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	68
Number of Gates	250000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-20°C ~ 85°C (TJ)
Package / Case	100-TQFP
Supplier Device Package	100-VQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn250-z2vqg100

Array Coordinates

During many place-and-route operations in the Microsemi Designer software tool, it is possible to set constraints that require array coordinates. Table 1-2 provides array coordinates of core cells and memory blocks for IGLOO and ProASIC3 devices. Table 1-3 provides the information for IGLOO PLUS devices. Table 1-4 on page 17 provides the information for IGLOO nano and ProASIC3 nano devices. The array coordinates are measured from the lower left (0, 0). They can be used in region constraints for specific logic groups/blocks, designated by a wildcard, and can contain core cells, memories, and I/Os.

I/O and cell coordinates are used for placement constraints. Two coordinate systems are needed because there is not a one-to-one correspondence between I/O cells and core cells. In addition, the I/O coordinate system changes depending on the die/package combination. It is not listed in Table 1-2. The Designer ChipPlanner tool provides the array coordinates of all I/O locations. I/O and cell coordinates are used for placement constraints. However, I/O placement is easier by package pin assignment.

Figure 1-9 on page 17 illustrates the array coordinates of a 600 k gate device. For more information on how to use array coordinates for region/placement constraints, see the *Designer User's Guide* or online help (available in the software) for software tools.

Table 1-2 • **IGLOO and ProASIC3 Array Coordinates**

Device		VersaTiles				Memory Rows		Entire Die	
		Min.		Max.		Bottom	Top	Min.	Max.
IGLOO	ProASIC3/ ProASIC3L	x	y	x	y	(x, y)	(x, y)	(x, y)	(x, y)
AGL015	A3P015	3	2	34	13	None	None	(0, 0)	(37, 15)
AGL030	A3P030	3	3	66	13	None	None	(0, 0)	(69, 15)
AGL060	A3P060	3	2	66	25	None	(3, 26)	(0, 0)	(69, 29)
AGL125	A3P125	3	2	130	25	None	(3, 26)	(0, 0)	(133, 29)
AGL250	A3P250/L	3	2	130	49	None	(3, 50)	(0, 0)	(133, 53)
AGL400	A3P400	3	2	194	49	None	(3, 50)	(0, 0)	(197, 53)
AGL600	A3P600/L	3	4	194	75	(3, 2)	(3, 76)	(0, 0)	(197, 79)
AGL1000	A3P1000/L	3	4	258	99	(3, 2)	(3, 100)	(0, 0)	(261, 103)
AGLE600	A3PE600/L, RT3PE600L	3	4	194	75	(3, 2)	(3, 76)	(0, 0)	(197, 79)
	A3PE1500	3	4	322	123	(3, 2)	(3, 124)	(0, 0)	(325, 127)
AGLE3000	A3PE3000/L, RT3PE3000L	3	6	450	173	(3, 2) or (3, 4)	(3, 174) or (3, 176)	(0, 0)	(453, 179)

Table 1-3 • **IGLOO PLUS Array Coordinates**

Device	VersaTiles				Memory Rows		Entire Die	
	Min.		Max.		Bottom	Top	Min.	Max.
IGLOO PLUS	x	y	x	y	(x, y)	(x, y)	(x, y)	(x, y)
AGLP030	2	3	67	13	None	None	(0, 0)	(69, 15)
AGLP060	2	2	67	25	None	(3, 26)	(0, 0)	(69, 29)
AGLP125	2	2	131	25	None	(3, 26)	(0, 0)	(133, 29)

Alternatively, Figure 2-7 shows how a microprocessor can be used with a voltage regulator's shutdown pin to turn the power supplies connected to the device on or off.
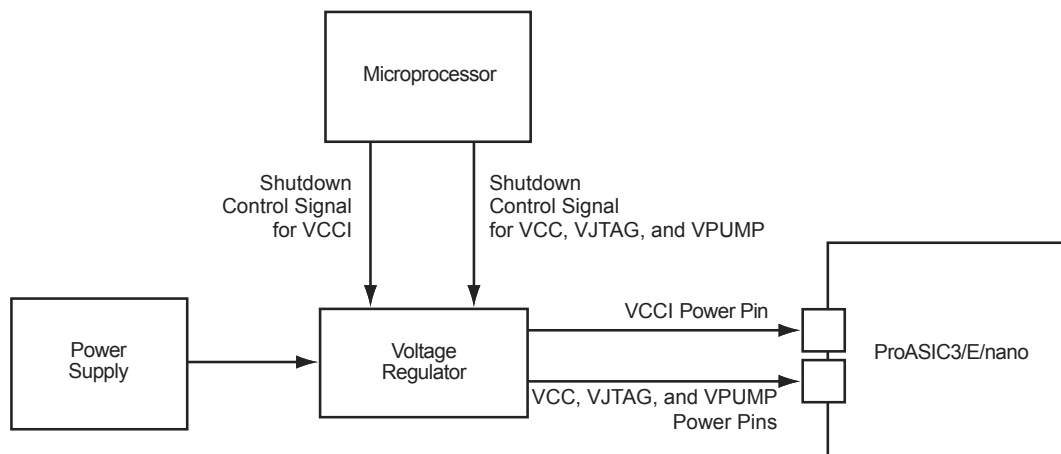

Figure 2-7 • **Controlling Power On/Off State Using Microprocessor and Voltage Regulator**

Though Sleep mode or Shutdown mode can be used to save power, the content of the SRAM and the state of the registers is lost when power is turned off if no other measure is taken. To keep the original contents of the device, a low-cost external serial EEPROM can be used to save and restore the device contents when entering and exiting Sleep mode. In the *Embedded SRAM Initialization Using External Serial EEPROM* application note, detailed information and a reference design are provided to initialize the embedded SRAM using an external serial EEPROM. The user can easily customize the reference design to save and restore the FPGA state when entering and exiting Sleep mode. The microcontroller will need to manage this activity, so before powering down VCC, the data must be read from the FPGA and stored externally. Similarly, after the FPGA is powered up, the microcontroller must allow the FPGA to load the data from external memory and restore its original state.

Conclusion

Microsemi ProASIC3/E and ProASIC3 nano FPGAs inherit low power consumption capability from their nonvolatile and live-at-power-up flash-based technology. Power consumption can be reduced further using the Static (Idle), User Low Static (Idle), Sleep, or Shutdown power modes. All these features result in a low-power, cost-effective, single-chip solution designed specifically for power-sensitive electronics applications.

Related Documents

Application Notes

Embedded SRAM Initialization Using External Serial EEPROM

http://www.microsemi.com/soc/documents/EmbeddedSRAMInit_AN.pdf

The following will happen during demotion of a global signal to regular nets:

- CLKBUF_x becomes INBUF_x; CLKINT is removed from the netlist.
- The essential global macro, such as the output of the Clock Conditioning Circuit, cannot be demoted.
- No automatic buffering will happen.

Since no automatic buffering happens when a signal is demoted, this net may have a high delay due to large fanout. This may have a negative effect on the quality of the results. Microsemi recommends that the automatic global demotion only be used on small-fanout nets. Use clock networks for high-fanout nets to improve timing and routability.

Spine Assignment

The low power flash device architecture allows the global networks to be segmented and used as clock spines. These spines, also called local clock networks, enable the use of PDC or MVN to assign a signal to a spine.

PDC syntax to promote a net to a spine/local clock:

```
assign_local_clock –net netname –type [quadrant|chip] Tn|Bn|Tn:Bm
```

If the net is driven by a clock macro, Designer automatically demotes the clock net to a regular net before it is assigned to a spine. Nets driven by a PLL or CLKDLY macro cannot be assigned to a local clock.

When assigning a signal to a spine or quadrant global network using PDC (pre-compile), the Designer software will legalize the shared instances. The number of shared instances to be legalized can be controlled by compile options. If these networks are created in MVN (only quadrant globals can be created), no legalization is done (as it is post-compile). Designer does not do legalization between non-clock nets.

As an example, consider two nets, net_clk and net_reset, driving the same flip-flop. The following PDC constraints are used:

```
assign_local_clock –net net_clk –type chip T3
assign_local_clock –net net_reset –type chip T1:T2
```

During Compile, Designer adds a buffer in the reset net and places it in the T1 or T2 region, and places the flip-flop in the T3 spine region (Figure 3-16).
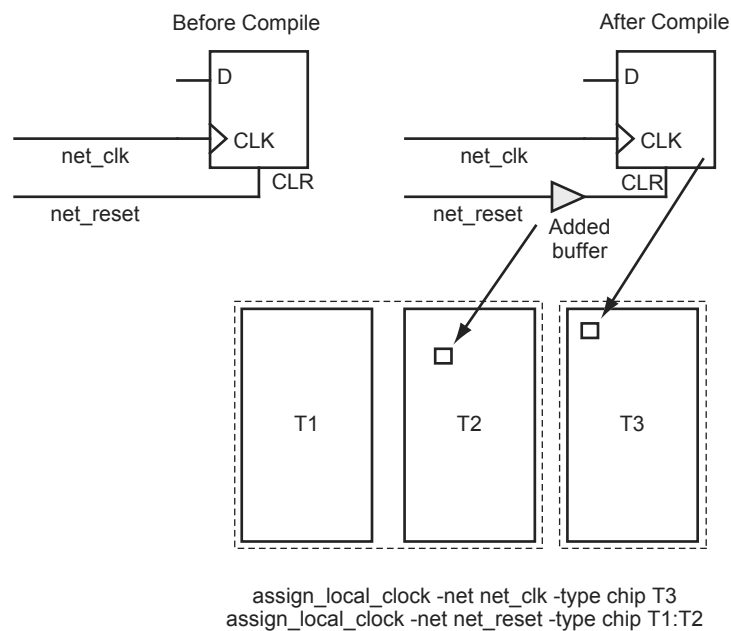

Figure 3-16 • **Adding a Buffer for Shared Instances**

Figure 3-18 • **Globals Management GUI in Designer**

3. Occasionally, the synthesis tool assigns a global macro to clock nets, even though the fanout is significantly less than other asynchronous signals. Select **Demote global nets whose fanout is less than** and enter a reasonable value for fanouts. This frees up some global networks from the signals that have very low fanouts. This can also be done using PDC.

4. Use a local clock network for the signals that do not need to go to the whole chip but should have low skew. This local clock network assignment can only be done using PDC.

5. Assign the I/O buffer using MVN if you have fixed I/O assignment. As shown in Figure 3-10 on page 45, there are three sets of global pins that have a hardwired connection to each global network. Do not try to put multiple CLKBUF macros in these three sets of global pins. For example, do not assign two CLKBUFs to GAA0x and GAA2x pins.

6. You must click **Commit** at the end of MVN assignment. This runs the pre-layout checker and checks the validity of global assignment.

7. Always run Compile with the **Keep existing physical constraints** option on. This uses the quadrant clock network assignment in the MVN assignment and checks if you have the desired signals on the global networks.

8. Run Layout and check the timing.

```
    wire VCC, GND;

    VCC VCC_1_net(.Y(VCC));
    GND GND_1_net(.Y(GND));
    CLKDLY Inst1(.CLK(CLK), .GL(GL), .DLYGL0(VCC), .DLYGL1(GND), .DLYGL2(VCC),
      .DLYGL3(GND), .DLYGL4(GND));

endmodule
```

Detailed Usage Information

Clock Frequency Synthesis

Deriving clocks of various frequencies from a single reference clock is known as frequency synthesis. The PLL has an input frequency range from 1.5 to 350 MHz. This frequency is automatically divided down to a range between 1.5 MHz and 5.5 MHz by input dividers (not shown in Figure 4-19 on page 84) between PLL macro inputs and PLL phase detector inputs. The VCO output is capable of an output range from 24 to 350 MHz. With dividers before the input to the PLL core and following the VCO outputs, the VCO output frequency can be divided to provide the final frequency range from 0.75 to 350 MHz. Using SmartGen, the dividers are automatically set to achieve the closest possible matches to the specified output frequencies.

Users should be cautious when selecting the desired PLL input and output frequencies and the I/O buffer standard used to connect to the PLL input and output clocks. Depending on the I/O standards used for the PLL input and output clocks, the I/O frequencies have different maximum limits. Refer to the family datasheets for specifications of maximum I/O frequencies for supported I/O standards. Desired PLL input or output frequencies will not be achieved if the selected frequencies are higher than the maximum I/O frequencies allowed by the selected I/O standards. Users should be careful when selecting the I/O standards used for PLL input and output clocks. Performing post-layout simulation can help detect this type of error, which will be identified with pulse width violation errors. Users are strongly encouraged to perform post-layout simulation to ensure the I/O standard used can provide the desired PLL input or output frequencies. Users can also choose to cascade PLLs together to achieve the high frequencies needed for their applications. Details of cascading PLLs are discussed in the "Cascading CCCs" section on page 109.

In SmartGen, the actual generated frequency (under typical operating conditions) will be displayed beside the requested output frequency value. This provides the ability to determine the exact frequency that can be generated by SmartGen, in real time. The log file generated by SmartGen is a useful tool in determining how closely the requested clock frequencies match the user specifications. For example, assume a user specifies 101 MHz as one of the secondary output frequencies. If the best output frequency that could be achieved were 100 MHz, the log file generated by SmartGen would indicate the actual generated frequency.

Simulation Verification

The integration of the generated PLL and CLKDLY modules is similar to any VHDL component or Verilog module instantiation in a larger design; i.e., there is no special requirement that users need to take into account to successfully synthesize their designs.

For simulation purposes, users need to refer to the VITAL or Verilog library that includes the functional description and associated timing parameters. Refer to the Software Tools section of the Microsemi SoC Products Group website to obtain the family simulation libraries. If Designer is installed, these libraries are stored in the following locations:

<Designer_Installation_Directory>\lib\vtl\95\proasic3.vhd

<Designer_Installation_Directory>\lib\vtl\95\proasic3e.vhd

<Designer_Installation_Directory>\lib\vlog\proasic3.v

<Designer_Installation_Directory>\lib\vlog\proasic3e.v

For Libero users, there is no need to compile the simulation libraries, as they are conveniently pre-compiled in the Model*Sim*® Microsemi simulation tool.

- Use quadrant global region assignments by finding the clock net associated with the CCC macro under the Nets tab and creating a quadrant global region for the net, as shown in Figure 4-33.

Figure 4-33 • **Quadrant Clock Assignment for a Global Net**

External I/O–Driven CCCs

The above-mentioned recommendation for proper layout techniques will ensure the correct assignment. It is possible that, especially with External I/O–Driven CCC macros, placement of the CCC macro in a desired location may not be achieved. For example, assigning an input port of an External I/O–Driven CCC near a particular CCC location does not guarantee global assignments to the desired location. This is because the clock inputs of External I/O–Driven CCCs can be assigned to any I/O location; therefore, it is possible that the CCC connected to the clock input will be routed to a location other than the one closest to the I/O location, depending on resource availability and placement constraints.

Clock Placer

The clock placer is a placement engine for low power flash devices that places global signals on the chip global and quadrant global networks. Based on the clock assignment constraints for the chip global and quadrant global clocks, it will try to satisfy all constraints, as well as creating quadrant clock regions when necessary. If the clock placer fails to create the quadrant clock regions for the global signals, it will report an error and stop Layout.

The user must ensure that the constraints set to promote clock signals to quadrant global networks are valid.

Cascading CCCs

The CCCs in low power flash devices can be cascaded. Cascading CCCs can help achieve more accurate PLL output frequency results than those achievable with a single CCC. In addition, this technique is useful when the user application requires the output clock of the PLL to be a multiple of the reference clock by an integer greater than the maximum feedback divider value of the PLL (divide by 128) to achieve the desired frequency.

For example, the user application may require a 280 MHz output clock using a 2 MHz input reference clock, as shown in Figure 4-34 on page 110.

FlashROM Generation and Instantiation in the Design

The SmartGen core generator, available in Libero SoC and Designer, is the only tool that can be used to generate the FlashROM content. SmartGen has several user-friendly features to help generate the FlashROM contents. Instead of selecting each byte and assigning values, you can create a region within a page, modify the region, and assign properties to that region. The FlashROM user interface, shown in Figure 5-10, includes the configuration grid, existing regions list, and properties field. The properties field specifies the region-specific information and defines the data used for that region. You can assign values to the following properties:

1. Static Fixed Data—Enables you to fix the data so it cannot be changed during programming time. This option is useful when you have fixed data stored in this region, which is required for the operation of the design in the FPGA. Key storage is one example.

2. Static Modifiable Data—Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration but could conceivably change in the future). This option enables you to avoid changing the value every time you enter new data.

3. Read from File—This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a text file with data for as many devices as you wish to program, and load that into the FlashPoint programming file generation software to get programming files that include all the data. SmartGen will optionally pass the location of the file where the data is stored if the file is specified in SmartGen. Each text file has only one type of data format (binary, decimal, hex, or ASCII text). The length of each data file must be shorter than or equal to the selected region length. If the data is shorter than the selected region length, the most significant bits will be padded with 0s. For multiple text files for multiple regions, the first lines are for the first device. In SmartGen, **Load Sim. Value From File** allows you to load the first device data in the MEM file for simulation.

4. Auto Increment/Decrement—This scenario is useful when you specify the contents of FlashROM for a large number of devices in a series. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is fed to the software.

Figure 5-10 • **SmartGen GUI of the FlashROM**

FIFO Flag Usage Considerations

The AEVAL and AFVAL pins are used to specify the 12-bit AEMPTY and AFULL threshold values. The FIFO contains separate 12-bit write address (WADDR) and read address (RADDR) counters. WADDR is incremented every time a write operation is performed, and RADDR is incremented every time a read operation is performed. Whenever the difference between WADDR and RADDR is greater than or equal to AFVAL, the AFULL output is asserted. Likewise, whenever the difference between WADDR and RADDR is less than or equal to AEVAL, the AEMPTY output is asserted. To handle different read and write aspect ratios, AFVAL and AEVAL are expressed in terms of total data bits instead of total data words. When users specify AFVAL and AEVAL in terms of read or write words, the SmartGen tool translates them into bit addresses and configures these signals automatically. SmartGen configures the AFULL flag to assert when the write address exceeds the read address by at least a predefined value. In a 2k×8 FIFO, for example, a value of 1,500 for AFVAL means that the AFULL flag will be asserted after a write when the difference between the write address and the read address reaches 1,500 (there have been at least 1,500 more writes than reads). It will stay asserted until the difference between the write and read addresses drops below 1,500.

The AEMPTY flag is asserted when the difference between the write address and the read address is less than a predefined value. In the example above, a value of 200 for AEVAL means that the AEMPTY flag will be asserted when a read causes the difference between the write address and the read address to drop to 200. It will stay asserted until that difference rises above 200. Note that the FIFO can be configured with different read and write widths; in this case, the AFVAL setting is based on the number of write data entries, and the AEVAL setting is based on the number of read data entries. For aspect ratios of 512×9 and 256×18, only 4,096 bits can be addressed by the 12 bits of AFVAL and AEVAL. The number of words must be multiplied by 8 and 16 instead of 9 and 18. The SmartGen tool automatically uses the proper values. To avoid halfwords being written or read, which could happen if different read and write aspect ratios were specified, the FIFO will assert FULL or EMPTY as soon as at least one word cannot be written or read. For example, if a two-bit word is written and a four-bit word is being read, the FIFO will remain in the empty state when the first word is written. This occurs even if the FIFO is not completely empty, because in this case, a complete word cannot be read. The same is applicable in the full state. If a four-bit word is written and a two-bit word is read, the FIFO is full and one word is read. The FULL flag will remain asserted because a complete word cannot be written at this point.

Variable Aspect Ratio and Cascading

Variable aspect ratio and cascading allow users to configure the memory in the width and depth required. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18. Low power flash device RAM can be configured as 1, 2, 4, 9, or 18 bits wide. By cascading the memory blocks, any multiple of those widths can be created. The RAM blocks can be from 256 to 4,096 bits deep, depending on the aspect ratio, and the blocks can also be cascaded to create deeper areas. Refer to the aspect ratios available for each macro cell in the "SRAM Features" section on page 137. The largest continuous configurable memory area is equal to half the total memory available on the device, because the RAM is separated into two groups, one on each side of the device.

The SmartGen core generator will automatically configure and cascade both RAM and FIFO blocks. Cascading is accomplished using dedicated memory logic and does not consume user gates for depths up to 4,096 bits deep and widths up to 18, depending on the configuration. Deeper memory will utilize some user gates to multiplex the outputs.

Generated RAM and FIFO macros can be created as either structural VHDL or Verilog for easy instantiation into the design. Users of Libero SoC can create a symbol for the macro and incorporate it into a design schematic.

Table 6-10 on page 147 shows the number of memory blocks required for each of the supported depth and width memory configurations, and for each depth and width combination. For example, a 256-bit deep by 32-bit wide two-port RAM would consist of two 256×18 RAM blocks. The first 18 bits would be stored in the first RAM block, and the remaining 14 bits would be implemented in the other 256×18 RAM block. This second RAM block would have four bits of unused storage. Similarly, a dual-port memory block that is 8,192 bits deep and 8 bits wide would be implemented using 16 memory blocks. The dual-port memory would be configured in a 4,096×1 aspect ratio. These blocks would then be cascaded two deep to achieve 8,192 bits of depth, and eight wide to achieve the eight bits of width.

Table 6-10 • **RAM and FIFO Memory Block Consumption**

Width			Depth									
			256		512	1,024	2,048	4,096	8,192	16,384	32,768	65,536
			Two-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port
1	Number Block		1	1	1	1	1	1	2	4	8	16 × 1
	Configuration		Any	Any	Any	1,024 × 4	2,048 × 2	4,096 × 1	2 × (4,096 × 1) Cascade Deep	4 × (4,096 × 1) Cascade Deep	8 × (4,096 × 1) Cascade Deep	16 × (4,096 × 1) Cascade Deep
2	Number Block		1	1	1	1	1	2	4	8	16	32
	Configuration		Any	Any	Any	1,024×4	2,048 × 2	2 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 2 Wide	8 × (4,096 × 1) Cascaded 4 Deep and 2 Wide	16 × (4,096 × 1) Cascaded 8 Deep and 2 Wide	32 × (4,096 × 1) Cascaded 16 Deep and 2 Wide
4	Number Block		1	1	1	1	2	4	8	16	32	64
	Configuration		Any	Any	Any	1,024 × 4	2 × (2,048 × 2) Cascaded Wide	4 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 4 Wide	16 × (4,096 × 1) Cascaded 4 Deep and 4 Wide	32 × (4,096 × 1) Cascaded 8 Deep and 4 Wide	64 × (4,096 × 1) Cascaded 16 Deep and 4 Wide
8	Number Block		1	1	1	2	4	8	16	32	64	
	Configuration		Any	Any	Any	2 × (1,024 × 4) Cascaded Wide	4 × (2,048 × 2) Cascaded Wide	8 × (4,096 × 1) Cascaded Wide	16 × (4,096 × 1) Cascaded 2 Deep and 8 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 8 Wide	64 × (4,096 × 1) Cascaded 8 Deep and 8 Wide	
9	Number Block		1	1	1	2	4	8	16	32		
	Configuration		Any	Any	Any	2 × (512 × 9) Cascaded Deep	4 × (512 × 9) Cascaded Deep	8 × (512 × 9) Cascaded Deep	16 × (512 × 9) Cascaded Deep	32 × (512 × 9) Cascaded Deep		
16	Number Block		1	1	1	4	8	16	32	64		
	Configuration		256 × 18	256 × 18	256 × 18	4 × (1,024 × 4) Cascaded Wide	8 × (2,048 × 2) Cascaded Wide	16 × (4,096 × 1) Cascaded Wide	32 × (4,096 × 1) Cascaded 2 Deep and 16 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 16 Wide		
18	Number Block		1	2	2	4	8	18	32			
	Configuration		256 × 8	2 × (512 × 9) Cascaded Wide	2 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 2 Wide	8 × (512 × 9) Cascaded 4 Deep and 2 Wide	16 × (512 × 9) Cascaded 8 Deep and 2 Wide	16 × (512 × 9) Cascaded 16 Deep and 2 Wide			
32	Number Block		2	4	4	8	16	32	64			
	Configuration		2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	8 × (1,024 × 4) Cascaded Wide	16 × (2,048 × 2) Cascaded Wide	32 × (4,096 × 1) Cascaded Wide	64 × (4,096 × 1) Cascaded 2 Deep and 32 Wide			
36	Number Block		2	4	4	8	16	32				
	Configuration		2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 4 Wide	16 × (512 × 9) Cascaded 4 Deep and 4 Wide	16 × (512 × 9) Cascaded 8 Deep and 4 Wide				
64	Number Block		4	8	8	16	32	64				
	Configuration		4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (1,024 × 4) Cascaded Wide	32 × (2,048 × 2) Cascaded Wide	64 × (4,096 × 1) Cascaded Wide				
72	Number Block		4	8	8	16	32					
	Configuration		4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded 4 Deep and 8 Wide					

Note: *Memory configurations represented by grayed cells are not supported.*

Low Power Flash Device I/O Support

The low power flash families listed in Table 7-1 support I/Os and the functions described in this document.

Table 7-1 • **Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO nano	Lowest power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
ProASIC3	ProASIC3 nano	Lowest cost 1.5 V FPGAs with balanced performance

Note: **The device name links to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Table 7-8 • **Hot-Swap Level 1**

Description	Cold-swap
Power Applied to Device	No
Bus State	–
Card Ground Connection	–
Device Circuitry Connected to Bus Pins	–
Example Application	System and card with Microsemi FPGA chip are powered down, and the card is plugged into the system. Then the power supplies are turned on for the system but not for the FPGA on the card.
Compliance of nano Devices	Compliant

Table 7-9 • **Hot-Swap Level 2**

Description	Hot-swap while reset
Power Applied to Device	Yes
Bus State	Held in reset state
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	–
Example Application	In the PCI hot-plug specification, reset control circuitry isolates the card busses until the card supplies are at their nominal operating levels and stable.
Compliance of nano Devices	Compliant

Related Documents

Application Notes

Board-Level Considerations

http://www.microsemi.com/soc/documents/ALL_AC276_AN.pdf

User's Guides

Libero SoC User's Guide

http://www.microsemi.com/soc/documents/libero_ug.pdf

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

SmartGen Core Reference Guide

http://www.microsemi.com/soc/documents/genguide_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the document.

Date	Changes	Page
August 2012	Figure 7-2 • I/O Block Logical Representation for Dual-Tile Designs (60 k,125 k, and 250 k Devices) was revised to indicate that resets on registers 1, 3, 4, and 5 are active high rather than active low (SAR 40698).	160
	The hyperlink for the *Board-Level Considerations* application note was corrected (SAR 36663).	181, 183
June 2011	Figure 7-2 • I/O Block Logical Representation for Dual-Tile Designs (60 k,125 k, and 250 k Devices) was revised so that the I/O_CLR and I/O_OCLK nets are no longer joined in front of Input Register 3 but instead on the branch of the CLR/PRE signal (SAR 26052).	160
	The following sentence was removed from the "LVCMOS (Low-Voltage CMOS)" section (SAR 22634): "All these versions use a 3.3 V–tolerant CMOS input buffer and a push-pull output buffer."	166
	The "5 V Input Tolerance" section was revised to state that 5 V input tolerance can be used with LVTTL 3.3 V and LVCMOS 3.3 V configurations. LVCMOS 2.5 V, LVCMOS 1.8 V, LVCMOS 1.5 V, and LVCMOS 1.2 V were removed from the sentence listing supported configurations (SAR 22427).	171

Volume Programming Services

Device Type Supported: Flash and Antifuse

Once the design is stable for applications with large production volumes, preprogrammed devices can be purchased. Table 10-2 describes the volume programming services.

Table 10-2 • **Volume Programming Services**

Programmer	Vendor	Availability
In-House Programming	Microsemi	Contact Microsemi Sales
Distributor Programming Centers	Memec Unique	Contact Distribution
Independent Programming Centers	Various	Contact Vendor

Advantages: As programming is outsourced, this solution is easier to implement than creating a substantial in-house programming capability. As programming houses specialize in large-volume programming, this is often the most cost-effective solution.

Limitations: There are some logistical issues with the use of a programming service provider, such as the transfer of programming files and the approval of First Articles. By definition, the programming file must be released to a third-party programming house. Nondisclosure agreements (NDAs) can be signed to help ensure data protection; however, for extremely security-conscious designs, this may not be an option.

- Microsemi In-House Programming

    When purchasing Microsemi devices in volume, IHP can be requested as part of the purchase. If this option is chosen, there is a small cost adder for each device programmed. Each device is marked with a special mark to distinguish it from blank parts. Programming files for the design will be sent to Microsemi. Sample parts with the design programmed, First Articles, will be returned for customer approval. Once approval of First Articles has been received, Microsemi will proceed with programming the remainder of the order. To request Microsemi IHP, contact your local Microsemi representative.

- Distributor Programming Centers

    If purchases are made through a distributor, many distributors will provide programming for their customers. Consult with your preferred distributor about this option.

Cortex-M1 Device Security

Cortex-M1–enabled devices are shipped with the following security features:

- FPGA array enabled for AES-encrypted programming and verification
- FlashROM enabled for AES-encrypted Write and Verify
- Fusion Embedded Flash Memory enabled for AES-encrypted Write

AES Encryption of Programming Files

Low power flash devices employ AES as part of the security mechanism that prevents invasive and noninvasive attacks. The mechanism entails encrypting the programming file with AES encryption and then passing the programming file through the AES decryption core, which is embedded in the device. The file is decrypted there, and the device is successfully programmed. The AES master key is stored in on-chip nonvolatile memory (flash). The AES master key can be preloaded into parts in a secure programming environment (such as the Microsemi In-House Programming center), and then "blank" parts can be shipped to an untrusted programming or manufacturing center for final personalization with an AES-encrypted bitstream. Late-stage product changes or personalization can be implemented easily and securely by simply sending a STAPL file with AES-encrypted data. Secure remote field updates over public networks (such as the Internet) are possible by sending and programming a STAPL file with AES-encrypted data.

The AES key protects the programming data for file transfer into the device with 128-bit AES encryption. If AES encryption is used, the AES key is stored or preprogrammed into the device. To program, you must use an AES-encrypted file, and the encryption used on the file must match the encryption key already in the device.

The AES key is protected by a FlashLock security Pass Key that is also implemented in each device. The AES key is always protected by the FlashLock Key, and the AES-encrypted file does NOT contain the FlashLock Key. This FlashLock Pass Key technology is exclusive to the Microsemi flash-based device families. FlashLock Pass Key technology can also be implemented without the AES encryption option, providing a choice of different security levels.

In essence, security features can be categorized into the following three options:

- AES encryption with FlashLock Pass Key protection
- FlashLock protection only (no AES encryption)
- No protection

Each of the above options is explained in more detail in the following sections with application examples and software implementation options.

Advanced Encryption Standard

The 128-bit AES standard (FIPS-192) block cipher is the NIST (National Institute of Standards and Technology) replacement for DES (Data Encryption Standard FIPS46-2). AES has been designed to protect sensitive government information well into the 21st century. It replaces the aging DES, which NIST adopted in 1977 as a Federal Information Processing Standard used by federal agencies to protect sensitive, unclassified information. The 128-bit AES standard has 3.4×10^{38} possible 128-bit key variants, and it has been estimated that it would take 1,000 trillion years to crack 128-bit AES cipher text using exhaustive techniques. Keys are stored (securely) in low power flash devices in nonvolatile flash memory. All programming files sent to the device can be authenticated by the part prior to programming to ensure that bad programming data is not loaded into the part that may possibly damage it. All programming verification is performed on-chip, ensuring that the contents of low power flash devices remain secure.

Microsemi has implemented the 128-bit AES (Rijndael) algorithm in low power flash devices. With this key size, there are approximately 3.4×10^{38} possible 128-bit keys. DES has a 56-bit key size, which provides approximately 7.2×10^{16} possible keys. In their AES fact sheet, the National Institute of Standards and Technology uses the following hypothetical example to illustrate the theoretical security provided by AES. If one were to assume that a computing system existed that could recover a DES key in a second, it would take that same machine approximately 149 trillion years to crack a 128-bit AES key. NIST continues to make their point by stating the universe is believed to be less than 20 billion years old.[1]

Figure 11-19 • **FlashLock Pass Key, Previously Programmed Devices**

It is important to note that when the security settings need to be updated, the user also needs to select the **Security settings** check box in Step 1, as shown in Figure 11-10 on page 248 and Figure 11-11 on page 248, to modify the security settings. The user must consider the following:

- If only a new AES key is necessary, the user must re-enter the same Pass Key previously programmed into the device in Designer and then generate a programming file with the same Pass Key and a different AES key. This ensures the programming file can be used to access and program the device and the new AES key.
- If a new Pass Key is necessary, the user can generate a new programming file with a new Pass Key (with the same or a new AES key if desired). However, for programming, the user must first load the original programming file with the Pass Key that was previously used to unlock the device. Then the new programming file can be used to program the new security settings.

Advanced Options

As mentioned, there may be applications where more complicated security settings are required. The "Custom Security Levels" section in the *FlashPro User's Guide* describes different advanced options available to aid the user in obtaining the best available security settings.

errors, but this list is intended to show where problems can occur. FlashPro4/3/3X allows TCK to be lowered from 6 MHz down to 1 MHz to allow you to address some signal integrity problems that may occur with impedance mismatching at higher frequencies. Customers are expected to troubleshoot board-level signal integrity issues by measuring voltages and taking scope plots.

Scan Chain Failure

Normally, the FlashPro4/3/3X Scan Chain command expects to see 0x1 on the TDO pin. If the command reports reading 0x0 or 0x3, it is seeing the TDO pin stuck at 0 or 1. The only time the TDO pin comes out of tristate is when the JTAG TAP state machine is in the Shift-IR or Shift-DR state. If noise or reflections on the TCK or TMS lines have disrupted the correct state transitions, the device's TAP state controller might not be in one of these two states when the programmer tries to read the device. When this happens, the output is floating when it is read and does not match the expected data value. This can also be caused by a broken TDO net. Only a small amount of data is read from the device during the Scan Chain command, so marginal problems may not always show up during this command. Occasionally a faulty programmer can cause intermittent scan chain failures.

Exit 11

This error occurs during the verify stage of programming a device. After programming the design into the device, the device is verified to ensure it is programmed correctly. The verification is done by shifting the programming data into the device. An internal comparison is performed within the device to verify that all switches are programmed correctly. Noise induced by poor signal integrity can disrupt the writes and reads or the verification process and produce a verification error. While technically a verification error, the root cause is often related to signal integrity.

Refer to the *FlashPro User's Guide* for other error messages and solutions. For the most up-to-date known issues and solutions, refer to http://www.microsemi.com/soc/support.

Conclusion

IGLOO, ProASIC3, SmartFusion, and Fusion devices offer a low-cost, single-chip solution that is live at power-up through nonvolatile flash technology. The FlashLock Pass Key and 128-bit AES Key security features enable secure ISP in an untrusted environment. On-chip FlashROM enables a host of new applications, including device serialization, subscription-based applications, and IP addressing. Additionally, as the FlashROM is nonvolatile, all of these services can be provided without battery backup.

Related Documents

User's Guides

FlashPro User's Guide

http://www.microsemi.com/soc/documents/flashpro_ug.pdf

Microsemi

---

13 – Core Voltage Switching Circuit for IGLOO and ProASIC3L In-System Programming

---

Introduction

The IGLOO® and ProASIC®3L families offer devices that can be powered by either 1.5 V or, in the case of V2 devices, a core supply voltage anywhere in the range of 1.2 V to 1.5 V, in 50 mV increments.

Since IGLOO and ProASIC3L devices are flash-based, they can be programmed and reprogrammed multiple times in-system using Microsemi FlashPro3. FlashPro3 uses the JTAG standard interface (IEEE 1149.1) and STAPL file (defined in JESD 71 to support programming of programmable devices using IEEE 1149.1) for in-system configuration/programming (IEEE 1532) of a device. Programming can also be executed by other methods, such as an embedded microcontroller that follows the same standards above.

All IGLOO and ProASIC3L devices must be programmed with the VCC core voltage at 1.5 V. Therefore, applications using IGLOO or ProASIC3L devices powered by a 1.2 V supply must switch the core supply to 1.5 V for in-system programming.

The purpose of this document is to describe an easy-to-use and cost-effective solution for switching the core supply voltage from 1.2 V to 1.5 V during in-system programming for IGLOO and ProASIC3L devices.

Microsemi.

14 – Microprocessor Programming of Microsemi's Low Power Flash Devices

Introduction

The Fusion, IGLOO, and ProASIC3 families of flash FPGAs support in-system programming (ISP) with the use of a microprocessor. Flash-based FPGAs store their configuration information in the actual cells within the FPGA fabric. SRAM-based devices need an external configuration memory, and hybrid nonvolatile devices store the configuration in a flash memory inside the same package as the SRAM FPGA. Since the programming of a true flash FPGA is simpler, requiring only one stage, it makes sense that programming with a microprocessor in-system should be simpler than with other SRAM FPGAs. This reduces bill-of-materials costs and printed circuit board (PCB) area, and increases system reliability.

Nonvolatile flash technology also gives the low power flash devices the advantage of a secure, low power, live-at-power-up, and single-chip solution. Low power flash devices are reprogrammable and offer time-to-market benefits at an ASIC-level unit cost. These features enable engineers to create high-density systems using existing ASIC or FPGA design flows and tools.

This document is an introduction to microprocessor programming only. To explain the difference between the options available, user's guides for DirectC and STAPL provide more detail on implementing each style.
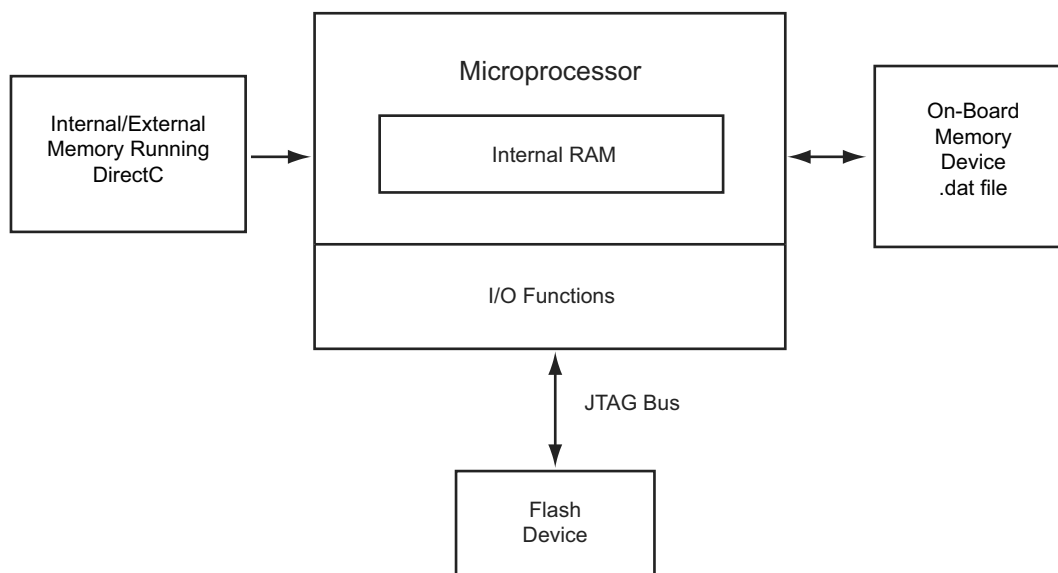

Figure 14-1 • **ISP Using Microprocessor**

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.