

# LAB 8 – GIAO TIẾP BỘ DMA

## 1. MỤC ĐÍCH

Thông qua bài thực hành này, sinh viên sẽ biết:

- Cấu trúc bộ DMA của Altera (các thanh ghi, chức năng...).
- Cách xây dựng hệ thống phần cứng bằng Qsys để giao tiếp với bộ DMA.
- Cách xây dựng chương trình C trên công cụ Nios II – EDS để giao tiếp với bộ DMA, và sử dụng ngắt của bộ DMA.

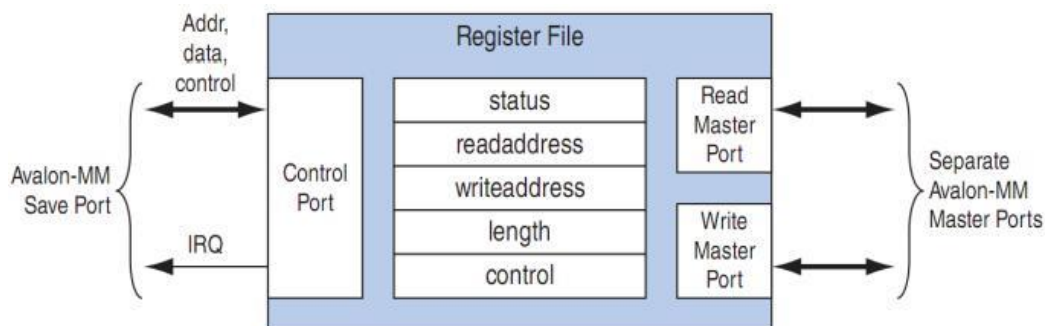
## 2. NỘI DUNG

### 2.1. Lý thuyết về bộ DMA

#### 2.1.1. Giới thiệu

DMA (Direct Memory Access) là một cơ chế truyền dữ liệu trực tiếp giữa hai hay nhiều thành phần trong hệ thống mà không thông qua CPU. Nhờ vậy, các quá trình truyền nhận dữ liệu có thể được thực hiện song song với tốc độ cao. Bộ DMA Controller của Altera thực hiện cơ chế DMA trong hệ thống Qsys.

DMA Controller tiến hành việc truyền dữ liệu từ địa chỉ nguồn (source address) sang địa chỉ đích (destination address). Địa chỉ nguồn hoặc đích có thể là địa chỉ của ngoại vi hoặc bộ nhớ. Ngoài ra, DMA Controller còn hỗ trợ báo hiệu ngắt khi quá trình DMA hoàn tất.



DMA Controller gồm hai Avalon-MM master ports (master read port và master write port) và một Avalon-MM slave port dùng để điều khiển DMA như hình bên trên. DMA Controller hỗ trợ truyền theo kiểu pipeline hoặc burst với độ rộng dữ liệu từ 1 – 16 bytes.

### 2.1.2. Thanh ghi của DMA

Module DMA bao gồm 5 thanh ghi cơ bản như bên dưới.

Offset	Register Name	Read/Write	31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	status (1)	RW	(2)										LEN	WEOP	REOP	BUSY	DONE
1	readaddress	RW	Read master start address														
2	writeaddress	RW	Write master start address														
3	length	RW	DMA transaction length (in bytes)														
4	—	—	Reserved (3)														
5	—	—	Reserved (3)														
6	control	RW	(2)		SOFTWARERESET	QUADWORD	DOUBLEWORD	WCON	RCON	LEEN	WEEN	REEN	I_EN	GO	WORD	HW	BYTE
7	—	—	Reserved (3)														

#### ❖ Thanh ghi “control”

Bit Number	Bit Name	Read/Write/Clear	Description
0	BYTE	RW	Specifies byte transfers.
1	HW	RW	Specifies halfword (16-bit) transfers.
2	WORD	RW	Specifies word (32-bit) transfers.
3	GO	RW	Enables DMA transaction. When the GO bit is set to 0 during idle stage (before execution starts), the DMA is prevented from executing transfers. When the GO bit is set to 1 during idle stage and the length register is non-zero, transfers occur.  If go bit is de-asserted low before write transaction complete, done bit will never go high. It is advisable that GO bit is modified during idle stage (no execution happened) only.
4	I_EN	RW	Enables interrupt requests (IRQ). When the I_EN bit is 1, the DMA controller generates an IRQ when the status register's DONE bit is set to 1. IRQs are disabled when the I_EN bit is 0.
5	REEN	RW	Ends transaction on read-side end-of-packet. When the REEN bit is set to 1, a slave port with flow control on the read side may end the DMA transaction by asserting its end-of-packet signal.
6	WEEN	RW	Ends transaction on write-side end-of-packet. WEEN bit should be set to 0.

Bit Number	Bit Name	Read/Write/Clear	Description
7	LEEN	RW	Ends transaction when the <code>length</code> register reaches zero. When this bit is 0, <code>length</code> reaching 0 does not cause a transaction to end. In this case, the DMA transaction must be terminated by an end-of-packet signal from either the read or write master port.
8	RCON	RW	Reads from a constant address. When <code>RCON</code> is 0, the read address increments after every data transfer. This is the mechanism for the DMA controller to read a range of memory addresses. When <code>RCON</code> is 1, the read address does not increment. This is the mechanism for the DMA controller to read from a peripheral at a constant memory address. For details, see the <b>Addressing and Address Incrementing</b> section.
9	WCON	RW	Writes to a constant address. Similar to the <code>RCON</code> bit, when <code>WCON</code> is 0 the write address increments after every data transfer; when <code>WCON</code> is 1 the write address does not increment. For details, see <b>Addressing and Address Incrementing</b> .
10	DOUBLEWORD	RW	Specifies doubleword transfers.
11	QUADWORD	RW	Specifies quadword transfers.
12	SOFTWARERESET	RW	Software can reset the DMA engine by writing this bit to 1 twice. Upon the second write of 1 to the <code>SOFTWARERESET</code> bit, the DMA control is reset identically to a system reset. The logic which sequences the software reset process then resets itself automatically.

#### ❖ Thanh ghi “status”

Bit Number	Bit Name	Read/Write/Clear	Description
0	DONE	R/C	A DMA transaction is complete. The <code>DONE</code> bit is set to 1 when an end of packet condition is detected or the specified transaction length is completed. Write zero to the <code>status</code> register to clear the <code>DONE</code> bit.
1	BUSY	R	The <code>BUSY</code> bit is 1 when a DMA transaction is in progress.
2	REOP	R	The <code>REOP</code> bit is 1 when a transaction is completed due to an end-of-packet event on the read side.
3	WEOP	R	The <code>WEOP</code> bit is 1 when a transaction is completed due to an end of packet event on the write side.
4	LEN	R	The <code>LEN</code> bit is set to 1 when the length register decrements to zero.

#### ❖ Thanh ghi “readaddress”.

Chỉ ra địa chỉ đọc đầu tiên của quá trình truyền nhận dữ liệu. Giá trị “readaddress” phải sắp hàng theo độ rộng dữ liệu được cấu hình ở thanh ghi “control” (bit 0, 1, 2, 10, 11).

❖ Thanh ghi “writeaddress”

Chỉ ra địa chỉ ghi đầu tiên của quá trình truyền nhận dữ liệu. Giá trị “writeaddress” phải sắp hàng theo độ rộng dữ liệu được cấu hình ở thanh ghi “control” (bit 0, 1, 2, 10, 11).

❖ Thanh ghi “length”.

Chỉ ra số lượng byte cần truyền nhận. Giá trị “length” phải là bội số của dữ liệu với độ rộng được cấu hình ở thanh ghi “control” (bit 0, 1, 2, 10, 11).

### **2.1.3. Hoạt động của DMA**

Khi bắt đầu hoạt động, bộ DMA sẽ đọc dữ liệu tại địa chỉ trong thanh ghi “readaddress”, và ghi giá trị đọc được vào địa chỉ trong thanh ghi “writeaddress”. Độ rộng dữ liệu của quá trình đọc ghi được cấu hình trong thanh ghi “control” ở các bit 0, 1, 2, 10, và 11. Quá trình DMA kết thúc khi số lượng byte đã được đọc ghi bằng với số lượng byte được cấu hình trong thanh ghi “length”, lúc này:

- Bit “LEN” và “DONE” của thanh ghi “status” sẽ bằng 1.
- Ngắt sẽ được sinh ra nếu bit “I\_EN” của thanh ghi “control” bằng 1.

Trong hàm xử lý ngắt DMA, xóa ngắt bằng cách xóa bit “LEN” và “DONE” của thanh ghi “status”.

## **2.2. Hệ thống phần cứng**

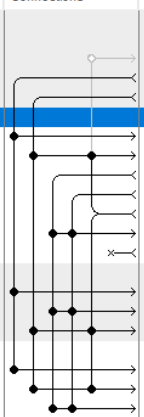
### **2.2.1. Tạo project Quartus**

Tạo project Quartus tên là “lab6”. Lưu ý đường dẫn thư mục project không được có khoảng trắng.

Với board DE2, chọn Family là Cyclone II, device là EP2C35F672C6.

### **2.2.2. Xây dựng hệ thống Qsys**

Xây dựng hệ thống phần cứng như hình bên dưới.

System Contents	Address Map	Clock Settings	Project Settings	Instance Parameters	System Inspector	HDL Example	Generation
Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<div>clk_0</div> <div>clk_in</div> <div>clk_in_reset</div> <div>clk</div> <div>clk_reset</div>	<div>Clock Source</div> <div>Clock Input</div> <div>Reset Input</div> <div>Clock Output</div> <div>Reset Output</div>	<div>clk</div> <div>reset</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0		
<input checked="" type="checkbox"/>		<div>nios2_qsys_0</div> <div>clk</div> <div>reset_n</div> <div>data_master</div> <div>instruction_master</div> <div>jtag_debug_module_re...</div> <div>jtag_debug_module</div> <div>custom_instruction_m...</div>	<div>Nios II Processor</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Master</div> <div>Avalon Memory Mapped Master</div> <div>Reset Output</div> <div>Avalon Memory Mapped Slave</div> <div>Custom Instruction Master</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0		
<input checked="" type="checkbox"/>		<div>onchip_memory2_0</div> <div>clk1</div> <div>s1</div> <div>reset1</div>	<div>On-Chip Memory (RAM or ROM)</div> <div>Clock Input</div> <div>Avalon Memory Mapped Slave</div> <div>Reset Input</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0	# 0x0001_0800	0x0001_0fff
<input checked="" type="checkbox"/>		<div>jtag_uart_0</div> <div>clk</div> <div>reset</div> <div>avalon_jtag_slave</div>	<div>JTAG UART</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Slave</div>	<div>Double-click to export</div> <div>Double-click to export</div> <div>Double-click to export</div>	clk_0	# 0x0001_1008	0x0001_100f

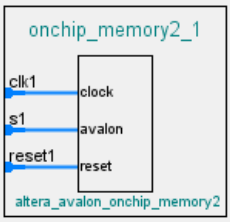
Thêm 1 bộ nhớ “onchip\_memory2\_1” thứ 2 vào hệ thống với cấu hình **128 Bytes** như bên dưới.

On-Chip Memory (RAM or ROM) - onchip\_memory2\_1

**On-Chip Memory (RAM or ROM)**  
altera\_avalon\_onchip\_memory2

**Block Diagram**

☐ Show signals



**Memory type**

Type: RAM (Writable) v

☐ Dual-port access

☐ Single clock operation

Read During Write Mode: DONT\_CARE v

Block type: AUTO v

**Size**

Data width: 32 v

Total memory size: 128 bytes

☐ Minimize memory block usage (may impact fmax)

**Read latency**

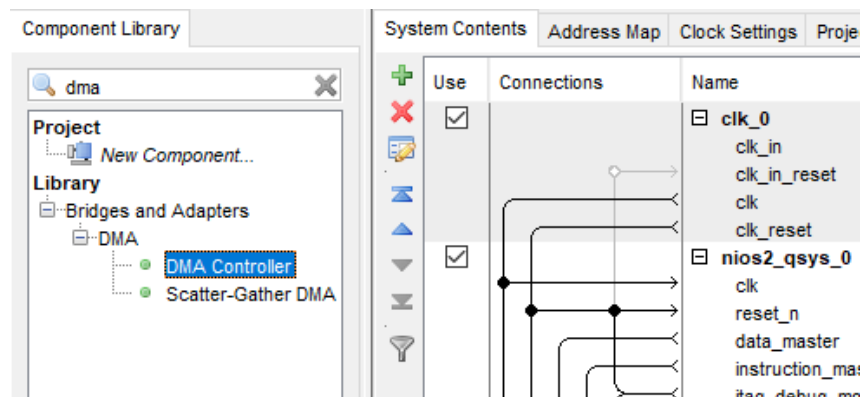
Slave s1 Latency: 1 v

Slave s2 Latency: 1 v

Kết nối các tín hiệu của bộ “onchip\_memory2\_1” vào hệ thống.

System Contents							
Address Map							
Clock Settings							
Project Settings							
Instance Parameters							
System Inspector							
HDL Example							
Generation							
Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		<b>clk_0</b> clk_in clk_in_reset clk clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to export Double-click to export	clk_0		
<input checked="" type="checkbox"/>		<b>nios2_qsys_0</b> clk reset_n data_master instruction_master jtag_debug_module_re... jtag_debug_module custom_instruction_m...	Nios II Processor Clock Input Reset Input Avalon Memory Mapped Master Avalon Memory Mapped Master Reset Output Avalon Memory Mapped Slave Custom Instruction Master	Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk] [clk] [clk] [clk] [clk]		IRQ 0 IRQ
<input checked="" type="checkbox"/>		<b>onchip_memory2_0</b> clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	Double-click to export Double-click to export Double-click to export	clk_0 [clk1] [clk1]	0x0001_0800 0x0000_8000	0x0001_0fff 0x0000_ffff
<input checked="" type="checkbox"/>		<b>jtag_uart_0</b> clk reset avalon_jtag_slave	JTAG UART Clock Input Reset Input Avalon Memory Mapped Slave	Double-click to export Double-click to export Double-click to export	clk_0 [clk] [clk]	0x0001_1008	0x0001_100f
<input checked="" type="checkbox"/>		<b>onchip_memory2_1</b> clk1 s1 reset1	On-Chip Memory (RAM or ROM) Clock Input Avalon Memory Mapped Slave Reset Input	Double-click to export Double-click to export Double-click to export	clk_0 [clk1] [clk1]	0x0000_0000	0x0000_007f

Trong cửa sổ thư viện, tìm bộ “DMA Controller” và thêm vào hệ thống.



Cấu hình bộ DMA với độ rộng **13 bits** như bên dưới, click “Finish”.

**DMA Controller**  
altera\_avalon\_dma

**Block Diagram**

☐ Show signals

**DMA Parameters** Advanced

**Transfer size**  
Width of the DMA length register (1-32) (bits):   
The maximum transaction size will be at least 8191.0 bytes.  
The maximum may be automatically increased to encompass the slave span.

**Burst transactions**  
☐ Enable burst transfers  
Maximum burst size (words):

**FIFO depth**  
Data transfer FIFO depth:   
Set the depth to at least twice the maximum read latency of the slave interface connected to the read master port of the DMA controller.  
A depth that is too low reduces transfer throughput.

**FIFO implementation**  
☐ Construct FIFO from registers  
Enable to construct FIFO from register, else default to using embedded memory blocks.

Kết nối các tín hiệu của bộ DMA vào hệ thống.

Lưu ý kết nối **tín hiệu ngắt** của DMA vào CPU NIOS II và chỉ kết nối tín hiệu **read\_master** và **write\_master** của DMA đến 2 bộ nhớ **onchip\_memory** để đọc data trực tiếp từ 2 bộ nhớ này

System Contents										
Address Map										
Clock Settings										
Project Settings										
Instance Parameters										
System Inspector										
HDL Example										
Generation										
Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Opcor	
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> clk_0	Clock Source	clk	clk_0					
		clk_in	Clock Input	clk						
		clk_in_reset	Reset Input	reset						
		clk	Clock Output							
		clk_reset	Reset Output							
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II Processor							
		clk	Clock Input		clk_0					
		reset_n	Reset Input		[clk]					
		data_master	Avalon Memory Mapped Master		[clk]			IRQ 0		
		instruction_master	Avalon Memory Mapped Master		[clk]			IRQ 31		
		jtag_debug_module_re...	Reset Output		[clk]					
		jtag_debug_module	Avalon Memory Mapped Slave		[clk]	0x0001_0800	0x0001_0fff			
		custom_instruction_m...	Custom Instruction Master		[clk]					
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)							
		clk1	Clock Input		clk_0					
		s1	Avalon Memory Mapped Slave		[clk1]	0x0000_8000	0x0000_ffff			
		reset1	Reset Input		[clk1]					
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART							
		clk	Clock Input		clk_0					
		reset	Reset Input		[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave		[clk]	0x0001_1148	0x0001_114f			
<input checked="" type="checkbox"/>		onchip_memory2_1	On-Chip Memory (RAM or ROM)							
		clk1	Clock Input		clk_0					
		s1	Avalon Memory Mapped Slave		[clk1]	0x0001_1080	0x0001_10ff			
		reset1	Reset Input		[clk1]					
<input checked="" type="checkbox"/>		dma_0	DMA Controller							
		clk	Clock Input		clk_0					
		reset	Reset Input		[clk]					
		control_port_slave	Avalon Memory Mapped Slave		[clk]	0x0001_1120	0x0001_113f			
		read_master	Avalon Memory Mapped Master		[clk]					
		write_master	Avalon Memory Mapped Master		[clk]					

Gán địa chỉ cho các module (System > Assign Base Addresses).

Gán độ ưu tiên ngắt cho các module (System > Assign Interrupt Number).

Hệ thống phân cứng đã hoàn thành, không còn thông báo lỗi. Save lại hệ thống với tên “nios\_sys”.

Chuyển sang tab “Generation”, click “Generate”.

### 2.2.3. Tích hợp hệ thống Qsys vào project Quatus

Thực hiện tương tự như bài thực hành trước.

Tạo file top module, đặt tên là “lab6.v” với nội dung như sau.



```

1  module lab6(CLOCK_50,KEY);
2
3  input CLOCK_50;
4  input [0:0] KEY;
5
6  nios_sys u0 (
7      .clk_clk      (CLOCK_50),
8      .reset_reset_n (KEY[0])
9  );
10
11 endmodule

```

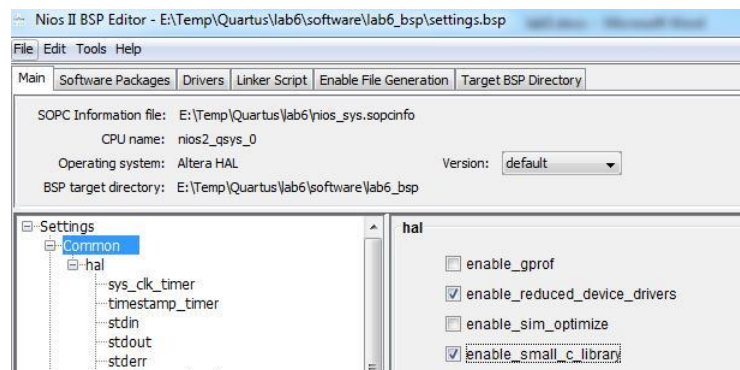
Build project Quartus và download hệ thống phần cứng xuống board.

### 2.3. Lập trình phần mềm

Tạo và đặt tên project là “lab6”.

Trong cửa sổ quản lý project, click chuột phải vào “lab6\_bsp”, chọn NIOS II > BSP Editor.

Tại tab “Main”, chọn “enable\_reduced\_device\_drivers” và “enable\_small\_c\_library” để giảm dung lượng thư viện chuẩn và driver. Click “Generate”, click “Exit”.



Trong thư mục “drivers/inc” của project “lab6\_bsp” có file “altera\_avalon\_dma\_regs.h”. Đây là file định nghĩa các thanh ghi DMA và các bit của các thanh ghi này. Chúng ta sẽ sử dụng file này để cấu hình DMA.

Code chương trình có nội dung như bên dưới.

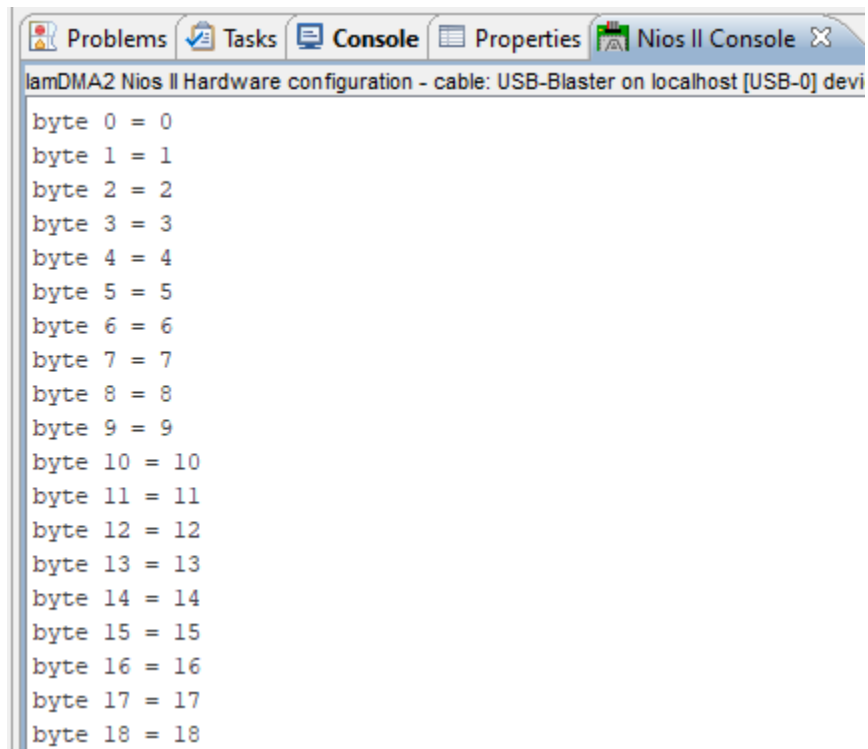


```

8#include <stdio.h>
9#include "system.h"
10#include "altera_avalon_dma_regs.h"
11#include "sys/alt_irq.h"
12
13// pdata0 points to a global array stored in onchip_memory2_0
14char pdata0[32] = {0,1,2,3,4,5,6,7,8,9,10,
15                  11,12,13,14,15,16,17,18,19,20,
16                  21,22,23,24,25,26,27,28,29,30,31};
17
18// pdata1 points to onchip_memory2_1
19char*pdata1 =(char*)(ONCHIP_MEMORY2_1_BASE);
20
21// Interrupt handler of DMA
22void DMA_ISR_Handler(void* isr_context){
23
24    int i;
25
26    // Read and print data in onchip_memory2_1
27    for(i=0;i<32;i++){
28        printf("byte %d = %d\n", i, pdata1[i]);
29    }
30
31    // Clear DMA interrupt bit
32    IOWR_ALTERA_AVALON_DMA_STATUS(DMA_0_BASE,0);
33 }
34
35// Initialize function of DMA
36void DMA_Init(void){
37
38    // De-init DMA
39    IOWR_ALTERA_AVALON_DMA_CONTROL (DMA_0_BASE,0);
40
41    // Source address is pdata0
42    IOWR_ALTERA_AVALON_DMA_RADDRESS(DMA_0_BASE,(int)pdata0);
43
44    // Destination address is pdata1
45    IOWR_ALTERA_AVALON_DMA_WADDRESS(DMA_0_BASE,(int)pdata1);
46
47    // Length is 32 bytes
48    IOWR_ALTERA_AVALON_DMA_LENGTH(DMA_0_BASE,32);
49
50    // Configure and Start DMA
51    IOWR_ALTERA_AVALON_DMA_CONTROL (DMA_0_BASE,ALTERA_AVALON_DMA_CONTROL_BYTE_MSK | // Byte transfer
52                                     ALTERA_AVALON_DMA_CONTROL_LEEN_MSK | // End transaction when length reach zero
53                                     ALTERA_AVALON_DMA_CONTROL_I_EN_MSK | // Interrupt enable
54                                     ALTERA_AVALON_DMA_CONTROL_GO_MSK); // Start DMA
55 }
56
57// entry point main()
58int main(void){
59
60    // Configure the DMA
61    DMA_Init();
62
63    // Register DMA's interrupt handler
64    alt_ic_isr_register(0, DMA_0_IRQ, DMA_ISR_Handler, (void*)0, (void*)0);
65
66    while(1);
67 }

```

Build và download phần mềm xuống board. Kiểm tra giá trị in ra màn hình console.



The screenshot shows a software window titled "Nios II Console" with a standard menu bar (Problems, Tasks, Console, Properties) and a title bar. The main content area displays the text "lamDMA2 Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] devi" followed by a list of 19 lines, each containing "byte" followed by an index and an equals sign and a value. The values range from 0 to 18, matching the indices. The text is rendered in a monospaced font.

```
byte 0 = 0
byte 1 = 1
byte 2 = 2
byte 3 = 3
byte 4 = 4
byte 5 = 5
byte 6 = 6
byte 7 = 7
byte 8 = 8
byte 9 = 9
byte 10 = 10
byte 11 = 11
byte 12 = 12
byte 13 = 13
byte 14 = 14
byte 15 = 15
byte 16 = 16
byte 17 = 17
byte 18 = 18
```

## **BÀI TẬP**

Bài 1:

Cấu hình bộ DMA để thực hiện “halfword transfer” và “word transfers” đọc 32 byte dữ liệu từ bộ nhớ “onchip\_memory2\_0” và ghi vào bộ nhớ “onchip\_memory2\_1”.