ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN BE-CO-HT

Gennum GN4124 to Wishbone bridge

April 2010

Edited by :
Simon DEPREZ

Checked by :

Abstract

Gennum is providing an IP core that can be used for hardware design with the chips GN412x. This documentation describe how to add a simple wishbone master to the IP and use it for a specific application.

Revision History

Version	Date	Notes
0.1	10-03-2010	Created
0.2	01-03-2010	First draft
0.2	14-04-2010	Add Wishbone master description
0.3	26-04-2010	Add Makefile changes
0.4	27-04-2010	Add Wishbone slave description

Contents

List of Figures

1 Gennum documentation

- GN412x PCI Express family Reference Manual (52624-0) :
  - Overview (p 11–12),
  - Timing diagrams (p 41, 52–55),
  - DMA core diagram (p 84);
- GN412x FPGA IP Hardware Design Guide (51860-2) :
  - Overview (p 7),
  - DMA sequencer limitations (p 9),
  - DMA core diagram (p 22),
  - Project register map (p 50);
- GN412x RDK Software Degign Guide (51859-1) : Describes the software furnished with the development kit;
- GN412x Simulation Test Bench User Guide (53716) : Tutorial for the simulation environment;
- GN412x BFM Reference Manual : Describes the Bus Functional Model used for the Testbench.

2 Wishbone master

This project is based on the Gennum FlexDMA IP core.

2.1 Application Attachment layer modifications

The Application attachment layer is the main part of the Gennum core. It contains DMA masters and the target controller. A Wishbone master interface is added.

2.1.1 Arbiter

Add an entry for `wbmaster` module. The file `arb.v` is modified to accept packets of `wbmaster` module. The added signals are :

- `wbm_arb_data` : input;
- `wbm_arb_valid` : input;
- `wbm_arb_dframe` : input;
- `wbm_arb_req` : input;
- `arb_wbm_gnt` : output.

2.1.2 Packet decoder

The packet decoder (`packet_decoder.v`) is not modified.

2.1.3 Wishbone master module

A Wishbone master based on Gennum Target module (`tar.v`) is added. It support non-burst read and write of 32 bit data.

Addresses between 100h and 1FFh are routed to the wishbone master (see `dma.v`). Can be redefined.

Wishbone signals are added to the application attachment layer entity (`dma.v`) :

- `wb_adr_o` : output;
- `wb_dat_i` : input;
- `wb_dat_o` : output;
- `wb_cyc_o` : output;
- `wb_sel_o` : output;
- `wb_stb_o` : output;
- `wb_we_o` : output;
- `wb_ack_i` : input.

This signals are also present in `wbmaster` module and replace original target interface(see `tar.v`).

When a request is addressed to the wishbone master, the module wait for an acknowledge or timeout. During the wishbone cycle, other requests are ignored.

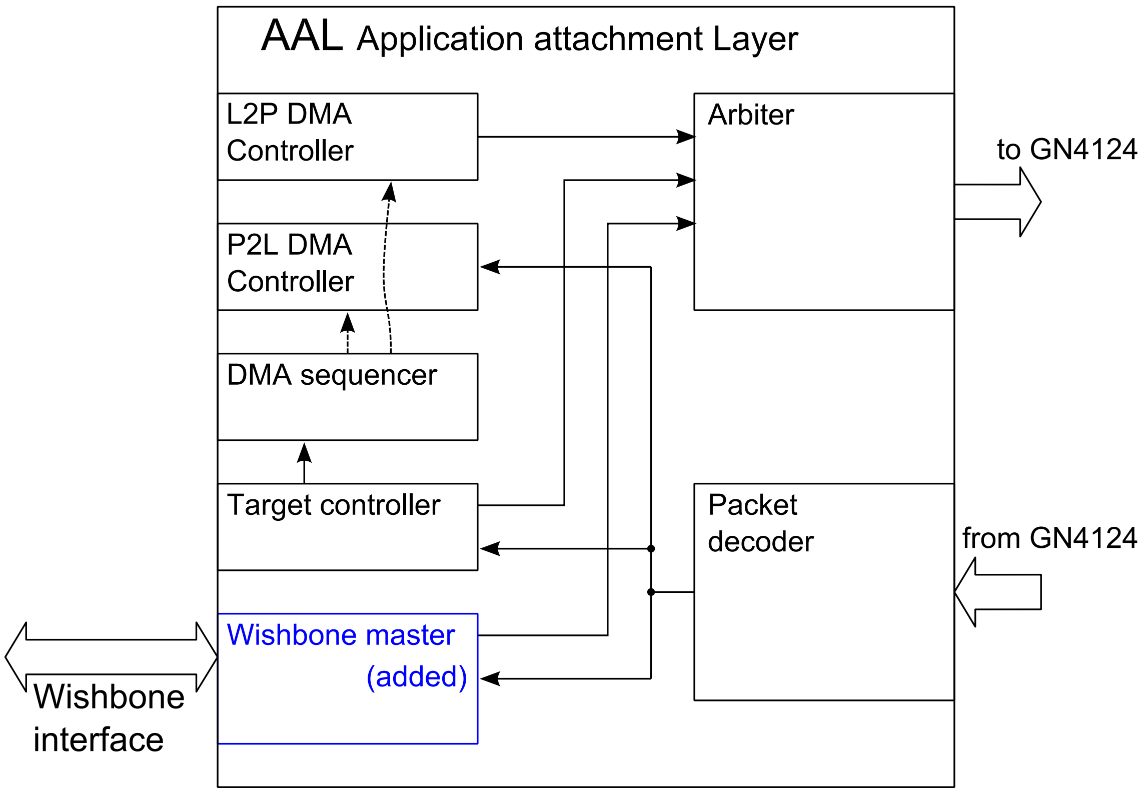

Figure 1: Gennum core with Wishbone master

2.1.4 Wishbone slave

A Wishbone slave module (`wb_debug.vhd`) was generated with `wbgen2` script with 3 registers of 8 bits for tests. It was connected to debug leds and buttons of the gennum kit and to the wishbone master.

The input description file for `wbgen2` is `wb_debug.wb` :

```
-- here comes our peripheral definition
peripheral {
-- short (human-readable) name for the peripheral.
  name = "Gennum card debug Wishbone slave core";
-- a longer description, if you want
  description = "A 8-bit output port";
-- name of the target VHDL entity to be generated
  hdl_entity = "wb_gpio_debug";
-- prefix for all the generated ports belonging to our peripheral
  prefix = "gpio";

-- Ouput register
  reg {
    name = "LED port";
    description = "Output port";
    prefix = "led";
    field {
      name = "Port output value";
      description = "Reflects the state of LEDs.";
      type = SLV;
      size = 8;
      access_bus = READ_WRITE;
      access_dev = READ_ONLY;
    };
  };

  reg {
    name = "DEBUG port";
    description = "Input port";
    prefix = "debug";
    field {
      name = "Port input value";
      description = "Reflects the state of the DEBUG switch.";
      type = SLV;
      size = 8;
      access_bus = READ_ONLY;
      access_dev = WRITE_ONLY;
    };
  };

  reg {
    name = "Test register";
    description = "Test register";
    prefix = "test";
    field {
      name = "Test register";
      description = "Test read and write register";
      type = SLV;
      size = 8;
      access_bus = READ_WRITE;
      access_dev = READ_ONLY;
    };
  };
};
```

3 Gennum Simulation Testbench

3.1 Some issues

3.1.1 Path error

The file `modelsim.ini` contain an absolute path reference to the file `vlog.opt` at line 256.

Change the line to :

```
OptionFile = vlog.opt
```

3.1.2 Error with address in BFM scripting using C

The `long long int` variables make errors with `printf` command if the MinGW (Windows) compiler is used. Replace `%016llX` by `%016I64X` in the file `./Test_Builder/lib/model.c` at lines 64, 73, 288 and 298.

3.1.3 Makefile

Rules are added for both Wishbone master `wbmaster.v` (line 51-53) and wishbone slave `wb_gpio.vhd` (line 49-51).

Dependencies are added in the top module (`lambo.v`, line 45) rule in the Makefile.

3.1.4 BFM script

Gennum provide a Testbench environment that emulate the local bus between the GN4124 chip and the FPGA. The Bus Functionnal Model is used to simulate PCI acces for de FPGA system.

The file `wbmaster_test.c` contains a script with read and write requests for the wishbone bus.

3.2 Simulation

Read the GN412x Simulation Test Bench User Guide (53716).

The file `./Test_builder/wbmaster_test.c` is a simple BFM script used for testbench with the wishbone master.

Open a terminal and use the command :

```
test wbmaster_test
```

In Modelsim, run the simulation :

```
do lambo.do
```

4 DMA controller

4.1 Gennum Pinto project

This project contain a DDR2 ram controller used for DMA transfer.

The project must be migrated to a new version of Xilinx ISE. If the file `C:\pinto\pinto.xcf` cannot be found by the synthesizer, change the path by click-right on `Synthesize - XST` and choose `Process properties`.

Summary

The wishbone master is simple and allow non-burst read or write of a 32 bit data.