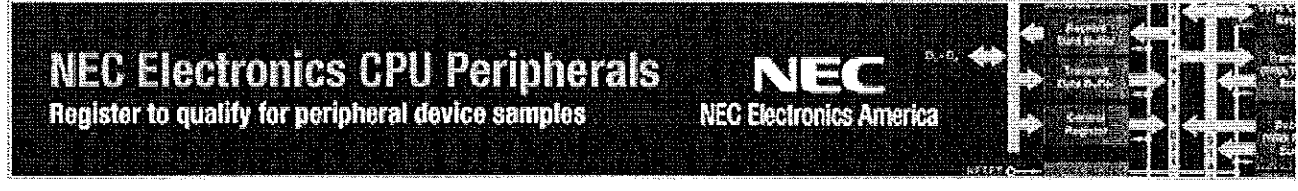




All Articles Products Cc

Site Search:

Welcome, Guest Home RSS Design Center Learning Center Product Center



Programmable Logic DesignLine > Design Center

How to interface FPGAs to microcontrollers

Neither standard product microcontrollers nor FPGAs were developed to communicate with each other efficiently, so interfacing the two can be a challenge.

By Rocendo Bracamontes Del Toro, Atmel

Page 1 of 3

Programmable Logic DesignLine
(07/30/2008 2:12 PM EDT)

As many as half of all embedded designs have an FPGA next to a microcontroller. The FPGA can be used to implement anything from glue logic, to custom IP, to accelerators for computationally intensive algorithms. By taking on some of the processing tasks, FPGAs help to improve system performance, thereby freeing up the MCU from cycle-intensive tasks. FPGAs also provide excellent performance characteristics and lots of flexibility to accommodate changing standards.

There are two basic implementations of MCU-plus-FPGA designs: putting a soft MCU core into the FPGA logic structure or using a standard product MCU with a discrete FPGA. Putting a soft core into the FPGA can be effective, but it can also be an expensive and power-hungry way to implement a microcontroller when compared to a standard product. This is especially true when using a 32-bit ARM-based core. As a result, only about one-third of FPGA-plus-MCU designs are implemented with an MCU core inside the FPGA logic. The remaining two-thirds consist of a standard product microcontroller next to a discrete FPGA.

Neither standard product microcontrollers nor FPGAs were developed to communicate with each other efficiently. They even use different languages. Thus, interfacing the two can be a challenge. FPGAs do not have any dedicated logic that communicates with microcontrollers. This logic module must be designed from scratch. Second, the communication between the microcontroller and FPGA is asynchronous. Special care is needed to resynchronize the MCU to the FPGA clock domain. Finally, there is an issue of bottlenecks, both at the interface and on the MCU bus. Transferring information between the MCU and the FPGA usually requires cycles on the MCU bus and

Rate this article

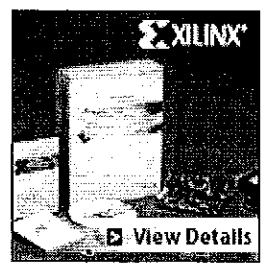
← WORSE | BETTER →
○ ○ ○ ○ ○
1 2 3 4 5

Submit

Print
Email
Reprints
SHARE

Related Companies

- ARM
- Atmel



Featured Jobs

Lowe's seeking Intel Engineer II in Mooresville, NC

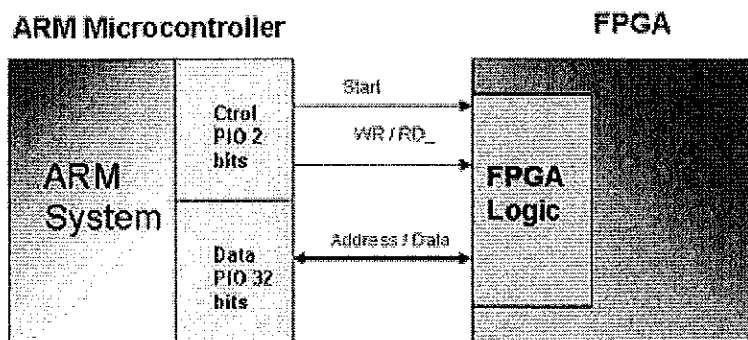
ON Semiconductor seeking Design Director in

usually ties up the resource (PIO or EBI) used to effect the transfer. Care must be taken to avoid bottlenecks with external SRAM or Flash and on the MCU bus.

There are basically three hardware options for interfacing the FPGA to the MCU: programmable I/O (PIO); external bus interface (EBI), if available; and, finally, a dedicated interface between built into the MCU between the advanced high-speed bus (AHB) and the FPGA. Which approach to use depends on the end application and the desired result.

PIO Interface

Interfacing the MCU to the FPGA via the PIO is relatively simple for simple data transfers, consisting of the transfer of 32 bits of address, 32 bits of data, and some control signals for control. This requires a 32-bit PIO and an additional 2-bits on another PIO (Fig 1).



1. PIO interface to FPGA.

(Click this image to view a larger, more detailed version)

For a transfer of data to the FPGA, the direction of the bidirectional buffers in the PIO must be set to output. The software algorithm that transfers data to the FPGA is as follows:

```
PIO_DATA = ADDRESS; // Pass the address to write
PIO_CTRL = START | WR; // Send start of address cycle
PIO_CTRL = CLEAR; // Clear PIO ctrl, this ends the address cycle
PIO_DATA = DATA; // Set data to transfer
PIO_CTRL = START; // Data is ready in PIO
PIO_CTRL = CLEAR; // This ends the data cycle
```

Reading from the FPGA is similar. Again, the direction of the buffer on the PIO must first be set to output and then change directions to input to read the data from the FPGA, the following code is executed:

```
PIO_DATA = ADDRESS; // Set the address to read
PIO_CTRL = START | RD; // Send start of address cycle
PIO_CTRL = CLEAR; // Clear PIO ctrl, this ends the address cycle
PIO_DATA_DIR = INPUT; // Set PIO-Data direction as input to receive the data
DELAY(WAIT_FOR_FPGA); // wait for the FPGA to send the data
DATA_FROM_FPGA = *PIO_DATA; // Read data from FPGA
```

The above algorithms are for a basic transfer, a more sophisticated algorithm is necessary to establish a proper communication between the ARM microcontroller and the FPGA. Special care is necessary to

Phoenix, AZ

1.

Northrop Grumman
seeking RF Systems
Engineer in Baltimore, MD

2.

3.

4.

Cirrus Logic seeking
Applications Engineer in
Austin, TX

in

5.

M

ITT Corporation seeking
Embedded Electronics
Engineer in Warminster,
PA

More jobs on
EETimesCareers

Sponsor



Actel IGLOO™ Flash FPGAs
SpeedWay Design Workshop
Join factory-certified field
application engineers from Avnet
Memec for a full-day hands-on
technical workshop devoted to
exploring low-power FPGA design
using Actel IGLOO FPGAs. Click
here for more info.

L

P

a

D

F

T

P

E

S

P

G

U

r

A

E

N

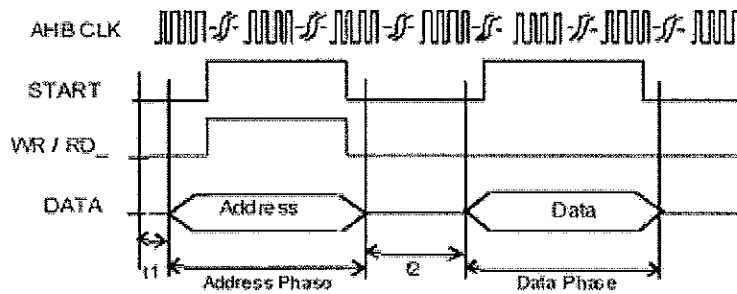
A

ensure the acknowledgment of data, e.g. no data has been lost due to speed or wait cycles on each side.

The access time is calculated as the sum of:

$$T_{\text{Access-PIO}} = t_1 + \text{address phase} + t_2 + \text{data phase}$$

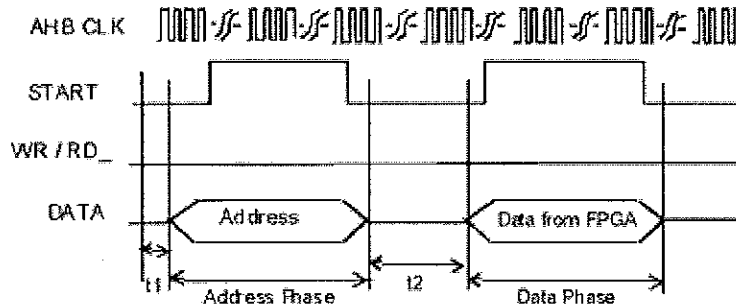
Using the GCC compiler with maximum optimizations, the system takes approximately 55 AHB cycles to perform the write operation to the FPGA (Fig 2).



2. PIO write to FPGA.

(Click this image to view a larger, more detailed version)

Assuming t_2 (wait for FPGA response ready) is also around 25 AHB cycles, the system takes approximately 85 AHB cycles for a read operation from FPGA (Fig 3).



3. PIO read from FPGA.

(Click this image to view a larger, more detailed version)

The interface from the MCU itself is fairly simple and straight forward. However, special logic must be implemented in the FPGA to decode all the traffic generated by the PIO. In the majority of cases, the traffic from the microcontroller is completely asynchronous. As a result, the FPGA must be able to oversample the control signals from the microcontroller; otherwise the FPGA will miss the time window and the traffic will not arrive at the final destination inside the FPGA.

Since the processor is the one in charge of keeping the PIO busy, there is an overhead of processing time. While the CPU is engaged in data transfers, it cannot do anything else. Thus, this solution has the potential to bog down system processing. DMA is not possible using a PIO interface, so the software programmer must limit data bandwidth to allow for other communication with the MCU. For example, if there is a routine that demands 100% of the processors cycles running

concurrently with a serial (SPI, USART or TWI) transfer to or from the FPGA, then one of these two processes must wait. If the data going to or coming from the FPGA is not buffered on time, it will probably be overrun by the next byte/word of data. In essence, the embedded processor becomes a glorified data mover.

Page 2: [next page](#)

[Page 1](#) | [2](#) | [3](#)

Please [login](#) or [register](#) here to post a comment



TechOnline Communities

[Audio DesignLine](#) | [Automotive DesignLine](#) | [CommsDesign](#) | [Digital Home DesignLine](#) | [DSP DesignLine](#) | [eProductCenter](#) | [Industrial Control DesignLine](#) | [Mobile Handset DesignLine](#) | [Planet Analog](#) | [Programmable Logic DesignLine](#) | [RF DesignLine](#) | [Teardown.com](#) | [TechOnline](#) | [Video/Imaging DesignLine](#)

EE Times

[United States](#) | [Asia](#) | [China](#) | [Europe](#) | [France](#) | [Germany](#) | [India](#) | [Japan](#) | [Korea](#) | [Taiwan](#)

Additional Network Sites

[Analog Europe](#) | [Automotive Designline Europe](#) | [DeepChip](#) | [Design & Reuse](#) | [Electronic Supply and Manufacturing](#) | [Industrial Control Europe](#) | [Microwave Engineering Europe](#) | [Portelligent](#) | [Power Management Designline Europe](#)

All materials on this site Copyright © 2008 TechInsights, a Division of United Business Media
[Privacy Statement](#) | [Your California Privacy Rights](#) | [Terms of Service](#)