Gateware Munition Interface Processor (GMIP)

Robert L. Riley, Jr., Munitions Directorate, Air Force Research Laboratory, Eglin AFB, FL (robert.riley@eglin.af.mil)

Daniel A. Perkins, Electronics and Avionics Systems, Battelle Memorial Institute, Columbus, OH (perkinsd@battelle.org)

Abstract

This paper presents the first version of the gateware munitions interface processor (GMIP) board. The GMIP is an FPGA-based, processor board that is specifically designed to handle communication over either a MIL-STD-1553 or a miniature munition store interface (MMSI) bus. Due to the flexibility of its FPGA processing technology, the board can also accommodate more compute-intense algorithms and applications (i.e. data link, sensor fusion, advanced signal and image processing). Such algorithms can be implemented while sustaining functionality of the necessary communication protocols that were previously mentioned. This initial version of the GMIP utilizes the Xilinx Virtex-II Pro technology.

1. Introduction

This paper describes the GMIP, a piece of hardware that has been designed for use as a reconfigurable processing, interface, and prototyping platform. Within this paper, an overview of the processor, its discrete components, test and evaluation results, and concepts for future work will be presented.

2. Overview of the GMIP

The Gateware Munitions Interface Processor (GMIP) provides a development platform for prototyping aircraft interfaces and processing applications. Utilizing reconfigurable technology, the GMIP applies powerful capability for implementing flexible high-speed designs. The GMIP utilizes the Xilinx Virtex-II Pro family of Field Programmable Gate Arrays (FPGAs) as a means for providing a parallel computing platform on a chip along with capability for hosting high-speed digital logic for aircraft interfacing. For other applications, the GMIP implements four channels of both EBR-1553 and CANBus hardware interfaces. For legacy applications, the GMIP implements two channels of MIL-STD-1553B interfaces. Both MMSI and MIL-STD-1553B require discrete control for power supply sequencing and independent control. These discrete interfaces are provided via the General Purpose IO Interface. For development and support, the GMIP implements a JTAG interface and a monitor/debug interface. Functional implementation of interfaces and application computation capability is accomplished with a high performance, high density Field Programmable Gate Array (FPGA). An FPGA Reconfiguration Controller is provided to allow mission loads and dynamic reconfiguration capability. DDR SDRAM, Static RAM and FLASH memory are provided to facilitate applications requiring runtime and non-volatile storage memory. A system description has been developed in order to design implementations in Viva®[1] and directly program the GMIP board from the Viva environment. Figure 1 shows a block diagram of the GMIP.

[1] Viva® is a registered trademark of Star Bridge Systems, Inc.

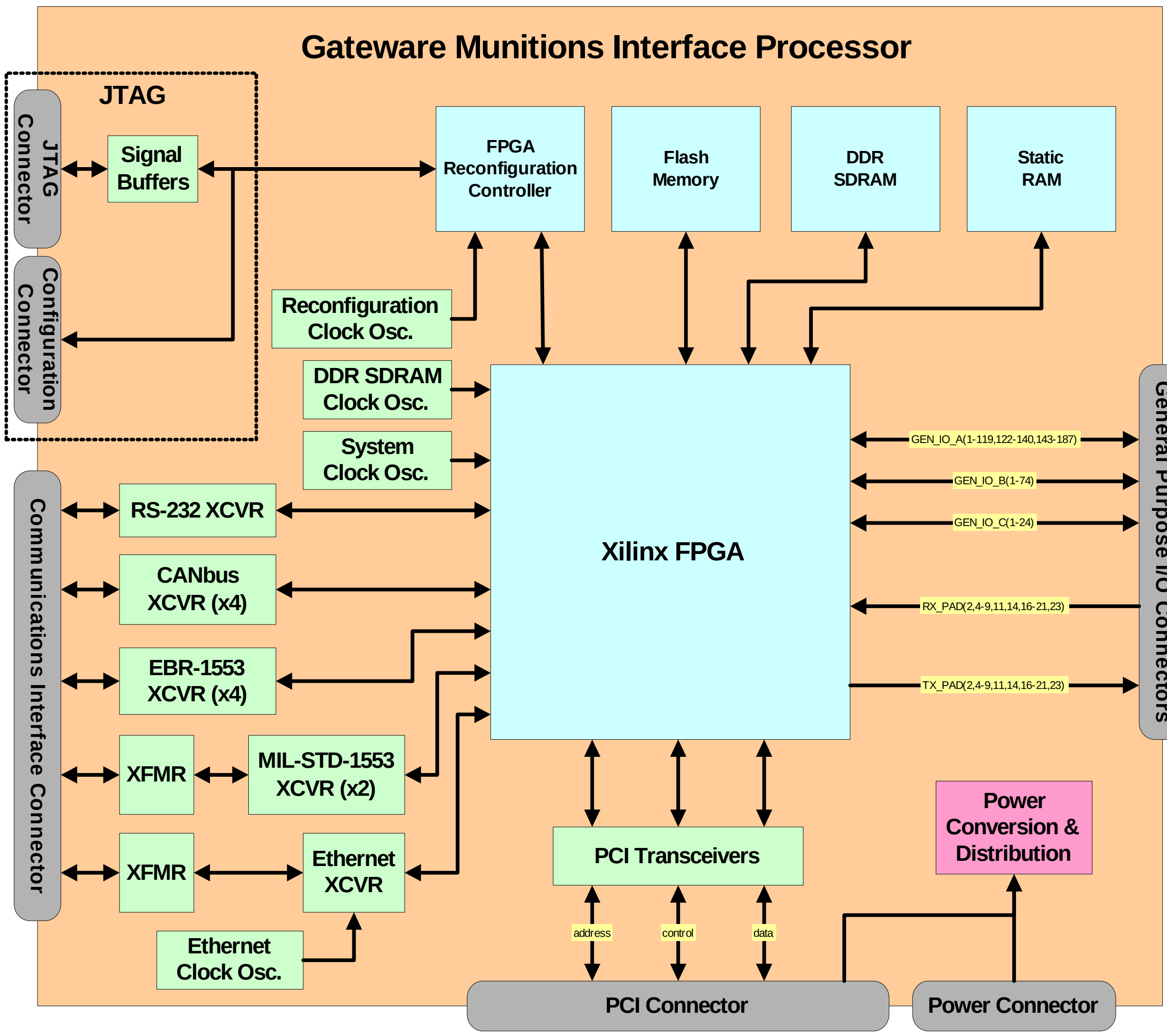

Figure 1: GMIP Block Diagram

3. Xilinx Virtex-II Pro FPGA

The Xilinx Virtex-II Pro FPGA performs the function of the central processing element on the GMIP. This device can be reconfigured either statically or dynamically ("on the fly") to perform any number of processing tasks along with the low level logic function implementation necessary to control the on board interfaces and external general purpose IO. The GMIP V2PRO exists in a 1152 pin Flip-Chip Fine-Pitch BGA package providing up to 692 IO pins depending on the device used. V2PRO devices compatible with this footprint may include up to 2 PowerPC hard-core processors, 16 RocketIO Transceivers, 8 Digital Clock Managers (DCMs), 4.176 Mbits of RAM, 232 Hardware Multipliers and 23,616 logic cell slices [2].

4. General Purpose IO (GPIO) Interface

The GPIO interface provides expansion capability to the GMIP. 287 discrete IO signals along with signaling for 8 RocketIO Interfaces are implemented across three high-density board-to-board connectors. This interface allows a developer to interface a custom daughter-card containing virtually any

2.5V compliant digital electronics. This custom daughter card could be used to add a memory, analog data acquisition circuit or an external interface like USB or Firewire. The 16 available RocketIO channel signaling sets are also brought out to this interface to allow use of these high-speed transceivers.

5. JTAG Interface

The JTAG interface is used for programming the GMIP Reconfiguration Controller discussed below or for configuring the V2PRO FPGA directly. Standard FPGA development tools include software that uses the parallel port on a PC along with a programming cable to implement the JTAG interface. The V2PRO may be configured directly across this interface in order to facilitate rapid development. The Reconfiguration Controller may also be programmed across this interface with FPGA bit-streams that are later used to configure the V2PRO FPGA. The electronics usually found in JTAG parallel programming cables are implemented directly on the GMIP. Because of this, the developer will only need a standard parallel cable connected to the DB25 connector on the GMIP to use the JTAG interface. A second interface connector brings out the JTAG and parallel configuration signals that interface with the Reconfiguration Controller. These signals are used to program the Reconfiguration Controller from an external source using the JTAG or parallel programming interface.

6. Reconfiguration Controller

The Reconfiguration Controller is used to reconfigure the V2PRO FPGA with any of eight bit-streams loaded into its internal Flash memory. The Reconfiguration Controller is a Multi Package Module (MPM) that contains a small Xilinx FPGA, Flash memory and a Configuration PROM. The Configuration PROM configures the internal FPGA on power-up. The internal FPGA handles programming the V2PRO FPGA with the appropriate bit-stream stored in the Flash memory. The Reconfiguration Controller is connected to the JTAG interface so that the internal Flash can be programmed with bit-streams by the development system. On power-up the Reconfiguration Controller configures the FPGA with the bit-stream held in position 1 in the Flash. The Reconfiguration Controller can be commanded by the V2PRO FPGA at any time after this to reconfigure the V2PRO FPGA with any of the stored bit-streams via the control interface. Using this interface, the V2PRO FPGA may self-reconfigure thus performing dynamic reconfiguration.

7. Flash Memory

The GMIP contains on-board Flash memory for use by user-developed applications that require non-volatile storage capability. Data stored in the Flash memory will be available after power cycling of the GMIP so that user applications may retain critical data between missions.

8. DDR SDRAM

The GMIP contains an on-board DIMM Socket for supporting DDR SDRAM memory. User applications may use DDR SDRAM as a means to quickly store and retrieve large amounts of data used during a mission.

9. Static RAM

The GMIP contains on-board static RAM for use by user-developed applications that require a fast temporary data storage capability. Data may be stored to and retrieved from the Static RAM very quickly without the need for memory refreshing and other “memory overhead” tasks usually associated with dynamic memory.

10. RS-232 Interface

The RS-232 Interface on the GMIP is included primarily for development purposes. This interface may be used as a means to communicate with a host PC application like a terminal emulator so that the user FPGA application may be easily monitored or controlled. The RS-232 interface is accessed via the DB9 connector on the GMIP using a standard serial cable.

11. CANBus Interface

The CANBus Interface on the GMIP is included primarily for user defined purposes. The CANBus Interface includes 4 CANBus transceivers that handle the low level CANBus signaling required by the standard.

12. EBR-1553 Interface

The EBR-1553 Interface on the GMIP is included primarily as a part of the MMSI implementation for Munitions, although, it could be used for user defined purposes. The EBR-1553 Interface includes 4 RS-485 transceivers that handle the low level RS-485 signaling required by the standard.

13. MIL-STD-1553B Interface

The MIL-STD-1553B Interface includes 4 MIL-STD-1553B transceivers along with 4 MIL-STD-1553B transformers that handle the low level signaling required by the standard.

14. Ethernet Interface

The GMIP contains an on-board Ethernet Interface for use by user-developed applications that require interfacing to Ethernet compliant devices. The Ethernet Interface includes the low level Ethernet transceiver that handles the low level Ethernet signaling required by the standard.

15. PCI Interface

The GMIP exists in a PCI form factor so that it may be installed in a PCI back plane and utilized as a PCI compliant card. The PCI interface includes PCI transceivers to support the necessary signaling required by the PCI standard.

16. Power

The GMIP requires 5V of input power. This 5V input is converted and distributed across the GMIP to power all of the on-board devices. This input power may be supplied by the input power connector or by the PCI interface. The input power connector is located such that if the GMIP is used in a PCI back-plane, it will be inaccessible. For this reason, any applications requiring a PCI configuration are dependant upon the available power provided by the PCI interface.

17. Clock Oscillators

The GMIP contains four clock oscillators:

Reconfiguration Clock Osc.	- Used exclusively by the Reconfiguration Controller
DDR SDRAM Clock Osc.	- Used by the FPGA as a reference clock for the DDR SDRAM. Should be used in conjunction with a digital clock manager (DCM) to provide a clock to the DDR SDRAM and a clock domain to the DDR SDRAM Control Logic.
System Clock Osc.	- General Purpose Clock used by the FPGA
Ethernet Clock Osc.	- Used exclusively by the Ethernet Transceiver

18. Switches

18.1 System RC Switch

The user may initiate a reconfiguration operation by pressing this switch. This switch drives the sys_rc signal as described earlier "HIGH". Note that the bit-stream selected by the rc_bitstr(2:0) pins will dictate which bit-stream in the reconfiguration controller flash is used to reconfigure the FPGA. Care should be taken in the design of the user FPGA application such that the rc_bitstr(2:0) pins default to a desired bit-stream position for a reset condition.

18.2 1553 Coupling Switches

The two on-board MIL-STD-1553B busses can be connected in a "Direct Coupled" or "Transformer Coupled" configuration as defined by the MIL-STD-1553B specification. The coupling for each bus is controlled by a toggle switch.

18.3 DIP Switches

An 8-position DIP switch is available on the GMIP for the purposes of user input. The DIP switch signals are connected directly to the V2PRO FPGA. A DIP switch in the "ON" position is read as a "LOW" by the FPGA while the "OFF" position is read as a "HIGH".

19. LEDs

19.1 PROG DONE LED

The PROG DONE LED is illuminated after a reconfiguration has completed. This LED will stay illuminated until the sys_rc pin has been driven high to initiate another reconfiguration.

19.2 DEV READY LED

The DEV READY pin is illuminated when the Reconfiguration Controller has been internally configured and is ready for operation. The V2PRO FPGA cannot be configured until this condition is met.

19.3 XMIT LED

The XMIT LED is illuminated when the Ethernet transceiver is performing a transmit operation.

19.4 LINK LED

The LINK LED is illuminated when the Ethernet transceiver recognizes that it is connected to a legitimate Ethernet bus.

19.5 100 MB/S LED

The 100 MB/S LED is illuminated when the Ethernet transceiver is operating at 100MB/S mode. Otherwise, the transceiver will be operating in 10MB/S mode.

19.6 STATUS LEDs

The 4 status LEDs indicate the status of the Reconfiguration Controller.

19.7 Dual 7 Segment LED

A dual 7 segment LED is available on the GMIP for the purposes of having display/monitor capability available to user FPGA applications. The Dual 7 Segment LED signals are connected directly to the V2PRO FPGA. Any of the segment LEDs driven opposite to the polarity signal for that 7 segment digit will be illuminated.

19.8 Power LEDs

The Power LEDs are illuminated corresponding to their respective power rails being active.

20. Battery Holder

The GMIP contains a 3V coin cell batter holder to power the V2PRO FPGA Decryptor Key Memory Backup Supply.

21. Power-On Reset Device

The GMIP contains a Power-On Reset Device that is connected to a user IO pin on the FPGA. The Power-On Reset signal goes “HIGH” When the 3.3V supply is available after a small time delay. This signal should be used to reset the DCMs in the V2PRO FPGA. Any logic driven by a DCM Clock output should be reset by the DCM Locked signal or its complement given the required logic reset polarity.

22. System Description Development

The system description for the GMIP was developed in order to utilize the Viva software environment for designing, synthesizing and calling the Xilinx tools for place and route of implementations for the GMIP board. The system description represents a software interpretation of the physical hardware which Viva uses for resource utilization, data communication, programming and design development on the GMIP. The current system description does not incorporate the Behavioral Communications System (BCS), which is Viva’s user interface for testing input and output signals. An upgrade to the system description in order to incorporate the BCS is currently planned for the future.

23. Porting EDIF & Block Memory (BRAM) Initialization

23.1 Method for use of IP Cores in Viva

The following process details the necessary steps for importing IP cores or externally synthesized designs into Viva. This process is more primitive than that which is described in [3], but accomplishes that task of importing multiple IP cores at a given time. It is expected that these cores exist in a format compatible with the Xilinx implementation tools such as .EDN or .NGC. It is also expected that each core is accompanied by a VHDL interface specification file that includes an Entity that defines the IO ports to the top level interface of the core. The following instructions show an example core called “system” that contains two referenced cores called “mf_core_toplevel” and “mf_core_bram”. These three cores are described here for completeness their netlists are not attached due to licensing limitations.

The imported cores should not contain any IO buffer primitive instantiations as these will be automatically added by Viva by way of the Viva system description. If the imported core references other IP Cores in it’s hierarchy, the netlist files for these cores must be added to the ‘VivaSystem’ directory along with the imported core netlist. The importation process is executed from a PHP script. The PHP script requires that PHP be installed. The PHP installer may be downloaded from [4].

23.1.1 Files

import.bat – A DOS batch file for running the PHP script.

update_brams.bat - A DOS batch file for updating bitsream files with executable program code and data.
parse.php – The PHP script file that handles the importation process.
system.vhd – The VHDL interface specification files for the “system” core.
system.ngc – The netlist for the “system” core.
mf_core_toplevel.ngc – The netlist for a referenced IP Core.
mf_core_bram.edn – The netlist for a referenced IP Core.
system.edn – The interface specification file converted to an EDIF format.
mixed_system_fpgastrings.txt – The output netlist containing references to the imported core.
wrapper_system.ipg – The Viva “wrapper” sheet created to allow instantiation of the core within Viva

23.1.2 Process

Define the VHDL interface specification file. The file may be the top level file of a VHDL design used to synthesis the core as long as the first Entity in the file defines the top level of the design.

Run the PHP script:

```
php –q <VHDL Interface Spec File> <Input FPGAStrings File> <System Name>
```

-q puts php in “quiet” mode so that it doesn’t print PHP specific dialog to the screen.

The input FPGAStrings file is the FPGAStrings file defined by the system and is currently used by the system description for accessing primitives.

The System Name is the name of the processing element within the system description (i.e. PE1). Copy the mixed_system_fpgastrings.txt file along with all the IP core netlist files to the VivaSystem directory. Open the wrapper_system.ipg sheet and create an object from the sheet. (Note that the current version of the core import script automatically creates this file as a Viva version 2.3 file. Opening and saving this file in version 2.4 should convert it.). Create a top level sheet and instantiate the wrapper_system.ipg object. Wire the wrapper_system object to the user design logic. Enable the System Editor and edit the FPGAStrings attribute of the FPGA object to point to the mixed_system_fpgastrings.txt file and proceed with compiling the design. Optionally update the output bitstream with bram initialization data (used for microprocessor based cores to embed program code and data.).

23.1.3 Background

Viva references primitive objects in a form that does not allow vectored interfaces. For this reason, the wrapper_system sheet is created to break all vectored inputs and outputs into single bit width ports. This interface specification is defined in the mixed_system_fpgastrings.txt file as a primitive. Within this object, a second cell is referenced that does contain vectored IO. The single bit ports are wired back to vectored ports so that the core netlist may then be referenced. When Viva runs a compile. It performs a hierarchical search finding the wrapper_system object, the wrapper_system primitive, the system cell with vectored IO and finally the system.ngc netlist. Since the termination leaf of this search is the netlist, Viva passes this netlist onto the Xilinx implementation tools where the netlist is added with all the referenced netlists and built into a bitstream.

The current version of the core import script was designed with a single core import in mind. Using the script to import more than one core into a design requires some modification to the resultant fpgastrings file. This could be corrected with a modification to the import script. As a work around, if the script is run twice with the output fpgastrings file from the first execution provided as an input to the second execution, a final fpgastrings file will be created with both cores referenced. However, the netlist cell definitions for each core will exist in a separate library. The library for each will be called designs. The Xilinx implementation tools will not recognize this as correct so the cell definitions must all be cut and pasted into the same library definition in the final output fpagstrings file.

23.2 Method Used for Initializing FPGA Block Memory (BRAM)

In order to initialize the BRAMs within a Xilinx device with program code and data, the bitstream must be updated with the initialization data. The process for this within a Viva development environment requires a number of iterations. First, the design must be compiled within Viva. The Xilinx implementation tools that are executed from within Viva will create a .bit bitstream file. A Xilinx utility called data2bram can be executed to update the bitstream file with the BRAM initialization data. Data2bram requires a .bit bitstream file and a .bmm BRAM definition file. The .bmm file details the placement information for the embedded BRAMs. Here is an example command line:

```
data2bram -bm system_bd.bmm -bt system.bit -bd little.elf tag my_bram1 my_bram2  -o b download.bit
```

system_bd.bmm – Defines BRAM placement information
system.bit – FPGA bitsream file
little.elf – binary program code file
my_bram1 & my_bram2 – Block RAMs to be initialized
download.bit – Output bitstream with initialized Block RAM Data.

The .bmm BRAM definition file will be automatically generated by the Bitgen application within the Xilinx implementation tools if and only if the ngdbuild application also within the Xilinx implementation tools is executed with a similar .bmm file as an input. The .bmm file that is provided as input to the ngdbuild application defines the BRAMs used in the design but not their placement information. This file is NOT generated automatically and must be manually created with the correct BRAM definition information. This information can be acquired by browsing the compiled design using the FPGA Editor utility. The FPGA Editor utility will display the correct naming of each of the BRAMs in a mapped design so that they can be defined in the .bmm file. These can also be reference from the map report file generated by the Xilinx mapper.

The batch file update_brams.bat is included as an example. This file calls data2bram with the .bmm and .bit file to produce a download.bit file that is updated with the BRAM initialization data and ready to download to the FPGA.

24. GMIP Evaluation Tests and Results

24.1 Demonstration #1: App X -- MIL-STD-1553B BC/Signal Filter#1

Figure 1 shows the set-up configuration for demonstration #1. In this set-up a data generation and 1553 bus controller application generates MIL-STD-1553B bus traffic based on inputs from the development PC over the RS-232 interface. The MIL-STD-1553B bus traffic is implemented with the Condor Engineering IP core and is used to drive a PDP display as shown. In parallel to this process a daughter-card on the GMIP does an A/D of an incoming analog signal. The GMIP implements a filter algorithm application and then does a D/A to get the signal back to the analog domain. The two signals can be compared on the oscilloscope to determine correct operation of the signal filter.

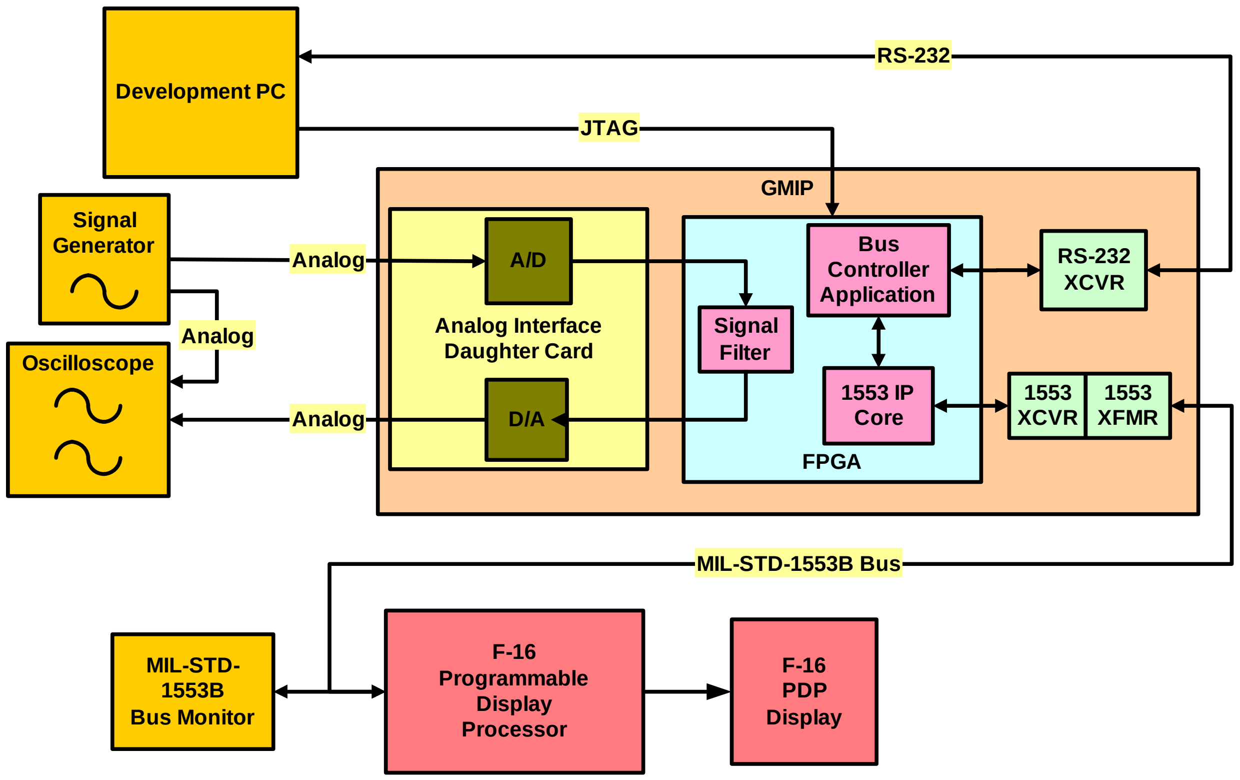

Figure 1. Demonstration #1 and #2 configuration

24.2 Demonstration #2: AppX ➔ App Y (Change Signal Filter from #1 to #2)

Demonstration #2 will be the same basic configuration as demonstration #1 (see Figure 1). This demonstration will show the utility of using VIVA to make an application change. In demo #2 the signal filter implementation will be changed using the VIVA development environment and a re-compilation and download will be done. This will show the complete process for the modification, integration and download using the GSE.

24.3 Demonstration #3: EBR-1553 Bus Controller/Remote Terminal

Figure 2 shows the set-up configuration for demonstration #3. This test set-up includes the use of two GMIPs. GMIP # 1 is acting as an EBR-1553 bus controller and GMIP #2 is an MMSI remote terminal. In this demonstration an A/D converter on GMIP #1 is used to generate data from a waveform. A GMIP #1 FPGA application collects this data, formats it for MMSI transmission and then uses an MMSI interface (IP core) to transmit the data. GMIP#2 receives the MMSI and reconstitutes the data for presentation to a digita-to-analog (D/A) converter. The output of the D/A is provided to an oscilloscope where it can be compared to the original waveform. This test shows a complete data flow thread through the EBR-1553 interface along with the real-time latency present within the interface. This test set-up is used to validate correct implementation of both an MMSI BC interface and an MMSI RT interface.

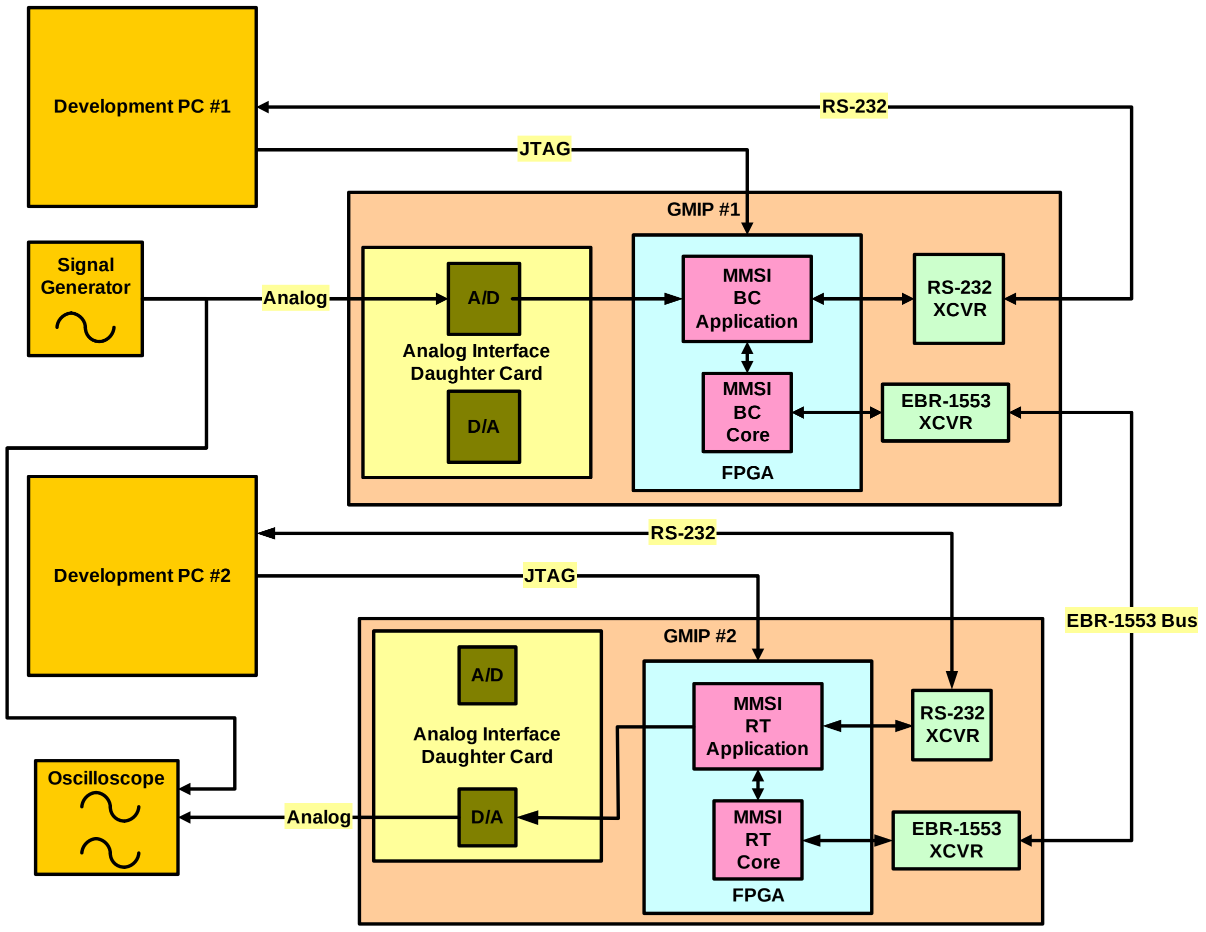

Figure 2. Demonstration #3 configuration

24.4 Demonstration #4: Dynamic reconfiguration

The GMIP features the ability to perform self-reconfiguration of the FPGA to support multiple mission objectives utilizing the same hardware resources. This capability is demonstrated by loading several different bit-streams into the System ACE MPM Reconfiguration Controller that are operation/functioned to identify themselves by bit-stream number across the RS-232 interface, assign a "next configuration" bit-stream for reconfiguration and perform self reconfiguration to cycle through all the bit-streams continuously.

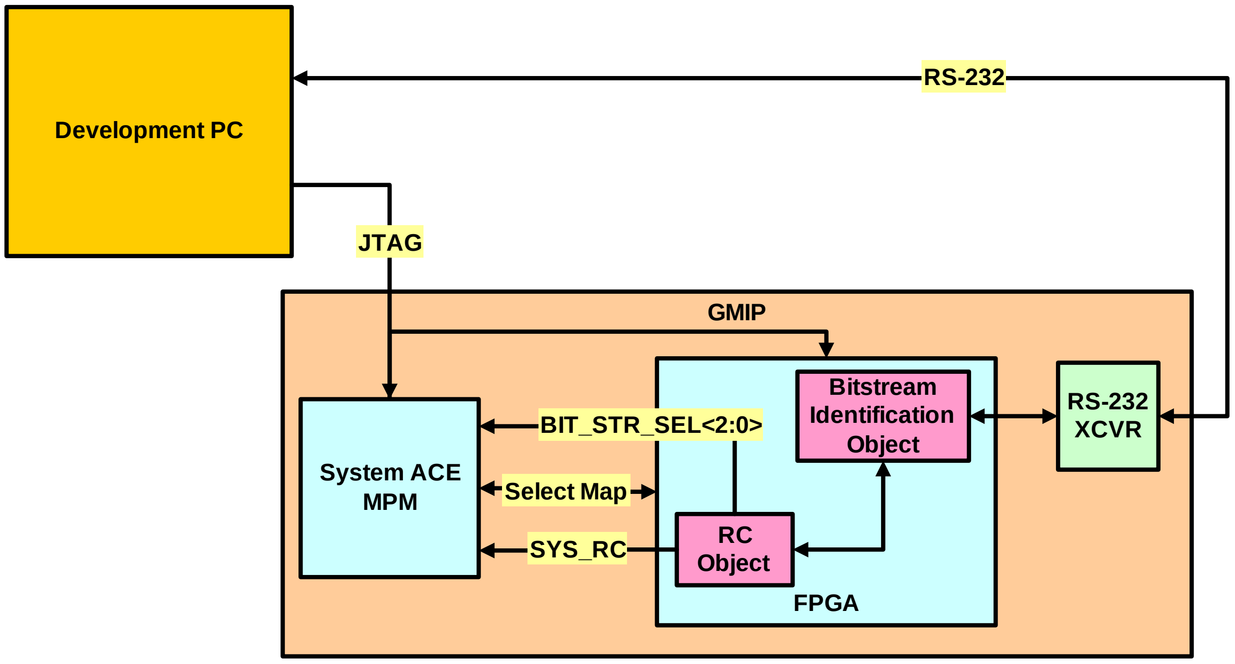

Figure 3. Demonstration #4 configuration

25. Future Work

This development board will be used for research in aircraft interfacing and processing towards "plug and play" technology. The goal would be to have a development platform to support research to maximize load-out.

25.1 Develop GMIP daughter-cards to support additional research

Example daughter-card/functions to define:

- Wireless (802.11, Bluetooth, 3G, etc)
- Video – to support capture of digital video and display on SVGA or better display
- Other daughter-cards may be identified based on future requirements

25.2 Implement demonstration munitions applications using GMIP

Reconfigurable computing using a Field Programmable Gate Array (FPGA) provides a very cost effective way to provide the increased processing power required for many of the functions that will be needed in near-term and long-term avionics embedded systems [1]. This increased processing power will lead to more effective individual load-out per host platform. Before this type of technology can be used in the field, these functions need to be demonstrated in the lab and compared to more traditional processing implementations. There will be more research in identifying ideal applications for hosting on the GMIP to demonstrate the utility of reconfigurable/parallel processing power of an FPGA based processor/interface card.

26. References

[1] Richard N. Pedersen. "Very High Performance Computing For Military Avionics Applications Using FPGAs", 22nd *Digital Avionics Systems Conference*, October 2003.

[2] Xilinx, Inc. *Virtex-II Pro Platform FPGA Handbook.* UG012 (version 1.0). January 2002.

[3] Sreesa Akella et. al. "Porting EDIF Netlists to the Viva Environment for Integrated Custom Computing Applications", *11th Military and Aerospace Programmable Logic Devices Conference*, September 2003.

[4] PHP: Hypertext Preprocessor. http://www.php.net