Six Ways to Replace a Microcontroller With a CPLD

With the advent of low-power CPLDs, low-power electronic product designers now have new options for implementing many of the functions traditionally performed by microcontrollers. This white paper discusses when it is advantageous to use a CPLD instead of a microcontroller, and when it makes sense to use a CPLD as a companion to a microcontroller.

Introduction

Tell a group of portable electronics designers that there is a low-power digital device that allows them to use a software program to reconfigure the operation of the hardware, and nine out of ten will assume that it is some form of a microcontroller. This is understandable. With a vast array of features and packages, abundance of software development tools, and huge library of application code, the ubiquitous microcontroller has found a home in nearly every type of portable application. However, with the arrival of low-power CPLDs, designers now have new options for implementing many of the functions traditionally performed by microcontrollers.

This white paper discusses when it is advantageous to use a CPLD instead of a microcontroller, and when a CPLD makes a good companion to a microcontroller. The examples given in this white paper are grouped into three categories, based on their function and level of complexity. The first category is I/O management, focusing on pin-level applications. The second category is port management, with the emphasis on the various interfaces between devices. The third category is system management, for applications that use a pin or port to control system-level functions.

Designers who are new to programmable logic will find many aspects of CPLD design to be similar to designing with traditional microcontrollers. A simplified description of CPLD design flow is as follows:

1. The design is written in a high-level language, such as Verilog or VHDL, using a software development tool.

2. The design is simulated for functional correctness.

3. The design is "fit" to a particular CPLD, providing the physical aspects such as resource utilization and timing paths.

4. The design is simulated for timing correctness.

5. The design is programmed into the physical device.

One major difference is the availability of sophisticated in-circuit emulators for microcontroller check out. However, once the nuances of the programmable technology are understood, microcontroller designers do well with CPLD design.

Examples of CPLDs Replacing Microcontrollers

The following section gives some of the many applications where a CPLD can cost-effectively replace a microcontroller.

I/O Management

When considering whether to use a CPLD or a microcontroller for I/O management, the quantity and the type of I/Os needed are two critical considerations. Microcontrollers enjoy a reputation as small and inexpensive, and certainly, there are many small, inexpensive microcontrollers from which a designer can choose. However, if an application requires a large quantity of general-purpose I/Os, often a CPLD can be cost-competitive with a microcontroller. Small, inexpensive microcontrollers are usually limited to a serial port and, at most, a few general-purpose I/O pins.

WP-01041-1.1

Designers discover that microcontrollers with dozens of I/Os are no longer so small and inexpensive. CPLDs, on the other hand, tend to be I/O intensive; it is not uncommon for a small form factor CPLD to have well over 50 I/Os. For example, the Altera® MAX® IIZ EPM240Z CPLD in a 5-mm-by-5-mm package has 80 I/Os. In addition to enjoying an I/O quantity advantage, in general, CPLDs are more flexible than microcontrollers. With some exceptions, the majority of CPLD I/Os can be used for any purpose.

Programmable Level Shifting

Many products require the use of logic devices of varying voltages. To support multi-voltage applications, designers frequently need to connect devices of differing voltage levels. This is rarely possible with a microcontroller because they have a limited number of I/O resources, which often operate from one voltage source. In contrast, CPLDs have a larger number of I/Os, which are grouped into multiple banks. Each I/O bank, in turn, is assigned a unique voltage source. Thus, creating a voltage-level shifter is merely a matter of grouping all the I/Os of one voltage in one bank and connecting the associated voltage reference to the power rail needed for those I/Os (Figure 1). While it is useful to be able to accomplish level shifting using a CPLD, an even greater advantage is derived from the power of programmability combined with the level shifting. For example, suppose an application calls for a LCD display that is not supported by the host processor and is not at the same voltage level. A CPLD could be used to provide voltage-level shifted timing control between the host processor and the LCD display.

Figure 1. Using an MAX IIZ CPLD to Perform Voltage-Level Shifting
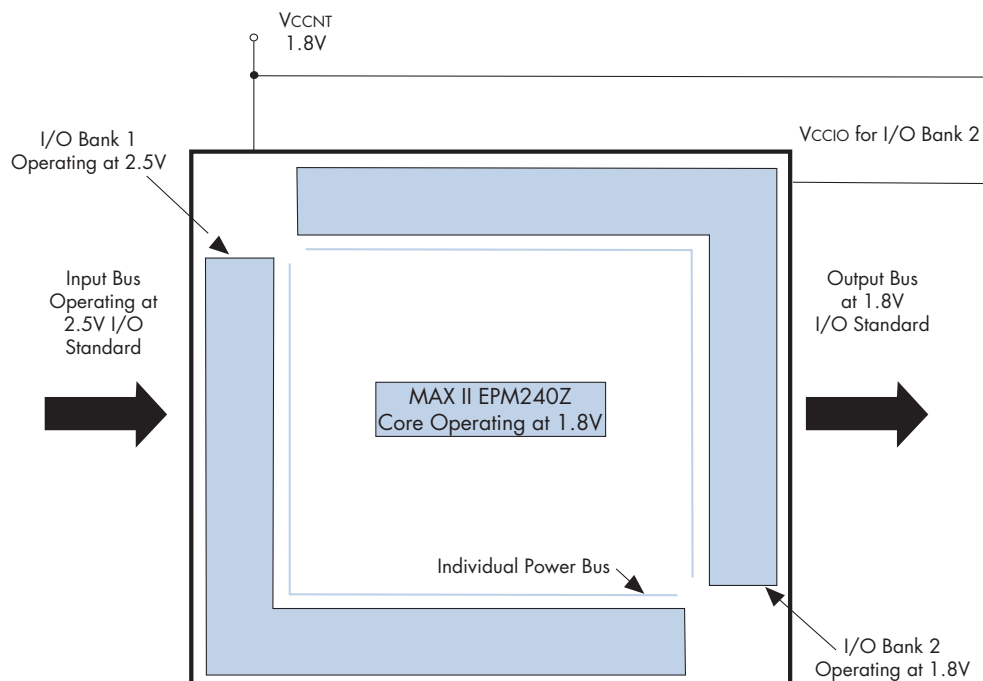
Pulse Width Modulation

Often, a designer chooses a microcontroller for a function, such as pulse width modulation (PWM), that can be accomplished using a CPLD. In PWM, the time period of the square wave is kept constant, while the time for which the signal remains high is varied or modulated. Thus, the duty cycle (t_{ON}) of the signal can be varied. PWM provides a powerful method for controlling analog circuits in a digital system. One method that is common in portable applications is using PWM to vary the intensity of an LED.

CPLDs do not have dedicated PWM circuitry, yet it is not difficult to implement a PWM output. For example, the MAX IIZ CPLD contains an internal oscillator that can be used as a source for the frequency, and counters can then be used to modulate this frequency.

Analog-to-Digital Converter

Designers often choose a microcontroller for its analog-to-digital converter (ADC) capabilities. However, in certain cases, such as keypad decoding, an ADC may not be necessary.

Figure 2 shows a basic switch array and an ADC. A set of resistors are connected in series between V_{CC} and GND, and a switch is connected to each resistor tap and a common node. If a switch is activated, the circuit generates a voltage proportional to the switch location in the resistor stack. To be used in a digital system, this analog signal must be converted to digital value, and often a microcontroller is chosen because it has a built-in ADC.

Figure 2. Analog Key Pad Array
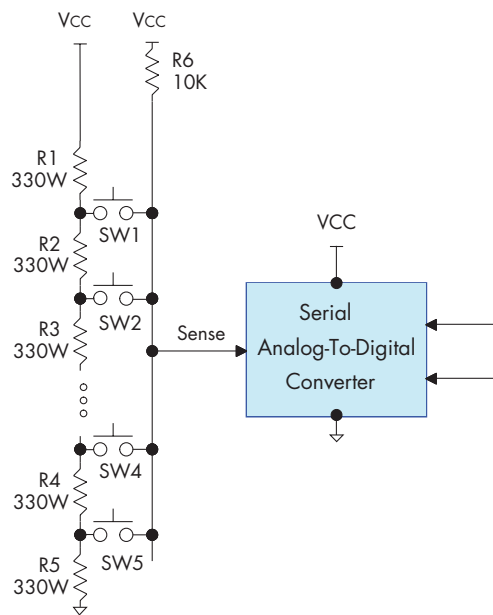
However, a CPLD is also an option. With the addition of a simple, low-cost external capacitor, the MAX IIZ CPLD can use its internal oscillator, Schmitt trigger I/O, and high-density arithmetic-programmable logic fabric to accomplish the analog-to-digital conversion.(1)

Power Sequencing

MAX IIZ devices are optimized for many system management functions such as power sequencing, which includes multi-voltage system power up and system reset, as well as chip-select generation. These two applications are often integrated into a single non-volatile, instant-on device. Multi-voltage system power sequencing requires a device to be instantly on and ready to manage the power-up sequencing of other devices on the PCB. Therefore a CPLD, which powers up within microseconds, is a better choice for power sequencing than a microcontroller, which often needs milliseconds to power up.

Figure 3 illustrates a typical MAX IIZ device power-sequencing application. As board density and the number of power planes on a board increases, the complexity of the power sequencing also increases. MAX IIZ CPLDs can easily manage the power sequencing for all levels of system complexity. Multiple power rails support different devices, and control logic is needed to manage the complete power-up sequence of each device. To ensure that accidental driving of these signals does not occur during power up, the MAX IIZ device is also used to control critical bus signals until the power up is complete. The JTAG port monitors the power sequence, storing errors and information upon power up. It can also be used to set break points in the power sequencing, which is useful during the debug phase.

Figure 3. Power Sequencing Using a CPLD
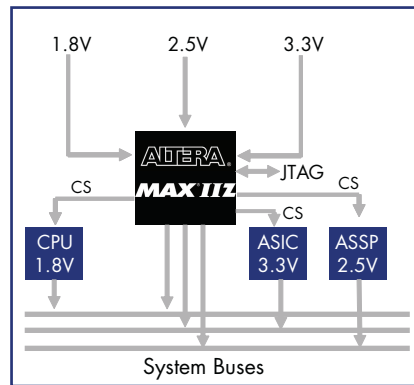
Watchdog Timer

Many system management applications require some form of timer. Designers may be surprised to discover that a CPLD can be used for many of the timer functions that are usually associated with microcontrollers. With a few discrete capacitors, resistors, diodes, and metal-oxide-semiconductor field-effect transistors (MOSFETs), a simple but powerful resistor-capacitor (RC) timer-based circuit can be built that powers up the CPLD at regular intervals. In the example circuit in Figure 4, the RC values are selected to create a 10-second timer. This basic timer can be extended by three external capacitors (C1, C2, and C3), which are used to create a simple non-volatile binary counter. Thus, an interval period from 10 seconds to 80 seconds can be fully implemented in a MAX IIZ EPM240Z CPLD, utilizing 19 percent of its logic.*(2)*

*Figure 4. Building a Timer-Based Power-Up Circuit for a MAX IIZ CPLD*
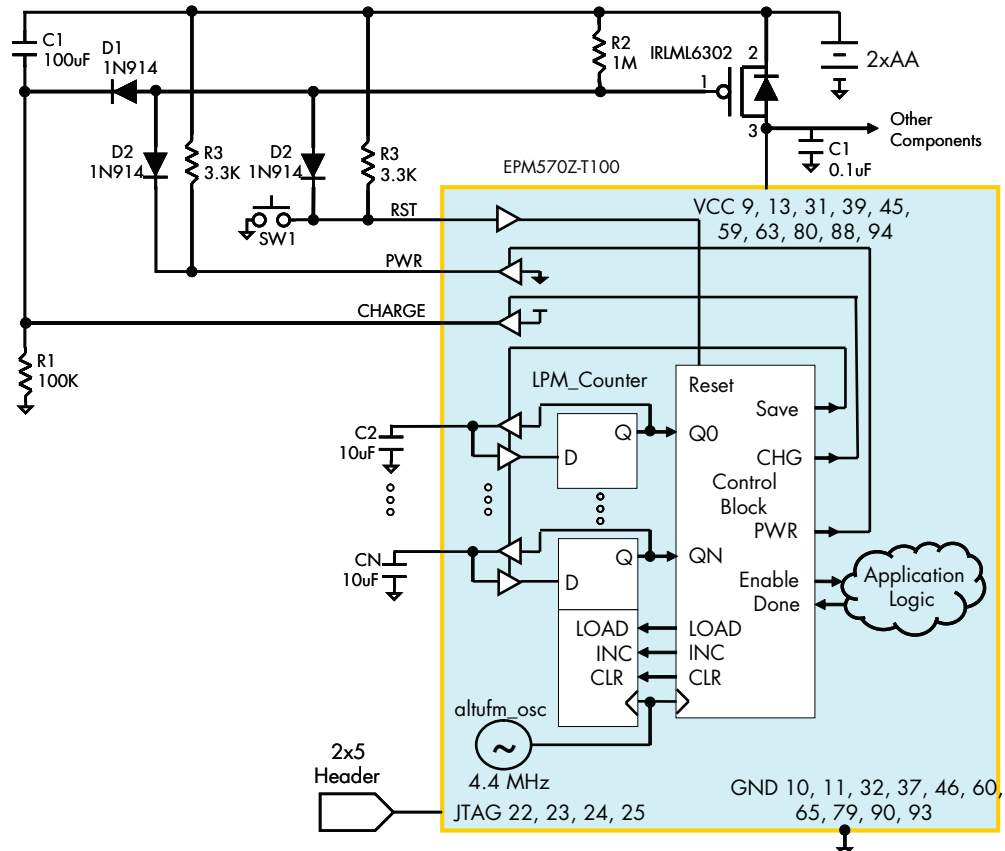
Examples of CPLD and Microcontroller Companionship

A CPLD does not always compete with a microcontroller. The following are several cases where the CPLD makes an excellent companion to a microcontroller.

GPIO Pin Expansion

In a common application known as general-purpose I/O pin expansion, many designers combine the programming capabilities of a small, inexpensive microcontroller with the general-purpose I/O resources of a CPLD. The CPLD builds a set of internal registers that can be accessed by the microcontroller through any available serial port, such as I²C or SPI (Figure 5). This allows the microcontroller to use existing I/O resources to expand its total I/O count. With this expanded I/O count, designers can use the CPLD for voltage-level shifting, which increases the utility of the CPLD. *(3)*

Figure 5. GPIO Pin Expansion
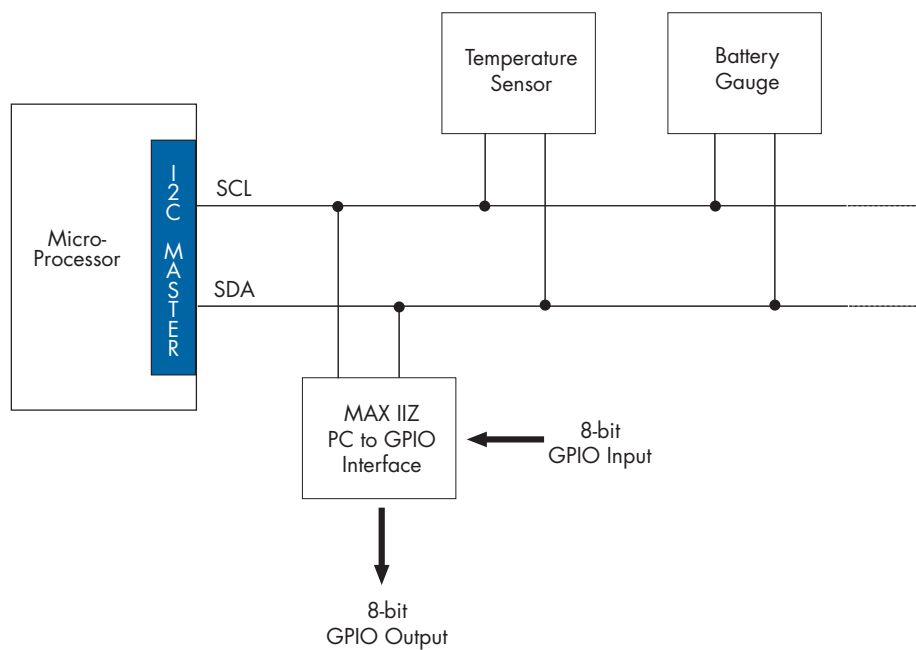
Port Management

Portable application designers often find a need to connect devices with differing I/O interfaces. This function is often referred to as bridging because the CPLD is used to form a bridge between the dissimilar interfaces. This section describes three such cases:

- Serial to serial - I²C to SPI
- Serial to parallel - SPI devices (serial) to a host processor (parallel)
- Parallel to parallel - host processor to CF+

In each of these examples, there are a few reasons why a CPLD might be a better choice than a microcontroller. For one, there is often a need for more I/Os than is cost-effectively available in a microcontroller. A microcode implementation may not be able to meet the necessary performance of the interface. In addition, the implementation may be far more burdensome to implement in microcode than in a CPLD's hardware.

Serial-to-Serial Conversion

Figure 6 illustrates how a CPLD can be used to bridge between two differing serial interfaces: I²C and SPI. This design can be implemented in a MAX IIZ EPM240Z CPLD, using about 43 percent of the available logic and six I/O pins. *(4)*

Figure 6. I²C to SPI Interface Using a MAX IIZ CPLD
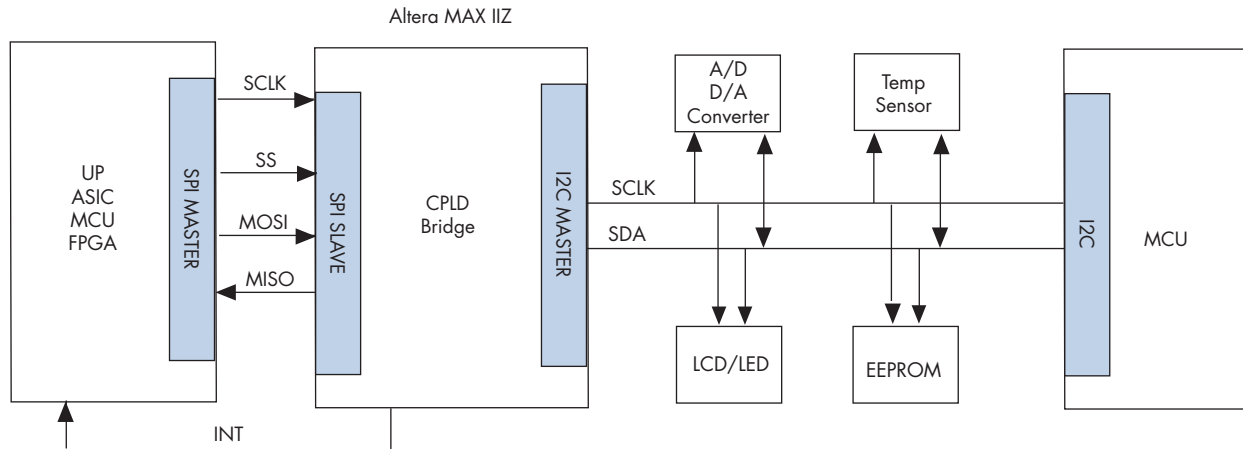
Serial-to-Parallel Conversion

Figure 7 shows a host processor interfaced to a SPI master, using a CPLD to implement this serial-to-parallel interface. This example creates a host-processor bus interface and a complete SPI master, and can be implemented in a MAX IIZ EPM240Z CPLD using about 30 percent of the available logic and 25 I/O pins. *(5)*

Figure 7. Host Processor to SPI Interface Using a MAX IIZ CPLD
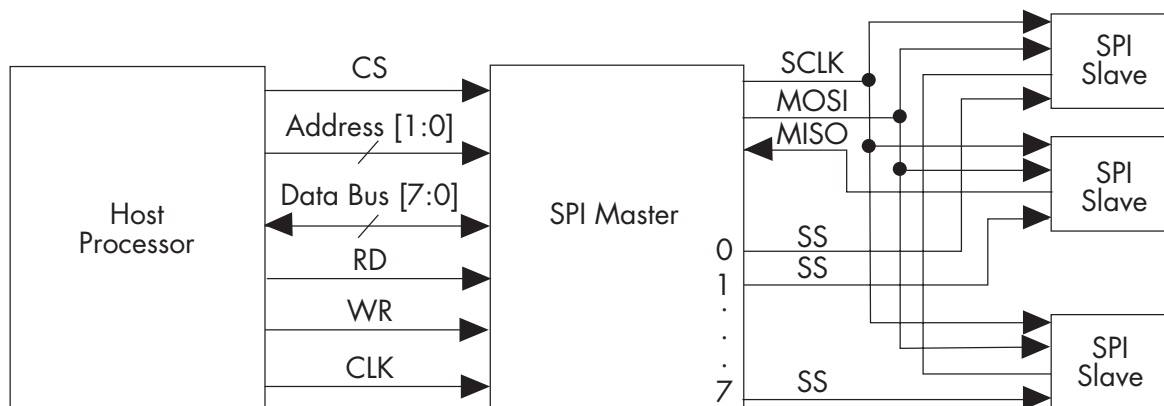
Parallel-to-Parallel Conversion

In Figure 8, a CPLD is used to bridge between two different parallel interfaces. This example implements a host processor bus interface to a Compact FLASH+ device. It can be implemented in a MAX II EPM240Z CPLD, using about 54 percent of the available logic and 45 I/O pins. *(6)*

Figure 8. Host Processor to CF+ Interface Using a MAX IIZ CPLD
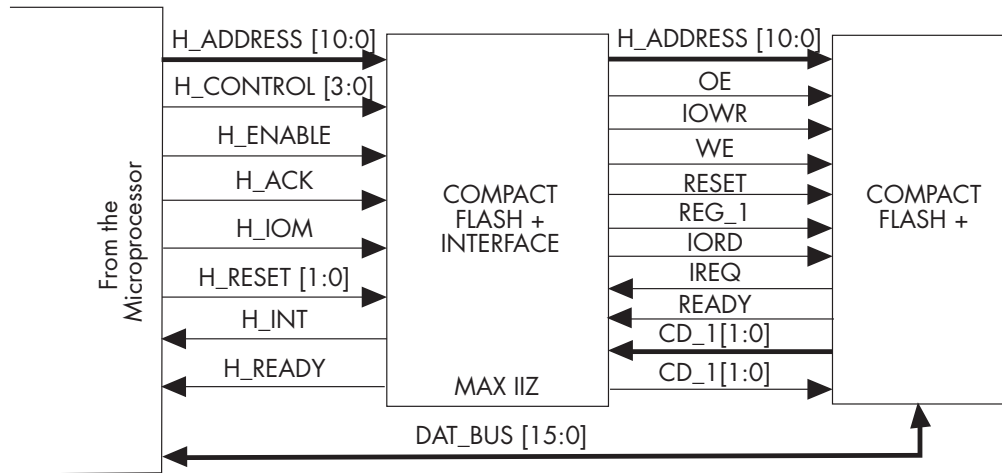
Conclusion

Traditionally, some form of microcontroller has been the only "programmable" logic available to the low-power electronics designer. However, with the introduction of low-power CPLDs, designers have new options for portable applications. This white paper shows multiple examples of how a low-power CPLD can be used in portable applications to replace or augment functions traditionally implemented in microcontrollers. As a result, low-power electronics designers have another set of problem-solving tools for portable applications, and a better ability to choose optimal devices when creating innovative products.

References

1. *AN 426: Using MAX II CPLDS as Analog Keyboard Encoders*:
   www.altera.com/literature/an/an426.pdf

2. *AN 491: Auto Start Using Altera MAX II CPLDS*:
   www.altera.com/literature/an/an491.pdf

3. *AN 494: GPIO Pin Expansion Using I²C Bus Interface in an Altera MAX II CPLD*:
   www.altera.com/literature/an/an494.pdf

4. *AN 486: SPI to I²C Using MAX II CPLDS*:
   www.altera.com/literature/an/an486.pdf

5. *AN 485: Serial Peripheral Interface (SPI) Master in Altera MAX II CPLDS:*
   www.altera.com/literature/an/an485.pdf

6. *AN 492: CF+ Interface Using Altera MAX II CPLDs*:
   www.altera.com/literature/an/an492.pdf

Further Information

- *Reduce Total System Cost in Portable Applications Using MAX II CPLDs*:
  www.altera.com/literature/wp/wp-01001-reduce-total-system-cost-in-portable-apps-using-max.pdf
- *Using Zero-Power CPLDs to Substantially Lower Power Consumption in Portable Applications:*
  www.altera.com/literature/wp/wp-01042-using-zero-power-cplds-to-lower-power-in-portable.pdf
- *AN 422: Power Management in Portable Systems Using MAX II CPLDs*:
  www.altera.com/literature/an/an422.pdf
- Free Quartus® II Web Edition Design Software:
  https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp
- Start Development with MAX II Development Kit:
  www.altera.com/products/devkits/altera/kit-maxii-1270.html
- Buy Device Samples:
  www.altera.com/corporate/contact/con-index.html
- Design Examples and Application Notes:
  www.altera.com/support/examples/max/exm-max.html

Acknowledgements