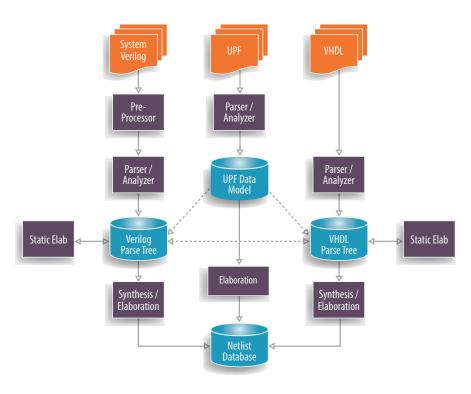
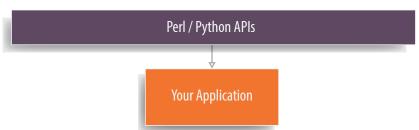


## **BUILD YOUR OWN RTL TOOLS**

with Verific's industry standard parsers and elaborators









Not every EDA application needs to be written in C++. Therefore Verific has enabled its industry standard System Verilog, UPF, and VHDL parsers with a complete Perl and Python interface. All regular Verific functionality is now available at your Perl or Python fingertips.

- Parse and analyze
- Elaborate
- Traverse and manipulate the parse tree and netlist
- Modify RTL and print out with comments and layout preserved
- Find, insert, remove, and change modules, ports, nets, etc..
- Keep / flatten hierarchy
- Group / ungroup

All through easy to understand Perl and Python APIs.

RTL modifications, debug insertion, design for test adjustments, interface changes, clock domain checks, you name it: It is all easily accomplished with Verific's parsers and data structures.

```
#!/usr/bin/perl
use Verific;
# this application parses a SystemVerilog design,
# finds all clock nets, and prints them to stdout
$file name = "example.sv";
Verific::veri file::AddIncludeDir("/usr/local/verilog") ;
$mode = $Verific::veri file::SYSTEM VERILOG ;
# Analyze the design. In case of failure return
if (!Verific::veri_file::Analyze($file_name, $mode)){
    exit(1);
# Elaborate all analyzed design units. In case of failure return
if (!Verific::veri file::ElaborateAll()) {
   exit(1);
# Get a handle to the top-level design
$top = Verific::Netlist::PresentDesign() ;
# Flatten down to primitives
$top->Flatten();
# Iterate over all DFF instances
$insts = $top->GetInsts();
$iter = $insts->Iterator("Instance");
for (my $inst = $iter->First; $iter < $iter->Size; $inst = $iter->Next) {
   if ($inst->Type() eq $Verific::PRIM DFF ||
        $inst->Type() eq $Verific::PRIM_DFFRS) {
        # Get clock net for this flipflop
        my $clock net = $inst->GetClock();
        # Use a Perl hash table to check if the net has occurred before
        # and if not, print to screen
        if (!defined($clocknets{$clock_net->Name()})) {
            # Have not seen this clock net before
            printf "-- Net %s is a clock net\n" , $clock_net->Name() ;
               $clocknets{$clock_net->Name()}=1;
        }
    }
# All done. Wasn't that verific!
```



