Expanding the FPSLIC I/O Area

Introduction

Atmel's AT94K FPSLIC™ integrates an FPGA with the AVR® 8-bit RISC processor. The communication between the AVR and the FPGA is using either 16 I/O locations or the Internal Dual-port RAM which is accessible both from the FPGA and the AVR. The 16 I/O locations are mapped into 4 windows of 4 I/O locations and only a single window is active at one time. This application note describes how to expand the number of I/O locations using a "linked" enable signal. All I/O registers are accessed using a linked list technique where the AVR first has to select the list, enabling the first entry. When an entry is accessed, the next entry in the list is automatically enabled afterwards. Thus a single chip select can be used to access a large number of I/O locations.

Description

The FPSLIC has five I/O locations which are used to access the I/O Area: the FISCR location is used to select the I/O window, and the 4 FISUx locations (x ∈ {A,B,C,D}) are used to generate 4 out of 16 chip selects (FIOSELn), see Table 1.

Table 1. FISUx Locations

FISCR	00	01	02	03
FISUA	FIOSEL0	FIOSEL1	FIOSEL2	FIOSEL3
FISUB	FIOSEL4	FIOSEL5	FIOSEL6	FIOSEL7
FISUC	FIOSEL8	FIOSEL9	FIOSEL10	FIOSEL11
FISUD	FIOSEL12	FIOSEL13	FIOSEL14	FIOSEL15

A peripheral using more than 4 I/O locations will suffer from an increased complexity in the driver if a direct mapping is used. In this mode, each I/O location is connected to one of the 16 chip selects. Since the I/O locations do not fit into a single window, the programmer has to maintain knowledge of the current value of the FISCR register to avoid writing to the wrong I/O window. When interrupts are introduced, the complexity level is even higher.

One way of handling the problem is to use a decoded internal address register. One I/O location is reserved for this purpose and whenever an internal register is to be accessed, its address must first be written to the address register. The drawback of this approach is that extra logic is needed and that two I/O cycles are always needed whenever a location is accessed, unless the same location is accessed twice. When interrupt routines are using the FPGA peripherals, the contents of the address register also need to be saved before use and restored afterwards. A software stack needs to be maintained, and the handling of the software stack must run with interrupts disabled, further adding to the overhead.

Programmable SLI

AT94K

Application Note

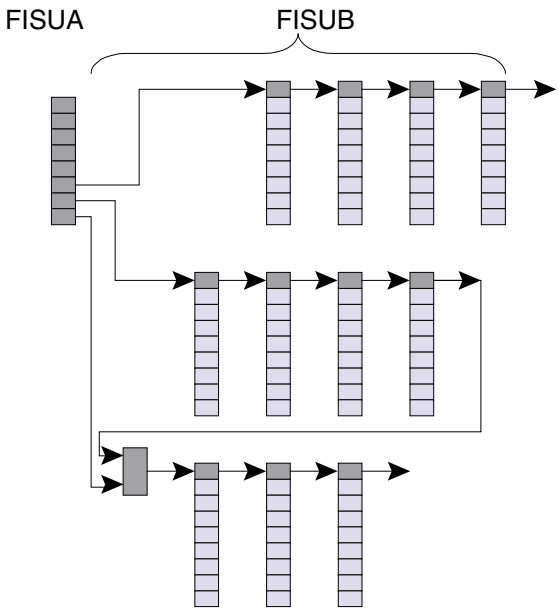
Rev. 3036A–FPSLI–03/02

The proposed approach implements the chip selects to each I/O location in multiple linked lists. Two I/O locations are again used, where the first is used to select which list, and the second is used to access the current entry in the list.

Thus no I/O location is accessible until a list has been selected. When the list has been selected, only the first entry in the list is immediately accessible. Accessing the current I/O location enables the next entry in the list, so the next access to the same I/O location will access the second entry in the list and so on.

Figure 1 shows how three linked lists can be accessed. The second list is a little bit special since it will select the third list automatically after the list has been exhausted. The approach is especially useful if multiple 16-, 24- or 32-bit registers are present in the FPGA.

Figure 1. Access to Three Linked Lists



The key addition to an I/O register is the “chip select out” signal. In the process block below, the register will be updated if both the “chip select in” and the write strobe are enabled. If the register is enabled, then the “chip select out” signal will be asserted at the same time.

```
if(RESET_Z = '0') then
      DATA(WIDTH-1 downto 0)  <= (others => '0');
      CSO<= '0';
elsif(RISING_EDGE(CLOCK)) then
      if(WRITE = '1') then
                if(CSI = '1') then
                          DATA(WIDTH-1 downto 0) <= DBUS(WIDTH-1 downto 0);
                          CSO<= '1';
                else
                          CSO<= '0';
                end if;
      end if;
end if;
```

Design Example

The enclosed VHDL® design example implements an interface to an external 16-bit memory bus with 24-bit addressing and two chip selects. This is an excellent opportunity to use the proposed approach.

The design contains three registers: a 24-bit address register using three 8-bit locations (A0,A1,A2), a 16-bit data register using two 8-bit locations (D0,D1), and an 8-bit control register using a single 8 bit location (C0).

The design requires the AVR to set up the registers and write another byte, starting a state machine which handles the memory interface timing.

Two lists are used, mainly for optimizing the driver for the read access. The first list contains all the registers in the following order (D0,D1,A0,A1,A2,C0) and is used to write data. The second list does not contain the data register, so the order becomes (A0,A1,A2,C0). Data is read using direct mapped chip selects.

The sequence to write data to the external memory becomes:

```
FISUA = DATA_CHAIN;              // Select the data register chain
FISUB = (data & 0xFF);           // Write lower byte to data holding register
FISUB = (data >> 8) & 0xFF;      // Write upper byte to data holding register
FISUB = (address & 0xFF);        // Write lower byte to address holding register
FISUB = (address >> 8) & 0xFF;   // Write mid byte to address holding register
FISUB = (address >> 16) & 0xFF; // Write upper byte to address holding register
FISUB = (WRL_Z | WRH_Z | CS1_Z);// Write to Control register
FISUB = (WRL_Z | WRH_Z | CS1_Z);// Start Write (data unimportant)
```

Thus only two I/O locations are used.

The sequence to read from the external memory does not use the data register chain, reducing the latency. The read state machine relies on the processor to generate wait states by inserting extra instructions between the start of the read cycle and the actual read. While these instructions write to FISUB, there is actually nothing reacting to the writing, and data is chosen to be the same as the previous write just to save code (it is already loaded to a register, so each C line maps to a single assembly line).

```
FISUA = ADDRESS_CHAIN;           // Select the address register chain
FISUB = (address >> 0) & 0xFF;
FISUB = (address >> 8) & 0xFF;
FISUB = (address >> 16)& 0xFF;
FISUB = (RD_Z | CS0_Z);          // Write to control register
FISUB = (RD_Z | CS0_Z);          // Start read  (data unimportant)
FISUB = (RD_Z | CS0_Z);          // wait        (data unimportant)
FISUB = (RD_Z | CS0_Z);          // wait        (data unimportant)
c = FISUC;                       // Read lower byte
return ((FISUD << 8) | c);       // Read upper byte and merge
```

The state machine controlling the read will maintain the read strobe and the chip select to the external memory active until it detects that the CPU has accessed FISUD.

The current read design uses all four locations (also FISUC and FISUD) to show that a combination of a linked list and normal is possible. It would be possible to have yet another list where the read target is changed.

```
FISUA = ADDRESS_CHAIN;          // Select the address register chain
FISUB = (address >> 0) & 0xFF;
FISUB = (address >> 8) & 0xFF;
FISUB = (address >> 16)& 0xFF;
FISUB = (RD_Z | CS0_Z);         // Write to control register
FISUB = (RD_Z | CS0_Z);         // Start read  (data unimportant)
FISUB = (RD_Z | CS0_Z);         // wait        (data unimportant)
FISUA = DATA_READ;              // wait        (data unimportant)
c = FISUB;                      // Read lower byte
return ((FISUB << 8) | c);      // Read upper byte and merge
```

This is not implemented in the VHDL code, but it does not present any difficulty.

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Microcontrollers
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Atmel Smart Card ICs
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Atmel Heilbronn
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

Atmel Programmable SLI Hotline
(408) 436-4119

Atmel Programmable SLI e-mail
fpslic@atmel.com

FAQ
Available on web site

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

© Atmel Corporation 2002.
Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

ATMEL® and AVR® are the registered trademarks of Atmel. FPSLIC™ is the trademark of Atmel.

VHDL® is the registered trademark of Cadence Design Systems, Inc. Other terms and product names may be the trademarks of others.

Printed on recycled paper.

3036A–FPSLI–03/02 xM