example_module_ro

Address width: 32

Data width: 32

Base address: 0xFFAA0000

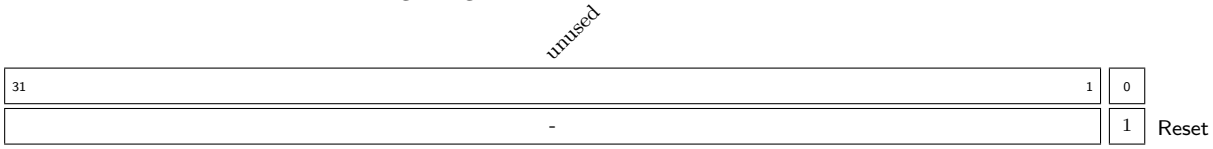
An example module that contain exclusively RO registers.

1 Register List

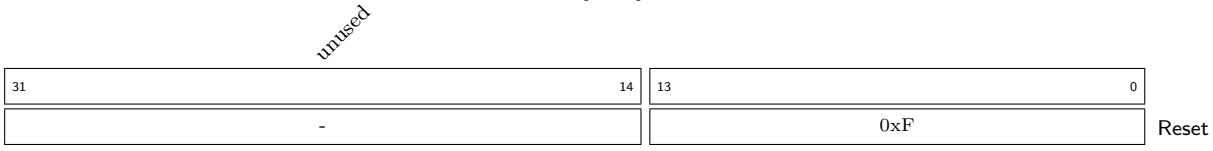
#	Name	Mode	Address	Type	Length	Reset
0	reg0	RO	0x00000000	SL	1	0x1
1	reg1	RO	0x00000004	SLV	14	0xF
2	reg2	RO	0x00000008	DEFAULT	32	0x0
3	reg3	RO	0x0000000C	FIELDS	24	0x0

2 Registers

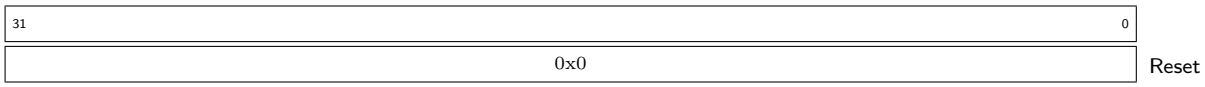
Register 2.1: REG0 - RO (0x00000000)
RO std_logic register that resets to 0x1 for some reason



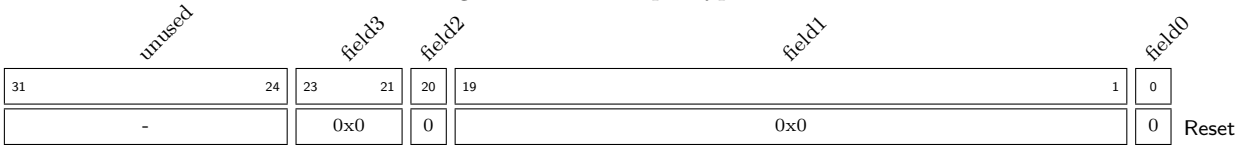
Register 2.2: REG1 - RO (0x00000004)
RO std_logic_vector[13:0] that resets to 0xF



Register 2.3: REG2 - RO (0x00000008)
Default RO register



Register 2.4: REG3 - RO (0x0000000C)
RO register with multiple types of fields



field0 std_logic field

field1 std_logic_vector[18:0] register

field2 std_logic field

field3 std_logic_vector[2:0] field

3 Example VHDL Register Access

All registers are bundled in records based on their mode. E.g. all RW registers are accessed through the record *bustype_rw_regs*. Access is also dependent on the type of register. All register of type SL, SLV and DEFAULT are all directly accessed by just specifying the mode record signal. E.g. the RW register *reg0* can be assigned a value like this (assuming AXI-bus):

```
axi_rw_regs.reg0 <= (others => '0');
```

Registers of type FIELD cannot be directly accessed without specification of a certain field. This is because the registers are implemented as a record in VHDL (thus a record of records). E.g. if the RO register *reg1* contains the field *field3* it can be accessed like this (assuming AXI-bus):

```
axi_ro_regs.reg1.field3 <= (others => '0');
```