Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

Embedded - FPGAs, or Field Programmable Gate Arrays, are advanced integrated circuits that offer unparalleled flexibility and performance for digital systems. Unlike traditional fixed-function logic devices, FPGAs can be programmed and reprogrammed to execute a wide array of logical operations, enabling customized functionality tailored to specific applications. This reprogrammability allows developers to iterate designs quickly and implement complex functions without the need for custom hardware.

Applications of Embedded - FPGAs

The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

Details	
Product Status	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	36864
Number of I/O	157
Number of Gates	250000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	256-LBGA
Supplier Device Package	256-FPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p250l-fgg256i

Introduction . . . 213
Low Power Flash Device I/O Support . . . 214
Pro I/Os—IGLOOe, ProASIC3EL, and ProASIC3E . . . 215
I/O Architecture . . . 220
I/O Standards . . . 223
I/O Features . . . 227
Simultaneously Switching Outputs (SSOs) and Printed Circuit Board Layout . . . 241
I/O Software Support . . . 242
User I/O Naming Convention . . . 245
Board-Level Considerations . . . 246
Conclusion . . . 248
Related Documents . . . 248
List of Changes . . . 249

9 I/O Software Control in Low Power Flash Devices . . . 251
Flash FPGAs I/O Support . . . 252
Software-Controlled I/O Attributes . . . 253
Implementing I/Os in Microsemi Software . . . 254
Assigning Technologies and VREF to I/O Banks . . . 264
Conclusion . . . 269
Related Documents . . . 269
List of Changes . . . 270

10 DDR for Microsemi's Low Power Flash Devices . . . 271
Introduction . . . 271
Double Data Rate (DDR) Architecture . . . 271
DDR Support in Flash-Based Devices . . . 272
I/O Cell Architecture . . . 273
Input Support for DDR . . . 275
Output Support for DDR . . . 275
Instantiating DDR Registers . . . 276
Design Example . . . 282
Conclusion . . . 284
List of Changes . . . 285

11 Programming Flash Devices . . . 287
Introduction . . . 287
Summary of Programming Support . . . 287
Programming Support in Flash Devices . . . 288
General Flash Programming Information . . . 289
Important Programming Guidelines . . . 295
Related Documents . . . 297
List of Changes . . . 298

12 Security in Low Power Flash Devices . . . 301
Security in Programmable Logic . . . 301
Security Support in Flash-Based Devices . . . 302
Security Architecture . . . 303
Security Features . . . 304
Security in Action . . . 308

Loading the Confi guration Register

The most important part of CCC dynamic configuration is to load the shift register properly with the configuration bits. There are different ways to access and load the configuration shift register:

- JTAG interface
- Logic core
- Specific I/O tiles

JTAG Interface

The JTAG interface requires no additional I/O pins. The JTAG TAP controller is used to control the loading of the CCC configuration shift register.

Low power flash devices provide a user interface macro between the JTAG pins and the device core logic. This macro is called UJTAG. A user should instantiate the UJTAG macro in his design to access the configuration register ports via the JTAG pins.

For more information on CCC dynamic reconfiguration using UJTAG, refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 363.

Logic Core

If the logic core is employed, the user must design a module to provide the configuration data and control the shifting and updating of the CCC configuration shift register. In effect, this is a user-designed TAP controller, which requires additional chip resources.

Specific I/O Tiles

If specific I/O tiles are used for configuration, the user must provide the external equivalent of a TAP controller. This does not require additional core resources but does use pins.

Shifting the Configuration Data

To enter a new configuration, all 81 bits must shift in via SDIN. After all bits are shifted, SSHIFT must go LOW and SUPDATE HIGH to enable the new configuration. For simulation purposes, bits <71:73> and <77:80> are "don't care."

The SUPDATE signal must be LOW during any clock cycle where SSHIFT is active. After SUPDATE is asserted, it must go back to the LOW state until a new update is required.

PLL Configuration Bits Description

Table 4-8 **Configuration Bit Descriptions for the CCC Blocks**

Config. Bits	Signal	Name	Description
<88:87>	GLMUXCFG [1:0][1]	NGMUX configuration	The configuration bits specify the input clocks to the NGMUX (refer to Table 4-17 on page 110).[2]
86	OCDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by the divider factor in Table 4-18 on page 111.
85	OBDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by a 0.5 factor (refer to Table 4-18 on page 111).
84	OADIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by certain 0.5 factor (refer to Table 4-16 on page 110).

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing* Tools > Report > CCC_Configuration *. The report contains the appropriate settings for these bits.*

Example of RAM Initialization

This section of the document presents a sample design in which a 4×4 RAM block is being initialized through the JTAG port. A test feature has been implemented in the design to read back the contents of the RAM after initialization to verify the procedure.

The interface block of this example performs two major functions: initialization of the RAM block and running a test procedure to read back the contents. The clock output of the interface is either the write clock (for initialization) or the read clock (for reading back the contents). The Verilog code for the interface block is included in the "Sample Verilog Code" section on page 167.

For simulation purposes, users can declare the input ports of the UJTAG macro for easier assignment in the testbench. However, the UJTAG input ports should not be declared on the top level during synthesis. If the input ports of the UJTAG are declared during synthesis, the synthesis tool will instantiate input buffers on these ports. The input buffers on the ports will cause Compile to fail in Designer.

Figure 6-10 shows the simulation results for the initialization step of the example design.

The CLK_OUT signal, which is the clock output of the interface block, is the inverted DR_UPDATE output of the UJTAG macro. It is clear that it gives sufficient time (while the TAP Controller is in the Data Register Update state) for the write address and data to become stable before loading them into the RAM block.

Figure 6-11 presents the test procedure of the example. The data read back from the memory block matches the written data, thus verifying the design functionality.

Figure 6-10 **Simulation of Initialization Step**

Figure 6-11 **Simulation of the Test Procedure of the Example**

Advanced I/Os—IGLOO, ProASIC3L, and ProASIC3

Table 7-2 and Table 7-3 show the voltages and compatible I/O standards for the IGLOO, ProASIC3L, and ProASIC3 families.

I/Os provide programmable slew rates (except 30 K gate devices), drive strengths, and weak pull-up and pull-down circuits. 3.3 V PCI and 3.3 V PCI-X can be configured to be 5 V–tolerant. See the "5 V Input Tolerance" section on page 194 for possible implementations of 5 V tolerance.

All I/Os are in a known state during power-up, and any power-up sequence is allowed without current impact. Refer to the "I/O Power-Up and Supply Voltage Thresholds for Power-On Reset (Commercial and Industrial)" section in the datasheet for more information. During power-up, before reaching activation levels, the I/O input and output buffers are disabled while the weak pull-up is enabled. Activation levels are described in the datasheet.

Table 7-2 **Supported I/O Standards**

IGLOO	**AGL015**	**AGL030**	**AGL060**	**AGL125**	**AGL250**		**AGL600**	**AGL1000**
ProASIC3	**A3P015**	**A3P030**	**A3P060**	**A3P125**	**A3P250/ A3P250L**	**A3P400**	**A3P600/ A3P600L**	**A3P1000/ A3P1000L**
Single-Ended								
LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V / 1.8 V / 1.5 V / 1.2 V LVCMOS 2.5 V / 5.0 V								
3.3 V PCI/PCI-X	–	–						
Differential								
LVPECL, LVDS, B-LVDS, M-LVDS	–	–	–	–				

I/O Banks and I/O Standards Compatibility

I/Os are grouped into I/O voltage banks.

Each I/O voltage bank has dedicated I/O supply and ground voltages (VMV/GNDQ for input buffers and VCCI/GND for output buffers). This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa. Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 7-3 shows the required voltage compatibility values for each of these voltages.

There are four I/O banks on the 250K gate through 1M gate devices.

There are two I/O banks on the 30K, 60K, and 125K gate devices.

I/O standards are compatible if their VCCI and VMV values are identical. VMV and GNDQ are "quiet" input power supply pins and are not used on 30K gate devices (Table 7-3).

Table 7-3 **VCCI Voltages and Compatible IGLOO and ProASIC3 Standards**

VCCI and VMV (typical)	**Compatible Standards**
3.3 V	LVTTL/LVCMOS 3.3, PCI 3.3, PCI-X 3.3 LVPECL
2.5 V	LVCMOS 2.5, LVCMOS 2.5/5.0, LVDS, B-LVDS, M-LVDS
1.8 V	LVCMOS 1.8
1.5 V	LVCMOS 1.5
1.2 V	LVCMOS 1.2

I/O Register Combining

Every I/O has several embedded registers in the I/O tile that are close to the I/O pads. Rather than using the internal register from the core, the user has the option of using these registers for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some architectural rules must be met. Provided these rules are met, the user can enable register combining globally during Compile (as shown in the "Compiling the Design" section on page 261).

This feature is supported by all I/O standards.

Rules for Registered I/O Function

1. The fanout between an I/O pin (D, Y, or E) and a register must be equal to one for combining to be considered on that pin.
2. All registers (Input, Output, and Output Enable) connected to an I/O must share the same clear or preset function:
   – If one of the registers has a CLR pin, all the other registers that are candidates for combining in the I/O must have a CLR pin.
   – If one of the registers has a PRE pin, all the other registers that are candidates for combining in the I/O must have a PRE pin.
   – If one of the registers has neither a CLR nor a PRE pin, all the other registers that are candidates for combining must have neither a CLR nor a PRE pin.
   – If the clear or preset pins are present, they must have the same polarity.
   – If the clear or preset pins are present, they must be driven by the same signal (net).
3. Registers connected to an I/O on the Output and Output Enable pins must have the same clock and enable function:
   – Both the Output and Output Enable registers must have an E pin (clock enable), or none at all.
   – If the E pins are present, they must have the same polarity. The CLK pins must also have the same polarity.

In some cases, the user may want registers to be combined with the input of a bibuf while maintaining the output as-is. This can be achieved by using PDC commands as follows:

```
set_io <signal name> -REGISTER yes ------register will combine
set_preserve <signal name> ----register will not combine
```

Weak Pull-Up and Weak Pull-Down Resistors

IGLOO and ProASIC3 devices support optional weak pull-up and pull-down resistors on each I/O pin. When the I/O is pulled up, it is connected to the VCCI of its corresponding I/O bank. When it is pulled down, it is connected to GND. Refer to the datasheet for more information.

For low power applications, configuration of the pull-up or pull-down of the I/O can be used to set the I/O to a known state while the device is in Flash*Freeze mode. Refer to the "Flash*Freeze Technology and Low Power Modes in IGLOO and ProASIC3L Devices" chapter of the *IGLOO FPGA Fabric User's Guide* or *ProASIC3L FPGA Fabric User's Guide* for more information.

The Flash*Freeze (FF) pin cannot be configured with a weak pull-down or pull-up I/O attribute, as the signal needs to be driven at all times.

Output Slew Rate Control

The slew rate is the amount of time an input signal takes to get from logic Low to logic High or vice versa.

It is commonly defined as the propagation delay between 10% and 90% of the signal's voltage swing. Slew rate control is available for the output buffers of low power flash devices. The output buffer has a programmable slew rate for both HIGH-to-LOW and LOW-to-HIGH transitions. Slew rate control is available for LVTTL, LVCMOS, and PCI-X I/O standards. The other I/O standards have a preset slew value.

The slew rate can be implemented by using a PDC command (Table 7-5 on page 179), setting it "High" or "Low" in the I/O Attribute Editor in Designer, or instantiating a special I/O macro. The default slew rate value is "High."

– The I/O standard of technology-specific I/O macros cannot be changed in the I/O Attribute Editor (see Figure 9-6).
– The user MUST instantiate differential I/O macros (LVDS/LVPECL) in the design. This is the only way to use these standards in the design (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
– To implement the DDR I/O function, the user must instantiate a DDR_REG or DDR_OUT macro. This is the only way to use a DDR macro in the design.

***Figure 9-6* • Assigning a Different I/O Standard to the Generic I/O Macro**

Performing Place-and-Route on the Design

The netlist created by the synthesis tool should now be imported into Designer and compiled. During Compile, the user can specify the I/O placement and attributes by importing the PDC file. The user can also specify the I/O placement and attributes using ChipPlanner and the I/O Attribute Editor under MVN.

Defining I/O Assignments in the PDC File

A PDC file is a Tcl script file specifying physical constraints. This file can be imported to and exported from Designer.

Table 9-3 shows I/O assignment constraints supported in the PDC file.

***Table 9-3* • PDC I/O Constraints**

Command	Action	Example	Comment
I/O Banks Setting Constraints			

set_iobank	Sets the I/O supply voltage, V_{CCI}, and the input reference voltage, V_{REF}, for the specified I/O bank.	`set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage]`
		
		`set_iobank Bank7 -vcci 1.50 -vref 0.75`
			Must use in case of mixed I/O voltage (V_{CCI}) design

set_vref	Assigns a V_{REF} pin to a bank.	`set_vref -bank [bankname] [pinnum]`
		
		`set_vref -bank Bank0 685 704 723 742 761`
			Must use if voltage-referenced I/Os are used

set_vref_defaults	Sets the default V_{REF} pins for the specified bank. This command is ignored if the bank does not need a V_{REF} pin.	`set_vref_defaults bankname`
		
		`set_vref_defaults bank2`	

Note: Refer to the Libero SoC User's Guide *for detailed rules on PDC naming and syntax conventions.*