Fastpath Logic™ Automatic infrastructure generation for modeling, design, and verification

Fastpath Logic's **chip specification language (CSL) compiler** automates generation of the C++ simulator, RTL testbench and RTL design code. The generated code supports a standard RTL and functional verification methodology. Designs are described at the architectural level in CSL.

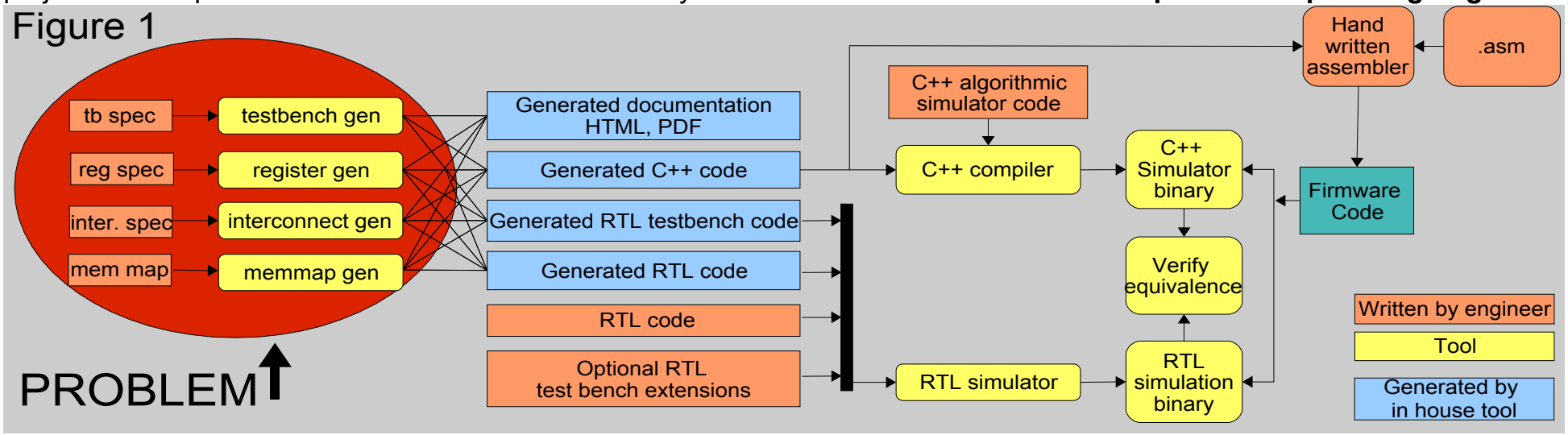
RTL coding tasks fall into two categories:

- Algorithmic: the datapaths and control logic that differentiate your implementation from the competition.
- Chip and verification infrastructure: the additional constants, structures, and connectivity required to connect the implementation of your algorithm to the rest of the chip, the system, the verification environment, and the application software.

The algorithm is what you create and implement for each project. The chip and verification infrastructure is what you have to develop to build and test your design. An important problem results from the fact algorithms change substantially if not completely from project to project. Each project needs to develop custom chip and verification infrastructure components. Chip and verification infrastructure component creation is repetitive and the method used to create the infrastructure does not change much from project to project. Fastpath Logic's **CSL compiler automates the chip and verification infrastructure generation tasks.** Moreover, a single input file facilitates architectural and design exploration.

A typical RTL and verification flow is shown in figure 1. Typically, different tools are used to generate the C++ simulator and RTL infrastructure code which leads to consistency problems in the generated code. **Creating the infrastructure components (blue) is a time consuming and error prone process. The process is complicated because infrastructure components require ongoing**

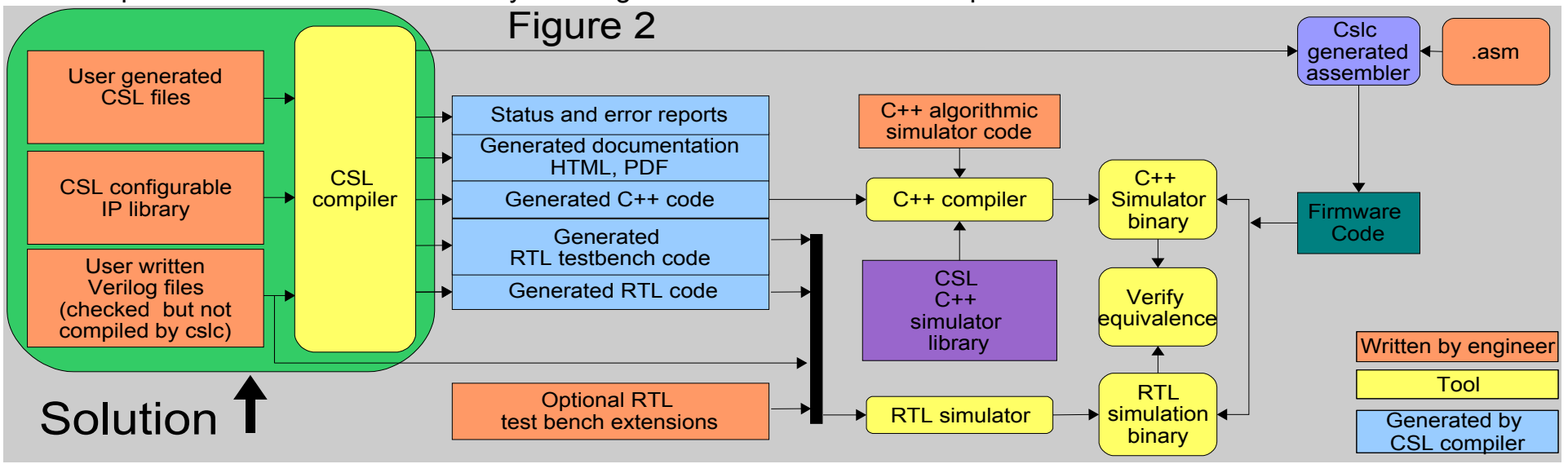
Figure 1



Fastpath Logic™ Automatic infrastructure generation for modeling, design, and verification

The Fastpath Logic solution is shown in figure 2. In contrast to the typical solution that uses many different tools and inputs, Fastpath Logic provides the CSL compiler that accepts a **single** input source. This compiler generates C++ and RTL infrastructure code (e.g. the infrastructure components for the C++ simulator, RTL testbench and RTL implementation). The infrastructure code is used to support the standard Silicon Valley chip design methodology. The CSL compiler also checks the CSL library code against the CSL infrastructure code. Because of the single input source, **the CSL compiler generates the infrastructure component interfaces and thus eliminates consistency problems and bugs** between RTL modules, C++ classes and RTL code. A partial list of the generated interfaces include vector and architectural state C++ writers and sockets, C++/RTL units and unit instantiations, RTL testbench memories for vectors and architectural state, RTL testbench comparators.

Figure 2



There are three major infrastructure component groups shown in figure 3 that need to be constructed to implement and verify a processor or an algorithmically intensive chip design.

1. The C++ simulator that models the behavior of the chip, generates the stimulus and expected vectors as well as the architectural state that is used to verify the design under test.

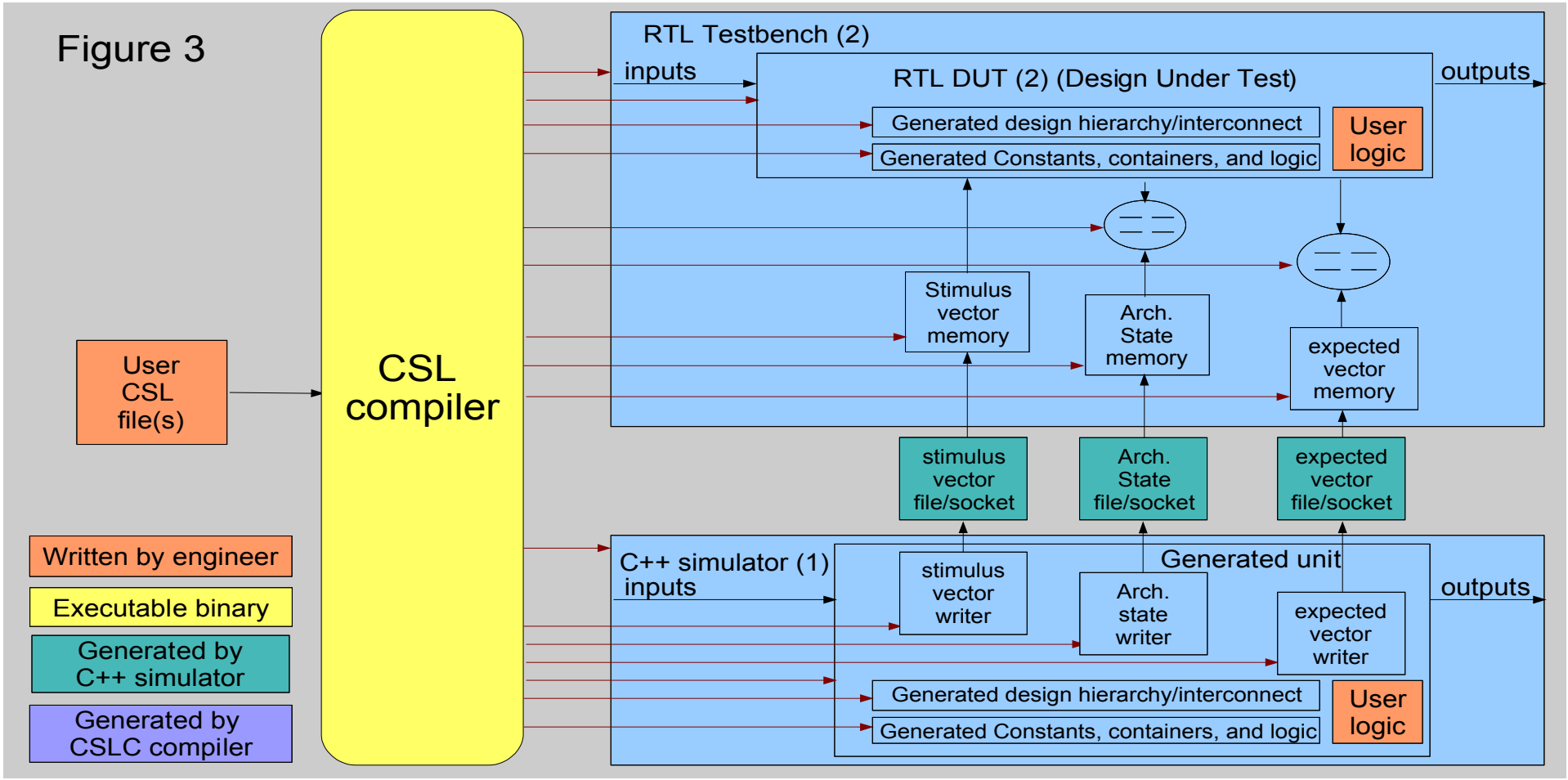
2. The RTL testbench is used to apply the stimulus vectors, compare the expected architectural state against the DUT architectural state, and compare the expected vectors and DUT output vectors.

3. The RTL design under test (DUT) design hierarchy and interconnect, the design memories and register sets, and the RTL model that implements the design.

These components facilitate the construction of the C++ simulator, RTL testbench and RTL implementation.

Fastpath Logic™ Automatic infrastructure generation for modeling, design, and verification

The automatically generated interfaces between the C++ simulator, RTL testbench, and RTL DUT infrastructure components are shown in the figure below. The user defines the chip specification language (CSL) file, the C++ simulator user logic (orange) and the RTL user logic (orange). The CSL compiler (yellow) compiles the CSL file and generates the infrastructure components (blue) for the C++ simulator, RTL testbench and DUT interconnect infrastructure components. The blue boxes in figure 3 are the generated infrastructure components. The C++ simulator executes tests which generate vectors (green) which are sent to the testbench from the simulator via files or a socket (future). The vectors are loaded into the RTL testbench (blue) and the design is simulated. The stimulus vectors drive the DUT inputs, and the expected vectors are compared against the DUT outputs.



Fastpath Logic™ Automatic infrastructure generation for modeling, design, and verification

RTL (Verilog and VHDL) is 20+ years old

- RTL is a low level detailed view of the design
- RTL is the input to the classic EDA tool flow
- RTL must be verified by simulators and test bench wrappers
- RTL can be generated by a compiler

Synthesis and timing analysis take less than 10% of the project time line. Projects pay big bucks for timing analyzers and synthesis tools. Verification now takes 50% of the project teams effort during the life of the project. Does verification get the tool budget needed to support its activities?

A significant part of the simulation, verification and design tasks involves building infrastructure to connect the components in the same domain (e.g. design interconnect) and in different domains (e.g. Connect C++ simulator to the test bench).

The verification infrastructure connects the tests, C++ simulators, test benches and RTL design code. If these infrastructure components are generated manually or using different tools, then inconsistent interfaces, different constant values, and differences between architectural state memories will likely result. These inconsistencies will occur between the infrastructure components in the C++ simulator, RTL testbench and design.

A single compiler that accepts a single input is the only way to guarantee automatic generation of consistent:

- interfaces and constants between different RTL modules
- interfaces and constants between different C++ simulator modules
- C++ simulator and testbench vectors
- C++ simulator and testbench architectural state memories
- Interfaces between RTL testbench components and DUT module interfaces

Benefits of FPL software

- The input specification is concise and automatically expands into detailed C++ and RTL code.
- Set up the infrastructure for a project quickly
- Make changes with minimal effort
- Reduce the amount of engineering time and resources spent on the design and verification tasks
- Create configurable IP
- Create configurable designs which can be easily modified in proliferation projects

Why build when you can buy?

Fastpath Logic, Inc.
PO Box 188
Palo Alto, CA, 94302-0188

Phone 650-331-0702
email info@fastpathlogic.com
web www.fastpathlogic.com