Many microcontroller and microprocessor chips limit the available I/O ports and pins to conserve pin counts and reduce package sizes. In designs that use these limited chips, you can use the Altera devices to implement I/O expanders. Using this method, you can increase the number of I/O pins without changing the microcontrollers or microprocessors in the design.

The design example in this document shows the method to implement microcontroller I/O expanders using these Altera devices:

- MAX® 10 FPGA
- MAX V
- MAX II
- MAX 3000A

**Related Information**

- **Design Example for MAX II, MAX V, and MAX 3000A**
  Provides the design files for this application note (AN 265).
- **Design Example for MAX 10**
  Provides the MAX 10 design files for this application note (AN 265).

## Advantages of Using Altera Devices as I/O Expanders

**Table 1: Advantages of Using Altera Programmable Devices as I/O Expanders**

For more information about MultiVolt I/O support and the number of I/O pins available for each device package, refer to the related information.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmable logic</td>
<td>Together with the I/O expander, you can integrate other functions in the programmable device. For example, you can integrate LED drivers or bus-bridging logic. Furthermore, with the user flash memory (UFM) in the MAX II, MAX V, or MAX 10 devices, you can also integrate functions that use memory. This capability allows you to reduce component count across the board.</td>
</tr>
<tr>
<td>Feature</td>
<td>Advantage</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MultiVolt I/O support</td>
<td>The Altera programmable devices support MultiVolt I/O capability. You can use this capability to interface with I/O pins with voltages of 3.3 V, 3.0 V, 2.5 V, 1.8 V, 1.5 V, and 1.2 V. Using this capability, you can implement voltage-level shifting to interface your newer 2.5 V, 1.8 V, 1.5 V, or 1.2 V with older 3.3 V devices.</td>
</tr>
<tr>
<td>Large number of I/O pins</td>
<td>The large number of I/O pins available in the Altera programmable devices make them ideal for implementing microcontroller I/O expansion. For example, with 500 I/O pins, you can implement 62 units of 8 bit ports.</td>
</tr>
<tr>
<td>In-system programmability</td>
<td>With in-system programmability (ISP), you have the added flexibility of reprogrammability on the board.</td>
</tr>
</tbody>
</table>

Related Information

- **MAX 10 FPGA Device Overview**
  Provides the number of maximum GPIO resources available for each device in the MAX 10 FPGA device family.
- **MAX 10 General Purpose I/O User Guide**
  Lists the I/O standards voltage and pin support information for each I/O standard supported by the MAX 10 FPGA device family.
- **Introduction, MAX II Device Handbook**
  Provides the number of maximum user I/O pins for each device in the MAX II device family.
- **MAX II Architecture, MAX II Device Handbook**
  Lists the MultiVolt I/O support for MAX II devices.
- **MAX 3000A Programmable Logic Device Family Data Sheet**
  Provides the number of maximum user I/O pins for each device and lists the MultiVolt I/O support for MAX 3000A devices.
- **MAX V Device Family Overview**
  Provides the number of maximum GPIO resources available for each device in the MAX V device family.
- **MAX V Architecture**
  Lists the MultiVolt I/O support for MAX V devices.

## Microcontroller I/O Expander Implementation

The I/O expander in this design example includes four ports—A, B, C, and D:

- You can program each port independently as I/O ports.
- Each port is 8 bits wide.
- Each port is bidirectional and acts as input port if tri-stated.
- To perform read and write operations, connect any of these ports with the microcontroller unit (MCU) port.
Pin Functions in the Microcontroller I/O Expander

Table 2: Input and Output Pins in Microcontroller I/O Expander Design Example

<table>
<thead>
<tr>
<th>I/O Name</th>
<th>Type</th>
<th>Description</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Input</td>
<td>Input clock signal to trigger the input data registers, output data registers, and I/O setting data registers.</td>
<td>Positive Edge</td>
</tr>
<tr>
<td>I/O Name</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| MCU port | Bidirectional | 8 bit bidirectional data lines that interface with the MCU data bus and transmit configuration data.  
- To write or read the 8-bit data to or from the I/O expander, use WR or RD signals with positive edge trigger.  
- To set the MCU port as data or address bus, use the MODE signal. |
| CS       | Input      | Input signal to select the device.                                         |
| RST      | Input      | Input signal to reset the device and all internal registers asynchronously. |
| CONF     | Input      | • Stores the address[5..0] from the MCU port on the rising edge of CLK into the configuration registers.  
- Enables the clock signal to the configuration registers.  
- Ignores data on address[7..6]. |
| LOAD     | Input      | • Stores the address[1..0] from the MCU port on the rising edge of CLK into the configuration registers.  
- Enables the clock signal to the configuration registers.  
- Ignores data on address[7..2]. |
| WR       | Input      | • Causes the data on the MCU port to be written to the enabled port (A, B, C, or D) on the rising edge of CLK.  
- Enables the clock signal to the output data registers |
| RD       | Input      | • Enables the data bus transfer to the MCU port from the enabled port (A, B, C, or D) on the rising edge of CLK.  
- Enables the clock signal to the input data registers. |
| MODE     | Input      | • When high, tri-states the MCU port.  
- When low, sets the MCU port as input or output based on the configuration registers. |
| Port A   | Bidirectional | 8-bit, general-purpose I/O port that can be programmed to either output or input mode. |
## Functional Description of the Microcontroller I/O Expander

The I/O expander operates as a slave that sends and receives data from and to the microcontroller through the MCU port.

### Table 3: Operation Modes of the Microcontroller I/O Expander

During read and write operations, set the `MODE` signal control pin to low.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>In read mode, the microcontroller receives the incoming data through the I/O expander.</td>
</tr>
<tr>
<td></td>
<td>• The selected port (A, B, C, or D) becomes an input port and the MCU port becomes an output port.</td>
</tr>
<tr>
<td></td>
<td>• When the RD and CS signals are low, the application enables the clock through the input data registers. On the rising edge of CLK, the I/O expander writes the data from the selected port to the input data registers.</td>
</tr>
<tr>
<td>Write</td>
<td>In write mode, the microcontroller transmits data out through the I/O expander.</td>
</tr>
<tr>
<td></td>
<td>• The selected port (A, B, C, or D) becomes an output port and the MCU port becomes an input port.</td>
</tr>
<tr>
<td></td>
<td>• When the WR and signals are low, the application enables the clock through the output registers. On the rising edge of the CLK signal, the I/O expander writes the data received at the MCU port to the output data register of the selected port.</td>
</tr>
<tr>
<td>Disabled</td>
<td>In this mode, the microcontroller cannot receive or transmit data through the I/O expander. When WR or RD signal is high, the application disables the CLK through the I/O data registers.</td>
</tr>
</tbody>
</table>
Microcontroller I/O Expander Configuration

To set the port as input, output, or disabled, you must configure the I/O expander.

The I/O expander configuration data is a set of six bit registers:

- Four bits of I/O ports configuration (bits 5, 4, 3, and 2)
- Two bits of enabled port address (bits 1 and 0)

### Table 4: I/O Expander Address Bus Configuration Function

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit</th>
<th>Function</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A7:A6</td>
<td>7–6</td>
<td>No function</td>
<td>—</td>
</tr>
</tbody>
</table>
| A5      | 5     | Configuration bit for Port D | • 0 = configured as input  
|         |       |                         | • 1 = configured as output   |
| A4      | 4     | Configuration bit for Port C | • 0 = configured as input  
|         |       |                         | • 1 = configured as output   |
| A3      | 3     | Configuration bit for Port B | • 0 = configured as input  
|         |       |                         | • 1 = configured as output   |
| A2      | 2     | Configuration bit for Port A | • 0 = configured as input  
|         |       |                         | • 1 = configured as output   |
| A1:A0   | 1–0   | Port address            | • 00 = enables port A          
|         |       |                         | • 01 = enables port B          
|         |       |                         | • 10 = enables port C          
|         |       |                         | • 11 = enables port D          |

When the MODE signal is high and CS signal is low, the design places the configuration data on the input address bus (MCU port).

To activate the new configuration, send a single pulse of the CONF signal for one clock period. When the CONF signal and CS signals are low, the clock is enabled through the configuration registers. As a result, the I/O expander writes the configuration data in the configuration registers. This transition sets selected configuration as the state of the I/O expander.

If you want to select which of the four ports (A, B, C, or D) to use without configuring the ports, assert the LOAD signal low while CS signal is low. At the rising edge of the CLK signal, only the enabled port address data bits are written to the configuration registers.

The configuration of the selected port (input or output) also configures the MCU port direction. For example, if you configure port A as input and enable it, the direction of the MCU port becomes output.
You cannot directly configure the input or output direction of the MCU port. However, if you need to access the configuration registers, use the \texttt{MODE} signal control pin to tri-state the MCU port and turn it into an input port.

The I/O expander design supports reset mode. When you assert the \texttt{RST} signal while the \texttt{CS} signal is low, all register contents are reset to zero asynchronously. All I/O expander including the MCU port are set to input mode, which is the initial state of the I/O expander.

**Related Information**

*Pin Functions in the Microcontroller I/O Expander* on page 3

---

**Resource Usage in Microcontroller I/O Expander Target Devices**

There are sufficient resources in the target devices to implement the design example and other logic.

**Table 5: Device Targets to Implement the Microcontroller I/O Expander Design Example**

This table lists the devices that you can target with the design example using the Quartus® II software and their resource utilization.

<table>
<thead>
<tr>
<th>Family</th>
<th>Device</th>
<th>Resource</th>
<th>Availability</th>
<th>Usage</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX 3000A</td>
<td>EPM3128ATC100</td>
<td>Macrocells</td>
<td>128</td>
<td>51</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flipflops</td>
<td>128</td>
<td>50</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O pins</td>
<td>80</td>
<td>48</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shareable expanders</td>
<td>128</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MAX II</td>
<td>EPM240T100C3</td>
<td>Logic elements</td>
<td>240</td>
<td>67</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flipflops</td>
<td>240</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O pins</td>
<td>80</td>
<td>48</td>
<td>60</td>
</tr>
<tr>
<td>MAX 10 FPGA</td>
<td>10M08SAE144C8G</td>
<td>Logic elements</td>
<td>8064</td>
<td>75</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flipflops</td>
<td>8064</td>
<td>59</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O pins</td>
<td>103</td>
<td>48</td>
<td>47</td>
</tr>
</tbody>
</table>

**Microcontroller I/O Expander Design Verification**

You can verify the functional and timing simulation of the example design on the target devices using the Quartus II software.

**Related Information**

*Functional Description of the Microcontroller I/O Expander* on page 5
Simulating Microcontroller I/O Expander Write Mode Timing

Figure 2: Write Mode Timing Simulation of the Microcontroller I/O Expander

1. With all the I/O pins in tri-state mode, initialize each port.
2. After initialization, configure port A and port B as output, and port C and port D as input.
3. Enable port B for write operation.
   When $WR$ asserts a low signal, the clock through the output data registers is enabled and creates a new output value (146) at the output port B from the MCU port.
4. Once the $LOAD$ signal asserts low, enable port A.
   When the $WR$ signal is low to enable the clock through the output data registers, the MCU port data (108) is created at the output port A.
Simulating Microcontroller I/O Expander Read Mode Timing

Figure 3: Read Mode Timing Simulation of the Microcontroller I/O Expander

1. With all the I/O pins in tri-state mode, initialize each port.
3. Enable port C for read operation.
   When \textit{RD} asserts a low signal, the clock through the input data registers is enabled and creates a new output value (214) at the MCU port from port C.
4. Once the \textit{LOAD} signal asserts low, enable port D.
   When the \textit{RD} signal is low to enable the clock through the input data registers, port D data (146) is created at the MCU port.

Microcontroller I/O Expander Application

You can modify the example design to target any 8-bit microcontroller such as the PIC and 8051 microcontrollers.
This figure shows the I/O expander applied to a microcontroller system to increase the number of I/O ports of the microcontroller.
Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| September 2014 | 2014.09.22 | • Added information for MAX 10 devices.  
• Added support for MAX V devices.  
• Added missing link to download the design example file for MAX V, MAX II, and MAX 3000A device families.  
• Added link to download the design example file for MAX 10 devices.  
• Removed tables that list the MultiVolt I/O support and number of maximum user I/O pins for the MAX II and MAX 3000A devices, and added links to related device-specific documents that list the information. The linked documents provide more details and contain the latest information for each device family.  
• Changed the figure that shows the I/O expander address bus configuration into a table for easier reference.  
• Removed the table that lists the port addresses configuration values and combined the information in the table that lists the I/O expander address bus configuration details.  
• Added a table that lists the design example target devices and combined the resource utilization information into this table for easy reference.  
• Updated the microcontroller I/O expander read and write timing simulation figures.  
• Restructured and rewrote all sections for clarity and style update.  
• Updated template. |
| March 2004   | 2.0     | Revision history information not available.                          |
| —           | 1.0     | Initial release.                                                      |