A CPU is Only as Good as its Ecosystem:

Turning RISC-V CPUs into Systems with FuseSoC

Olof Kindgren
Qamcom Research & Technology
FOSSi Foundation
Spring 2022 RISC-V week - Paris 2022.05.05
IP cores

Traditional HDL designs are built from IP cores.

Ideally, a product should be built upon a foundation of existing commodity IP cores with the value-add on top.

This is how software products are normally developed.
IP cores come from four sources.

A non-trivial design normally use a mix of these.

- In-house developed
- 3rd party proprietary
- Platform-provided
- 3rd party open source
What is FuseSoc?
What is FuseSoc?

FuseSoC is a package manager…
What is FuseSoC?

FuseSoC is a package manager...

...and a build tool abstraction for HDL
What is a core?
What is a core?

It looks like you’re making an IP core. Would you like help?

- Get help with making the IP core
- Just make the IP core without help
- Don’t show me this tip again
Core description files describe properties of the core that the EDA tools need e.g.

- files
- parameters
- tool options
Core description files

Core description files can be stored separately from the core, in which case they contain info on how FuseSoC can find the core files.

```plaintext
... targets:
  nexys_a7:
    default_tool: vivado
    filesets: [rtl, nexys_files]
    tools: [vivado: {part : xc7a100tcsg324-1}]
    toplevel: corey_top
  tb:
    default_tool: modelsim
    filesets: [rtl, tb]
    toplevel: corey_tb
    tools:
      modelsim:
        vlog_options: [-timescale=1ns/1ns]
      xsim:
        xelab_options: [--timescale=1ns/1ns]
```
What is a core?

Cores list their immediate dependencies which forms dependency trees. FuseSoC resolves the dependencies to find a set of versions that satisfies all requirements.
What is a core?

Core description files can be grouped into core libraries for easier management.
What is a core?

Targets describes ways to use the core and its dependencies. Some targets, like a simulation target, can support several tools.

```plaintext
... targets:
  nexys_a7:
    default_tool: vivado
    filesets: [rtl, nexys_files]
    tools: [vivado: {part : xc7a100tcs324-1}]
    toplevel: corey_top
  tb:
    default_tool: modelsim
    filesets: [rtl, tb]
    toplevel: corey_tb
    tools:
      modelsim:
        vlog_options: [-timescale=1ns/1ns]
      xsim:
        xelab_options: [-timescale=1ns/1ns]
```
FuseSoC reads the core description file, creates the appropriate tool setup files and optionally runs the tool.

FUSESOC RUN...

... --target=sim --tool=verilator corey
... --target=nexys_a7 corey
... --target=de0_nano corey
... --target=sim --tool=icarus corey
## Build or buy (or get for free)

<table>
<thead>
<tr>
<th>In-house solution</th>
<th>FuseSoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>MAINTENANCE</td>
</tr>
<tr>
<td>Build</td>
<td>Buy/Get for free</td>
</tr>
<tr>
<td>Build</td>
<td>FEATURE GROWTH</td>
</tr>
<tr>
<td>Build</td>
<td>Build/buy</td>
</tr>
<tr>
<td>Build</td>
<td>TRAINING</td>
</tr>
<tr>
<td></td>
<td>Build/Buy/Get for free</td>
</tr>
</tbody>
</table>
Existing open source libraries

- fusesoc-cores 77
- orpsoc-cores 100
- openpiton 66
- opentitan 129
- optimsoc 102

+ many smaller libraries e.g. CVA6, picorv32, PULP, serv, microwatt, SweRV

+ many proprietary libraries

Most prominent open source silicon projects already use or have started looking at using FuseSoC
Tomorrow

- Increased adoption
- Expanded base library
- Additional EDA tool support
- Public package pool (à la pypi)
Future

- Industry standard
- World-class documentation
Future

- Industry standard
- World-class documentation
Why use FuseSoc?

Increase reuse

Target different tools and devices with the same core description files
Share cores between different working groups
Reuse cores between projects

Reduce cost

Lower maintenance and on-boarding costs
Battle-proven base functionality that can be easily extended

Focus on your core business, not your cores.
Backup slides
Core description files (generators)

EDA tools only speak (system)verilog (and VHDL). All other file types must be generated before being sent to the EDA tool.
EDA tools only speak (system)verilog (and VHDL). All other file types must be generated before being sent to the EDA tool.

Generators can be shared between cores
Core description files (generators)

Other examples:
- C $readmemh files
- IP-XACT toplevels
- CPU configuration
- memory maps

Generators:
- ramycramface.core
- ramgenerator.core
- generated.core

Generator parameters:
- RAM compiler
- verilog
Core description files (the other stuff)

Parameters
  Tool-agnostic descriptions of `define, parameters, plusargs, generics…
  Controllable from command-line

Script hooks
  Inject custom scripts before and after setup, build and run

VPI
  Tool-agnostic way of compiling VPI libraries
  No DPI yet (but coming)

Probably more stuff I forgot

[Ask me about FuseSoC]
SweRV

- RTL
- Testbench
- Documentation
- Other stuff
SweRVolf

A basic SoC built around SweRV

- Primarily intended for FPGA prototyping
- Portable between devices
- Wide simulator support
- Extendable
- Modular
- Easy to use
- Zephyr OS support
swerv.core

RTL
Testbench
Documentation
Other stuff
SweRVolf dependencies

- axi_mem_if.core
- axi_node.core
- swerv.core
- vlog_tb_utils.core
- axi.core
- common_cells.core
Core description files (generators)