HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
www.hunteng.co.uk
www.hunt-dsp.com

The HERON FPGA7 Example3

Rev 1.1 R. Williams 29-04-05

The HERON-FPGA7 is a module that has a Virtex-II Xilinx FPGA and 256Mbytes of SDRAM memory.

Most users will use the HERON-FPGA7 to provide a custom storage capability using the SDRAM to store large amounts of data. The module may also use the FPGA to process the data stored in the SDRAM.

With 256Mbytes of SDRAM memory available the HERON-FPGA7 is ideally suited for creating a large FIFO, with the memory interface of the SDRAM providing the storage for the FIFO.

With Example3 the memory is split into two parts, each 128Mbytes in size. Each half of the memory is used to create an individual FIFO. The result is two 128Mbyte FIFOs interfaced to two separate pairs of HERON input and output FIFOs.

As such Example3 is highly suitable for applications that require buffering between high speed data sources like A/D converters of IO modules, and slower non-real-time host PC interfaces such as PCI.

History

Example revision 1.0	26-09-03	Developed from Example3 for the HERON-FPGA5
Example revision 1.1	29-04-05	Removed reference to specific ISE versions

What the Bit-stream Does

Example3 (SDRAM FIFO) for the HERON-FPGA7 is supplied on the HUNT ENGINEERING CD, and web site. The FPGA source code is supplied along with a bit-stream that can be loaded directly onto the HERON-FPGA7.

The bit-stream provided is for the 3M-gate FPGA. For those users who are interested in using this example as it is to implement two large FIFOs, only the smallest FPGA is required as only a small proportion of the 3M FPGA is required to implement this function.

However, for users of the 6M-gate and 8M-gate HERON-FPGA7 the example project can be used as a starting place for creating your own SDRAM based design that includes extra functionality inside the FPGA.

Please note that because this project uses hardly any of the FPGA the routing delays when using one of the larger FPGA devices becomes significant. That is, design nets that traverse a large section of the device will have a large routing delay, and with a larger FPGA this distance means that the delay becomes more significant at the operating speeds of Example 3.

It has been found that this example when built for the –4 speed grade 6M-gate and 8M-gate devices fails the timing constraints of the SDRAM clock by between 0.5 to 1.0ns. However it should be noted that as extra logic is added between the various stages of the design, the operating speed will increase as routing delays drop between neighbouring blocks of logic. For users of the –5 or –6 speed grade devices, the example will build successfully for all sizes of device.

If you make changes to the project and re-build it you can change the functionality to be whatever you want, but if you use the supplied bit-stream you need to know what it is doing. This document describes that for you.

The standard clock soldered to “User Osc3” of the HERON-FPGA7 is 100Mhz.

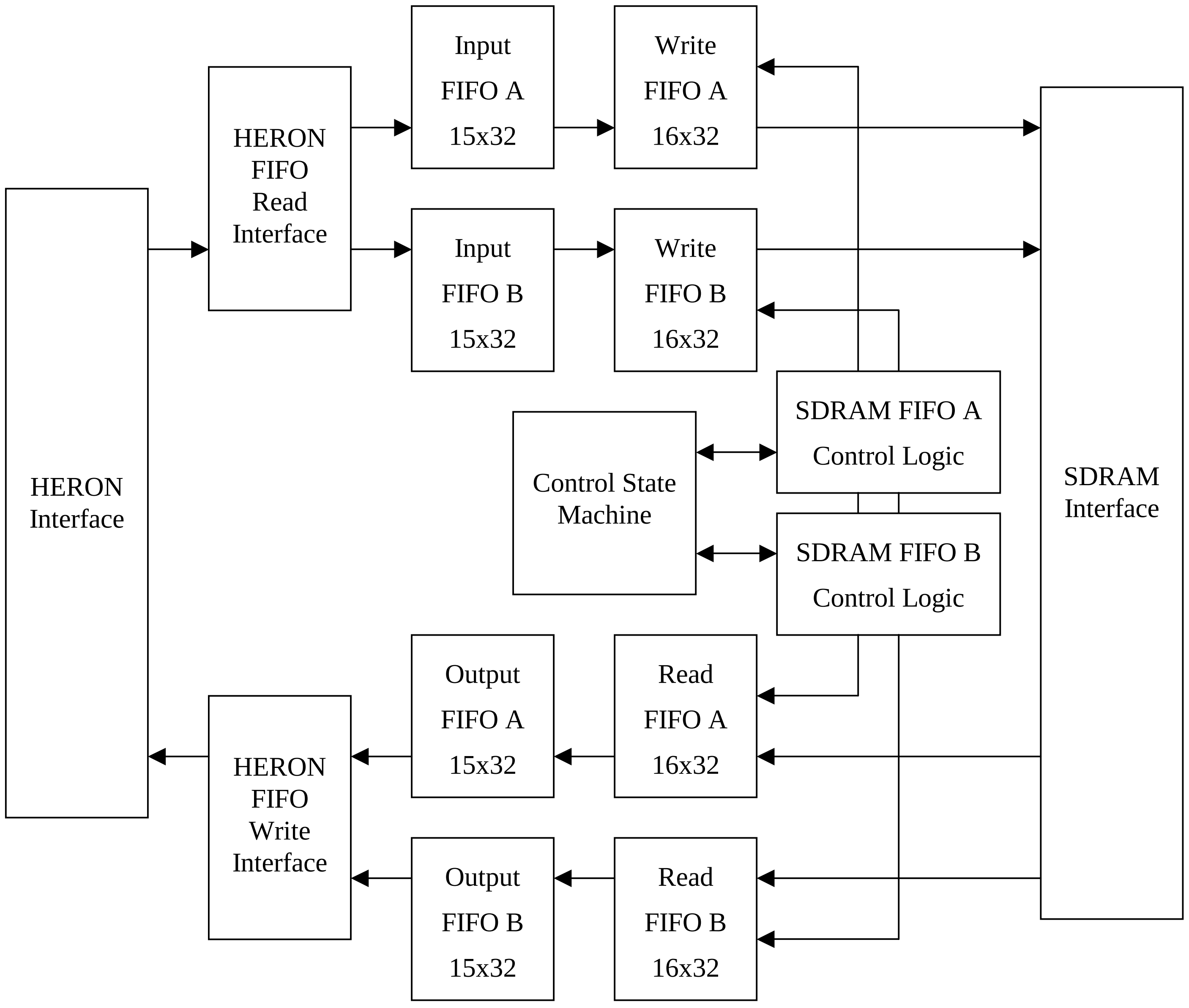
Example3 always uses this clock to drive the fundamental clock source of the SDRAM. This clock source is synthesized up to 133MHz inside the SDRAM interface component of the Hardware Interface Layer.

This clock is also used to drive the FIFO clock directly, or to drive the FIFO clock after being divided by 2.

For an HEPC8 the HERON FIFO clocks must be driven at less than 60Mhz, whereas for the HEPC9 they must be driven at between 60Mhz and 100Mhz. For this reason there are different bit-streams supplied for HEPC8 and HEPC9 as follows.

2v3000ff1152.hcb	HERON-FPGA7 fitted to HEART based carrier e.g HEPC9 (100Mhz FIFO clocks)
2v3000ff1152_pc8.hcb	HERON-FPGA7 fitted to HEPC8 (50Mhz FIFO clocks)

FUNCTIONAL BLOCK DIAGRAM



SDRAM FIFO Operation

The HERON-FPGA7 provides a 256Mbyte bank of SDRAM memory. This memory is organised as 32-bit wide memory with 64Mlocations. The SDRAM FIFO example splits this memory into two equal halves. Each half is used to create an independent FIFO of 32Mlocations by 32-bits. The first FIFO is SDRAM FIFO A, and the second FIFO is SDRAM FIFO B.

All data from HERON Input FIFO 0 is passed to SDRAM FIFO A. Data output from SDRAM FIFO A is passed to HERON Output FIFO 0. Similarly, all data from HERON Input FIFO 1 is passed to FIFO B, and data output from that FIFO is passed to HERON Output FIFO 1.

Between each HERON input FIFO and SDRAM FIFO are two small FIFOs. The first small FIFO is used to decouple the data being input at the FIFO clock frequency from the 133MHz SDRAM clock domain. This first small FIFO is a 15x32 asynchronous FIFO generated using the Core Generator.

The second small FIFO is a synchronous FIFO that directly interfaces to the SDRAM interface of the USER_AP entity.

Between each SDRAM FIFO and HERON output FIFO are again, two more small FIFOs. The first is a synchronous FIFO to directly interface to the SDRAM, and the second is the CoreGen 15x32 FIFO used to decouple the 133MHz SDRAM clock from the FIFO clock.

With four small FIFOs associated with each SDRAM FIFO there is a total FIFO depth from HERON input FIFO to HERON output FIFO of 33554494 words.

The SDRAM interface of the USER_AP entity presents both a write port and a read port. However, at any one point in time, only a read or write may be in progress. This is due to the fact that the external SDRAM devices have one common data bus for both memory reads and memory writes.

Where data is being transferred through both SDRAM FIFOs at the same time a control function is required to decide which FIFO has access to the SDRAM.

There are four possible transfers that can occur at any one time. Two memory writes for either SDRAM FIFO A or SDRAM FIFO B, and two memory reads for either FIFO. There is a single control state machine in Example3 that is used to govern this access to the SDRAM. This state machine will rotate between all read and write requests for both FIFOs in a fair manner. As the data flow for any particular FIFO slows, the next transfer is selected. To ensure fairness, there is also a time-out that limits how long one FIFO is reading or writing the memory.

Four LEDs are used to represent the state of both SDRAM FIFOs, as follows:

- LED 0 is illuminated when SDRAM FIFO A is empty
- LED 1 is illuminated when SDRAM FIFO A is full
- LED 2 is illuminated when SDRAM FIFO B is empty
- LED 3 is illuminated when SDRAM FIFO B is full

For each LED there is a small counter that ensures the LED is illuminated long enough to be seen, even if the full or empty condition lasts for only a few clock cycles.

LED 4 will always flash to indicate that the system FIFO clock FCLK_G is running.

Typical Use of the Example Bit-stream

The typical use of the example bit-stream is to provide two independent FIFOs, each over 32Mlocations in depth, and 32-bits wide. The first SDRAM FIFO reads and writes from HERON FIFO 0, and the second SDRAM FIFO reads and writes from HERON FIFO 1.

Therefore, the HERON-FPGA7 can be placed in a system with other modules and by correctly connecting FIFOs from surrounding modules to FIFO 0 and FIFO 1 of the HERON-FPGA7, two large data buffers can be inserted in the flow of data through the system.

Where are the Bit-streams and Example Program?

The bit streams for this example can be found on the HUNT ENGINEERING CD in the directory `\fpga\fpga7v1\Sdram_Fifo(ex3)`. The name of the bitstream reflects the Xilinx FPGA part number and the Carrier board type, as explained earlier in this document.

An easier way to navigate to the correct directory is to select the “Files” link next to the “SDRAM FIFO” link under the IP sections of the CD browser.

The source files for the FPGA example can be found in the ‘`src`’ sub-directory. The sources in the ‘`\fpga7v1\common`’ directory are also required. A project for the Xilinx ISE development tools is provided for you in the ‘`ISE`’ sub-directory.

FPGA Example Code

You should understand the HUNT ENGINEERING VHDL support for HERON modules before looking at this section. If you do not then please review Example1 again (the ‘Getting Started’ example for FPGA modules).

As you are expected to be already familiar with Example1, this section only discusses the points that are unique to example3.

In the USER-AP entity for Example3, contained in the file ‘user_ap3.vhd’, there are options that allow you to select the FIFO clock frequency. Just as with Example1, you can select if the 100Mhz input clock is divided by 2 to provide the FIFO clock frequency. This allows example2 to be able to clock the FIFOs at 50Mhz (suitable for HEPC8) or 100Mhz (suitable for HEPC9).

You must also remember to set the HIGH_FCLK_G option to show if this clock is higher or lower than 60Mhz.

For example2 the correct options for an HEPC8 are:

DIV2_FCLK	FCLK_G_DOMAIN	HIGH_FCLK_G	HIGH_FCLK_RD	HIGH_FCLK_WR
True	True	False	n/a	n/a

This will set both input and output FIFOs to be clocked at 50Mhz.

For example2 the correct options for an HEPC9 are :-

DIV2_FCLK	FCLK_G_DOMAIN	HIGH_FCLK_G	HIGH_FCLK_RD	HIGH_FCLK_WR
False	True	True	n/a	n/a

This will set both input and output FIFOs to be clocked at 100Mhz.

You need to consider the timing constraints that are defined in the ‘.ucf’ file for your design. Actually if you use a time specification that is more strict than needed there is no problem, so the standard ‘.ucf’ file have the FIFO clocks specified as 100Mhz, along with the SDRAM clock defined as 133MHz. If the project builds (as Example3 does) with this specification it is still guaranteed to work at lower clock speeds. If you add new clock nets into your design then you need to add new timing constraints into your design.