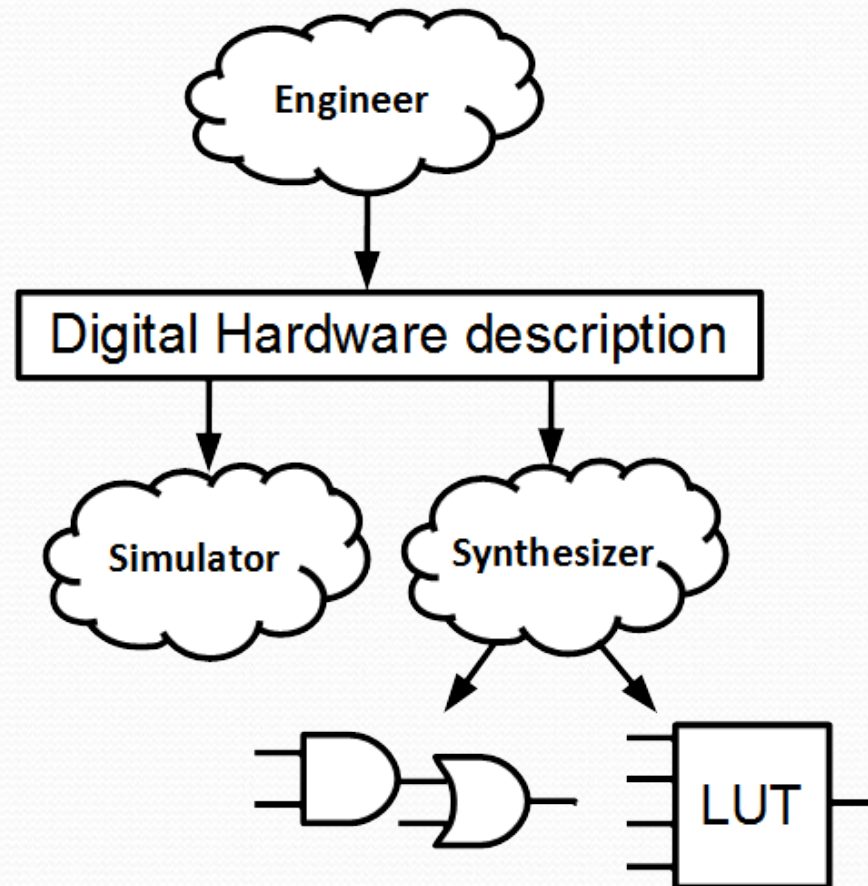




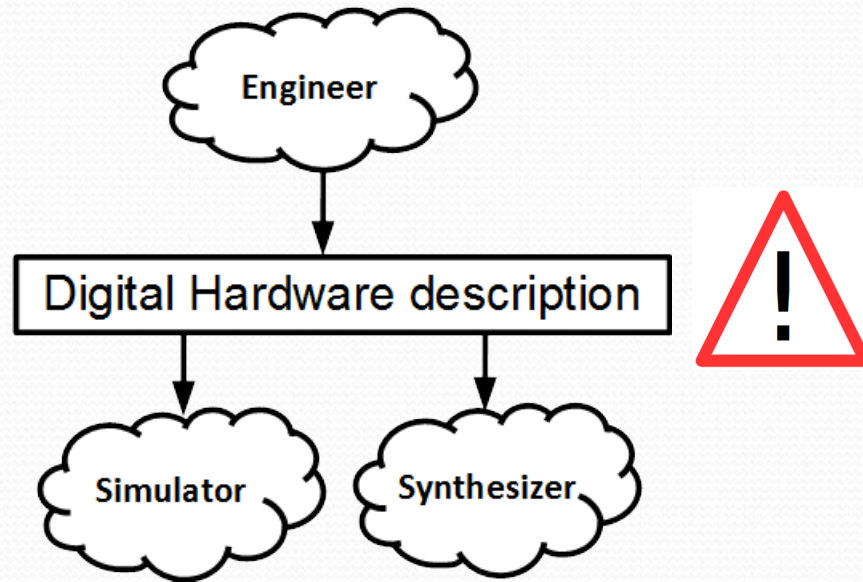
SpinalHDL

An alternative hardware description language

The Digital hardware flow



Issue



- The digital hardware description is done by using VHDL or Verilog
 - They are relatively low level
 - They are very verbose
 - They were not designed for this purpose

Solution Idea

- Using an intermediate hardware description language which
 - Generates VHDL/Verilog
 - Is dedicated for this purpose
 - Introduces new description paradigms
 - Is free and open source

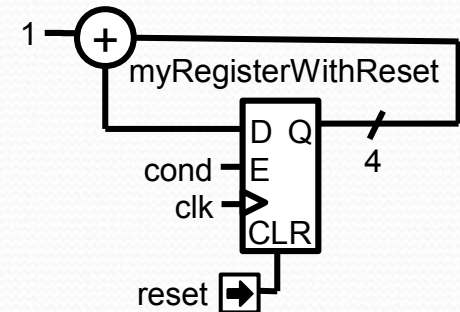
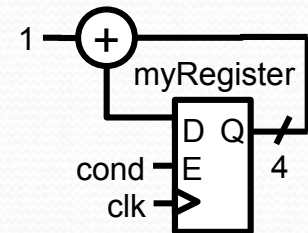
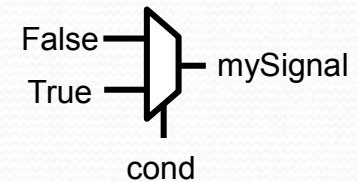
SpinalHDL introduction

- Open source , started in december 2014
- Focus on RTL description
- Thinked to be interoperable with existing tools
 - It generates VHDL/Verilog files
 - Existing IPs can be integrated as blackboxes
- Abstraction level :
 - You can design things similarly to VHDL/Verilog
 - If you want to, you can use many abstraction utilities and also define new ones

Dedicated syntax

```
val mySignal           = Bool
val myRegister         = Reg(UInt(4 bits))
val myRegisterWithReset = Reg(UInt(4 bits)) init(0)
```

```
mySignal := False
when(cond) {
  mySignal           := True
  myRegister         := myRegister + 1
  myRegisterWithReset := myRegisterWithReset + 1
}
```



SpinalHDL => 10 lines

VHDL => 30 lines

Interface support

```
val config = Axi4Config(addressWidth = 32,  
                        dataWidth    = 32,  
                        idWidth     = 4)  
val axiBus = Axi4(config)
```



```
signal axiBus_aw_valid : std_logic;  
signal axiBus_aw_ready : std_logic;  
signal axiBus_aw_addr : unsigned(31 downto 0);  
signal axiBus_aw_id : unsigned(3 downto 0);  
signal axiBus_aw_region : std_logic_vector(3 downto 0);  
signal axiBus_aw_len : unsigned(7 downto 0);  
signal axiBus_aw_size : unsigned(2 downto 0);  
signal axiBus_aw_burst : std_logic_vector(1 downto 0);  
signal axiBus_aw_lock : std_logic_vector(0 downto 0);  
signal axiBus_aw_cache : std_logic_vector(3 downto 0);  
signal axiBus_aw_qos : std_logic_vector(3 downto 0);  
signal axiBus_aw_prot : std_logic_vector(2 downto 0);  
signal axiBus_w_valid : std_logic;  
signal axiBus_w_ready : std_logic;  
signal axiBus_w_data : std_logic_vector(31 downto 0);  
signal axiBus_w_strb : std_logic_vector(3 downto 0);  
signal axiBus_w_last : std_logic;  
signal axiBus_b_valid : std_logic;  
signal axiBus_b_ready : std_logic;  
signal axiBus_b_id : unsigned(3 downto 0);  
signal axiBus_b_resp : std_logic_vector(1 downto 0);  
signal axiBus_ar_valid : std_logic;  
signal axiBus_ar_ready : std_logic;  
signal axiBus_ar_addr : unsigned(31 downto 0);  
signal axiBus_ar_id : unsigned(3 downto 0);  
signal axiBus_ar_region : std_logic_vector(3 downto 0);  
signal axiBus_ar_len : unsigned(7 downto 0);  
signal axiBus_ar_size : unsigned(2 downto 0);  
signal axiBus_ar_burst : std_logic_vector(1 downto 0);  
signal axiBus_ar_lock : std_logic_vector(0 downto 0);  
signal axiBus_ar_cache : std_logic_vector(3 downto 0);  
signal axiBus_ar_qos : std_logic_vector(3 downto 0);  
signal axiBus_ar_prot : std_logic_vector(2 downto 0);  
signal axiBus_r_valid : std_logic;  
signal axiBus_r_ready : std_logic;  
signal axiBus_r_data : std_logic_vector(31 downto 0);  
signal axiBus_r_id : unsigned(3 downto 0);  
signal axiBus_r_resp : std_logic_vector(1 downto 0);  
signal axiBus_r_last : std_logic;
```

SpinalHDL => 4 lines

VHDL => 39 lines

Other features

- A proper support of
 - Design parameterization
 - Functions definition
- Many integrated checks
 - No latch
 - No combinatorial loops
 - No unwanted clock domain crossing
- And also
 - Object Oriented programming
 - Functional programming
 - Possibility to define new abstraction level
 - Meta-hardware description capabilities
 - Free IDE to make things easier

The technology



- SpinalHDL is an internal DSL (Domain Specific Language)
- Advantages
 - All feature the host language are inherited
 - All tools the host language are inherited
 - It makes the SpinalHDL compiler simpler
- Scala as host programming language
 - Flexible enough to have a natural syntax
- How it works
 - You compile and run your SpinalHDL hardware description
 - Each usage of SpinalHDL syntax contribute to build a netlist
 - Then this netlist go through some transformation and checks
 - Finally it is flushed into a VHDL or a Verilog file



Our buisness model

- No fees to use the language
 - No licence
 - No royalties
- We provide the commercial support
 - Presentation
 - Training
 - Consulting

Some links

- Completely open source :
 - <https://github.com/SpinalHDL/SpinalHDL> 
- Online documentation :
 - <https://spinalhdl.github.io/SpinalDoc/>
- Ready to use base project :
 - <https://github.com/SpinalHDL/SpinalBaseProject>
- Communication channels :
 - spinalhdl@gmail.com
 - <https://gitter.im/SpinalHDL/SpinalHDL> 
 - <https://github.com/SpinalHDL/SpinalHDL/issues>