MultiBoot Trigger

A minimalist approach to MultiBoot in Spartan-3AN, Spartan-3A and Spartan-3A DSP.

Reference design provided for immediate use with an
XC3S700AN on the Spartan-3AN Starter Kit

Ken Chapman

Xilinx Ltd

Limitations

Limited Warranty and Disclaimer. These designs are provided to you "as is". Xilinx and its licensors make and you receive no warranties or conditions, express, implied, statutory or otherwise, and Xilinx specifically disclaims any implied warranties of merchantability, non-infringement, or fitness for a particular purpose. Xilinx does not warrant that the functions contained in these designs will meet your requirements, or that the operation of these designs will be uninterrupted or error free, or that defects in the Designs will be corrected. Furthermore, Xilinx does not warrant or make any representations regarding use or the results of the use of the designs in terms of correctness, accuracy, reliability, or otherwise.

Limitation of Liability. In no event will Xilinx or its licensors be liable for any loss of data, lost profits, cost or procurement of substitute goods or services, or for any special, incidental, consequential, or indirect damages arising from the use or operation of the designs or accompanying documentation, however caused and on any theory of liability. This limitation will apply even if Xilinx has been advised of the possibility of such damage. This limitation shall apply not-withstanding the failure of the essential purpose of any limited remedies herein.

This design module is **not** supported by general Xilinx Technical support as an official Xilinx Product. Please refer any issues initially to the provider of the module.

Any problems or items felt of value in the continued improvement of this reference design would be gratefully received by the author.

Ken Chapman
Senior Staff Engineer
Spartan Applications Specialist
email: chapman@xilinx.com

ΣXILINX®

Design Overview

MultiBoot is the ability each Spartan-3 Generation device has to reconfigure itself when desired to perform a different set of functions. For example, a communications radio would be able to replace the logic only used during transmission with that only required for receiving and vice versa with the benefit of achieving all the functionality required by the transceiver in a smaller device consuming less power. Alternatively, the Spartan device could remotely upload an upgrade to its current design and then MultiBoot to that revised configuration image.

The Spartan-3A devices, comprising of the Sparrtan-3A, Spartan-3AN and Spartan-3A DSP, offer significant flexibility when invoking MultiBoot reconfigurations. However, this reference design provides an optimised macro to demonstrate how to implement the most fundamental scheme. This will be of particular interest to users of the Spartan-3AN and hence this reference design has been provided primarily for the Spartan-3AN Starter Kit. Anyone looking for greater flexibility is advised to study the more advanced 'MultiBoot Control' reference design provided for the Spartan-3A Starter Kit.

Each Spartan-3AN device is the combination of a Spartan-3A FPGA and an SPI Flash memory. The principle purpose of the internal Flash memory is to hold the configuration image for the FPGA which is loaded automatically by the FPGA after power is applied. The exciting thing is that the initial configuration image occupies less than half of the Flash memory leaving between 78k and 1531k bytes of Flash memory for you to use in your own application. Although you may have data uses for that flexible non-volatile memory, the potential exists for every Spartan-3An devices to hold a second complete configuration image allowing the complete FPGA design to be changed and yet still have some non-volatile memory remaining for user data if required.

In this reference design you will see how to program a Spartan-3AN device with two configuration images and how to make a designs that can switch between these two images automatically. The 'multiboot_trigger' macro provided can then be used directly in your own Spartan-3A, 3AN and 3ADSP designs.

This reference contains two designs 'A' and 'B'. Both are extremely simple circuits which drive the 8 LEDs on the Starter Kit indicating when MultiBoot reconfiguration will take place at 8 second intervals. Obviously your own designs will implement significantly more functions but even these simple designs demonstrate the MulitiBoot concept.
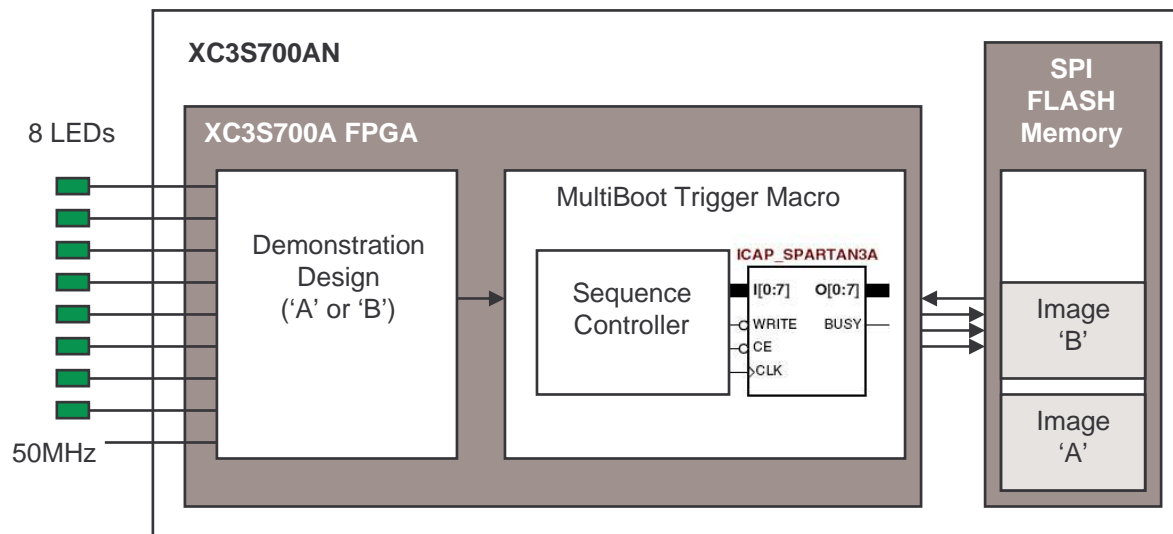
In this reference design you will be shown how to program the internal Flash with two configuration images using iMPACT.

XILINX®

Board Setup

+5v supply
Don't forget to switch on!

'PROG_B' press button
(will always force design
'A' to be reloaded).

USB cable.
Cable plus devices on board
provide essentially the same
functionality as a 'Platform
Cable USB' which is used in
conjunction with iMPACT
software to program the
Spartan-3AN with MultiBoot
configuration images.

Configuration
'DONE' LED

Initialisation
'INIT' LED

Spartan
XC3S700AN

** VERY IMPORTANT **
The Spartan-3AN device must
be of a production quality for
MultiBoot operation. The last
line of the device markings
should be '4C' or similar.
Devices marked '4CES'
indicate early engineering
samples which do not
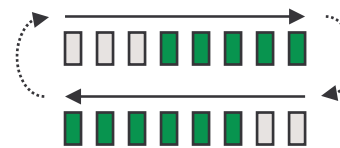MultiBoot as described.

Set to
'ISF'

Note that MODE selection
must be to the internal FLASH
memory. Setting MODE pins
to JTAG is not necessary and
doing so would prevent
MultiBoot from working.

The 'A' and 'B' reference designs drive the LED's turning them off
over a period of 8 seconds. MultiBoot occurs as the last LED turns off.

'A' LEDs turn off from left to right.

'B' LEDs turn off from right to left.

Programming the XC3S700AN

Later we will look at the designs and preparation of the programming file but for now you can take the provided '**multiboot_designs_a_and_b.mcs**' and program the XC3S700AN immediately with the complete MultiBoot demonstration. This provides you with a known working reference which you can then attempt to replicate using the source files provided. It highly recommended that you do this before embarking on your own unique designs.

Obviously you need to have the Xilinx ISE tools including iMPACT installed on your PC before proceeding further. This reference design is *not* recommended for the novice user. Please take time to become familiar with the board and the ISE tools before investigating MultiBoot further.

The really easy way this time!

Just to make the set up really easy for this particular demonstration using the Spartan-3AN Starter Kit, a batch file has been provided which will run iMPACT and instruct it to erase and program the XC3S700AN on your board.
Set up the board as shown on the previous page.
Unzip all the files provided in this reference design into a folder of your choice.
Locate and double click on '**program_xc3s700an_multiboot_designs_a_and_b.bat**'.
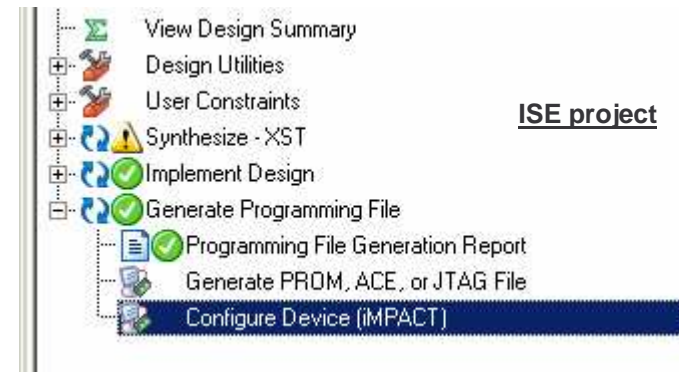You can expect this process to take somewhere between one and two minutes to complete because it involves erasing and programming of the internal Flash memory. A DOS window will provide an indication of progress and the DONE LED will be illuminated on the board during the programming procedure. Once the device is programmed the MultiBoot demonstration design should start to work automatically (if not, check the MODE pin settings again).

The normal way!

Normally when using iMPACT you will control iMPACT via the GUI displays. You can start iMPACT from any ISE project or from the start menu on your PC.

PC Start menu

Start -> All Programs -> Xilinx ISE -> Accessories -> iMPACT.

ISE project

- View Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
  - Programming File Generation Report
  - Generate PROM, ACE, or JTAG File
  - Configure Device (iMPACT)

Regardless of how you start iMPACT, first set up the board as shown on the previous page. Also unzip all the files provided in this reference design into a folder of your choice so that you know where to locate the **multiboot_designs_a_and_b.mcs** file. The next three pages walk you through the set up of iMPACT to program the XC3S700AN on your board. Hopefully you are already familiar with using iMPACT but the slight differences associated with programming the Spartan-3AN internal Flash memory may be new to you.

ΣXILINX®

Programming the XC3S700AN

These pages show iMPACT screen shots indicating the steps required to assign and program an Spartan-3A device with an MCS file.
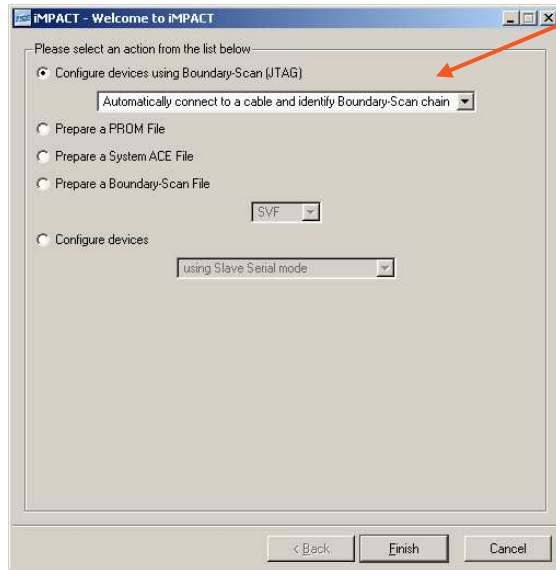
Select Boundary Scan (JTAG) with the 'Automatically connect' option and then 'Finish'

Display should show the two BSCAN capable devices on the board and then prompt you for the configuration file for the xc3s700an. (if not, right click on the device and select 'Assign New Configuration File...' from the options).
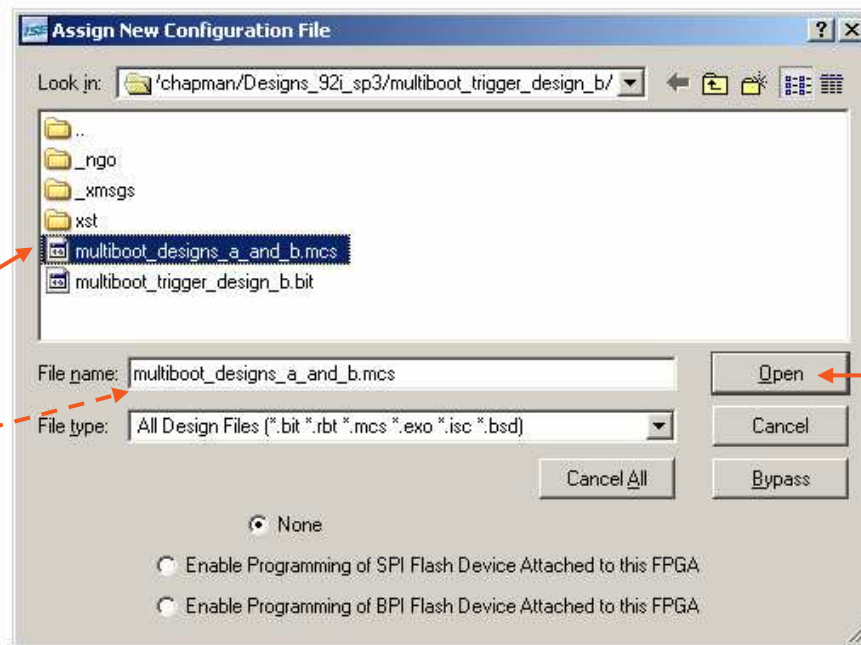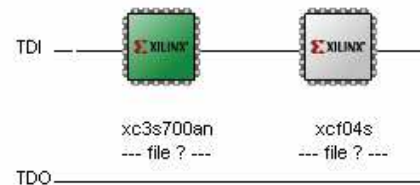
Navigate to the folder containing the 'multiboot_designs_a_and_b.mcs' file provided with the reference design and select it.

Then click 'Open'.

XILINX®

Programming the XC3S700AN
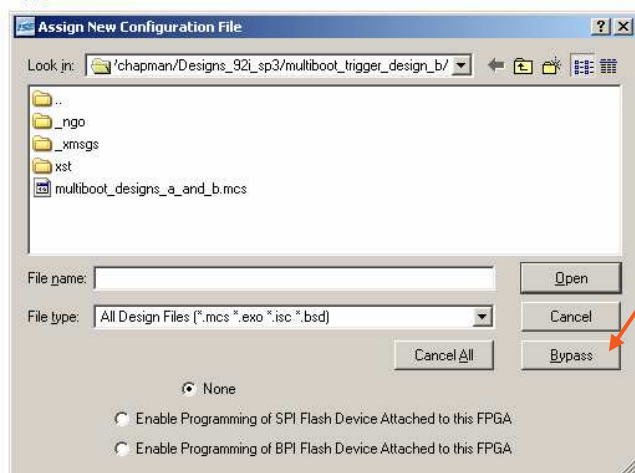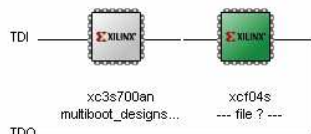
Then when prompted for a configuration image for the xcf04s device select 'Bypass'.

The chain should now look complete with the all important MCS file assigned to the xc3s700an ready for programming.
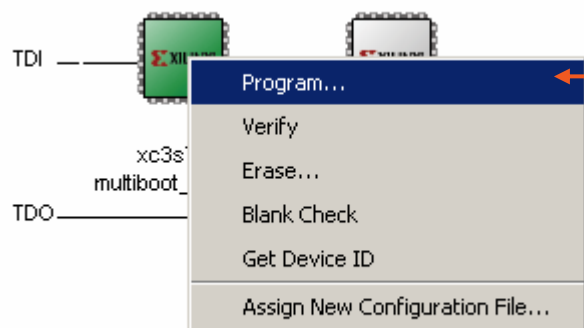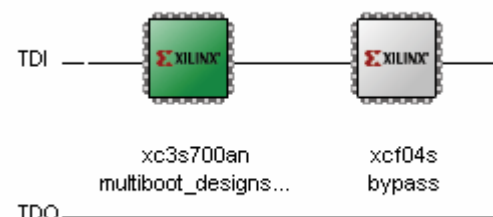
Right click on the xc3s700an device and select 'Program…'.

Programming the XC3S700AN
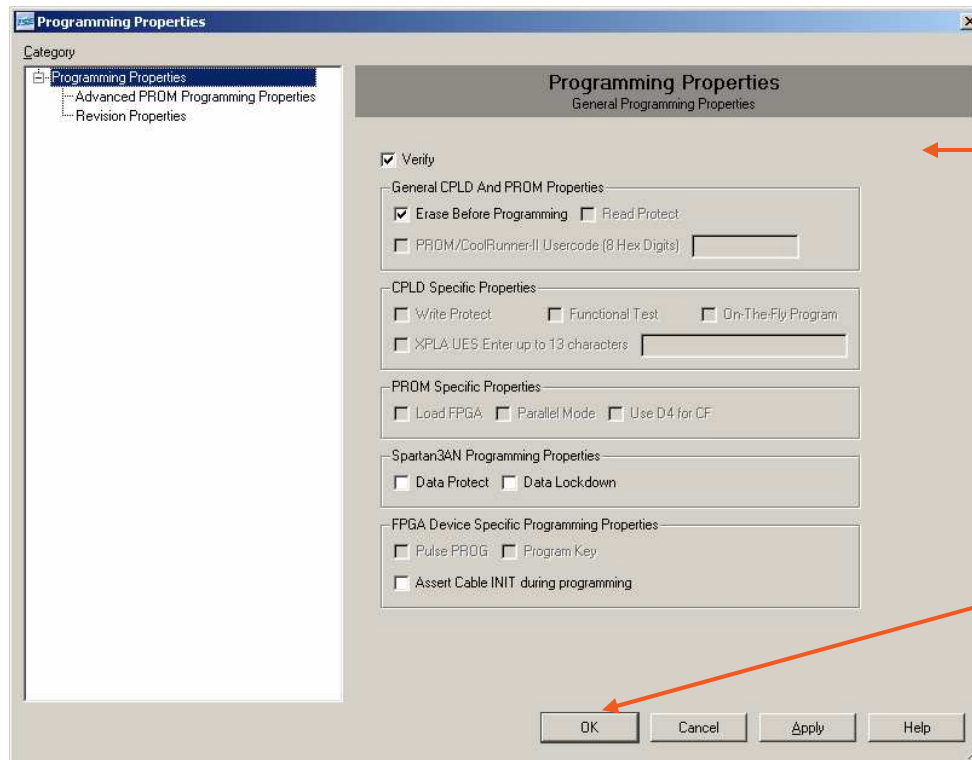
'Verify' increases the time the programming procedure will take but it is reassuring!

'OK' to start programming.

You can expect the erase, program and verify process to take approximately two minutes to complete. The DONE LED will be illuminated on the board during this time. Once the device is programmed the MultiBoot demonstration design should start to work automatically (if not, check the MODE pin settings again).
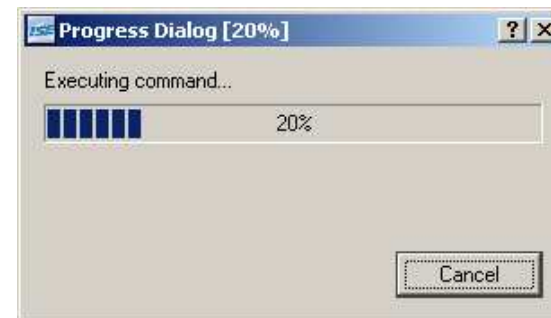
What is the board doing?

With the Spartan-3AN device programmed, your board should be demonstrating automatic reconfiguration of the Spartan-3AN every 8 seconds.

Power Applied ⟹ Design 'A'

LD7 ... LD0

+ 1 Second
+ 2 Seconds
+ 3 Seconds
+ 4 Seconds
+ 5 Seconds
+ 6 Seconds
+ 7 Seconds
+ 8 Seconds

MultiBoot

Design 'B'

+ 1 Second
+ 2 Seconds
+ 3 Seconds
+ 4 Seconds
+ 5 Seconds
+ 6 Seconds
+ 7 Seconds
+ 8 Seconds

MultiBoot

When power is applied to the board (or the PROG_B button is pressed), the Spartan-3AN device configures from its internal Flash memory starting at address zero. This results in reference design 'A' being loaded and starting to operate.

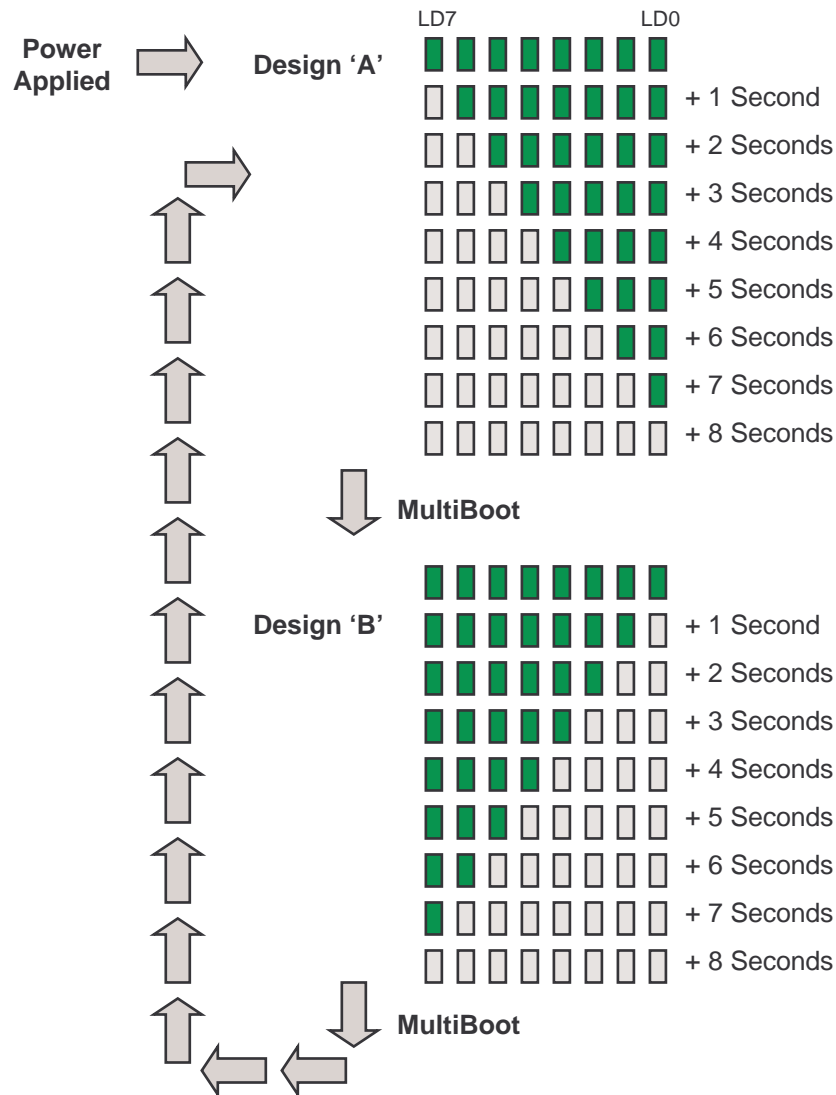Design 'A' initially turns on all LEDs and then over a period of 8 seconds (determined by dividing the 50MHz oscillator) turns one LED off at a time as shown. As the last LED (LD0) is turned off, design 'A' triggers a MulitiBoot reconfiguration from the internal Flash memory starting at address '0C0000'.

INIT   DONE

Looking *very* carefully you should see the INIT LED flash as the device is first cleared. A rather more definite blink off of the DONE LED is the ~75ms the Spartan-3AN takes to completely reconfigure with reference design 'B' .

Reference design 'B' is very similar to design 'A' but the LEDs are tuned off in the opposite direction over the next 8 seconds. Although the difference is small, it should be appreciated that the difference is a result of a different configuration of the hardware circuits and representative of what could be significant differences in a much larger practical application.
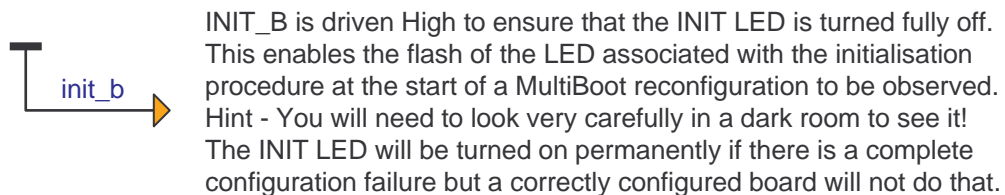
INIT   DONE

As the last LED (LD7) is turned off, design 'B' triggers a MulitiBoot reconfiguration back to a design 'A' located at address zero in the internal Flash.

XILINX®

Design 'A'

'multiboot_trigger_design_a.vhd'

Although shown here for completeness, the precise details of design 'A' are not important when understanding MultiBoot; it is mainly a simple circuit driving the LEDs. Design 'B' is not illustrated because the differences are small and again irrelevant to MultiBoot. However, the connection and use of the 'multiboot_trigger.vhd' macro in both designs will be of value when using the macro to initiate reconfiguration in your own designs.

INIT_B is driven High to ensure that the INIT LED is turned fully off. This enables the flash of the LED associated with the initialisation procedure at the start of a MultiBoot reconfiguration to be observed. Hint - You will need to look very carefully in a dark room to see it! The INIT LED will be turned on permanently if there is a complete configuration failure but a correctly configured board will not do that.

The 50MHz clock is divided by 50,000,000 to provide pulses at one second intervals which are used to enable an 8-bit shift register. The shift register is initialised to all '**1**'s and a logic '0' is shifted down the chain (up the chain in design 'B') such that over a period of 8 seconds the corresponding LED's connected to the shift register are turned off. As the last bit of the shift register goes Low, a High level is provided to the 'trigger' input of the 'multiboot_trigger' macro and a MultiBoot reconfiguration is initiated.

XILINX
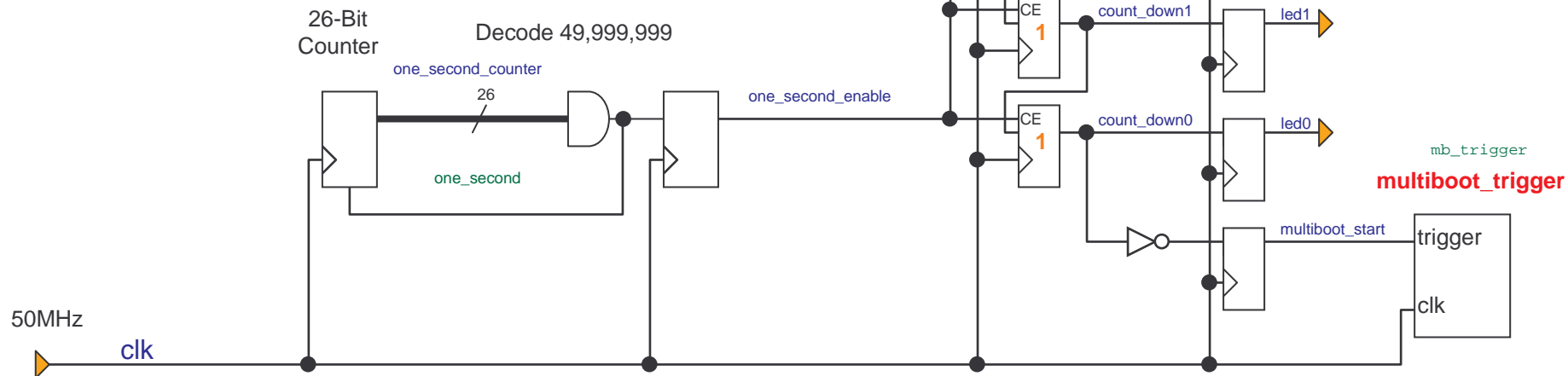
MultiBoot Trigger Circuit Diagram

This circuit occupies just 10 slices due to the efficiency of the SRL16E mode of the look up tables.

The 'i' and 'o' ports of the ICAP primitive are defined with bit '0' as the most significant bit and bit '7' as the least significant bit. To make the design more natural the bit order is reversed just prior to connection.

12×8-bit registers = 96 flip-flops.
Correct synthesis results in just 4 slices containing 8 SRL16E shift register look up tables and 8 flip-flops.
Hexadecimal values represent the **initial value** for each register loaded as part of device configuration.

config_port
icap_spartan3A

| 00 | 00 | 00 | 20 | 0E | 00 | A1 | 30 | 99 | AA | 00 | 00 |

icap_byte11 icap_byte10 icap_byte9 icap_byte8 icap_byte7 icap_byte6 icap_byte5 icap_byte4 icap_byte3 icap_byte2 icap_byte1 icap_byte0

icap_write_en[11:0]   12 flip-flops ccorrectly synthesised result in 1 SRL16E and 1 flip-flop. **Initial value** for each flip-flop.

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

icap_write
icap_ce
icap_clk

write
ce   busy
clk

Note active Low write and clock enable controls.

trigger

clk

icap_prepare   icap_clk   icap_next_en

MultiBoot Trigger 11

XILINX®

MultiBoot Command Sequence

As shown in this table from UG332, the most simple way to initiate a MultiBoot reconfiguration is to write the 8-byte sequence AA,99,30,A1,00,0E,20,00 to the ICAP port (ICAP_SPARTAN3A primitive). The address of the configuration image must be defined when generating the configuration image with Bitgen (see later).

The 'multiboot_trigger.vhd' macro (diagram on previous page) writes this sequence to the ICAP port as indicated in the waveforms below.
There are three points of interest which may not be immediately obvious:-

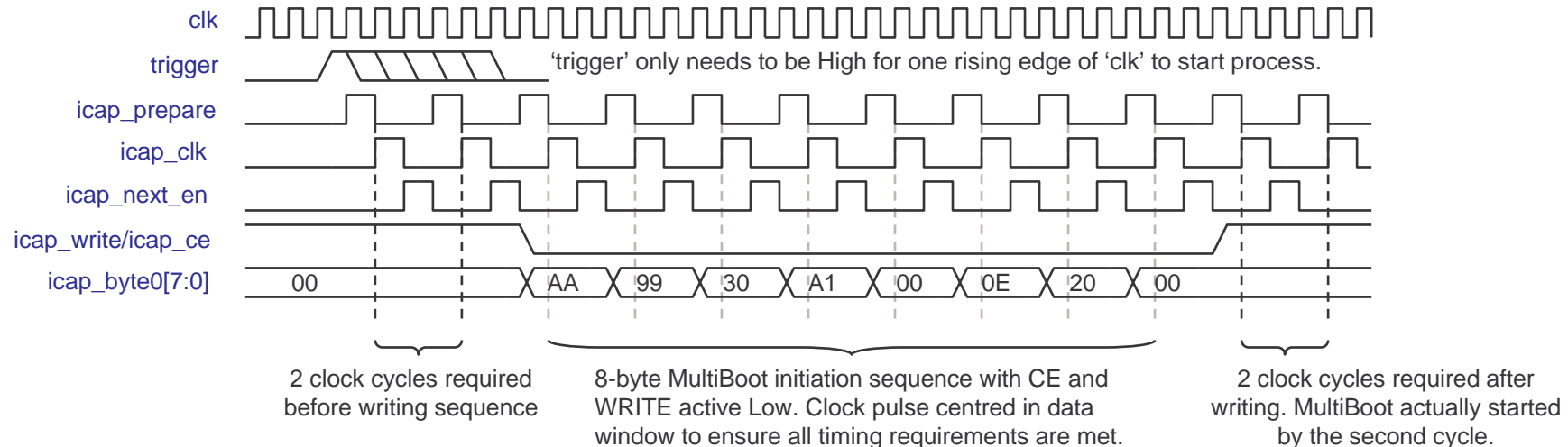Two additional clock cycles are required before and after writing of the sequence.

A MultiBoot reconfiguration is *always* from a Flash memory. However, JTAG and iMPACT will often be used to configure the device during design development and in such cases JTAG and ICAP effectively compete for access of the configuration circuits. Generating and applying *only* the required number of clock pulses to ICAP when initiating MultiBoot the operation will be reliable even when using JTAG.

Spartan-3 Generation Configuration User Guide
UG332 (v1.3) November 21, 2007

Table 14-2: **Command Sequence to Initiate MultiBoot from a Preloaded Address**

CLK Cycle	Command	High or Low Byte	D0	D1	D2	D3	D4	D5	D6	D7	Hex
1	SYNC WORD	High	1	0	1	0	1	0	1	0	0xAA
2		Low	1	0	0	1	1	0	0	1	0x99
3	Type 1 Write CMD (1 Word)	High	0	0	1	1	0	0	0	0	0x30
4		Low	1	0	1	0	0	0	0	1	0xA1
5	REBOOT Command	High	0	0	0	0	0	0	0	0	0x00
6		Low	0	0	0	0	1	1	1	0	0x0E
7	No Op	High	0	0	1	0	0	0	0	0	0x20
8		Low	0	0	0	0	0	0	0	0	0x00

If, for any reason the sequence does not trigger a MulitBoot the first time, this macro repeats the trigger sequence continuously. It should be noted that the device must fully complete its start up sequence before ICAP becomes available. When configuring in master mode this will happen very quickly but when configuring using iMPACT and JTAG then start up can take several milliseconds to complete. So although a design should provide adequate time before triggering MultiBoot, this macro will trigger reconfiguration at the first possible opportunity due to its repeating sequence.

2 clock cycles required before writing sequence

8-byte MultiBoot initiation sequence with CE and WRITE active Low. Clock pulse centred in data window to ensure all timing requirements are met.

2 clock cycles required after writing. MultiBoot actually started by the second cycle.

XILINX®

Preparing Configuration (BIT) Files

It is vital for correct MulitiBoot reconfiguration that each design knows the start address at which the next configuration is located in the Flash memory. Whilst it is possible to specify that address by writing an extended command sequence to the ICAP port, the minimalist scheme implemented by the 'multiboot_trigger' macro only commands the MultiBoot reconfiguration to begin and therefore the address has to be specified an alternative way.

So this technique requires the configuration image (BIT file) to be generated with the start of configuration address embedded within it. This is achieved using a special option when executing the Bitgen utility. The following pages show how to specify the address but first we need to know what address to specify for each design in a MultiBoot system. Note that the Bitgen option results in a fixed address being associated with each design and therefore the MultiBoot will always be to the same location from a given design.

Hint – In cases where an active design will make a decision concerning which one of several images should be loaded next, then the address needs to be specified locally in order to make that choice. This more advanced scheme requiring extended command sequences is explored in the 'MultiBoot Control' reference design.

Defining the correct address for each design.

Regardless of the type of Flash memory used, the initial design read by a Spartan-3A device following power up (or PROG_B) is that located at address zero. The address is a 24-bit value and therefore represented by the 6-digit hexadecimal value '000000'. In this reference design, the initial image contains design 'A'.

Design 'A' then needs to know the location of design 'B'. In theory, the second configuration image this can be located anywhere in the providing adequate space in the Flash memory. However, there are recommendations, and in the case of the Spartan-3AN devices, these become almost definite rules because deviation from them will not correlate with the files generated in ISE or the programming performed by iMPACT. So for Spartan-3AN, look at the memory allocation tables for each device shown in the Spartan-3AN FPGA In-System Flash User Guide (UG333).

The memory allocation table for the XC3S700AN shows that the second configuration image (design 'B' in this case) starts at the 24-bit address '0C0000'. Therefore design 'A' must be defined with a next configuration address of '0C0000'. Since we then want design 'B' to MultiBoot back to design 'A', design 'B' must be defined with a next configuration address of '000000'.
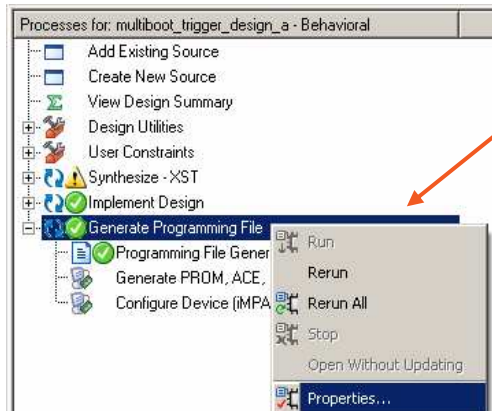
DESIGN 'A'
Start = 000000
Boot to 0C0000

DESIGN 'B'
Start = 0C0000
Boot to 000000

Table 2-6: XC3S700AN In-System Flash Memory Allocation

Allocation	Sector		Default Addressing Mode		Optional Power-of-2 Addressing Mode	
		Page	Address	Page	Address	
Bitstream 'A'	0	a	0	0x00_0000	0	0x00_0000
		b	...	0x00_1000	...	0x00_0800
	..	...	. . .	...	. . .	
	5	1,293	0x0A_1A00	1,334	0x05_3600	
First available user data space (page aligned)	5	1,294	0x0A_1C00	1,335	0x05_3700
		...	. . .	...	. . .	
		1,535	0x0B_FE00	1,535	0x05_FF00	
2nd MultiBoot Bitstream, or available for user data space 'B'	6	1,536	0x0C_0000	1,536	0x06_0000
	...	...	. . .	...	. . .	
	11	2,829	0x16_1A00	2,890	0x0B_4A00	
Second available user data space (page aligned)	11	2,830	0x16_1C00	2,871	0x0B_4B00
		...	. . .	...	. . .	
		3,071	0x17_FE00	3,071	0x0B_FF00	
User data space (sector aligned)	12	3,072	0x18_0000	3,072	0x0C_0000
		...	. . .	...	. . .	
		3,327	0x19_FE00	3,327	0x0C_FF00	
	13	3,328	0x1A_0000	3,328	0x0D_0000	
		...	. . .	...	. . .	
		3,583	0x1B_FE00	3,583	0x0D_FF00	
	14	3,584	0x1C_0000	3,584	0x0E_0000	
		...	. . .	...	. . .	
		3,839	0x1D_FE00	3,839	0x0E_FF00	
	15	3,840	0x1E_0000	3,840	0x0F_0000	
		...	. . .	...	. . .	
		4,095	0x1F_FE00	4,095	0x0F_FF00	

**XILINX®**

Preparing Configuration (BIT) Files

Having successfully implemented a design (in this case design 'A' is being prepared), right click on 'Generate Programming File' In the ISE Project Navigator window and select 'Properties'.

This will open the following dialogue box….

The 'General Options' menu should be the first shown, but if not, select it by clicking here.

Ensure that the 'CRC' box is checked as shown as this helps MultiBoot to continue operating even if a sporadic error should occur during configuration.
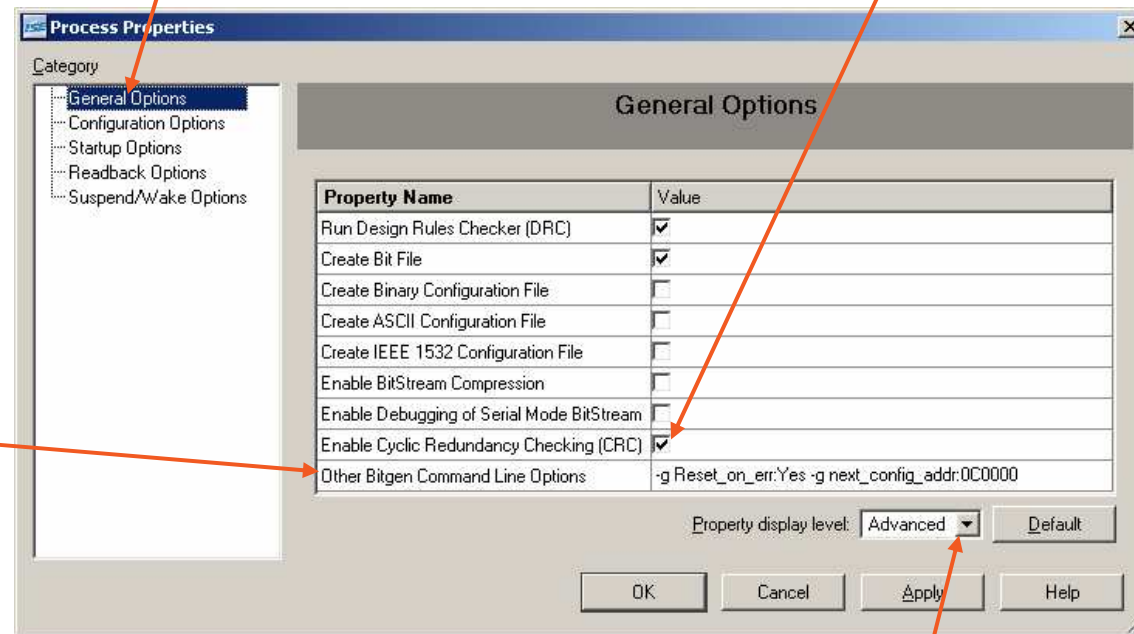
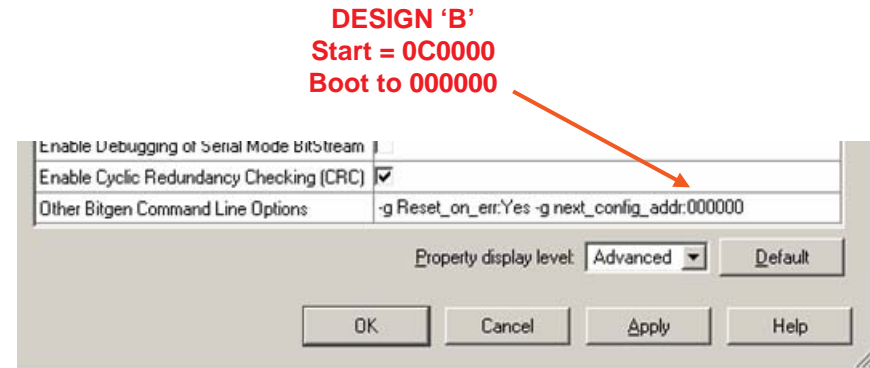In the 'Other Bitgen Command Line Options' window enter the text string shown.
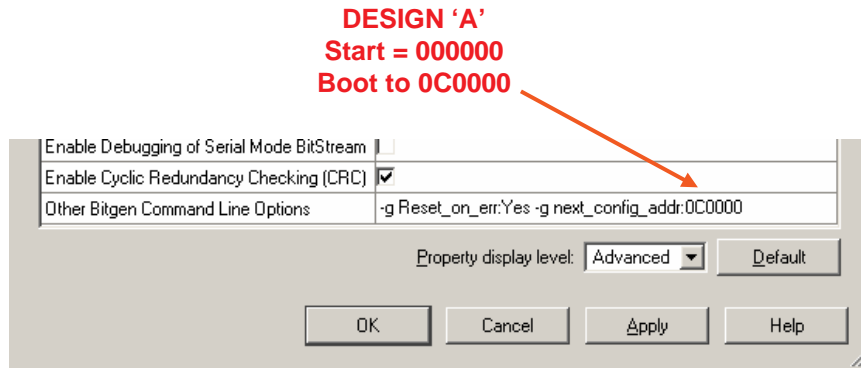
The first option again helps MultiBoot to continue operating should there be any errors during configuration. The second option states the 24-bit address for the next configuration.

-g Reset_on_err:Yes -g next_config_addr:0C0000

<u>Hint</u> – If you do not see the 'Other Bitgen Command Line Options' window then select the 'Advanced' option here.

XILINX®

Preparing Configuration (BIT) Files

Although each design contains the identical 'multiboot_trigger' macro, the configuration image (BIT file) for each design must be prepared to specify a different next configuration address corresponding with the memory allocation table from UG333.

DESIGN 'A'
Start = 000000
Boot to 0C0000

DESIGN 'B'
Start = 0C0000
Boot to 000000

Enable Debugging of Serial Mode BitStream	
Enable Cyclic Redundancy Checking (CRC)	✔
Other Bitgen Command Line Options	-g Reset_on_err:Yes -g next_config_addr:0C0000

Property display level: Advanced ▾   Default

OK   Cancel   Apply   Help

Enable Debugging of Serial Mode BitStream	
Enable Cyclic Redundancy Checking (CRC)	✔
Other Bitgen Command Line Options	-g Reset_on_err:Yes -g next_config_addr:000000

Property display level: Advanced ▾   Default

OK   Cancel   Apply   Help

After the BIT file is generated, the Bitgen report file can be reviewed to confirm that the required address has been included. You should also see the options associated with automatic error recovery and that the ICAP port has been enabled (this should happen automatically because of the ICAP primitive being included in the designs). Below are abstracts from the Bitgen report files for each designs showing the items most relevant to MultiBoot operations.

multiboot_trigger_design_a.bgn

```
+----------------------+----------------------+
| CRC                  | Enable**             |
+----------------------+----------------------+
+----------------------+----------------------+
| Reset_on_err         | Yes                  |
+----------------------+----------------------+
+----------------------+----------------------+
| next_config_addr     | 0C0000               |
+----------------------+----------------------+
+----------------------+----------------------+
| ICAP_Enable          | Yes                  |
+----------------------+----------------------+
```

multiboot_trigger_design_b.bgn

```
+----------------------+----------------------+
| CRC                  | Enable**             |
+----------------------+----------------------+
+----------------------+----------------------+
| Reset_on_err         | Yes                  |
+----------------------+----------------------+
+----------------------+----------------------+
| next_config_addr     | 000000               |
+----------------------+----------------------+
+----------------------+----------------------+
| ICAP_Enable          | Yes                  |
+----------------------+----------------------+
```

XILINX®

Optimised MultiBoot

Since we are going to be reconfiguring the device as part of eth system operation, it makes sense to minimise the time taken to reconfigure. This can easily be achieved by specifying the highest configuration rate clock supported by the internal Flash memory of the Spartan-3AN device. So also make the adjustments shown here when generating BIT files.
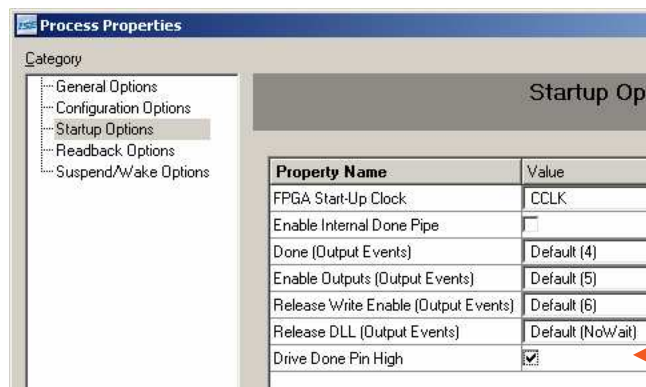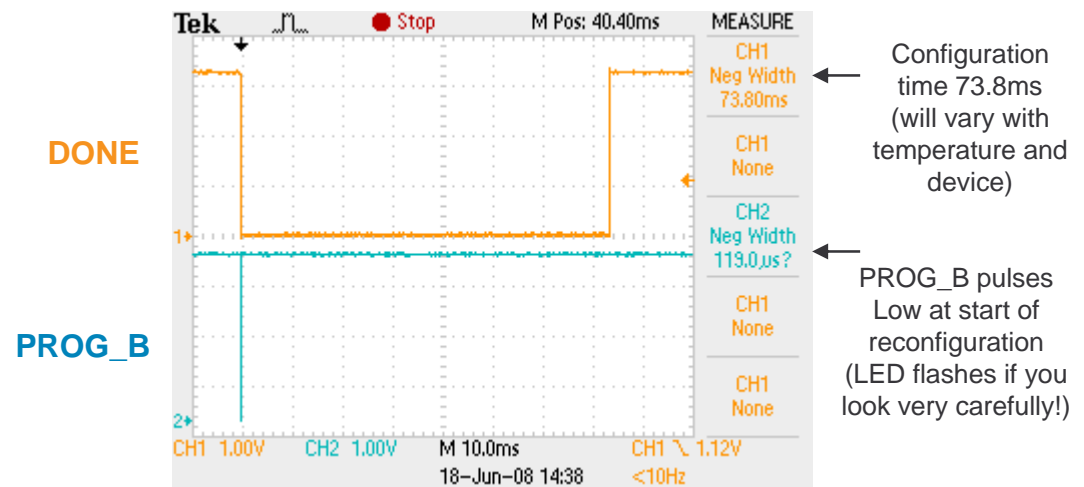
'Configuration Options' menu
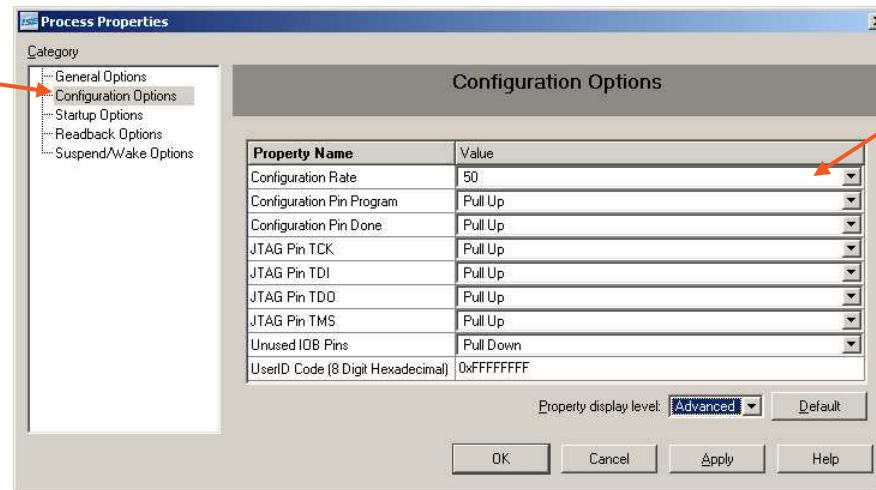
Increase to 50MHz for fastest configuration rate supported by a Spartan-3AN device.

Note that the VS[2:0] pins on the Spartan-3AN device must all be High (default when left unconnected) as these define that the 'fast read' command will be used when the Flash is accessed during configuration.

It is also a general recommendation that the 'Drive Done Pin High' option be used with Spartan-3AN devices and this can be enabled under the 'Startup Options' menu.

This oscilloscope trace captures the MultiBoot reconfiguration of the XC3S700AN using these configuration settings.

DONE

PROG_B

Configuration time 73.8ms (will vary with temperature and device)

PROG_B pulses Low at start of reconfiguration (LED flashes if you look very carefully!)

XILINX®

Preparing the Programming MCS File

Using iMPACT it is possible to directly program a Spartan-3AN with a configuration image BIT file. However, that would only program a single image at address zero whereas for a MultiBoot system we need to program two configuration images in different locations. The way to provide iMPACT with both configuration images and specify their order is by first creating a PROM definition file (MCS). The following pages show how to prepare this MCS file using the two previously prepared BIT files.
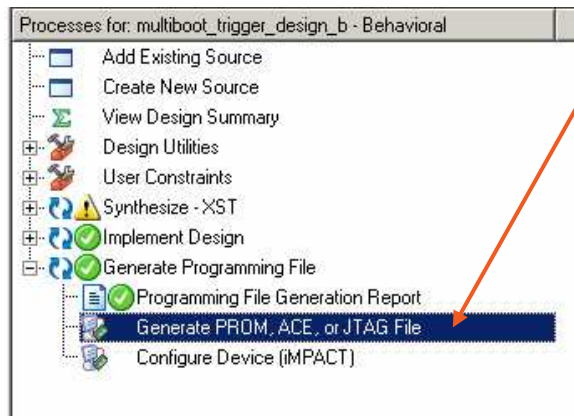
In the ISE Project Navigator select the 'Generate PROM, ACE, or JTAG File' option. (Alternatively you can invoke this mode from iMPACT if it is already open).
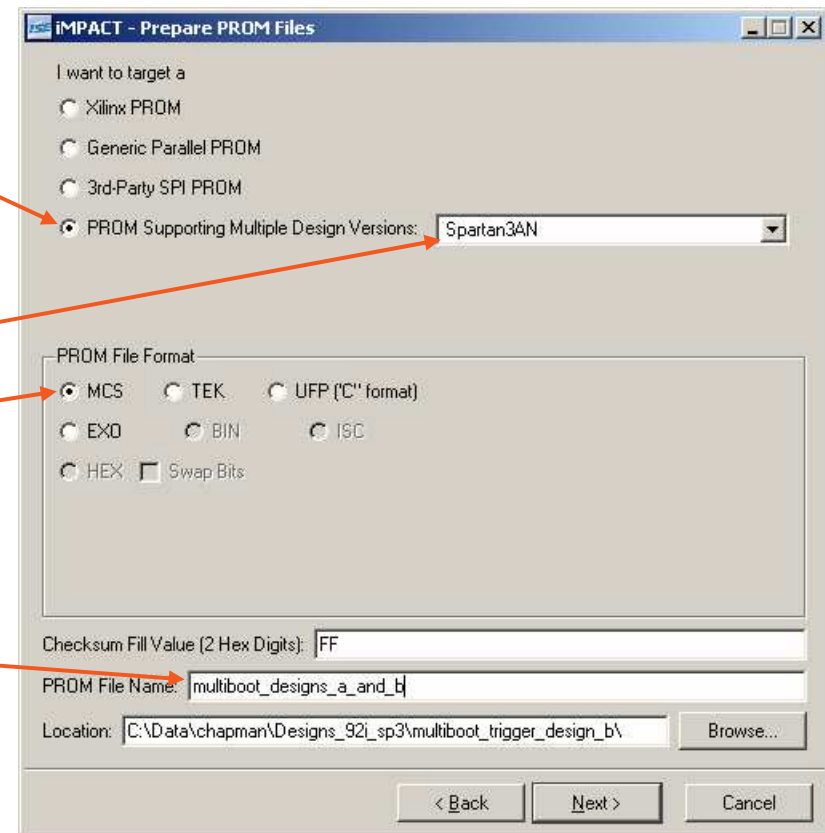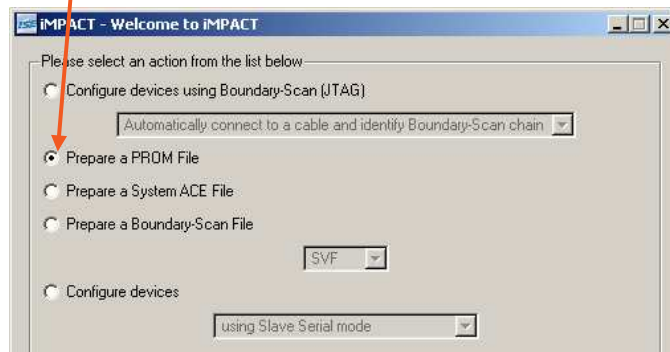
Select this option for a MultiBoot programming file.
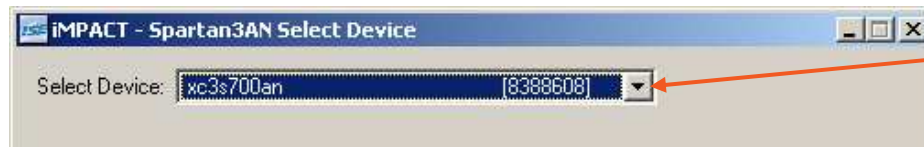
Select Spartan-3AN from the drop down.

MCS is default

'Prepare a PROM File'

Provide a suitable name for the output file

Preparing the Programming MCS File

Select the appropriate Spartan-3AN device from the drop down. This effectively defines the memory allocation table for the Flash that is shown in the memory allocation tables contained in UG333.

There always has to be an initial image at address zero. This is where we will be putting design 'A' in the next stage.

Check this box so that iMPACT knows that you want to include a second MultiBoot configuration image. This is where we will be putting design 'B' in the next stage.

A confirmation screen is then displayed. – click 'Finish' to continue


As discussed earlier, iMPACT only works with a pre-defined and fixed memory allocation table. So the address of the second configuration is displayed automatically when using a Spartan-3AN device.

Is that the right address?

This screen shot is indicating that the start address of the second configuration image is 405504. This is a decimal number unlike the 6-digit hexadecimal values we needed to specify when preparing the BIT files or seen when looking at the memory allocation table in UG333. However, when we convert this value to hexadecimal we get '063000' which is rather different to the '0C0000' we would expect to see. Unfortunately this is where you will have to ignore what you see displayed because iMPACT really will use the address '0C0000' from the memory allocation table.

The underlying cause of confusion is that the Flash memory has a non-linear address map formed of memory pages that contain a non-power-of-2 number of bytes and the calculator inside iMPACT is based on a traditional linear memory. The 'use Power of 2 for Start Address' check box is also related to this but this topic is beyond the scope of this reference design. This reference design is a minimalist approach to MulitiBoot and providing you run with the settings shown here then everything works as expected.
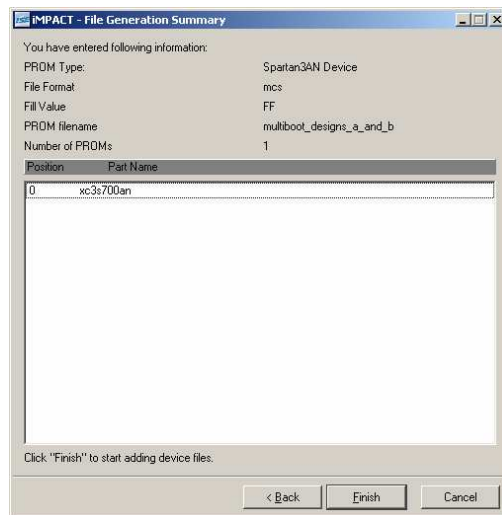
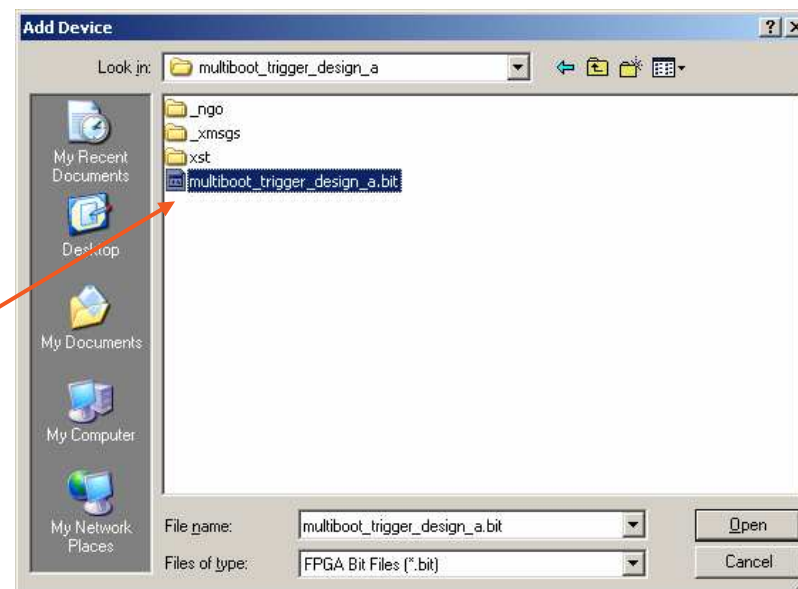**XILINX®**

Preparing the Programming MCS File

At this point you are prompted to specify the FIRST configuration image which will be located at address zero.

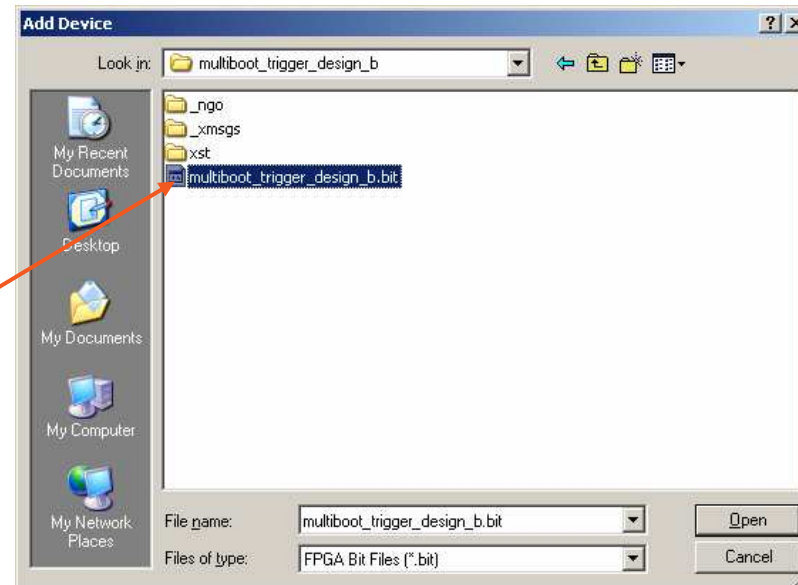So navigate to the BIT file for design 'A', select it and then click on 'Open'.

Then you are prompted to specify the SECOND configuration image which will be located at address 0C0000.

So navigate to the BIT file for design 'B', select it and then click on 'Open'.

Preparing the Programming MCS File

Add Device

You have completed the device file entry.
Click 'Ok' to continue

OK

The contents of the programming file are now specified but we still need to generate the actual MCS file.

Click the 'Generate File' option.

(Alternatively right click in the main window and select the 'Generate File' option from the pop-up).
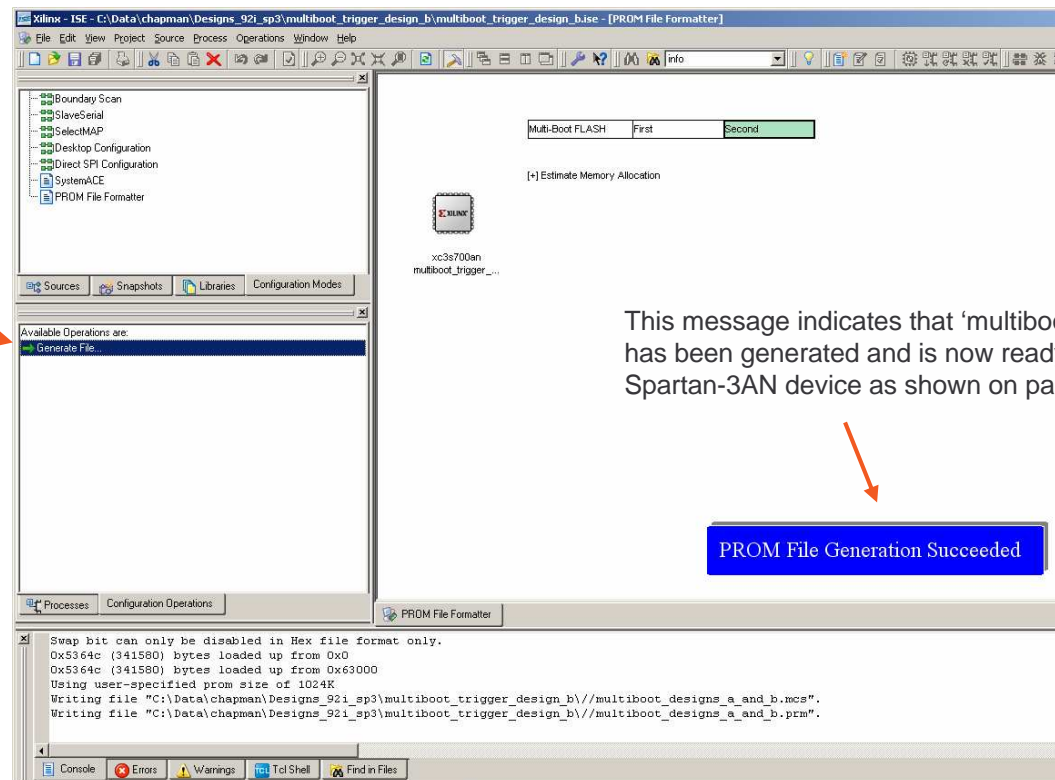
This message indicates that 'multiboot_designs_a_and_b.mcs' has been generated and is now ready to be programmed into the Spartan-3AN device as shown on pages 5 to 8 of this document.

PROM File Generation Succeeded

Swap bit can only be disabled in Hex file format only.
0x5364c (341580) bytes loaded up from 0x0
0x5364c (341580) bytes loaded up from 0x63000
Using user-specified prom size of 1024K
Writing file "C:\Data\chapman\Designs_92i_sp3\multiboot_trigger_design_b\//multiboot_designs_a_and_b.mcs".
Writing file "C:\Data\chapman\Designs_92i_sp3\multiboot_trigger_design_b\//multiboot_designs_a_and_b.prm".

XILINX®

Reference Design Files

The following files are provided with this reference design.
The '**multiboot_trigger.vhd**' (or 'multiboot_trigger.v') macro can be used unmodified in your own designs.

Design 'A'

multiboot_trigger_design_a.vhd — Top level file and description of hardware for design 'A'.

multiboot_trigger_design_a.ucf — I/O constraints file for Spartan-3AN Starter Kit and timing specification for 50MHz clock.

multiboot_trigger.vhd — The MultiBoot Trigger macro including ICAP primitive

multiboot_trigger_design_a.bit — The configuration image for design 'A' with next configuration address set to 0C0000 during Bitgen

Design 'B'

multiboot_trigger_design_b.vhd — Top level file and description of hardware for design 'B'.

multiboot_trigger_design_b.ucf — I/O constraints file for Spartan-3AN Starter Kit and timing specification for 50MHz clock.

multiboot_trigger.vhd — The MultiBoot Trigger macro including ICAP primitive

multiboot_trigger_design_b.bit — The configuration image for design 'B' with next configuration address set to 000000 during Bitgen

Design 'A+B'

multiboot_designs_a_and_b.mcs — The final XC3S700AN programming file containing the configuration images of both design 'A' and design 'B'.

program_xc3s700an_multiboot_designs_a_and_b.bat — Batch file to program the XC3S700AN on the Spartan-3AN Starter kit with the MCS file above.

Verilog

multiboot_trigger.v — Verilog version of the MultiBoot Trigger macro for Verilog users Thank you to Nick Sawyer (Xilinx) for this variant.

XILINX®

MultiBoot Trigger Macro

As shown in the 'A' and 'B' reference designs, the 'multiboot_trigger.vhd' macro is easy to include in your own designs. The macro includes the ICAP port and writes the required command sequence to initiate a MultiBoot reconfiguration using only 10 logic slices.

Component declaration of the MultiBoot Trigger macro

```
component multiboot_trigger
  Port (   trigger : in std_logic;
              clk : in std_logic);
  end component;
```

The 'multiboot_trigger.vhd' and 'multiboot_trigger.v' macros can be used without modification in any Spartan-3A, Spartan-3AN and Spartan-3A DSP device.

Instantiation of the MultiBoot Trigger with typical signal names

```
mb_trigger: multiboot_trigger
port map(  trigger => multiboot_start,
              clk => clk);
```

Verilog Instantiation of the MultiBoot Trigger with typical signal names

```
multiboot_trigger mb_trigger (
                .trigger (multiboot_start),
                .clk     (clk_));
```

Signals

clk	INPUT	The 'clk' input will typically be connected to the main (or only) free running clock within your design. Actual frequency is not important but a clock between 5 and 100MHz is typical. The clock should be sourced from a clock buffer but it is normal practice for the main clock in any design to be automatically implemented in this way by the synthesis and ISE tools).
trigger	INPUT	The 'trigger' input should remain Low until a MultiBoot reconfiguration is required. Drive 'trigger' High for at least one rising edge of 'clk' to start the MultiBoot process. MultiBoot will actually take place on the 34th rising edge of 'clk' following the detection of 'trigger' being High.

Remember that you must set the next configuration address using the Bitgen option **-g next_config_addr:AAAAAA** where AAAAAA is the 24-bit start address of the configuration address to be loaded expressed as a 6-digit hexadecimal value.

XILINX®