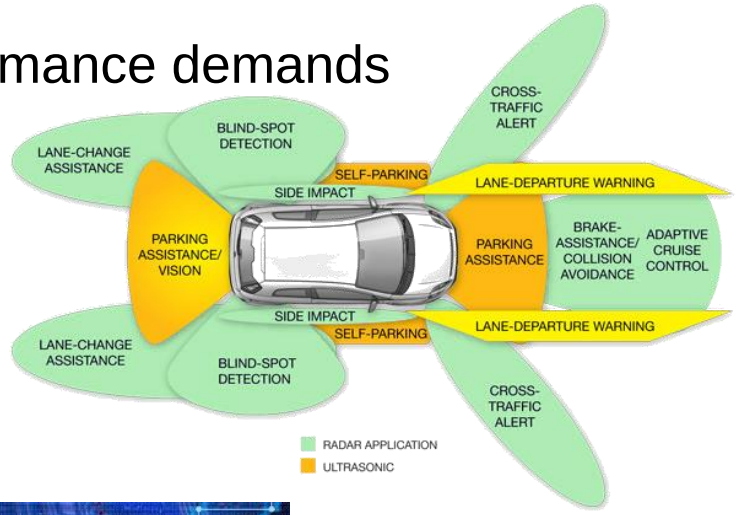
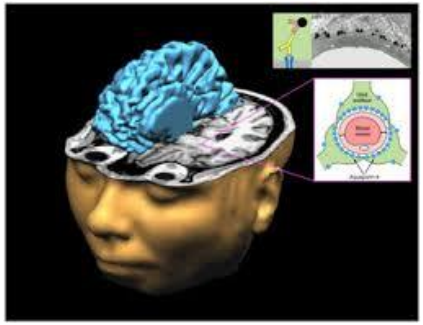
High Level Synthesis (HLS)

Industry Trends

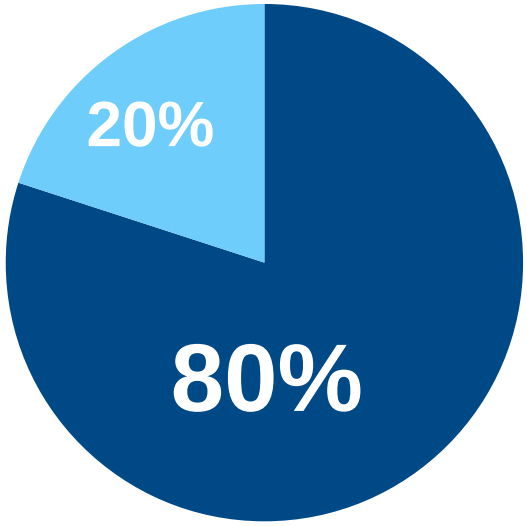
- Increasing product functionality and performance demands
- Smaller time-to-market window
- Shorter product lifecycle
- Limited personnel resources



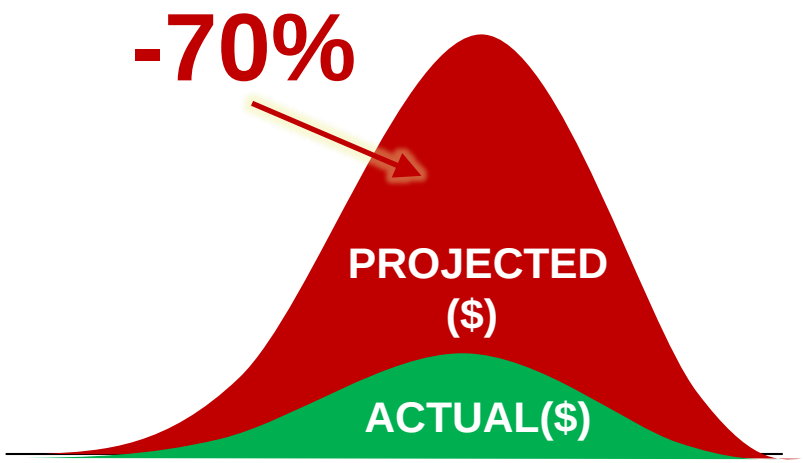


Developing FPGAs Takes Too Long!

- 80% of customers missed their production deadline
  - Driven by change in requirements and delays in their FPGA development
- Production delay can cost them as much as 70% of their potential volume and revenue



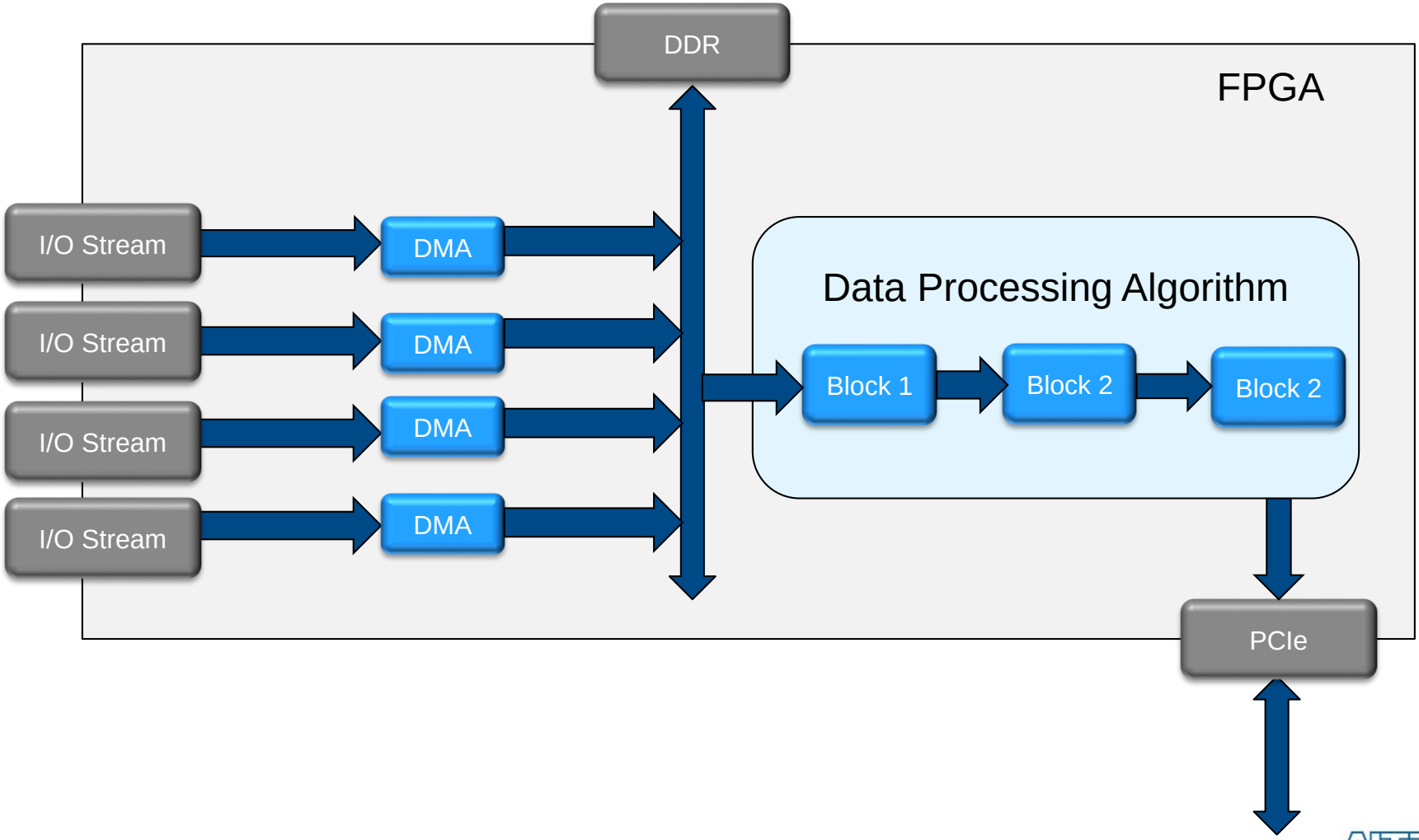
Delay
On time



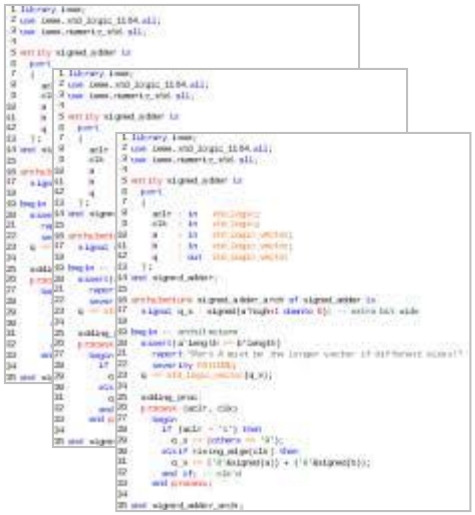
- Ramifications of not making improvements:

Delays → up to 70% in Revenue Loss

What Do FPGA Developers Build?

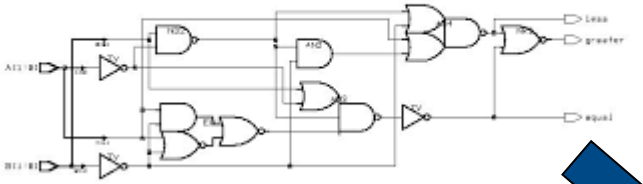


How Do They Build It?

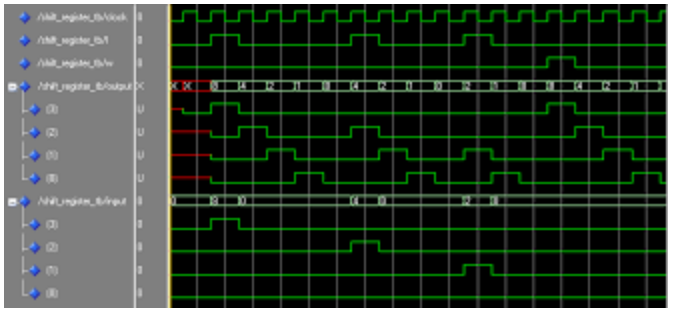


Write HDL

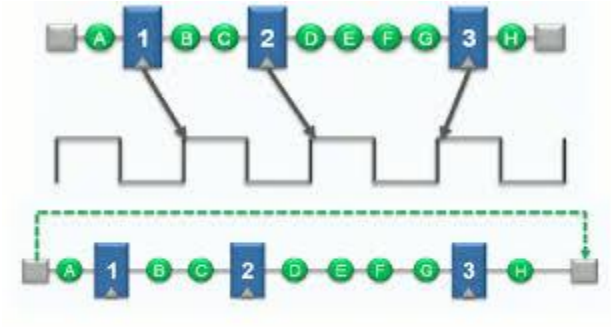
Synthesize

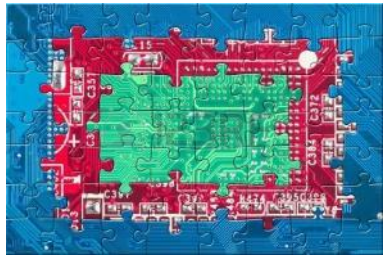


Simulate



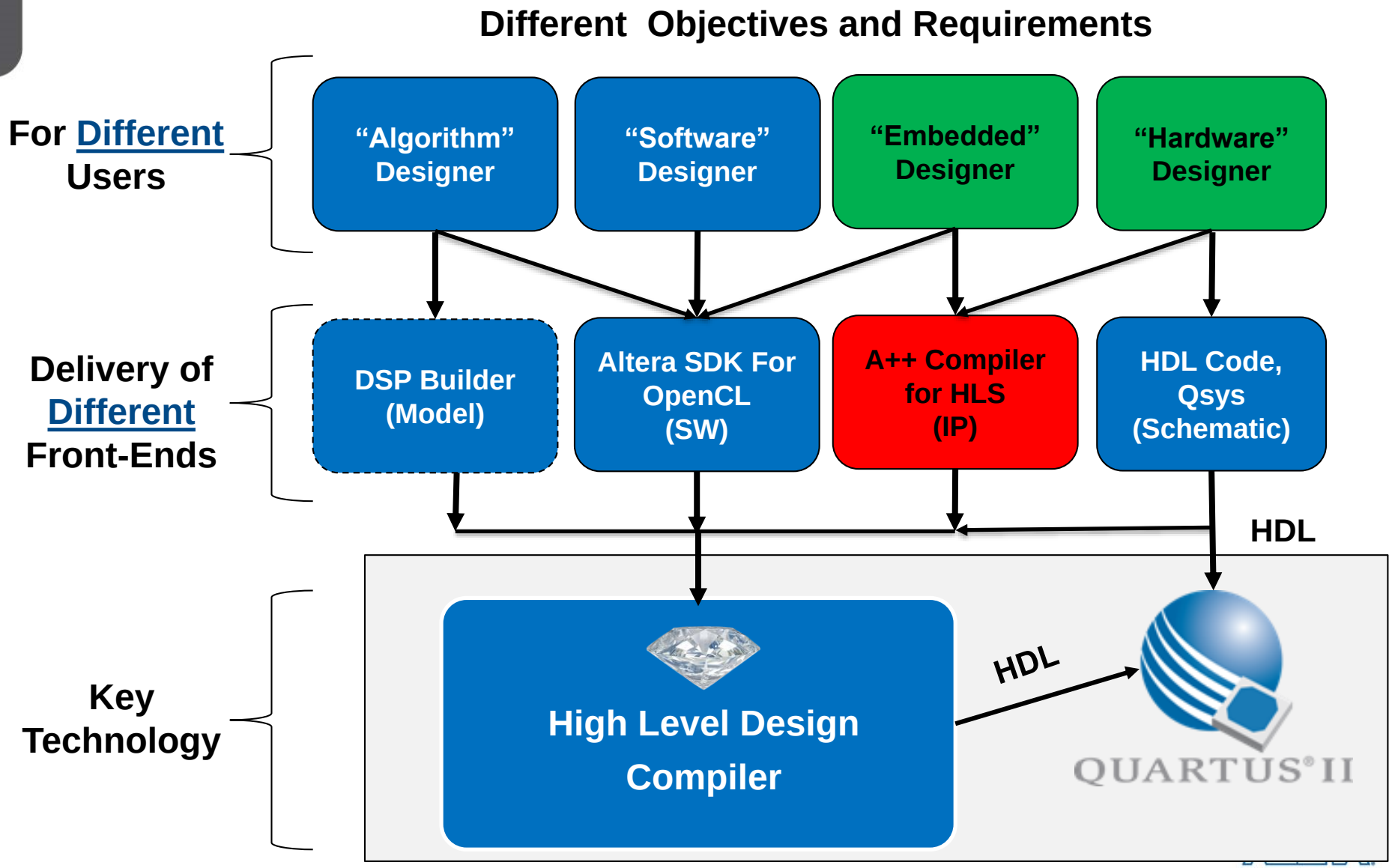
Close Timing



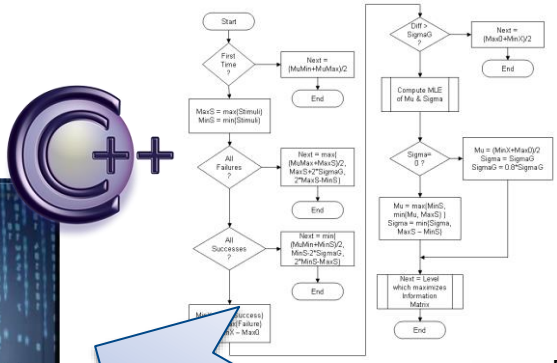


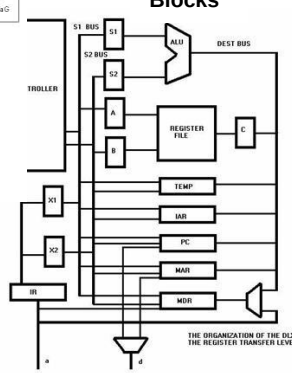
Repeat

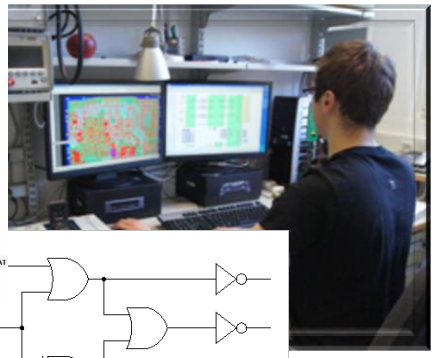
What Tools Do They Use?

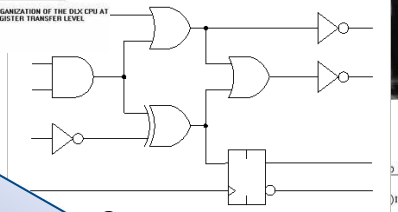


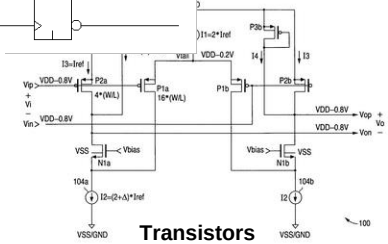
So, Why HLS?







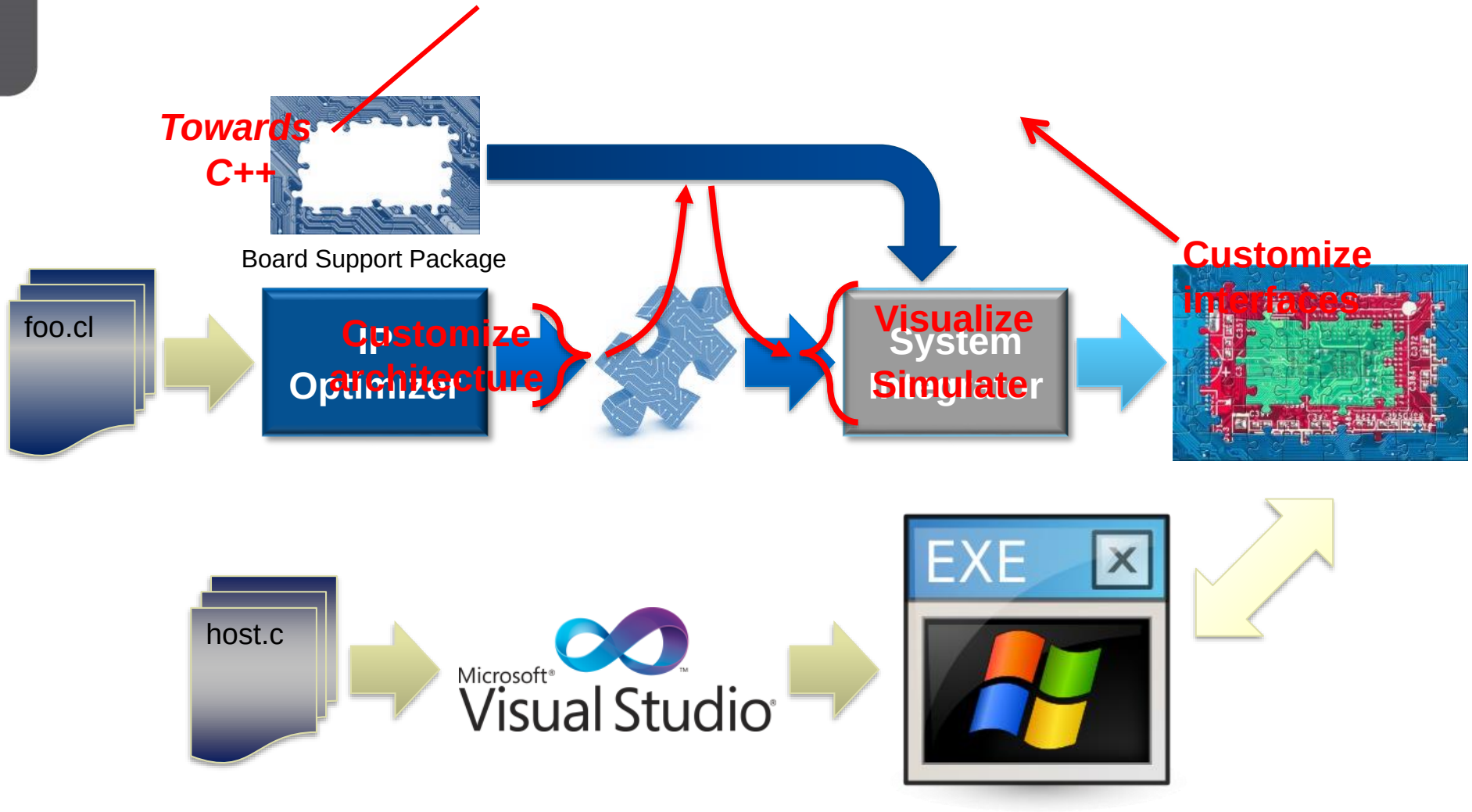




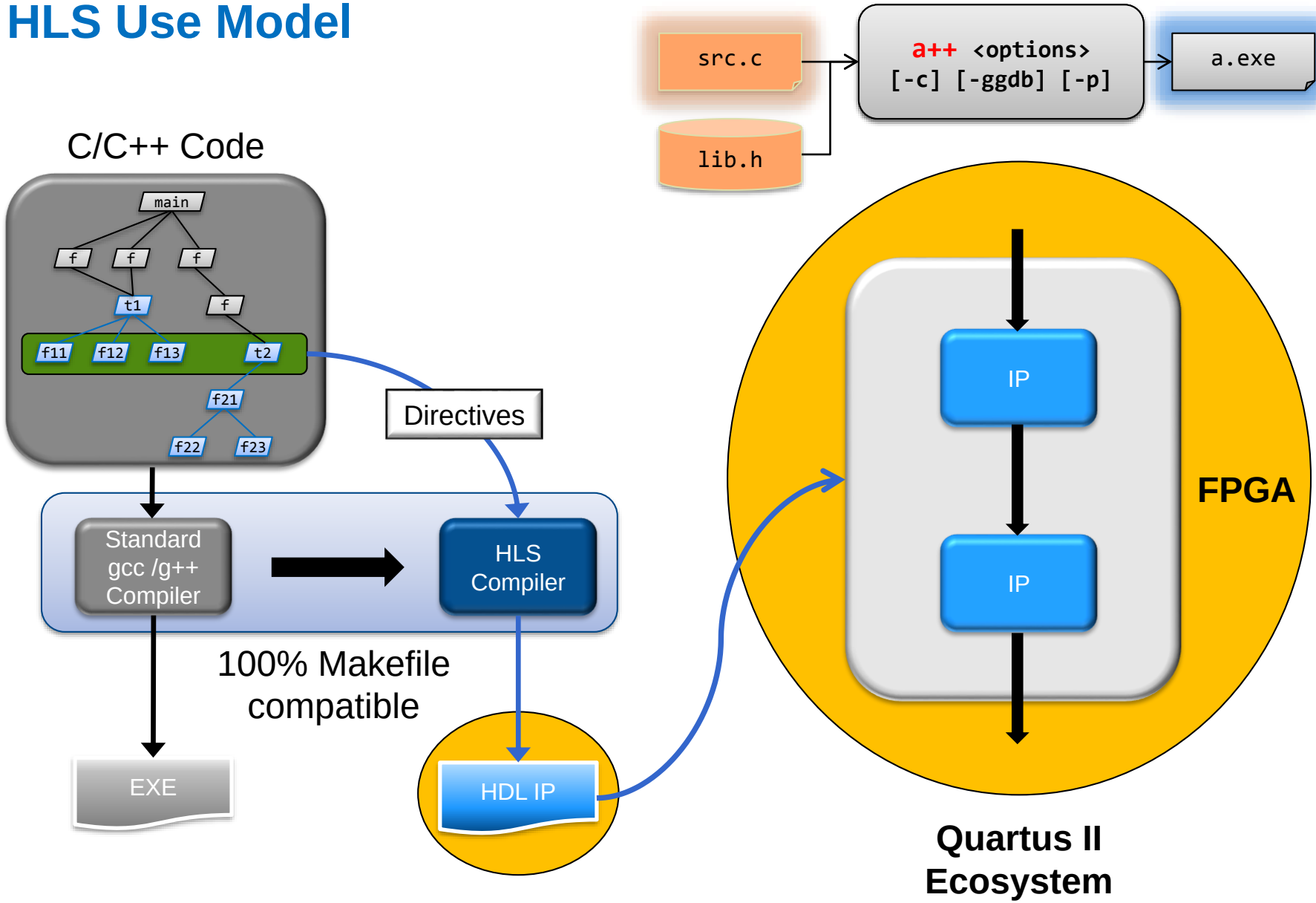
Increase Productivity
Higher Level of Abstraction

- Debugging software is much faster than hardware
- Many functions are easier to specify in software than RTL
- Simulation of RTL takes thousands times longer than software
  - Creating test benches takes a long time
  - Updates to code requires changes to testbench to verify
  - Simulation of PCIe at 1fs for 500ms = *Enough said!*
- Design Exploration is much easier and faster in software

How Does a++ Compiler for HLS Differ From OpenCL?

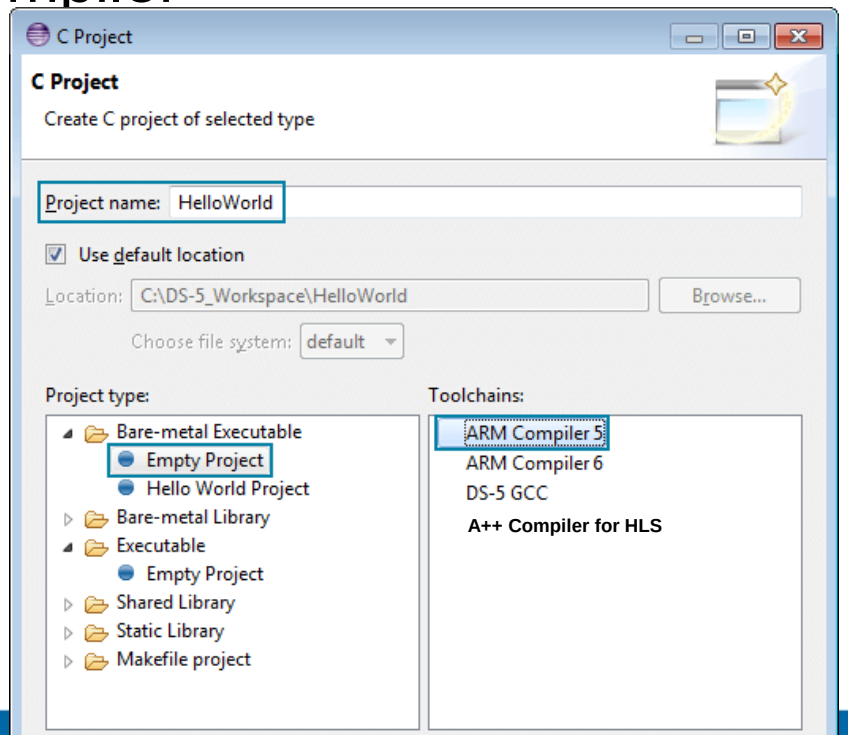


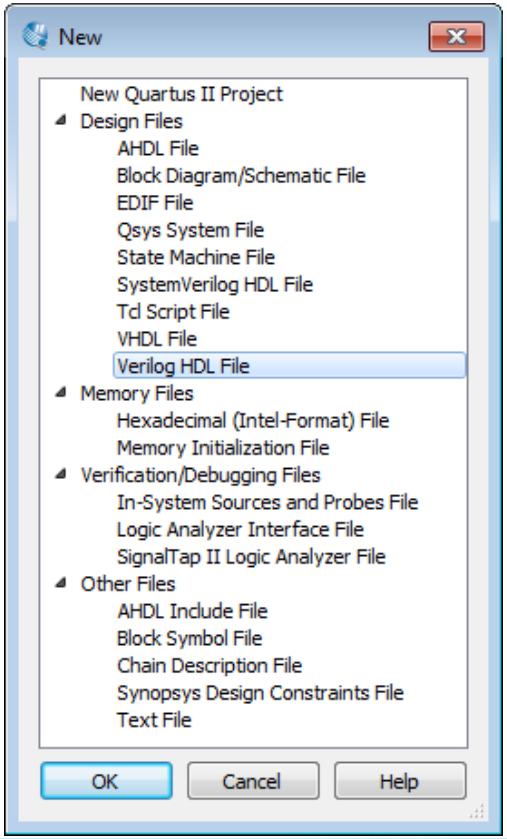
HLS Use Model



Integration into FPGA Development Tools

- Quartus II HDL integration
- Qsys IP integration tool
- DSP Builder Advanced Blockset model based tool
- OpenCL BSP
- SoC EDS compiler





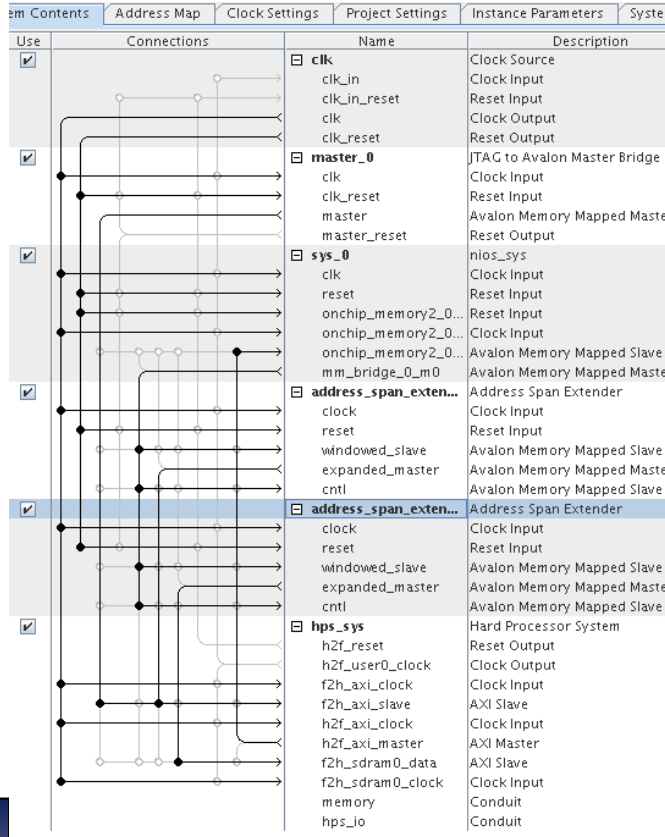
OpenCL vs a++ Compiler Summary

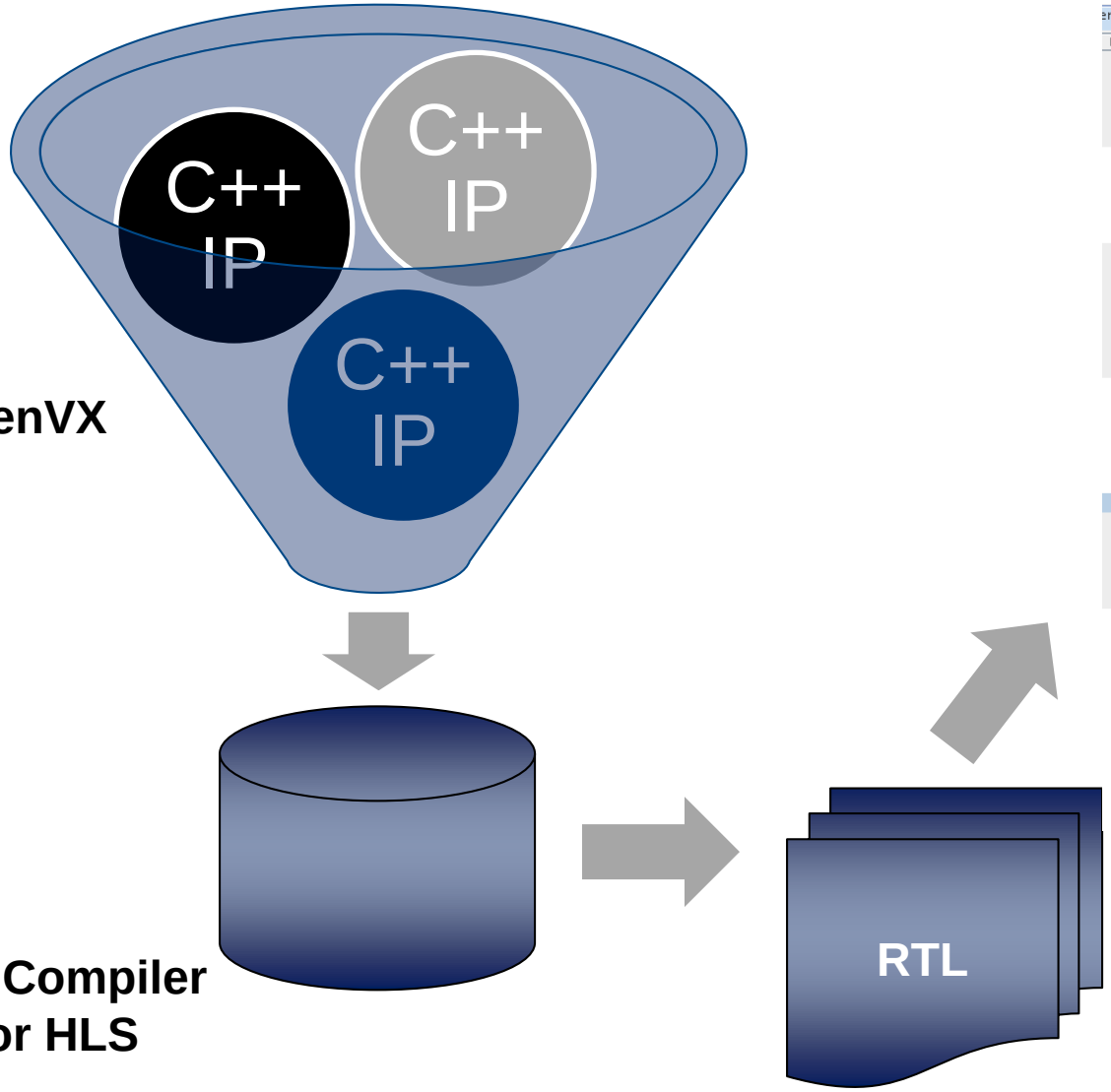
A++ Compiler
for HLS

Targets CPU, GPU and FPGAs	→	Targets FPGAs
Target user is HW or SW	→	Target user is HW
Implements FPGA in software development flow	→	Implements FPGA in traditional FPGA development flow
Performance is determined by resources allocated	→	Performance is defined and amount of resource to achieve is reported
Builds the entire FPGA system	→	Builds an IP block
Host Required	→	Host is optional

Can Also Be Wrapped With Higher Level Flows

Qsys



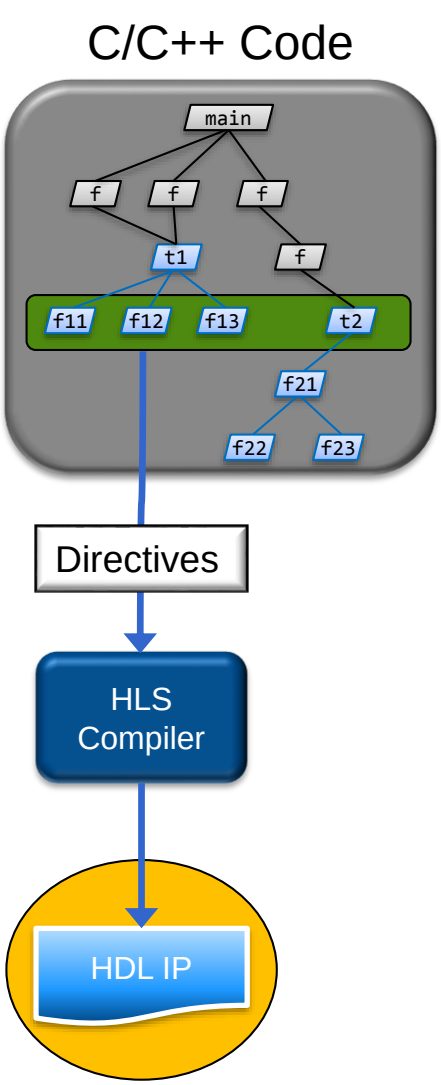


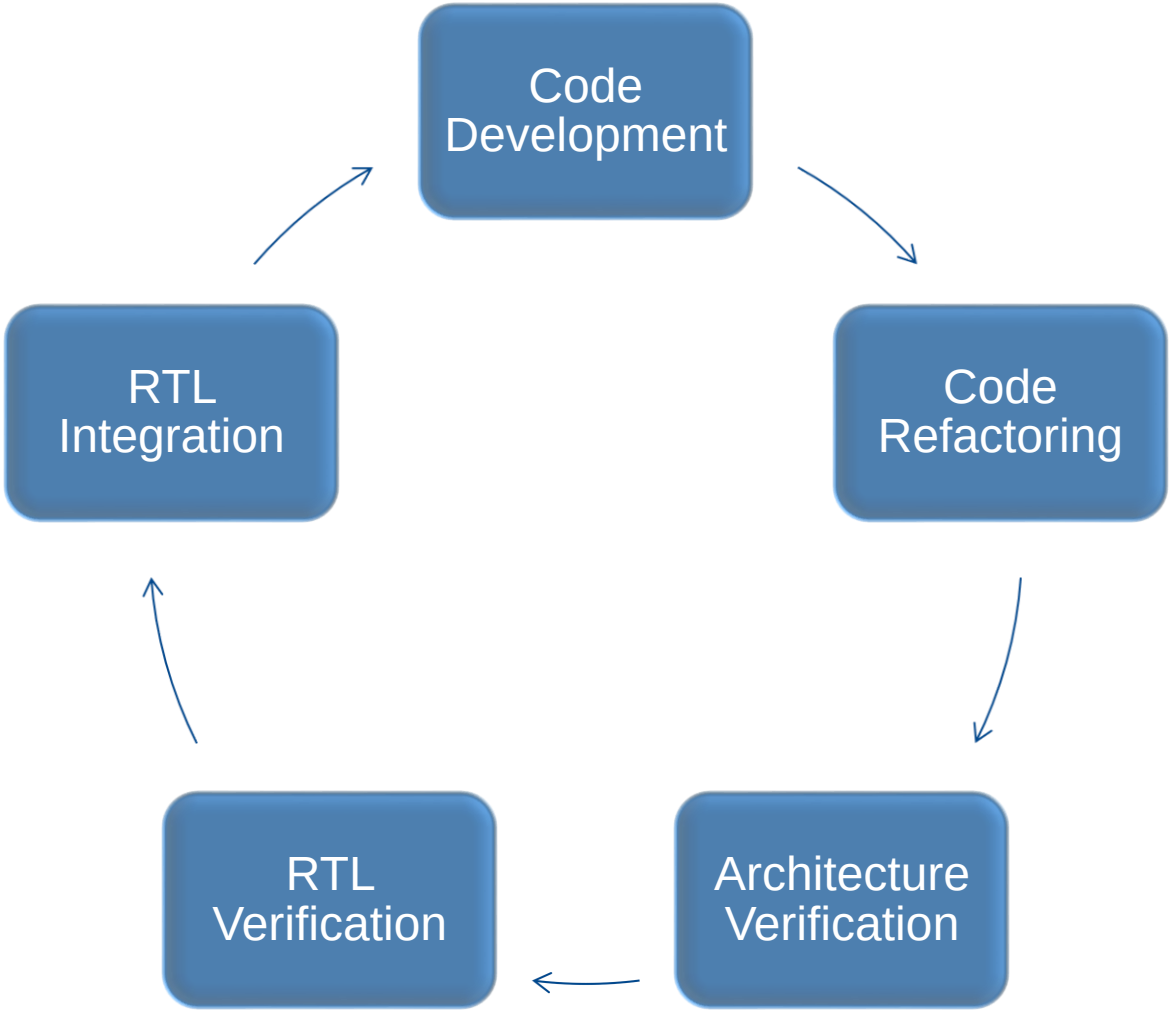
OpenVX

A++ Compiler for HLS

ALTERA
now part of Intel

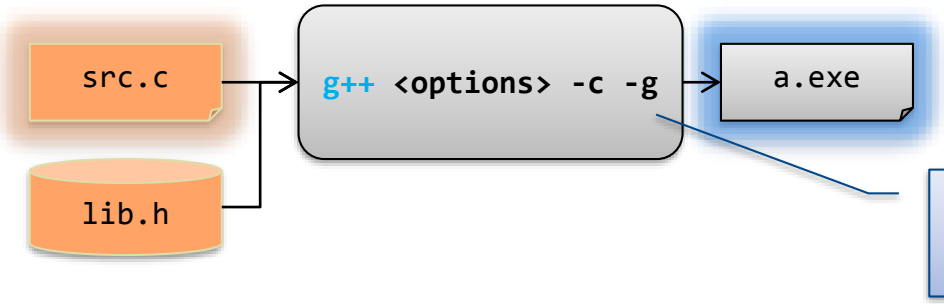
HLS Development Phases



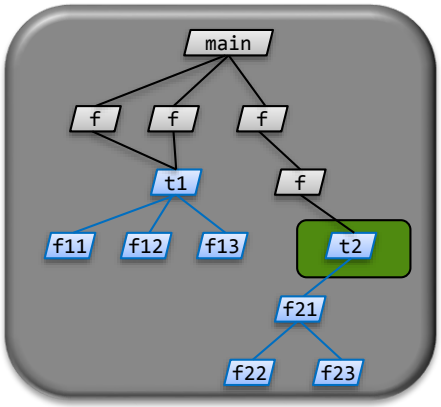


Code Development

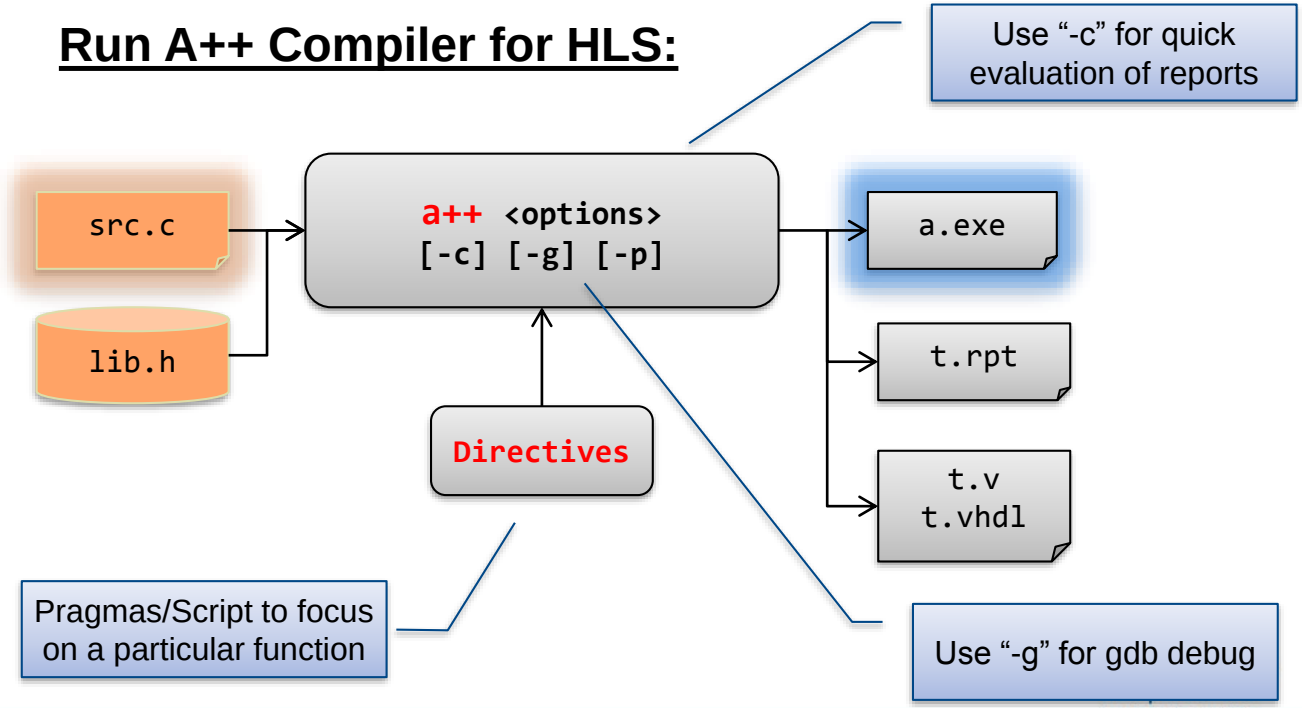
Develop with C/C++:



C/C++ Code

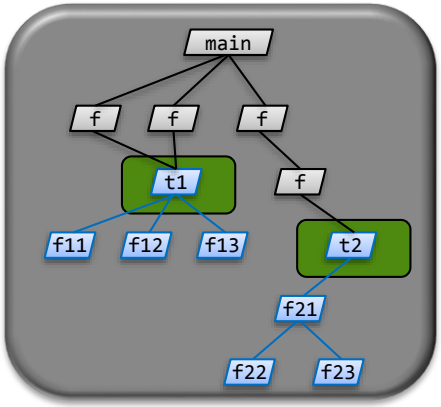


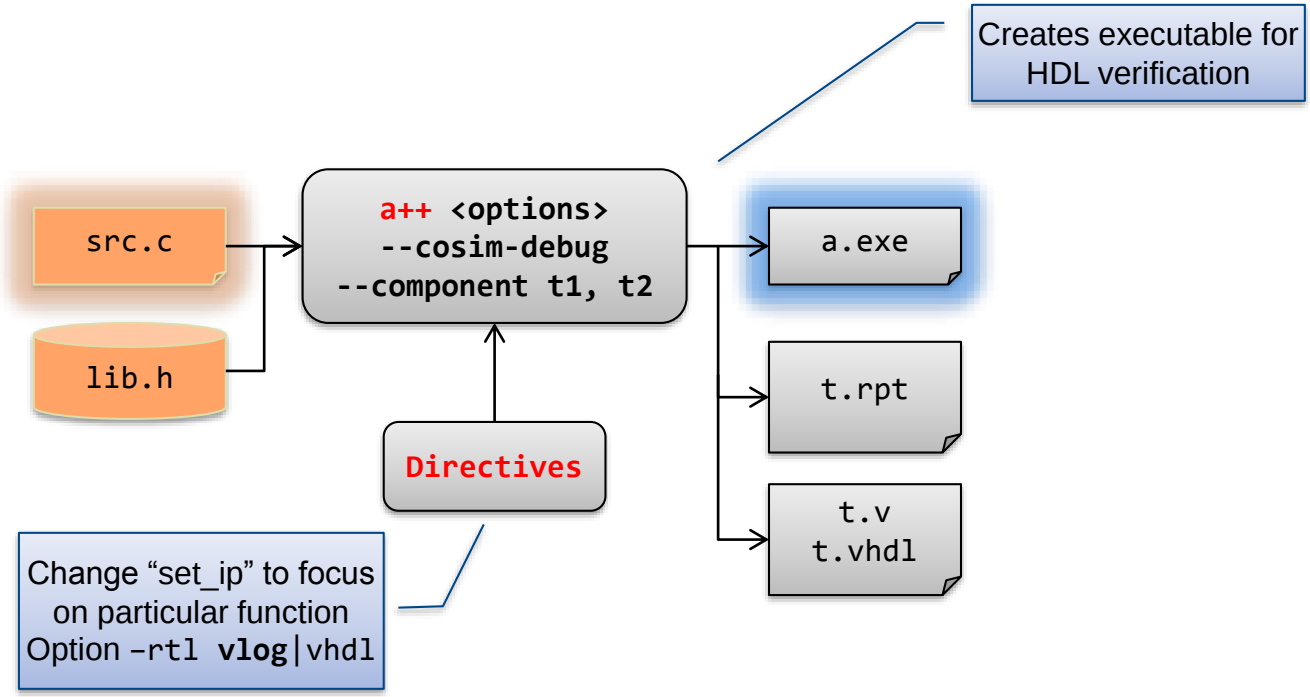
Run A++ Compiler for HLS:



Automated HDL Verification

C/C++ Code



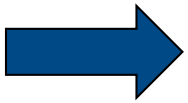


Automated Flow:

- stub out and rename
- Emulate local clock and reset
- Manage HDL Simulator

Component/Testbench Partitioning Example

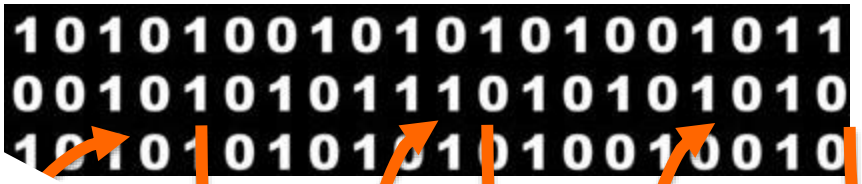
```
#include "HLS/stdio.h"
int impl(int a, int b) {
  int res=a+b;
  printf ("x86:%d+%d=%d\n",a,b,res);
  return res;
}

int component accelerate(int x, int y){
  return impl(x,y);
}

int main(){
  int inp=3;
  int res=accelerate(inp, inp);
  printf("%s: %d+%d=%d\n",
         ((res=impl(inp,inp)) ?
         "Pass":"Fail"),inp,inp,res);
  return 0;
}
```



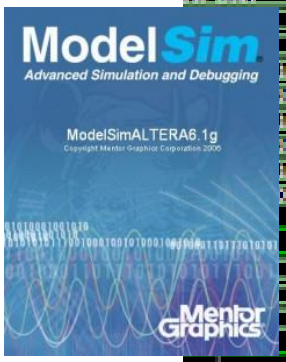
- Main() becomes testbench for component accelerate()
- When accelerate() is called, the vectors are pushed into modelsim

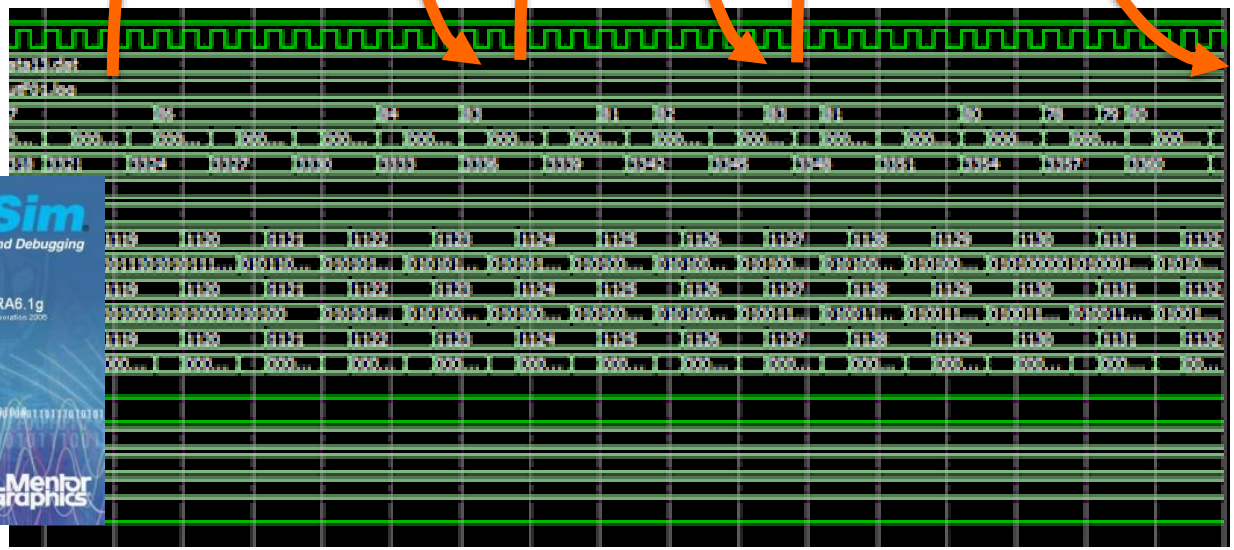
Automated Verification of IP

component
a_sim.so



testbench
a.out





Release Summary

Release	Now	Q1 2017	17.1
HLS Status	Private Beta	Public Beta	Production

- Targeting Data path and control logic optimizations of high speed designs
- Initial production release is not targeting low sample rate designs

A++ Compiler for High Level Synthesis Summary

- Allows FGPA developers to increase their productivity and time to market
  - Simulating C code is thousands of time faster than HDL
- Leverages standard C/C++ development environment
  - Much less pragmas than competition
- Produces a validated IP block from C/C++ code
- Compiler integrates into all our tool flows
  - Qsys, DSP Builder, Quartus, Simulink, SoC EDS, etc.
- Compiler integrates into IDE environment like typical compilers
  - Eclipse, Clion
- Quality of results on par with HDL performance and 10% of area
  - Use model: you define desired performance, compiler reports amount of resources to achieve