"Software product that requires user guide will not sell" - Amir

Register File Generator

The goal is to make simplest to write/use/maintain register file generator.
"Less is more".
The input format is text file, editable by vim or emacs. Dont allow complex directives. Use minimal number of directives. Most registers defined in one line. While register can have fields, don't define just one field. It is almost stupid.

Registers and Fields appear on the header under their names. Unique fields keep their names. If You must use same field names in several registers, Register name will be prefixed. Unless suffix/prefix is defined from that register.

chip line

First line in any register definition file is chip line.

- **chip** Regfilename **apb wid**=32 **addrwid**=20 **reset**=async **empty**=0xdeadbeaf
  wid= is data bus width
  addrwid= defines address bus width
  reset= can be async or sync
  empty= defines data returned on read from not used location.

register line

- **reg** Regname **wid=**32 **access**=rw **reset**=0x232 **desc**=Descr **ready**=true **prefix**=aa **suffix**=aa **fields**=external

REG parameters

width (or wid) is register width. It can be any number, including big numbers like 500. In case there are fields, wid parameter is not required.
access can be rw (or wr) , ro , rw_pulse, ro_pulse, w1c or external, added now also "wo" : for encryption keys, which you dont want to be read by software.
access=w1c is clear on write ones. Incoming read-only bus sets (by ORing with register)) bits in flipflop register. reading it does nothing. writing to it will clear all bits active in PWDATA.
access=external output read/write pulsed strobes, dataout and datain. It is "DIY register".
description Has either no spaces, or enclosed in "..." .
reset is relevant to writable registers.

ready=true is relevant to ro/rw_pulse registers. specific ready signal to this reg will stall the APB.
prefix or **suffix** is relevant to fields of this register, if the field can be double used.
fields=external causes the fields split to be written in separate file. It makes easier to pass wide busses through hierarchies.

FIELDS

field lines follow their register line.
Fields dont have access. The access (read/write/...) is determined by the register.
Fields should have width parameter and may have description.

field Name **wid**=8 **align=**16 **desc**=DESC
align is misnomer. It is the number of bits the field will be allocated in register. Active bits are defined by wid (or synonym width). Usefull for keeping fields on byte boundaries.

field gap (field with name "gap") acts also as empty bits insertion.

Please refrain from defining REG with a single FIELD. it is ugly, not to say smart challenged.

TEMPLATE and INSTANCE

1. **template** Tname
   reg ...
   reg ...
   reg ....
   endtemplate

Template works like macro.

2. **instance** Tname Iname
   instance will expand the template, where all registers and fields will be prefixed with Iname.

GAP

gap align=1024 **abs**=0x1000 **wid=**32
defined jump in address. Only one flag per gap is allowed : align ,abs or wid.

EXTERNAL

ram Name **wid**=32 **depth=1024 desc**=Description
Ram adds interface to external ram module. it adds read/write/addr/rdata/wdata ports.
Width of ram cannot be wider that apb bus.
reg Name **access=external wid**=23

similar interface to ram, just without depth.

Files generated

to run it : **pybin/regfile.py** *SOME*.**regfile**

Verilog RTL file is generated. Additional several formats are generated - include files and documentation.

look for examples in examples/ directory.

Here is one example:

```
chip ready_rgf apb reset=async wid=32 addrwid=16 empty=0xdeadbeaf
# this is comment
reg ronly access=ro_pulse wid=32 ready=true
reg ronly2 access=ro_pulse wid=16 ready=true
reg rega wid=32 access=rw reset=0x123456 desc="first register to work"
// this is also comment
reg control0 wid=32 access=rw reset=0xcc00 desc=control.register
reg statusa wid=24 access=ro
field dma0 wid=2 align=8
field cpu1 wid=8
field spi0 wid=3 desc=spi.status

reg regb wid=24 access=rw_pulse reset=0x0
field odma wid=2 align=8
field ocpu wid=8
field ospi wid=3 desc=spi.status

end
```

POINTS

- register can have unlimited width. It will be split between several addresses as needed. No coherency is built in.
- access rw_pulse will generate pulse when last location of register is written. This can be used to push to a fifo or enforce coherency.
- access ro_pulse also produces pulse when last location is read. To catch coherency start reading from the last address and use the pulse to catch the whole bus.

questions/suggestions: Ilia greenblat@mac.com