Wishbone SoC Reference Design for EFM-02

The efm02_soc reference design demonstrates, how to connect FPGA IP cores like peripheral modules and memory to a host PC via the on-board Cypress FX-3 super-speed USB3.0 controller. It uses the CESYS UDK3 that comes with the EFM02 module. It is ideally suited as a starting point for user designs.

The host software can read and write all functional blocks of the reference design (i.e. DDR2 SDRAM, Flash memory or General Purpose IO ports), by calling the read and write functions of the API. The referenced addresses are embedded in the UDK3 protocol, transferred serial over the USB (Universal Serial Bus) and converted into Wishbone bus-cycles. This way, the host-software can access registers or memory locations in the FPGA by addressing them. You can easily expand the efm02_soc design by adding your own blocks to the Wishbone bus or modifying existing ones.

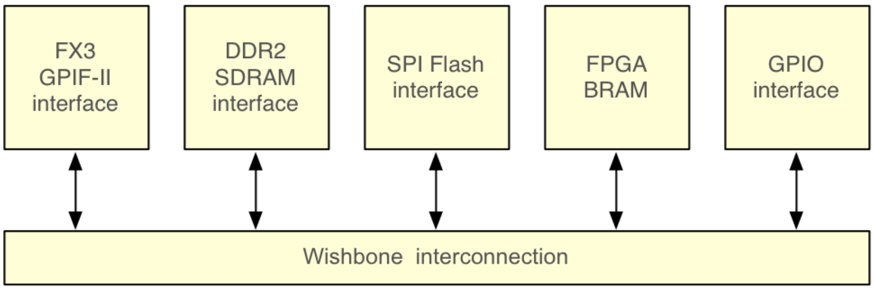
The reference design comes in VHDL source code. Depending on the target device (SLX45 or SLX150), the Xilinx(tm) ISE Webpack or ISE Design Suite can be used to generate FPGA bitfiles.

Features

The efm02_soc reference design contains the follwing basic elements:

- A clock manager to generate independent clocks for the FX-3 GPIF interface and the Wishbone bus with its peripheral modules.
- Wishbone SoC interconnect Architecture, 32 Bit data bus
- FX-3 GPIF-II interface, Wishbone busmaster
- DDR2 SDRAM interface based on Xilinx(tm) MIG Memory Interface Generator with a Wishbone wrapper.
- SPI Flash Memory interface with Wishbone wrapper.
- FPGA BRAM interface with Wishbone wrapper.
- GPIO interface with Wishbone wrapper.

EFM-02 reference design



Quick start

Obtaining the efm02_soc design

The efm02_soc design files are packed in the archive `efm02-soc-reference-design.zip`.
Use the login information, you received with your board to download the archive from www.cesys.com

Synthesis and generating bit file

The efm02_soc design can be synthesized with XILINX(tm) ISE Design suite 14.7 or later. Depending on the FPGA placed on your EFM02 board, you can either use the free WebPACK Edition (for SLX45) or the Embedded Edition (SLX100, SLX150).
Please consult www.xilinx.com to obtain Xilinx (tm) Design Software.

Testing the efm02_soc design

Install the Application "UDK3PerfMon.exe". It is part of the UDK3 and comes free with the EFM02 module. Configure the module with the design efm02_soc_top, or the optimized version efm_perf_top. To evaluate data transfer rates of different targets like BlockRam and DDR2 memory, enter the targets Wishbone bus address into UDK3PerfMon.exe For details see Cesys user guide ug104.

You can also use the PYTHON scripts that come with the UDK3 to communicate with the efm02_soc design or write a application in one of the supported languages. The UDK3 contains some source-code examples. udk3-udkapi-and-examples-1.0.zip

Files and Directory structure

File	Directory structure
`efm02/gpif_design`	This directory contains no files in the free efm02_soc design. The project files for Cypress GPIF(tm)II designer are only available as part of the UDK3 source code package (sold seperatly). You do not need this files to develop applications for the EFM02 module because they are already compiled into the FX-3 firmware that comes with the CESYS UDK3.
`efm02/ipcore_dir`	Project files for Xilinx(tm) CORE Generator. The efm02_soc uses CORE Generator FIFOs and Clock management IPs.
`efm02/src`	Source files of the reference designs. There are separate top_level designs and constraint files for SLX45 and SLX150. The other source files are identical for all FPGA densities.
`efm02/XC6SLX150`	Project files for ISE Project Navigator 14.7 targeting SLX150. `efm02_soc_top_slx150.xise`: full soc design project `efm02_perf_top_slx150.xise`: soc design project with only FX-3 and BRAM on the Wishbone-bus to demonstrate the maximum USB performance. The "perf" design runs with 100MHz on the GPIF (clk_gpif) and 180MHz on the rest of the design (sys_clk). The full "soc" design runs with100 MHz on the GPIF (clk_gpif) and 50 MHz on the rest of the design (sys_clk).
`efm02/XC6SLX45`	Project files for ISE Project Navigator 14.7 targeting SLX45.

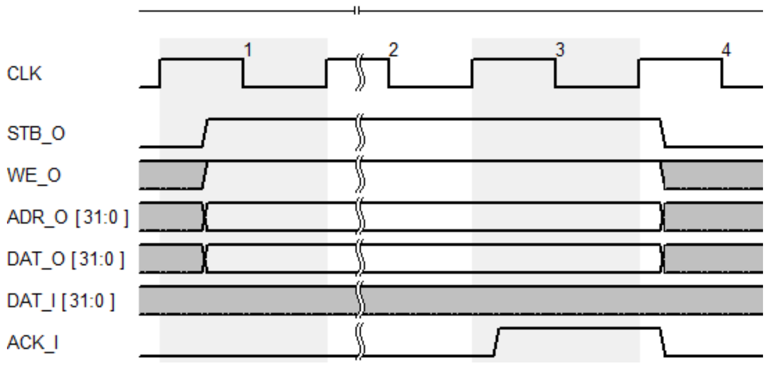
To implement the design, use the GUI of the XILINX(tm) Project Navigator Version 14.7 or later. Command-line users can produce a tcl-file with "Project -> Generate Tcl script..".

Wishbone bus

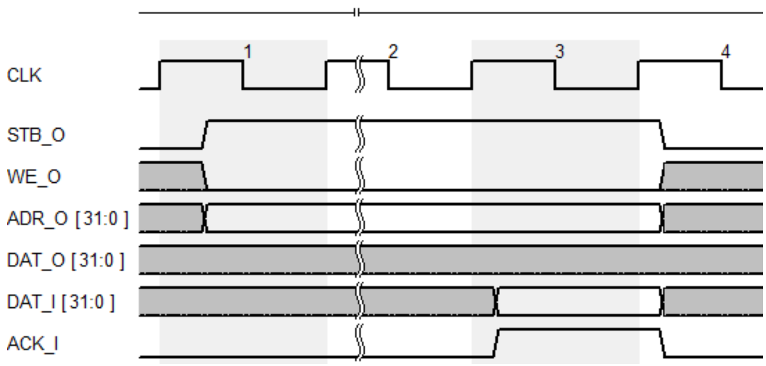
The reference design makes use of the WISHBONE System-on-Chip (SoC) Interconnect Architecture. The WISHBONE standard is not copyrighted, and is in the public domain. It may be freely copied and distributed by any means. Furthermore, it may be used for the design and production of integrated circuit components without royalties or other financial obligations.
The details are on http://opencores.org/opencores,wishbone

Wishbone transactions

In the efm02_soc reference design, all devices of the WISHBONE system are implemented to support only SINGLE READ / WRITE Cycles:



A Wishbone Master Transaction (Write)



A Wishbone Slave Transaction (Read)

The WISHBONE signals in these illustrations and explanations are shown as simple bit types or bit vector types, but in the VHDL code these signals could be encapsulated in extended data types like arrays or records.

Example:

```
port map
(
...
ACK_I =>
intercon.masters.slave(2).ack,
…
```

In this example, Port ACK_I is connected to signal ack of element 2 of array slave, of record masters, of record intercon.

Files and modules that are somehow related to the WISHBONE system are labeled with the prefix “wb_”. Wishbone slaves are prefixed with “sl_”. Wishbone master modules are prefixed with the additional prefix “ma_”.

Calling the software API-functions `ReadRegister()`, `WriteRegister()` lead to one and `ReadBlock()`, `WriteBlock()` to several consecutive WISHBONE single cycles. Bursting is not implemented in the efm02_soc reference design. The address can be incremented automatically in block transfers. You can find details on enabling/disabling the burst mode and address auto-increment mode in the CESYS application note AN101 - UDK3 Transfer Protocol and Cesys user guide UG101 – UDK3 API specification.

CESYS USB transfer protocol is converted into one or more WISHBONE data transaction cycles. So the FX-3 becomes a master device in the internal WISHBONE architecture.

Input signals for the WISHBONE master are labeled with the postfix “_I”, output signals with “_O”.

WISHBONE signals driven by the master	
STB_O	strobe, qualifier for the other output signals of the master, indicates valid data and control signals
WE_O	write enable. Indicates, if a write or read cycle is in progress
ADR_O[31:2]	32-Bit address bus, the software uses BYTE addressing, but all internal WISHBONE accesses are DWORD (32-Bit) aligned. So address LSBs [1:0] are discarded.
DAT_O[31:0]	32-Bit data out bus for data transportation from master to slaves

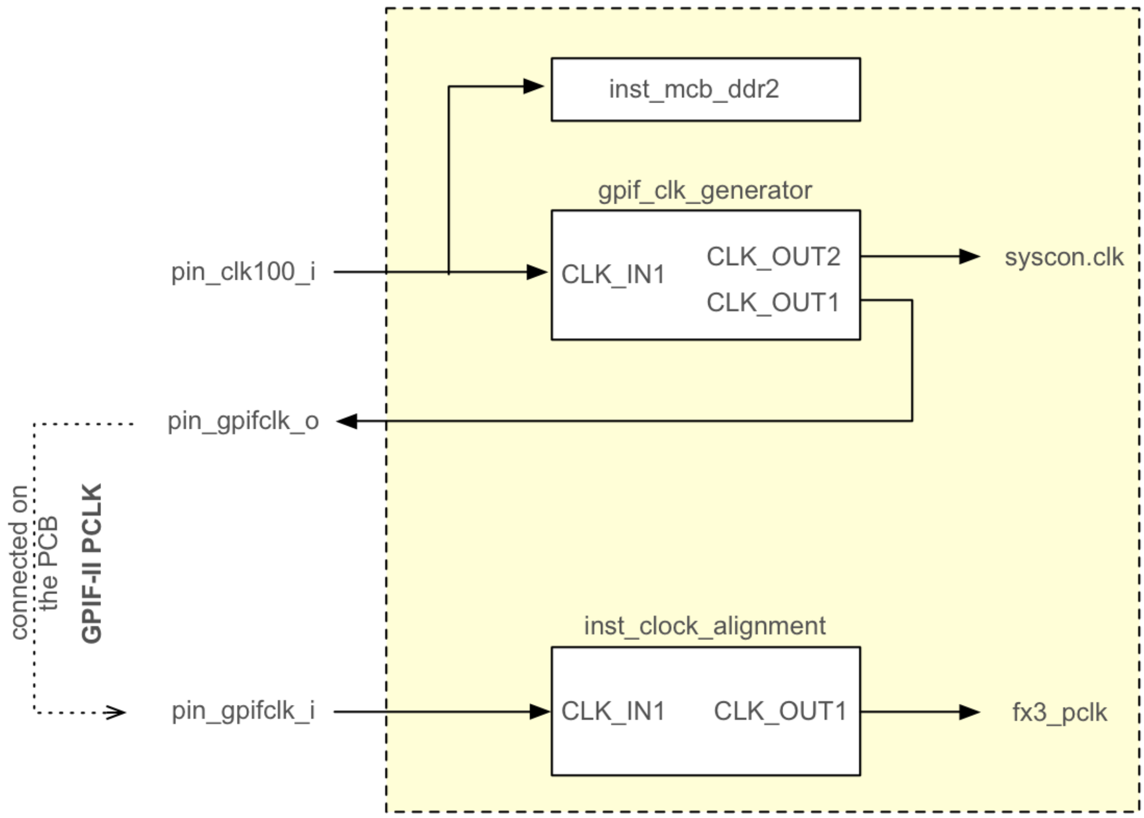
WISHBONE signals driven by slaves	
DAT_I[31:0]	32-Bit data in bus for data transportation from slaves to master
ACK_I	handshake signal, slave devices indicate a successful data transfer for writing and valid data on bus for reading by asserting this signal, slaves can insert wait states by delaying this signal, it is possible to assert ACK_I in first clock cycle of STB_O assertion using a combinatorial handshake to transfer data in one clock cycle Registered feedback handshake should be used in applications, where maximum data throughput is not needed, because timing specs are easier to meet.

The complete Wishbone bus specification and free IP modules can be found on:
http://opencores.org/opencores,wishbone

Clocking

The clock source for the efm02_soc reference design is the 100 MHz Oscillator Y1 at `pin_clk100_i` (LOC = W12). This clock is used by the DDR2 memory controller instance `inst_mcb_ddr2` and by the instance `gpif_clk_generator`.
The `gpif_clk_generator` provides clocks for the FX-3 GPIF-II interface and the Wishbone interconnect architecture. Clock signals `syscon.clk` and `pin_gpifclk_o` can have independent clock frequencies as required by the design.

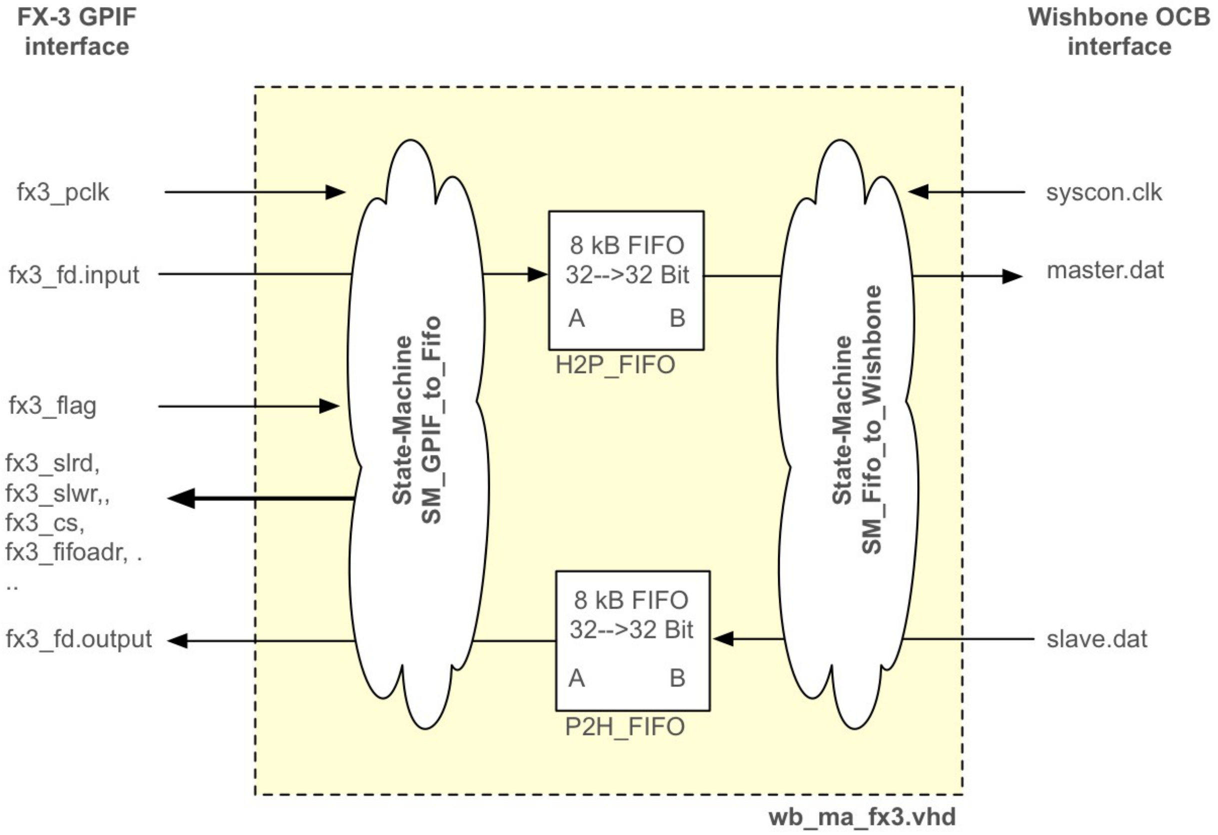
The GPIF-II clock (PCLK) frequency is set to 100 MHz in the efm02_soc reference design. This is the maximum allowed by the FX-3 specifications.
The FX-3 GPIF-II PCLK signal is driven by `pin_gpifclk_o` (LOC = W11).



At `pin_gpifclk_i` (LOC = Y12), the efm02_soc reference design expects the externally reefed clock signal PCLK of the GPIF-II interface. To compensate routing and buffer delays, it is aligned by `inst_clock_alignment`. The aligned clock *f*`x3_pclk` sources the GPIF - Wishbone interface `wb_ma_fx3.vhd`

USB 3.0 Interface

The Cypress EZ-USB® FX3™ is a SuperSpeed USB 3.0 peripheral controller.
Its configurable General Programmable Interface (GPIF™ II) is connected to the Spartan-6 FPGA. The efm02_soc reference design uses 32 bidirectional data bits, 4 FIFO status signals, 2 address signals and some control signals of this interface.



Two state machines and a asynchronous Fifo for each direction are used to connect the Wishbone OCB interface to the FX-3 GPIF II interface. The Fifos are generated by the XILINX ISE Core Generator. Because they are asynchronous, the GPIF PCLK frequency and the syscon.clk frequency can be choosen independently.
State machine `SM_GPIF_to_Fifo` moves blocks of data between the FX-3 Fifos and the FPGA Fifos. State machine `SM_Fifo_to_Wishbone` works on the other side of the Fifos, interprets the data blocks according to the UDK3 protocol and generates appropriate Wishbone OCB bus cycles.

State machine SM_GPIF_to_Fifo

The state machine `SM_GPIF_to_Fifo` moves data between the FPGA Fifo and FX-3 Fifo.
The GPIF II FIFO-status signals `fx3_flag` (full, watermark, empty) and control signals (`fx3_slrd, fx3_slwr, fx3_cs, ...`) have strict timing requirements and some clock cycles latency[1]. This are the challenges when designing a performant interface.
To satisfy the FX-3 setup and hold timing requirements, all signals related to the emf02_soc GPIF interface are directly sourced or sinked by FPGA Fip-Flops.
To avoid problems arising from the latency of the FX-3 status signals, the efm02_soc reference design does not monitor FX-3 status signals during data transfers. They are only monitored in the idle state to decide whether a transfer can be initiated.
The efm02_soc reference design uses a fixed block size of 8 kByte. This has been chosen as a compromise to maximize the transfer rate and minimize the response time when reading from the FPGA [2]. No short packets and no zero-lenght packets are needed.

Data transfer Host to Peripheral (H2P)

Socket 0 / thread 0 is used for transfers from host to peripheral (fx3_fifoadr is "00").
In Cypress GPIF II designer, flag(0) is connected to "thread0 DMA ready flag" which gives this signal the meaning "FX3 is empty and it can not give any data".
flag(1) is connected to "thread0 watermark flag". The watermark level is set to 8 kByte which gives this signal the meaning "FX3 is able to deliver at least one buffer".
Logically, the watermark flag is sufficient to decide whether the transfer of a block can be initiated but it is only valid in conjunction with the empty flag.

Data transfer Peripheral to Host (P2H)

Socket 2 / thread 2 is used for transfers from host to peripheral (fx3_fifoadr is "10").
In Cypress GPIF II designer, flag(3) is connected to "thread2 DMA ready flag" which gives this signal the meaning "FX3 is full".
flag(4) is connected to "thread2 watermark flag". The watermark level is set to buffer-size minus 8 kByte which gives this signal the meaning "FX3 is able to consume at least one 8k buffer".
Logically, the watermark flag is sufficient to decide whether the transfer of a block can be initiated but it is only valid in conjunction with the empty flag.

1 Details on cypress.com : AN65974 - Designing with the EZ-USB® FX3™ Slave FIFO Interface
2 Another idea is to run 2 endpoints with a big fixed block size to transfer data streams with with high rate and 2 endpoints with a small fixed block size for register read and write (not implemented).

State machine SM_Fifo_to_Wishbone

Data transfers over the USB bus are streamed. There are endpoints, but no individual addresses. Data transfers on the Wishbone bus are address oriented. The "translation" between streamed and address oriented transfers is implemented by inserting headers in the payload data according to the UDK3 protocol (see Cesys application note AN101 for details). On the host side, the software API inserts the headers.
On the device side, the state machine `SM_Fifo_to_Wishbone` interprets the headers and initiates appropriate Wishbone bus cycles.
This way, calling read or write functions of the API will result in read and write transaction on the Wishbone bus.

DDR2 SDRAM Interface

The DDR2 SDRAM memory controller is created by the Xilinx Memory Interface Generator (MIG).

The file `wb_sl_mcb.vhd` contains the Wishbone bus interface for the memory controller.

The memory controller core was generated using this parameters:

```
CORE Generator Options:
   Target Device              : xc6slx45-fgg484
   Speed Grade                : -3
   HDL                        : vhdl
   Synthesis Tool             : Foundation_ISE

If any of the above options are incorrect, please click on "Cancel", change
the Project Options in XPS, and re-run MIG from the MPMC GUI

MIG Output Options:
   Component Name             : mcb_ddr2
   No of Controllers          : 1



/*******************************************************/
/*                  Controller 3                       */
/*******************************************************/
Controller Options :
   Memory                  : DDR2_SDRAM
   Interface               : NATIVE
   Design Clock Frequency  : 3000 ps (333.33 MHz)
   Memory Type             : Components
   Memory Part             : MT47H128M16XX-25E
   Equivalent Part(s)      : MT47H128M16HG-25E;MT47H128M16RT-25E
   Row Address             : 14
   Column Address          : 10
   Bank Address            : 3
   Data Mask               : enabled

Memory Options :
   Burst Length                       : 4(010)
   CAS Latency                        : 5
   DQS# Enable                        : Enable
   DLL Enable                         : Enable-Normal
   OCD Operation                      : OCD Exit
   Output Drive Strength              : Fullstrength
   Outputs                            : Enable
   Additive Latency (AL)              : 0
   RDQS Enable                        : Disable
   RTT (nominal) - ODT                : 50ohms
   High Temparature Self Refresh Rate : Disable
```

```
User Interface Parameters :
   Configuration Type     : Two 32-bit bi-directional and four 32-bit
unidirectional ports
   Ports Selected         : Port0
   Memory Address Mapping : ROW_BANK_COLUMN

   Arbitration Algorithm  : Round Robin

   Arbitration            :
      Time Slot0 : 0
      Time Slot1 : 0
      Time Slot2 : 0
      Time Slot3 : 0
      Time Slot4 : 0
      Time Slot5 : 0
      Time Slot6 : 0
      Time Slot7 : 0
      Time Slot8 : 0
      Time Slot9 : 0
      Time Slot10: 0
      Time Slot11: 0

FPGA Options :
   Class for Address and Control        : II
   Class for Data                       : II
   Memory Interface Pin Termination     : CALIB_TERM
   DQ/DQS                               : 25 Ohms
   Bypass Calibration                   : enabled
   Debug Signals for Memory Controller  : Disable
   Input Clock Type                     : Single-Ended
```

GPIO Interface

The available user IO signals are mapped to eight 32 bit ports in the address space. The direction of each I/O signal can be configured as input or output by setting the direction bits in eight 32 bit direction registers. The registers are mapped to the FPGA I/O signals continuously with some unused signals at the end. There are 8*32 = 256 possible I/O signals but only 191 (LX150) respective 161 (LX45) are needed. Some of the unused signals are wired to control the user's Led.

User's LED

One of the three LEDS is controllable by the FPGA design. The default behavior is “blinking” to signal that the FPGA is configured, has clock and does not stuck in Reset. GPIO Output 254 switches between blinking mode and user-controllable Led state. GPIO Output 255 switches the Led on or off when it is in user-controllable state.

Flash memory Interface

The low level flash controller for SPI FLASH memory supports reading and writing of four bytes at one time and erasing the whole memory.

Address map

The base addresses of the efm02_soc modules are defined in the file `efm02_soc_pkg.vhd`:

```
MCB_BASEADR         0x0000 0000
BRAM_BASEADR        0x1000 0000
CFG_FLASH_BASEADR   0x2000 0000
GPIO_BASEADR        0x4000 0000
```

The software can access DDR2 SDRAM, FPGA block ram, flash memory and the IO-signals of the module using the addresses defined in this package when making a read or write call to the API.

Timing Constraints

According to the FX-3 datasheet (CYUSB301X_001-52136.pdf revised MAY 31, 2013), the timing constraints of the GPIF-II interface between FX3 – FPGA are:

```
NET "pin_clk100_i" TNM_NET = "pin_clk100_i";
TIMESPEC TS_pin_clk100_i = PERIOD "pin_clk100_i" 10 ns HIGH 50 %;

NET "pin_gpifclk_i" TNM_NET = "pin_gpifclk_i";
TIMESPEC TS_pin_gpifclk_i = PERIOD "pin_gpifclk_i" 10 ns HIGH 50 %;

# following specs are for PCLK = 100 MHz  (10ns)

# FX3 clock to data out (tCO) = 8 ns
# FX3 clock to data out hold (tDOH) = 2 ns
TIMEGRP "FD" OFFSET = IN 2 ns VALID 4 ns BEFORE "pin_gpifclk_i";

# FX3 data in to clock setup time (tDS) = 2ns
# FX3 data in to clock hold time (tDH) = 0.5ns
TIMEGRP "FD" OFFSET = OUT 8 ns VALID 8.5 ns AFTER "pin_gpifclk_i";

#
# FX3 clock to control out propagation delay (tCTLO) = 8ns
# FX3 clock to control out hold (tCOH) = 0ns
# The following constraint would normally apply:
# TIMEGRP "FLAGS" OFFSET = IN 2 ns VALID 2 ns BEFORE "pin_gpifclk_i" RISING;
# Because we treat FLAGS in the design as if they were completly
# asynchronous signals (2-stage synchronisation FFs to internal
# clock), their timing is TIG
TIMEGRP "FLAGS" TIG;

#
# FX3 control input to clock setup time (tS) = 2ns
# FX3 control input to clock hold time (tH) = 0.5ns
TIMEGRP "CTRL"  OFFSET = OUT 8 ns VALID 8.5 AFTER "pin_gpifclk_i";
```

This constraints are easily met by -3 speedgrade Spartan-6 FPGAs.
ToDo: what about other speedgrades?

Copyright Notice

Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by Cesys, and to the maximum extent permitted by applicable law:

(1) THESE MATERIALS ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND CESYS HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE;

and

(2) Cesys shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Cesys had been advised of the possibility of the same.

CRITICAL APPLICATIONS

CESYS products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of Cesys products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.

CESYS Gesellschaft für angewandte Mikroelektronik mbH
Zeppelinstrasse 6a
D - 91074 Herzogenaurach
Germany

Revision history

v1.0	February, 17 2014	Initial release.
v.1.1	April, 07 2014	Modified, Layout modified. (jk)

Table of contents