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

Using HERON modules with FPGAs to connect to FPDP

Rev 2.0 R.Williams 17-12-02

The HERON-FPGA and HERON-IO families are ranges of HERON modules with FPGAs, often combined with some interface capability. The HERON-FPGA family in particular provides an FPGA along with a large number of signals that are routed to general-purpose connectors. As such these modules are highly suited to applications that connect to FPDP.

Some of the modules in the HERON-FPGA range include a Differential ECL driver & receiver chip. The main reason for providing this chip is to allow the module to be connected to clocks such as the PSTROBE PECL clock provided by the Front Panel Data Port (FPDP) standard.

For modules such as the HERON-FPGA4 that do not have an ECL driver & receiver chip FPDP can still be used. With these modules the FPDP clock is simply connected directly to the Virtex-II FPGA, which must implement the LVPECL_33 I/O standard in the input IOB connected to the PSTROBE signal.

This example shows how a user can connect the FPGA to FPDP to gather or send data from that bus. Because this is only really interesting when combined with other processing in the FPGA program, it is not offered as Intellectual property that allows direct use of a module for an FPDP interface, but merely as a demonstration of how you would make this connection.

If however a user wishes to use an FPGA module simply as an interface between FPDP and a DSP, they should discuss this with HUNT ENGINEERING, as it would be very simple to make such an interface if the requirements can be tightly defined.

History

Rev 1.0	First written (in schematic form)
Rev 2.0	Converted the example to VHDL

What is FPDP?

FPDP is a simple interface that has been defined for interconnecting equipment from different vendors. Data acquisition boards and DSP boards for example. It is a 32 bit interface that can be clocked at up to 40Mhz, allowing the transfer of data at 160Mbytes/sec.

It uses a simple ribbon cable interface, which has a single ended strobe and a differential PECL strobe defined. It is recommended that the differential version of the strobe signal is used.

FPDP can be used in framing modes, and even in time division multiplexing to use the same FPDP connection to carry data from several sources.

For the full specification see http://www.ics-ltd.com/login_a.htm

Electrical Connection

The first consideration that must be made when interfacing to FPDP is what kind of electrical connection is required to use it. Only when correct electrical connection has been made can the FPGA be programmed to receive data from the FPDP interface.

The physical FPDP bus can be connected to the I/O pins of an FPGA, but the FPDP signals are TTL level signals that are driven using FCT type logic. According to the FPDP standard, a receiver *should* terminate some of these signals. The HERON-FPGA modules do not have these terminations, but using the differential Strobe signal should work well with short cables.

As the FPDP signals are TTL they cannot be directly connected to an FPGA that is not 5V tolerant (like the Virtex II devices). The HERON-FPGA2 and Spartan-II based HERON-FPGA3S all use 5V tolerant FPGA devices and can therefore be used very simply with FPDP.

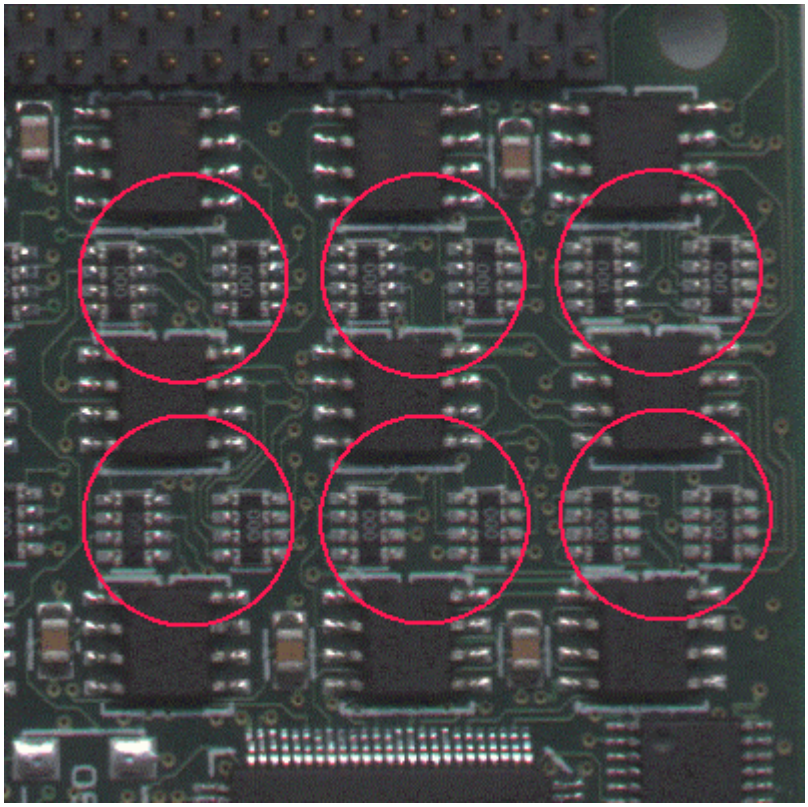
However, the processing requirements of your application may dictate that a Virtex-II FPGA be used. Virtex-II based modules like the HERON-FPGA3V and HERON-FPGA4V have the possibility to fit 100R series resistors into the I/O lines at build time. The combination of these resistors and the over-voltage protection on these modules make it safe to connect FPDP to these modules, if the 100R series resistors are fitted.

The following sections discuss what is required if you intend to use the HERON-FPGA3V or HERON-FPGA4V with FPDP. For those who are not using the Virtex-II based modules you may skip these sections.

Using FPDP with a Virtex-II HERON-FPGA3V

If you intend to use the Virtex-II HERON-FPGA3V module to implement FPDP, please ensure that you specify the inclusion of 100R series resistors for your board when you place the order.

The location of the resistor packs is shown in the picture below (the picture shows the underside of the board). The resistor packs are normally fitted as 0R resistors (marked with 000 on the pack). For the use of 5V signals these must be fitted as 100R (package marked 101) to the sites for Connectors A, B and C of the HERON-FPGA3 such that each signal has 100R in series between the FPDP cable and the FPGA.



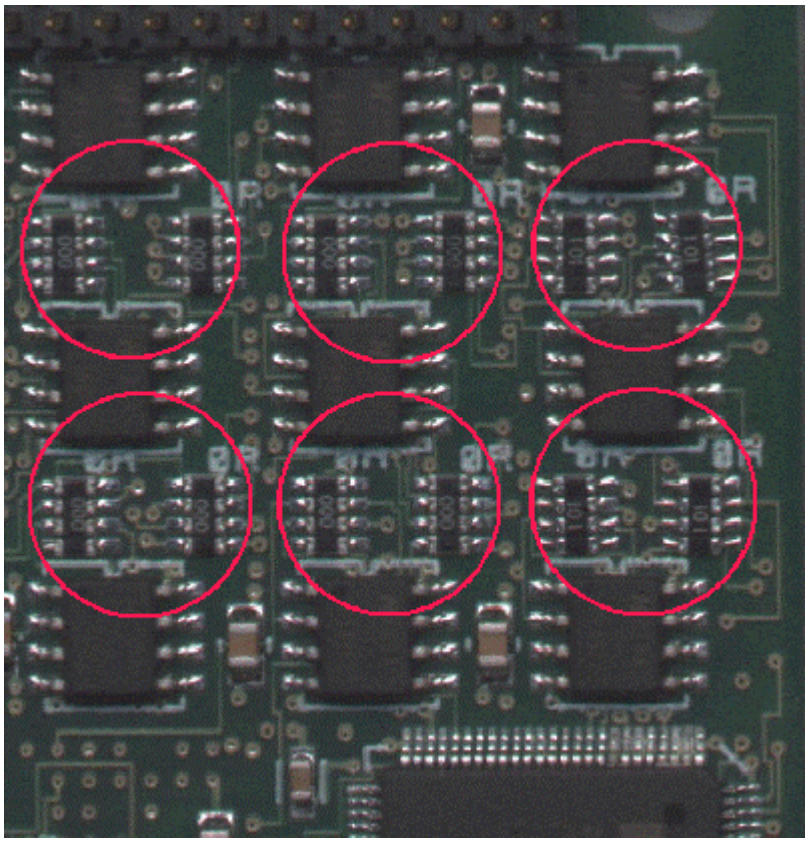
If when you ordered your Virtex-II HERON-FPGA3V module you did not inform HUNT ENGINEERING that you would be interfacing to 5V FPDP signals, then your board will not have these resistors fitted.

In this case you can either contact HUNT ENGINEERING to discuss having the resistors fitted at the factory, or alternatively you can fit the resistors yourself. Please note, although HUNT ENGINEERING is happy for you to fit the resistors yourself, any damage done to the board in doing so is not covered under the warranty.

Using FPDP with a Virtex-II HERON-FPGA4V

If you intend to use the Virtex-II HERON-FPGA4V module to implement FPDP, please ensure that you specify the inclusion of 100R series resistors for your board when you place the order.

The location of the resistor packs is shown in the picture below (the picture shows the underside of the board). The resistor packs are normally fitted as 0R resistors (marked with 000 on the pack). For the use of 5V signals these must be fitted as 100R (package marked 101) to the sites for Connectors A, B and C of the HERON-FPGA4 such that each signal has 100R in series between the FPDP cable and the FPGA.



If when you ordered your Virtex-II HERON-FPGA4V module you did not inform HUNT ENGINEERING that you would be interfacing to 5V FPDP signals, then your board will not have these resistors fitted.

In this case you can either contact HUNT ENGINEERING to discuss having the resistors fitted at the factory, or alternatively you can fit the resistors yourself. Please note, although HUNT ENGINEERING is happy for you to fit the resistors yourself, any damage done to the board in doing so is not covered under the warranty.

VHDL Example : Interfacing to FPDP Raw Data

This document is provided along with two VHDL source files that demonstrate interfacing to raw FPDP data.

The FPDP standard works on several levels. At the lowest level there are electrical connection considerations and cabling issues. At the next level, logic is required that will present the signals provided on an FPDP cable to an internal FPGA interface that can process the incoming data. At the final level, logic is required that will decode packets and frames of data from the incoming data stream.

The final level of packet and frame decoding is highly system specific and is outside the scope of this document. That part of FPDP interface design is left for the user to implement.

The remainder of this document deals with the example VHDL module, and how this module can be used to interface the signals presented on an FPDP cable to a user's 'application specific' VHDL design.

The FPDP Signals

The FPDP interface is quite simple, being made up of a 32-bit data path and associated control signals. These control signals include a Strobe signal used to clock the data and signals used for indicating valid data and frame synchronisation.

The control signals are:

- PSTROBE — PECL data strobe
- DIR* — Asserted low by the data source
- DVALID* — Asserted low by the data source when the data bus has valid data
- NRDY* — Asserted low by the receiver when it is not ready to receive data
- SUSPEND* — Asserted by the receiver to show that there is soon to be a condition where it cannot receive data (the sender can send up to 16 more words)
- SYNC* — Asserted low by the data source during the last data word of a frame. Intended for use by the receiver to synchronise data capture
- PIO1, PIO2 — Programmable I/O lines for implementing user defined functions

Note: * indicates active low signals

The Example VHDL 'FPDP' Component

When the FPGA is being used to receive data from an FPDP source, this makes the FPGA the receiver and the FPDP source the transmitter. As such, the FPGA interface must connect to the PSTROBE, DIR*, DVALID*, SUSPEND*, SYNC* and D(31:0) as inputs, and drive the NRDY* signal as an output.

It is up to the user to define if the programmable I/O signals PIO1 and PIO2 are to be used, and whether they are to be inputs or outputs, to or from the FPGA. In our example, they have been configured as inputs, but this can be simply changed to create outputs instead.

The example VHDL that has been provided illustrates how the FPGA should be used when connecting to FPDP as a receiver.

There are two VHDL files provided as part of the FPDP example. The first file ‘fpdp.vhd’ contains one entity named ‘FPDP’ which can be used in your design to connect to FPDP. The second file ‘fpdp.vho’ contains instantiation templates that allow you to quickly cut and paste the required component declaration and component instantiation lines into your own design source.

The FPDP entity declaration is shown below:

```
entity FPDP is
   port (
      -- FPDP Control Inputs and Output
      PSTROBE_PIN  : in  std_logic;
      DVALID_PIN   : in  std_logic;
      SYNC_PIN     : in  std_logic;
      SUSPEND_PIN  : in  std_logic;
      DIR_PIN      : in  std_logic;
      PIO1_PIN     : in  std_logic;
      PIO2_PIN     : in  std_logic;
      NRDY_PIN     : out std_logic;
      NRDY_PIN_EN  : out std_logic;
      -- FPDP Data Inputs
      DIN_PINS     : in  std_logic_vector(31 downto 0);
      -- FPDP Control Signals
      FPDP_CLOCK   : out std_logic;
      FPDP_DVALID  : out std_logic;
      FPDP_SYNC    : out std_logic;
      FPDP_SUSPEND : out std_logic;
      FPDP_DIR     : out std_logic;
      FPDP_PIO1    : out std_logic;
      FPDP_PIO2    : out std_logic;
      FPDP_READY   : in  std_logic;
      -- FPDP Data Signals
      FPDP_DIN     : out std_logic_vector(31 downto 0)
   );
end FPDP;
```

The FPDP component has ten ports that must be connected to signals that form the I/Os of the device. That is, each of these ports must be directly connected to signals in the USER_AP interface which connect to user I/Os.

The ten ports are PSTROBE_PIN, DVALID_PIN, SYNC_PIN, SUSPEND_PIN, DIR_PIN, PIO1_PIN, PIO2_PIN, NRDY_PIN, NRDY_PIN_EN and DIN_PINS.

The following VHDL code fragment (part of the component instantiation) shows an example of the connections that would be made when using the HERON-FPGA3.

```
PSTROBE_PIN  => QTTL,
DVALID_PIN   => CONN_C_IN(2),
SYNC_PIN     => CONN_C_IN(3),
SUSPEND_PIN  => CONN_C_IN(6),
DIR_PIN      => CONN_C_IN(7),
PIO1_PIN     => CONN_C_IN(4),
PIO2_PIN     => CONN_C_IN(5),
NRDY_PIN     => CONN_C_OUT(8),
NRDY_PIN_EN  => CONN_C_EN(8),
DIN_PINS     => FPDP_DIN_PINS,
```

In this example the external ECL driver/receiver chip of the HERON-FPGA3 is being used for the PSTROBE clock input. The output of the ECL receiver buffer is provided through the USER_AP signal 'QTTL'. The signals CONN_C(2) up to CONN_C(7) on Connector C are used as inputs, and the signal CONN_C(8) is used as an output. Note, the FPDP component uses both the CONN_C_OUT(8) connector output and the CONN_C_EN(8) output buffer enable signal to implement a tri-stateable output for the FPDP NRDY signal.

The signal FPDP_DIN_PINS has been defined as a std_logic_vector of 32-bits in size in this example, and is assembled from 32 signals input on Connectors A, B and C as follows:

```
FPDP_DIN_PINS <= CONN_C_IN(1 downto 0) & CONN_B_IN & CONN_A_IN;
```

The remaining component ports that all start with the string 'FPDP_' are the internal FPDP signals that are intended to be used the user application. Any synchronous element in your design that uses the FPDP control or data signals must be clocked using a clock net driven by the FPDP_CLOCK output of the FPDP component.

Using the Example 'FPDP' Component

It is now up to the user to decide how to process the data output by the FPDP component. This decision will be based on the data format being driven onto the FPDP bus, and how the data processing will deal with that data. However, regardless of the logic used to connect to this component the following points should be remembered.

1. The FPDP_CLOCK pin must be used to clock logic connected to the registered FPDP data bus FPDP_DIN, and the four registered control signals FPDP_DIR, FPDP_DVALID, FPDP_SUSPEND and FPDP_SYNC. The rising edge of the FPDP_CLOCK signal must be used.
2. The FPDP_READY signal is asynchronous, so therefore does not need to be clocked using FPDP_CLOCK. The FPDP_READY component port should be driven high during normal operation, and set low when no more data can be received.
3. The registered FPDP_DVALID signal is active low. It will be asserted low during a clock cycle where the registered 32-bit data bus contains valid data. This means that the FPDP_DVALID signal can be inverted and used to drive an active high clock enable input on the next stage of logic.
4. The FPDP_SYNC signal is active low. It will be asserted low during the last valid data word of a received frame. The receiving logic must use a logic 0 on this signal to reset input logic such the next valid data that is received is treated as the first data of a new frame.

A time-specification must be applied to the design that specifies the clock rate of the FPDP_CLOCK net. This time-specification should be set to 40MHz, which is the maximum FPDP clock rate.

The 'FPDP' Component Implementation

The PSTROBE clock signal must connected to as an input, and inside the FPGA must be used to drive a clock net that is driven by a global clock buffer. Inside the FPDP component, a BUFG component is used to buffer this clock net. The output of the global clock buffer drives the FPDP_CLOCK port, and should be used by all synchronous logic elements that interface to the FPDP signals.

The control signals DIR*, DVALID*, SUSPEND* and SYNC* are all registered with input flip-flops inside the FPDP component. These flip-flops are clocked using the FPDP_CLOCK signal. The 32-bits of data are also registered with input flip-flops inside the FPDP component. Again, the input flip-flops are all clocked using the FPDP_CLOCK signal.

For the NRDY* output control signal, which is defined as open collector, an output buffer with tri-state control is required (for example, the component OBUFT). With the FPDP component, this is implemented using the OBUFT components that appear in the design tree's top source module 'top.vhd'. The input of the OBUFT is driven with a logic '0' by connecting the NRDY_PIN port of the FPDP component to a digital I/O connector output (e.g. CONN_C_OUT(8)). The tri-state control associated with that output must then be driven with the state of the FPDP_READY signal. This is done by connecting the NRDY_PIN_EN port to the corresponding digital I/O tri-state enable (e.g. CONN_C_EN(8)). This simulates the use of an open collector driver.

When the FPDP_READY signal is high, the output buffer will be tri-stated. When the ready signal is low, the buffer will be enabled, forcing the NRDY* signal low. Therefore, the FPDP_READY signal should be driven high inside the FPGA when data can be received and driven low when data cannot be received.

FPDP Cabling

How you choose to connect the FPDP cable to your FPGA is largely up to you. However, what is provided below is an example that can be used as a starting point for your cabling solution.

This cabling solution may be used with the HERON-FPGA2 and HERON-FPGA3 as shown. If you are using the HERON-FPGA4 module, you will need to modify the STROBE connection so that it uses two devices I/Os. This change will also require that an LVPECL_33 IOSTANDARD input buffer is used in the source module 'top.vhd' in place of the default buffer types. A VHDL code example that does this is included at the end of the 'fpdp.vho' instantiation template.

Remember, if you are using Virtex-II based modules you must check that the correct series resistors are being used and that the resistors that are fitted match the connections used. Please refer to the section 'Electrical Connection' for more information on the requirements of the series resistors for Virtex-II boards.

The FPDP cable usually uses a fine pitch ribbon cable, and a connector of the type 8825E-080-175 (KEL no strain relief), 8825R-080-175 (KEL with strain relief) or P25E-080S-TG (Robinson Nugent).

Signal	FPDP Cable Core	HERON-FPGA Connector Pin
GND	1	N/C
STROBE	2	N/C
GND	3	N/C
GND	4	N/C
GND	5	N/C
GND	6	C19
NRDY*	7	C18
GND	8	C17
DIR*	9	C16
GND	10	N/C
RESERVED	11	N/C
GND	12	C15
SUSPEND*	13	C14
GND	14	N/C
GND	15	N/C
GND	16	C13
PIO2	17	C12
GND	18	C11
PIO1	19	C10
GND	20	N/C
RESERVED	21	N/C
GND	22	N/C
RESERVED	23	N/C
GND	24	Serial I/O - GND
PSTROBE	25	Serial I/O - DECL
GND	26	Serial I/O - GND
PSTROBE*	27	Serial I/O - DECLB
GND	28	C9
SYNC*	29	C8
GND	30	C7
DVALID*	31	C6
GND	32	C5

D31	33	C4
D30	34	C2
GND	35	C1
D29	36	B30
D28	37	B28
GND	38	B27
D27	39	B26
D26	40	B24
GND	41	B21
D25	42	B22
D24	43	B20
GND	44	B19
D23	45	B18
D22	46	B16
GND	47	B13
D21	48	B14
D20	49	B12
GND	50	B11
D19	51	B10
D18	52	B8
GND	53	B5
D17	54	B6
D16	55	B4
GND	56	B3
D15	57	B2
D14	58	A30
GND	59	A29
D13	60	A28
D12	61	A26
GND	62	A23
D11	63	A24
D10	64	A22
GND	65	A21
D09	66	A20
D08	67	A18
GND	68	A15
D07	69	A16
D06	70	A14
GND	71	A11
D05	72	A12
D04	73	A10
GND	74	A9
D03	75	A8
D02	76	A6
GND	77	A5
D01	78	A4
D00	79	A2
GND	80	A1

** The differential ECL clock results in a single input to the FPGA for the HERON-FPGA2 & 3.