European Organization for Nuclear Research
Organisation Europeenne pour la Recherche Nucleaire

wbgen2

A simple Wishbone slave core generator

Version: **20100223**

Tomasz Włostowski
CERN BE-Co-HT

1 Introduction

wbgen2 is a Lua script for generating VHDL Wishbone slave cores from a register set description provided by the user. By the "slave core" we mean a HDL entity which is connected to Wishbone bus on one side, and on the other side it provides ports for accessing memory mapped registers, FIFOs and RAMs, as shown on figure 1.

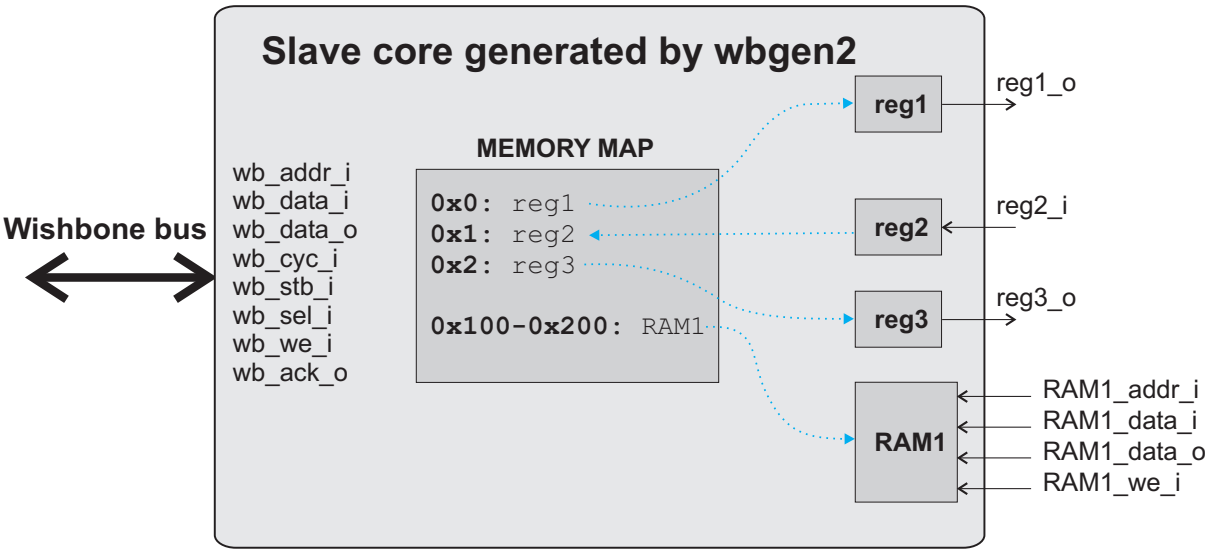

Figure 1: An example of wbgen2-generated slave core

The main features of wbgen2 are listed below:

- Generation of VHDL code for slaves consisting of memory mapped registers, FIFO registers and RAMs
- Automatic minimal address space generation
- Generation of C header files containing addresses consistent with the VHDL core
- Customizable register types, with multiple access options and multiple clocking schemes
- Support for most common VHDL types
- (optional) automatic instantiation and wiring of slave core into the VHDL design
- (optional) documentation generator.

2 wbgen2 slave description files

In order to generate the slave core, user must provide a slave description file, which tells wbgen2 what he wants to have inside the core. Each slave description file contains a tree-like structure, describing the peripheral's register layout for a single Wishbone peripheral.

Slaves may contain registers, FIFO registers and RAM memories. Registers and FIFO registers consist of fields (see figure 2), RAMs are accessible as plain, synchronous memories.

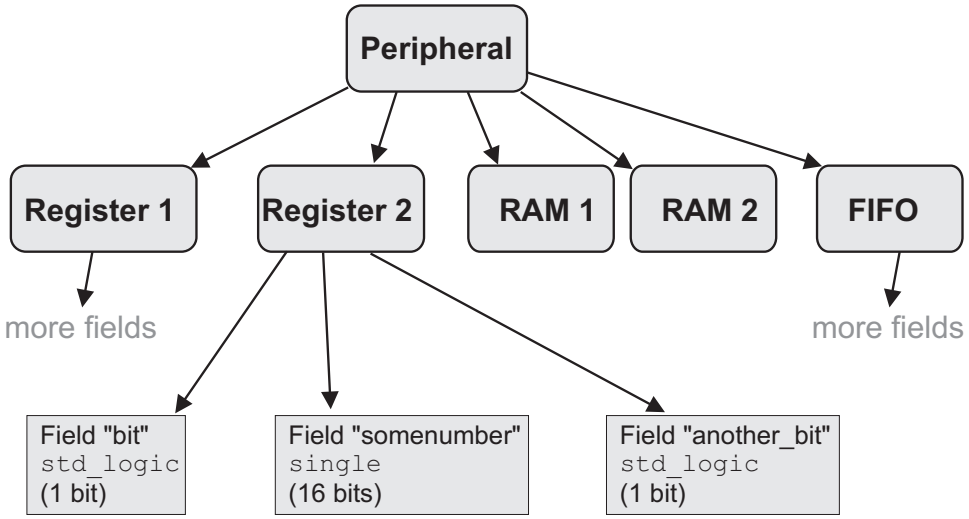

Figure 2: Structure of slave cores generated by wbgen2

2.1 Slave description syntax

Slave description files have C-like syntax. Each file may contain the description of a single slave core. The description must begin with *peripheral* block, which contains one or more of *reg*, *fiforeg* or *ram* subblocks. Each *reg* and *fiforeg* subblock must contain at least one *field*. Inside each block, there is a list of attributes. The listing below shows a dummy description file layout:

```
peripheral {
  name = "My peripheral";
  c_prefix = "periph1";
  hdl_prefix = "periph1";
  reg {
    name = "My register";
    prefix = "myreg";
    description = "A longer description";
    field {
      name = "My field 1";
      prefix = "field1";
      type = type-of-the-field; // BIT, SLV, etc...
    };
-- more fields here...
  };
  fiforeg {
    name = "My FIFO register";
    prefix = "myram";

    field { ... };
  };
  ram {
    name = "My RAM";
    prefix = "myram";
    size = 1024;

  };
};
```

2.2 Common attributes

There are few attributes, which are common for all types of blocks in the description file:

name (mandatory)	contains a short (single line) human-readable name for the peripheral/register/field. This name is not used directly in code generation (except for the code comments).
c_prefix **hdl_prefix** **prefix** (mandatory)	contains a short prefix for each block which is used for generation of VHDL port/signal names and C macros. Names are generated by concatenating the prefixes. In the example shown above, the signal name of field "My field 1" would be *periph1_myreg_field1*. The format of **prefix** value must follow the language syntax rules and your coding style. Note that you can provide either separate prefixes for C/HDL languages (**c_prefix, hdl_prefix**) or a single **prefix** for both.
description (optional)	a longer description of the block, used by the documentation generator.

Table 1: Attributes common for all description blocks

2.3 PERIPHERAL block attributes

Peripheral block is the top-level block in the description file.
Block-specific attributes:

hdl_entity (mandatory)	name of the VHDL entity of the slave core to be generated.

Table 2: Attributes specific for **peripheral** blocks

2.4 REG block attributes

Reg block describes a single memory-mapped register. Each **reg** block must contain one or more **field** blocks. Available field types are listed in table 4.
Block-specific attributes:

align = val (optional)	Alignment value for the field address. When given, wbgen2 will align the address of this register to the nearest multiple of *val*. See also figure 3.

Table 3: Attributes specific for **reg** blocks

Type	**Description**
BIT	VHDL single bit of type `std_logic`
SLV	VHDL field of type `std_logic_vector`
SIGNED	VHDL field of type `signed`
UNSIGNED	VHDL field of type `unsigned`
MONOSTABLE	VHDL field of type `std_logic`, generating a single-cycle positive pulse upon bus write of '1'.

PASS_THROUGH	special field, for which wbgen2 will generate only the address decoding logic which provides "wr" signal asserted high for a single WB clock cycle upon each write to the register. The written value will be fed to the corresponding SLV output directly from the Wishbone bus (just wires, no registers in between).

Table 4: Possible field types for **reg** blocks

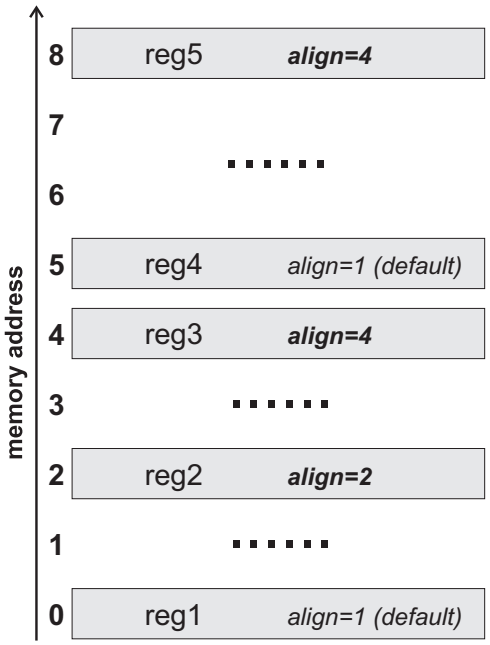

Figure 3: Register alignment

2.5 FIELD block

Field block describes a single register field. There are several types of fields, shown in table 4. It's the most elementary block in the design.
Block-specific attributes:

type (mandatory)	type of the field. See table 4
size (mandatory for: SLV, optional for SIGNED, UNSIGNED)	size of the field in bits. For SIGNED and UNSIGNED types it's interchangeable with **range** attribute.
range = {min,max} (optional for SIGNED, UNSIGNED)	minimal and maximal field value. When provided, wbgen2 will automatically calculate the necessary number of bits.

access_bus **access_dev** (optional)	field access flags. **access_bus** defines how the field can be accessed from the Wishbone bus, **access_dev** defines how the field can be modified by the HDL entity in which the slave core is instantiated. Access flags can have one of these values: READ_ONLY, WRITE_ONLY, READ_WRITE. For the possible access combinations refer to table **??**. The default value is READ_WRITE (from the bus) and READ_ONLY (from the device).
access (optional)	can be used instead of **access_bus** and **access_dev** to define access rights. See table **??**.
align = num (optional)	when given, the bit offset at which field will be allocated, will be aligned to integer multiple of **num**. Default value is 1 (no alignment).
clock (optional)	can be used to provide a clock port name if the field needs to operate in clock domain other than Wishbone bus clock. wbgen2 will automatically provide the necessary synchronization logic. Clock names are automatically appended to slave core entity port list. If no **clock** attribute is provided, wbgen2 defaults to Wishbone bus clock.
load (optional for RW/RW fields)	this attribute is applicable only to RW/RW-accessed fields (e.g. the fields which are writable both from the bus and the device). There are two possible values, indicating where the field register will be physically placed (see figure **??**): *LOAD_EXT* - the field register is placed outside the slave core. Upon bus write operation, slave outputs the new value to the output port and asserts the "load" signal for a single clock cycle. The device has to handle these signals and update the value of the register respectively. *LOAD_INT* - the field register is placed inside the slave core. When the device wants to update it's value, it passes it to certain input port and asserts the load signal high. *not implemented yet.*

Table 5: Attributes specific for **field** blocks

3 Registers

wbgen2 supports various types of memory mapped I/O registers and fields:

- SLV, SIGNED, UNSIGNED and BIT standard registers
- Bus-synchronous (operating with the same clock as the WB bus) or asynchronous (using externally supplied clock). wbgen2 automatically provides all necessary synchronization logic.
- MONOSTABLE registers which generate positive pulse upon write of '1'. '
- PASS_THROUGH registers
- Different access configurations
- Single register can contain fields of different types, clocks and access.

Figures 4 and **??** show all possible register field types.

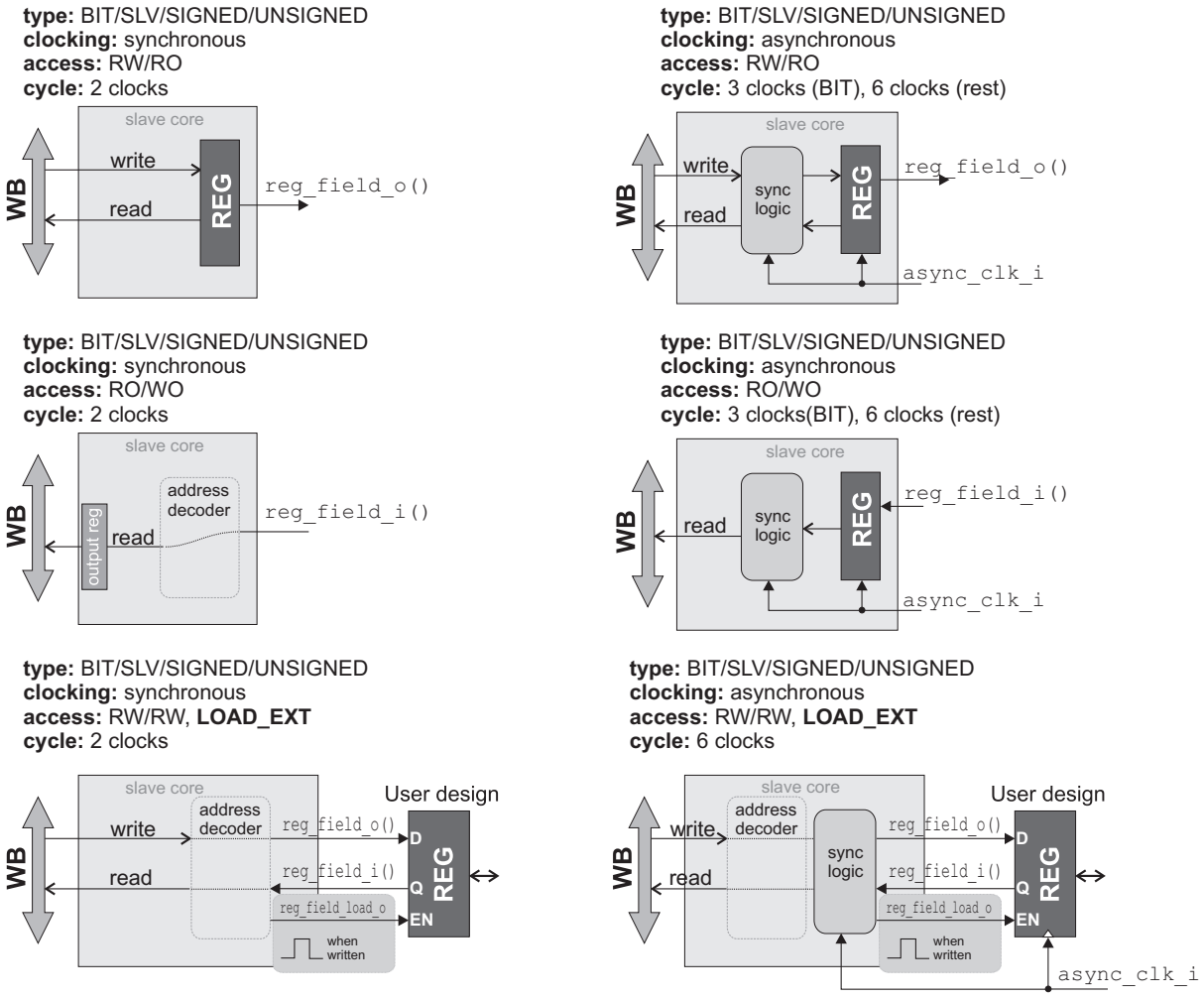

Figure 4: wbgen2 register type cheatsheet - standard registers