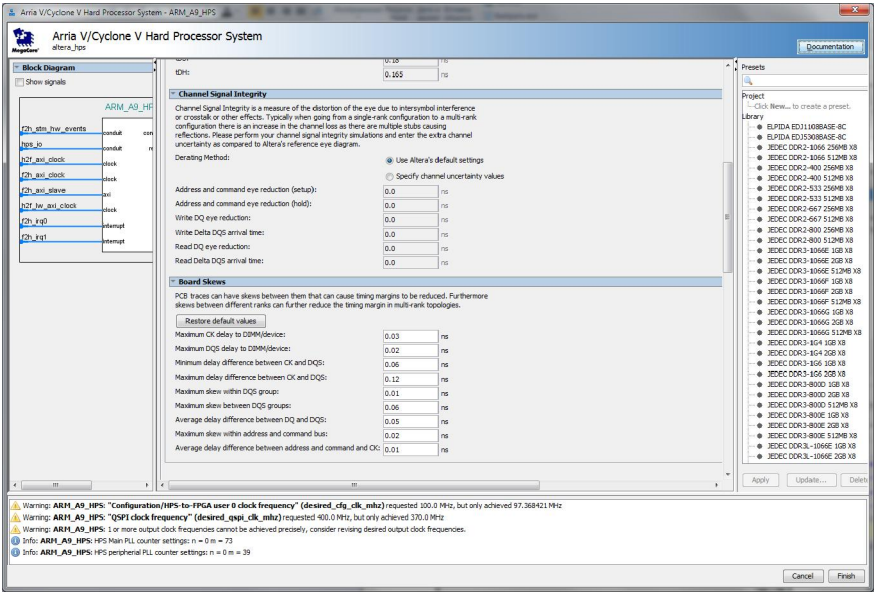


Рисунок 15. Налаштування інтерфейсу SDRAM.

12. Натисніть ***Finish***.

13. Додайте до системи модуль ***JTAG to Avalon Master Bridge.*** Він знаходиться у папці ***Basic Functions -> Bridges and Adaptors -> Memory Mapped.*** Залиште його налаштувння за замовченням (рисунок 16).

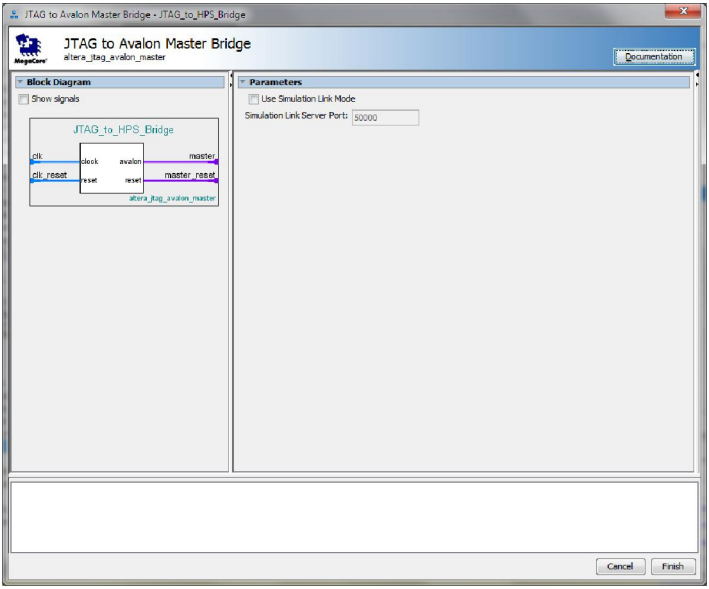


Рисунок 16. Налаштування ***JTAG to Avalon Master Bridge.***

14. Додайте до системи модуль ***On-Chip Memory (RAM or ROM)*** та налаштуйте його у відповідності до рисунку 17.

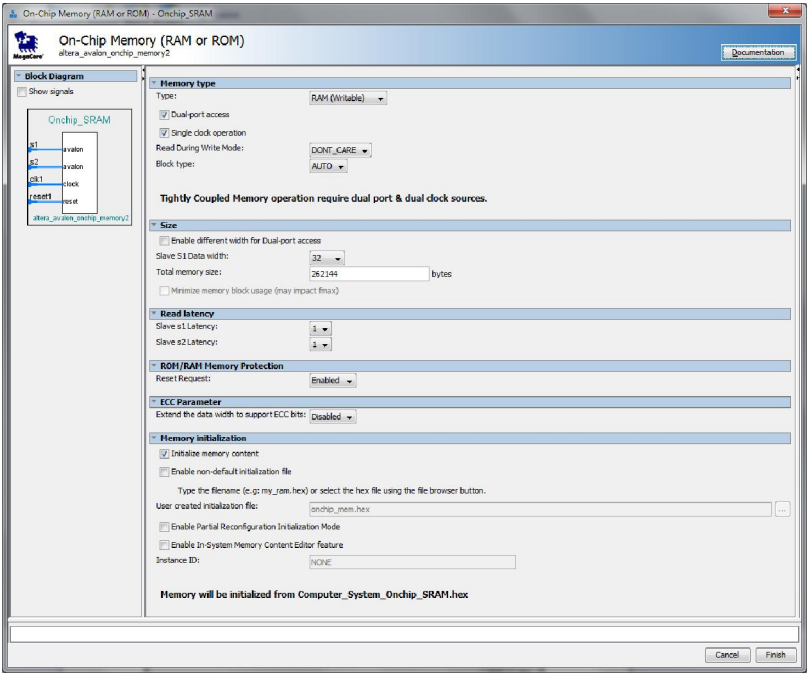


Рисунок 17. Налаштування ***On-Chip Memory (RAM or ROM).***

15. Додайте до системи два модулі ***PIO (Parallel I/O)*** та налаштуйте їх у відповідності до рисунків 18 та 19.

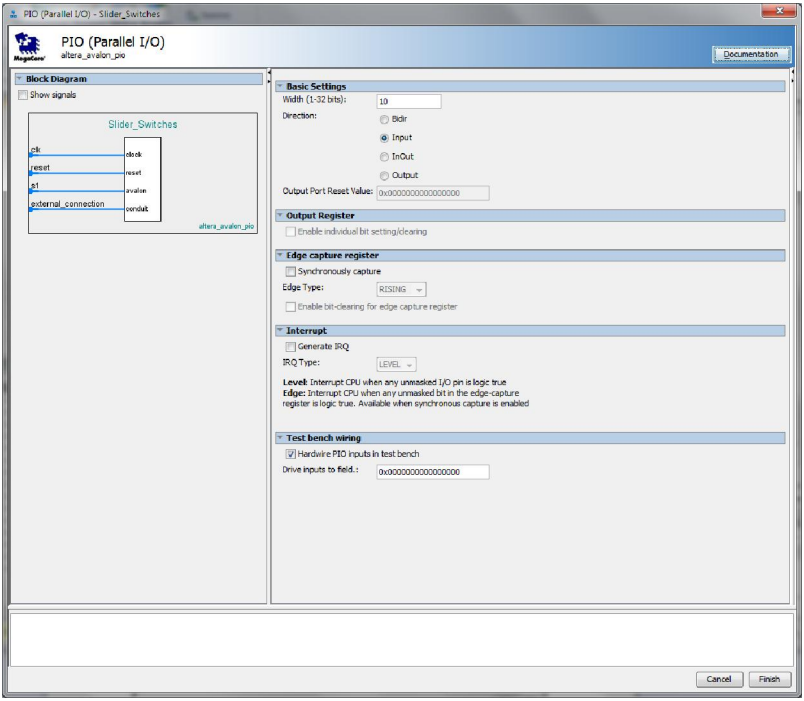


Рисунок 18. Налаштування порту ***Slider_Switches.***

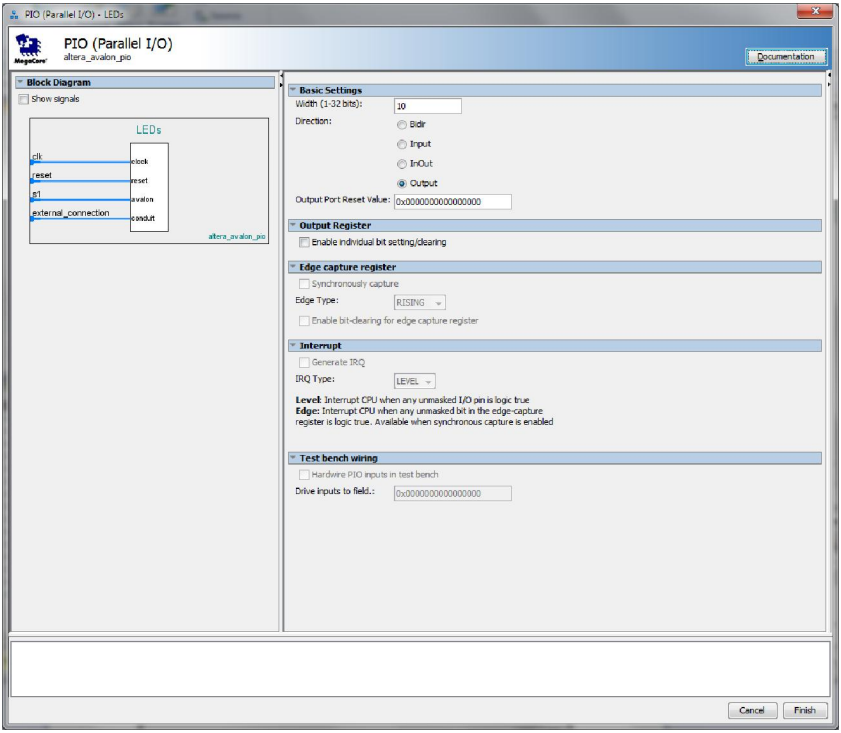


Рисунок 19. Налаштування порту ***LEDs.***

16. Додайте до системи модуль ***System ID Peripheral*** та залиште його налаштування за замовченням (рисунок 20). Він знаходиться у папці ***Basic Functions -> Simulation; Debug and Verification -> Debug and Performance.***

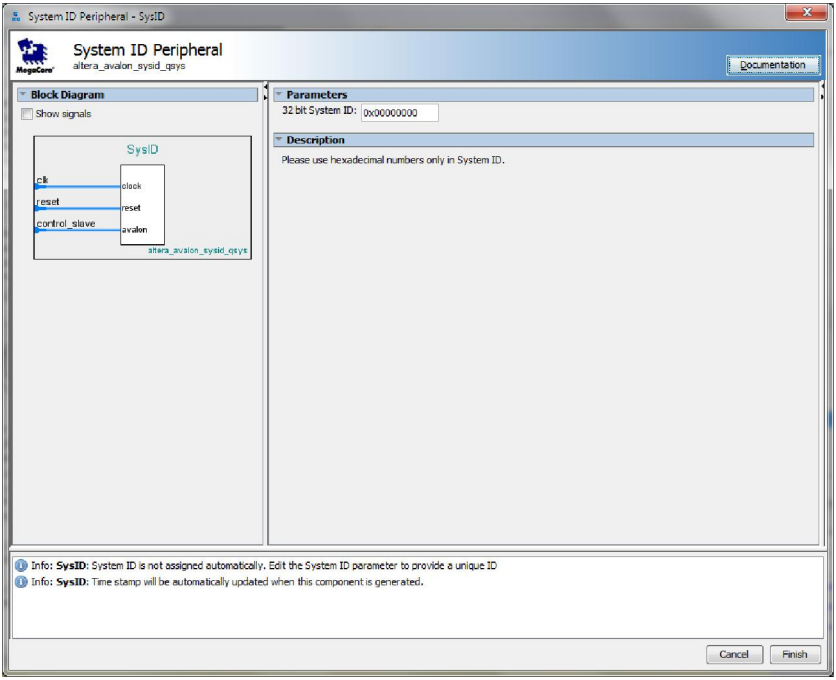


Рисунок 20. Налаштування ***System ID Peripheral.***

17. Зробіть з’єднання компонентів системи у відповідності до рисунку 21. Змініть назви компонентів системи. Визначте зовнішні сигнали (стовпчик ***Export***). Зверніть увагу на базові адреси компонентів.

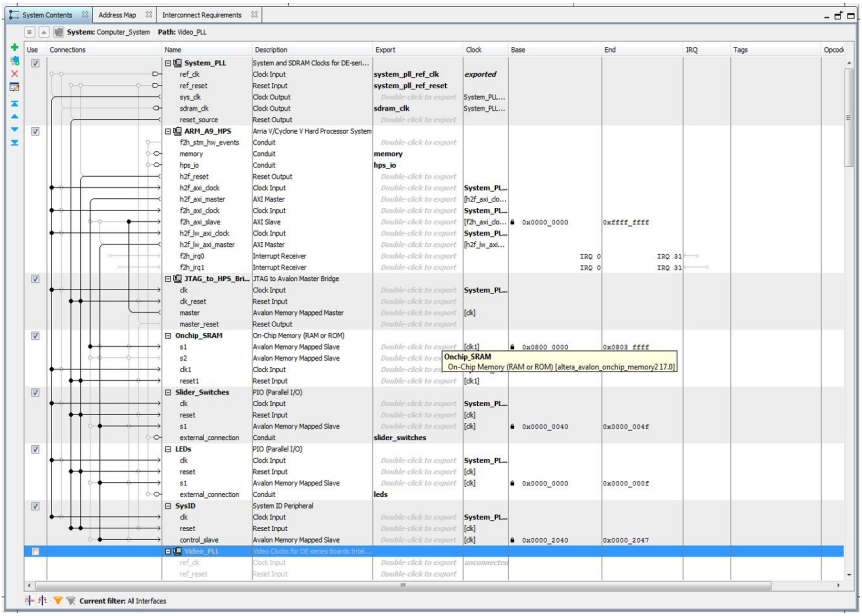


Рисунок 21. З’єднання компонентів системи.

18. Збережіть систему під назвою ***Computer_System.***
Тепер пояснимо, як FPGA взаємодіє з HPS частиною. Між ними є 3 мости:

F2H - FPGA-to-HPS bridge;
H2F - HPS-to-FPGA bridge;
LWH2F - Lightweight HPS-to-FPGA bridge.

F2H дозволяє FPGA звертатись до складових HPS (наприклад USB, або CAN контролери), при цьому FPGA буде master, а HPS — slave. Його слово може мати 32, 64, або 128 біт.

LWH2F використовується для доступу ARM-ядром до IP-cores, що генеруються у FPGA. Слово має розрядність 32 біти. НЕ МОЖНА використовувати для доступу до пам’яті.

H2F використовується для доступу ARM-ядром до пам’яті через FPGA. Його слово може мати 32, 64, або 128 біт.

19. Далі оберіть команду ***Generate → Generate HDL***. Для ***Create simulation model*** оберіть ***None***. Натисніть *Generate* та дочекайтеся сповіщення ***Generate completed***.

20. Поверніться до головного вікна середовища ***Quartus Prime***. Наразі необхідно долучити до проекту створену ***Computer_System***. Для цього у вікні ***Project Navigator → Files*** (зліва) натисніть правою кнопкою миші на піктограмі ***Files*** та оберіть команду ***Add/Remove Files in Project*** з контекстного меню. Відкриється вікно додавання файлів. Додайте файли ***директорія проекту/ Computer_System / synthesis/ Computer_System.v*** та ***Computer_System.qip***.

21. У файлі верхнього рівня ієрархії ***comp_arm.v*** підключіть сигнали до згенерованої системи:

```
Computer_System The_System (
.system_pll_ref_clk_clk          (CLOCK_50),
.system_pll_ref_reset_reset      (1'b0),
.slider_switches_export          (SW),
.leds_export                     (LEDR),
.sdram_clk_clk                   (DRAM_CLK),
.memory_mem_a                    (HPS_DDR3_ADDR),
.memory_mem_ba                   (HPS_DDR3_BA),
.memory_mem_ck                   (HPS_DDR3_CK_P),
.memory_mem_ck_n                 (HPS_DDR3_CK_N),
.memory_mem_cke                  (HPS_DDR3_CKE),
.memory_mem_cs_n                 (HPS_DDR3_CS_N),
.memory_mem_ras_n                (HPS_DDR3_RAS_N),
.memory_mem_cas_n                (HPS_DDR3_CAS_N),
.memory_mem_we_n                 (HPS_DDR3_WE_N),
.memory_mem_reset_n              (HPS_DDR3_RESET_N),
.memory_mem_dq                   (HPS_DDR3_DQ),
.memory_mem_dqs                  (HPS_DDR3_DQS_P),
.memory_mem_dqs_n                (HPS_DDR3_DQS_N),
.memory_mem_odt                  (HPS_DDR3_ODT),
.memory_mem_dm                   (HPS_DDR3_DM),
.memory_oct_rzqin                (HPS_DDR3_RZQ),
.hps_io_hps_io_gpio_inst_GPIO35  (HPS_ENET_INT_N),
```

```
.hps_io_hps_io_emac1_inst_TX_CLK   (HPS_ENET_GTX_CLK),
.hps_io_hps_io_emac1_inst_TXD0     (HPS_ENET_TX_DATA[0]),
.hps_io_hps_io_emac1_inst_TXD1     (HPS_ENET_TX_DATA[1]),
.hps_io_hps_io_emac1_inst_TXD2     (HPS_ENET_TX_DATA[2]),
.hps_io_hps_io_emac1_inst_TXD3     (HPS_ENET_TX_DATA[3]),
.hps_io_hps_io_emac1_inst_RXD0     (HPS_ENET_RX_DATA[0]),
.hps_io_hps_io_emac1_inst_MDIO     (HPS_ENET_MDIO),
.hps_io_hps_io_emac1_inst_MDC      (HPS_ENET_MDC),
.hps_io_hps_io_emac1_inst_RX_CTL   (HPS_ENET_RX_DV),
.hps_io_hps_io_emac1_inst_TX_CTL   (HPS_ENET_TX_EN),
.hps_io_hps_io_emac1_inst_RX_CLK   (HPS_ENET_RX_CLK),
.hps_io_hps_io_emac1_inst_RXD1     (HPS_ENET_RX_DATA[1]),
.hps_io_hps_io_emac1_inst_RXD2     (HPS_ENET_RX_DATA[2]),
.hps_io_hps_io_emac1_inst_RXD3     (HPS_ENET_RX_DATA[3]),
.hps_io_hps_io_qspi_inst_IO0       (HPS_FLASH_DATA[0]),
.hps_io_hps_io_qspi_inst_IO1       (HPS_FLASH_DATA[1]),
.hps_io_hps_io_qspi_inst_IO2       (HPS_FLASH_DATA[2]),
.hps_io_hps_io_qspi_inst_IO3       (HPS_FLASH_DATA[3]),
.hps_io_hps_io_qspi_inst_SS0       (HPS_FLASH_NCSO),
.hps_io_hps_io_qspi_inst_CLK       (HPS_FLASH_DCLK),
.hps_io_hps_io_gpio_inst_GPIO61    (HPS_GSENSOR_INT),
.hps_io_hps_io_gpio_inst_GPIO40    (HPS_GPIO[0]),
.hps_io_hps_io_gpio_inst_GPIO41    (HPS_GPIO[1]),
.hps_io_hps_io_gpio_inst_GPIO48    (HPS_I2C_CONTROL),
.hps_io_hps_io_i2c0_inst_SDA       (HPS_I2C1_SDAT),
.hps_io_hps_io_i2c0_inst_SCL       (HPS_I2C1_SCLK),
.hps_io_hps_io_i2c1_inst_SDA       (HPS_I2C2_SDAT),
.hps_io_hps_io_i2c1_inst_SCL       (HPS_I2C2_SCLK),
.hps_io_hps_io_gpio_inst_GPIO54    (HPS_KEY),
.hps_io_hps_io_gpio_inst_GPIO53    (HPS_LED),
.hps_io_hps_io_sdio_inst_CMD       (HPS_SD_CMD),
.hps_io_hps_io_sdio_inst_D0        (HPS_SD_DATA[0]),
.hps_io_hps_io_sdio_inst_D1        (HPS_SD_DATA[1]),
.hps_io_hps_io_sdio_inst_CLK       (HPS_SD_CLK),
.hps_io_hps_io_sdio_inst_D2        (HPS_SD_DATA[2]),
.hps_io_hps_io_sdio_inst_D3        (HPS_SD_DATA[3]),
.hps_io_hps_io_spim1_inst_CLK      (HPS_SPIM_CLK),
.hps_io_hps_io_spim1_inst_MOSI     (HPS_SPIM_MOSI),
.hps_io_hps_io_spim1_inst_MISO     (HPS_SPIM_MISO),
```

```
.hps_io_hps_io_spim1_inst_SS0      (HPS_SPIM_SS),
.hps_io_hps_io_uart0_inst_RX       (HPS_UART_RX),
.hps_io_hps_io_uart0_inst_TX       (HPS_UART_TX),
.hps_io_hps_io_gpio_inst_GPIO09    (HPS_CONV_USB_N),
.hps_io_hps_io_usb1_inst_D0        (HPS_USB_DATA[0]),
.hps_io_hps_io_usb1_inst_D1        (HPS_USB_DATA[1]),
.hps_io_hps_io_usb1_inst_D2        (HPS_USB_DATA[2]),
.hps_io_hps_io_usb1_inst_D3        (HPS_USB_DATA[3]),
.hps_io_hps_io_usb1_inst_D4        (HPS_USB_DATA[4]),
.hps_io_hps_io_usb1_inst_D5        (HPS_USB_DATA[5]),
.hps_io_hps_io_usb1_inst_D6        (HPS_USB_DATA[6]),
.hps_io_hps_io_usb1_inst_D7        (HPS_USB_DATA[7]),
.hps_io_hps_io_usb1_inst_CLK       (HPS_USB_CLKOUT),
.hps_io_hps_io_usb1_inst_STP       (HPS_USB_STP),
.hps_io_hps_io_usb1_inst_DIR       (HPS_USB_DIR),
.hps_io_hps_io_usb1_inst_NXT       (HPS_USB_NXT)
);
```

22. Збережіть файл.
23. Імпортуйте призначення контактів вводу/виводу з файлу ***comp_arm.qsf.***
24. Виконайте компіляцію проекту.

2. **Розробка програмного забезпечення.**

1. Запустіть ***Intel FPGA Monitor Program***.
2. Оберіть меню ***File → New Project***. Збережіть проект під власною назвою у директорії ***app_software***. Оберіть архітектуру ***ARM Cortex-A9***. Натисніть ***Next***.
3. На вкладинці ***Specify the system*** вкажіть наступні параметри:

Select a system = <Custom system>

System description file =
директорія проекту/Computer_System.sopcinfo

FPGA programming (SOF) file =
директорія проекту /output_files/comp_arm.sof

Preloader = DE1-SoC

4. Натисніть ***Next.***
5. На вкладинці ***Specify a program type*** укажіть значення ***Program Type = Assembly Program***. Натисніть ***Next.***
6. На вкладинці ***Specify program details*** натисніть ***Add,*** та оберіть файли програми ***address_map_arm.s*** та ***simple_program.s .***. Вкажіть ***Start symbol = _start***. Натисніть ***Next.***
   Цей приклад використовувався у лабораторній роботі № 3 – у ньому виконувалося копіювання стану перемикачів до регістру світлодіодів. Додатково, у цьому прикладі додаються семисегментні індикатори, які будуть відображати число, еквівалентне поточному стану світлодіодів.
7. На вкладинці ***Specify system parameters*** укажіть наступні значення

Host connection = DE-SoC [3-2]

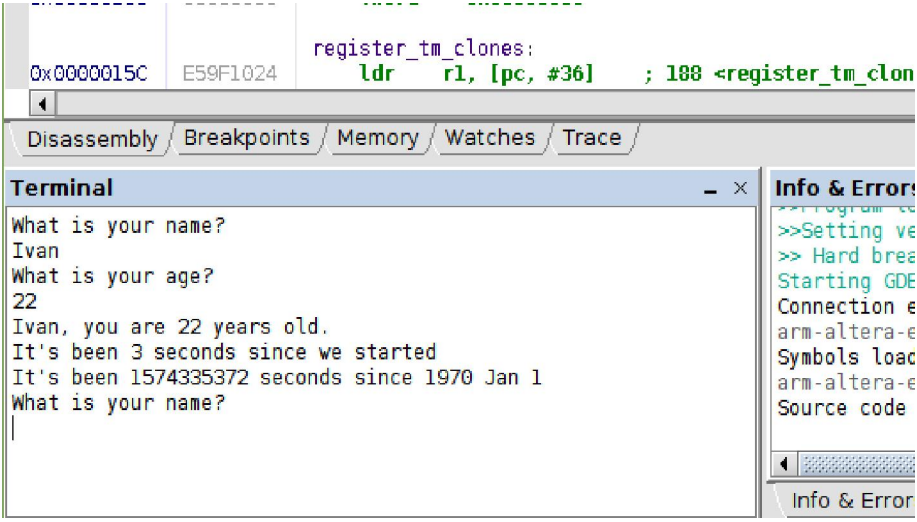
Processor = ARM_A9_HPS_arm_a9_0

Terminal device = Semihosting

8. На останній вкладинці натисніть ***Finish***, або ***Save***. На запит чи завантажувати систему на плату відповідайте ***Yes.***
9. Після сповіщення про успішне завантаження, у вікні ***Intel FPGA Monitor Program —*** оберіть меню ***Actions → Compile & Load.***
10. Запустіть програму на виконання у безперервному (зелений трикутник), або покроковому режимі (жовта стрілочка) та спостерігайте її виконання.
    Переконайтеся, що червоні світлодіоди змінюють свій стан відповідно до стану перемикачів при виконанні інструкції

***stbio R4, 0(R3)*,** а на семи сегментних індикаторах відображається вірне значення.

11. Зупиніть виконання програми.
12. Другим проектом у ***Monitor_Program*** буде програма мовою С, основана на прикладі «робота з Semihosting».
13. Створіть новий проект в окремій директорії. Повторіть налаштунки попереднього проекту – за виключенням того, що проект ґрунтується на скомпільованій програмі AXF, написаної мовою С.
14. Додайте до проекту файл ***semihosting_example.axf.*** Впевніться, що всі додаткові файли (надаються до лабораторної роботи) теж знаходяться у папці проекту.
15. Завантажте систему, скомпілюйте програму та запустіть її на виконання.
    Semihosting — це спеціальний механізм, який забезпечує програмам, що виконуються на ARM-процесорі, доступ до сервісу дебаґера. Стандартні бібліотеки функцій модифіковані для роботи з ***Semihosting***. Наприклад, ***printf()*** та ***scanf()***, які зазвичай працюють з терміналом комп'ютера, на якому запущена програма, що їх викликає, будуть працювати з терміналом програми-дебаґера.
16. Якщо програма працює правильно, то можно буде запустити діалог (надати відповіді на запитання у консольному вікні) які надходять з процесора (рисунок 21).
    Рисунок 21. Робота програми ***Semihosting***



3. Індивідуальне завдання

Внесіть модифікацію до програми ***Semihosting***: нехай відповідь виводиться на світлодіоди. Наприклад, додайте коди:

volatile int * led_ptr = (int *)LEDR_BASE;
****led_ptr = age;***

у потрібних місцях. Коди можуть знаходитись не разом.

Якщо усе зроблено правильно, то після відповіді на запитання "*What is your age?*" вказане число буде виведене на LEDR[7:0] та HEX0-1.