SpinalHDL intro for new users

Andreas Wallner (andreasWallner)

About me

- **Me: Andreas Wallner**
- **> 10 years at Infineon**
  - Whatever I say is my private opinion and not connected to my employer
- **GitHub / Gitter: andreasWallner**

SpinalHDL setup

- **Java, scala & sbt**
- **For simulation/formal we recommend**
  - Verilator
  - oss-cad-suite
- **Recently updated:**

https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Getting%20Started/Install%20and%20setup.html

Components

- The base of all hardware
- has inputs/outputs
- contains hardware
- Can be instantiated in other components/top

Biggest differences to classical HDLs

- **(Generative library)**
  - Running our “program” (generator) “elaborates” our netlist
- **Implicit clock domains (clock, reset, power, …)**
  - One default CD
  - Always one current CD
- **Wire vs. Register is explicit**
  - Latches detected during generation
- **Strictly typed**

Coding

- **Get SpinalTemplateSbt**
  - https://github.com/SpinalHDL/SpinalTemplateSbt
  - Adapt project/package name & sbt settings
  - Run in sbt
    - `runMain intro.Basics`
    - `~runMain intro.DefaultClockDomain`
- **`Let's make a counter and simulate`**
    - `runMain intro.CounterSim`

Operators

Op	“world”	Usage
=	scala	Assign scala variables – generator runtime
:=	hardware	Hardware signal assignment – creates HW connections
<>	hardware	Intelligent hardware assignment
#=	simulation	Drive value during simulation
>> >> </<...	?	… type dependent
/=	hardware	Hardware

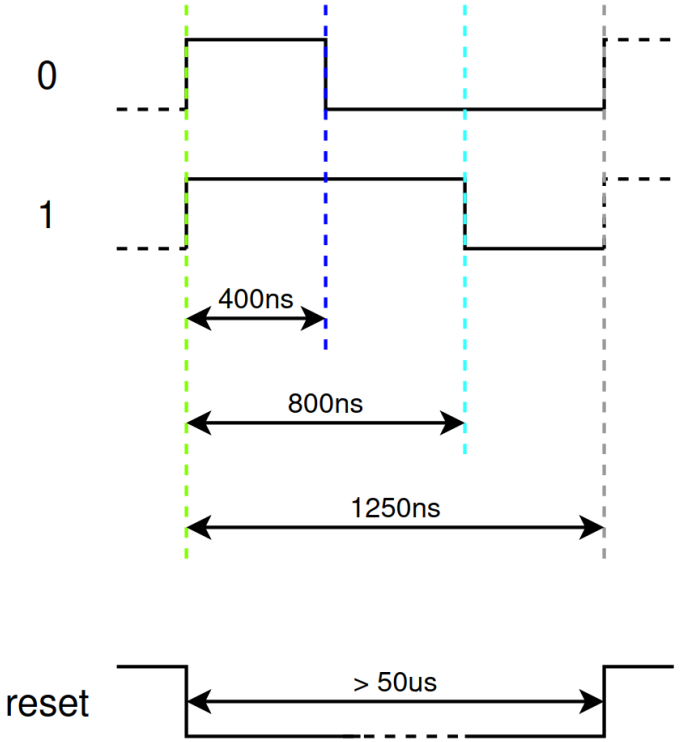
https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Semantic/assignments.html#assignments

Types

- **scala: `Boolean`, `Int`, `Long`, `BigInt`, `String`,...**
- **spinal**
  - Bool, Bits (`std_ulogic[_vector]`, `wire/reg`)
  - UInt, SInt, ... (`[un]signed`, `wire/reg`)
  - SpinalEnum (enumerated type, SV `enum`)
  - Bundle (`~record`, `~struct`)
  - Vec (`~array`, ~unpacked array)

https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Data%20types/index.html

Excursion: WG2812



https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

Coding

- **Component sending data**
  - Would not write it like that – written to demo
  - `runMain intro.SimpleWG2812Sim`

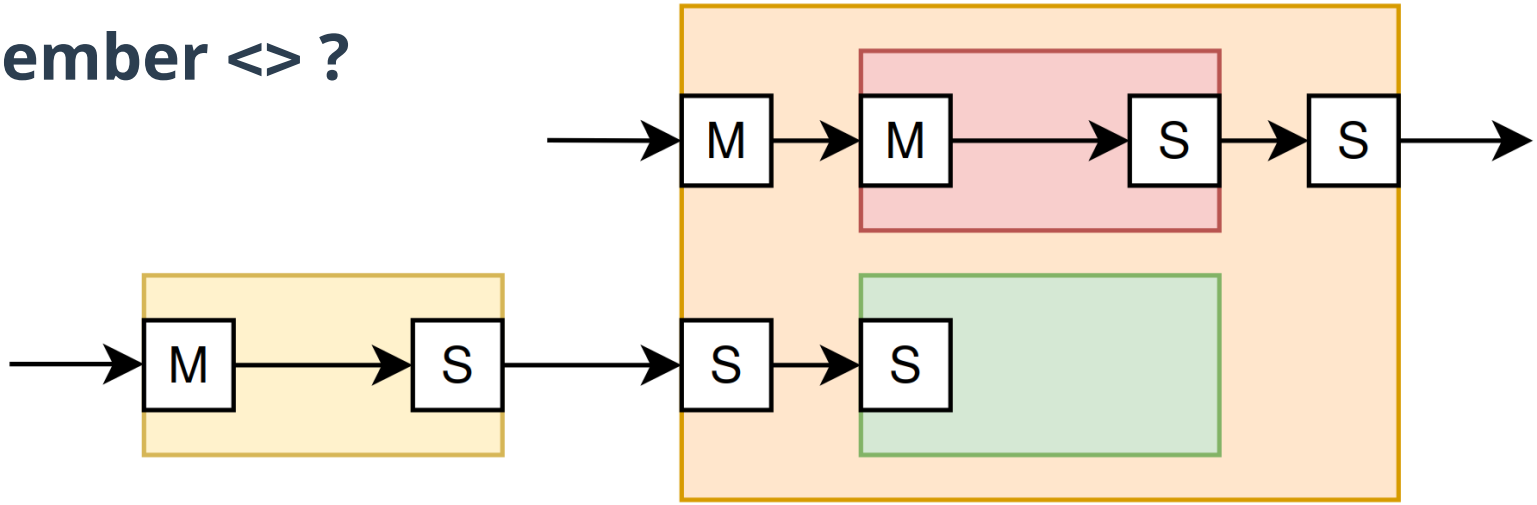
Higher order types

- **Recurring patterns, e.g. handshake**
- **Pack into a reusable, configurable type**
  - Less documentation needed (conventions)
  - Less typing
  - Library of common utilities (FIFOs, Arbiter, test driver, etc.)
- **Major Types: Flow / Stream / Fragment**

https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Libraries/stream.html
https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Libraries/flow.html

Port directions

- `in` / `out` / `inout`
- `master` / `slave` (also for Bus like AHB, etc.)
- Remember <> ?



https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Data%20types/bundle.html#master-slave

Coding

- **Use a Stream of Fragments for our transmitter**
- **One transaction per color**
- **Mark last fragment for last color in update cycle**

  - `runMain intro.FragmentWG2812`

Next steps

- Calculate timings from ClockDomain info
- Use FSM library
- Better test cases (scalatest)
- ...

Resources

- Code for this talk https://github.com/andreasWallner/SpinalHDL-intro-talk-2023
- English documentation https://spinalhdl.github.io/SpinalDoc-RTD/master/index.html
- 中文 documentation https://github.com/thuCGRA/SpinalHDL_Chinese_Doc
- Gitter chat english
  https://gitter.im/SpinalHDL/SpinalHDL?utm_source=badge&utm_medium=badge&utm_campaign=pr-badge&utm_content=badge
- Gitter chat 中文
  https://gitter.im/SpinalHDL-CN/community?utm_source=badge&utm_medium=badge&utm_campaign=pr-badge&utm_content=badge
- Bootcamp https://github.com/jijingg/Spinal-bootcamp
- Workshop https://github.com/SpinalHDL/SpinalWorkshop
- the source code ;-)


QUESTIONS?