

# High Level Synthesis (HLS)

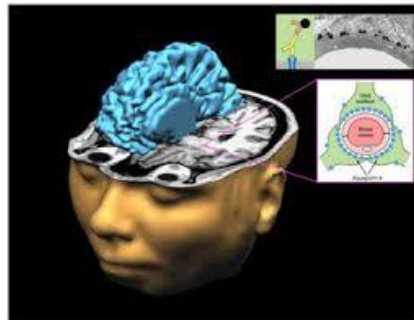
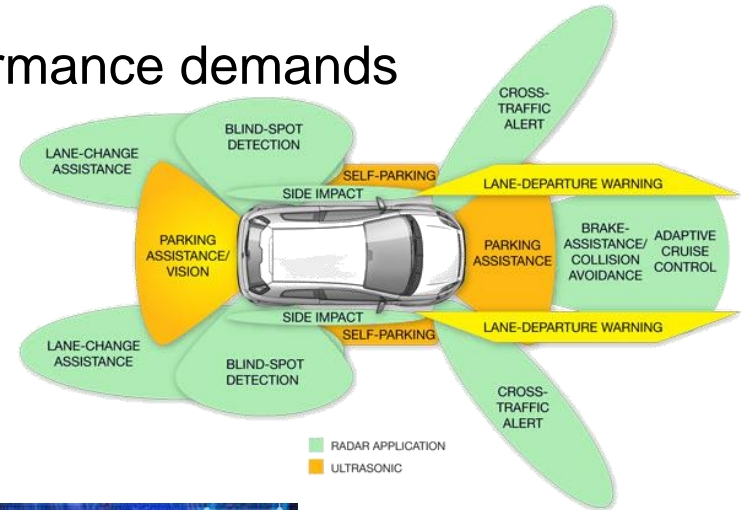
The Altera logo is rendered in a blue, outlined, sans-serif font. The letters are bold and have a consistent stroke width. A registered trademark symbol (®) is located at the top right of the letter 'A'.

**ALTERA**®

now part of Intel

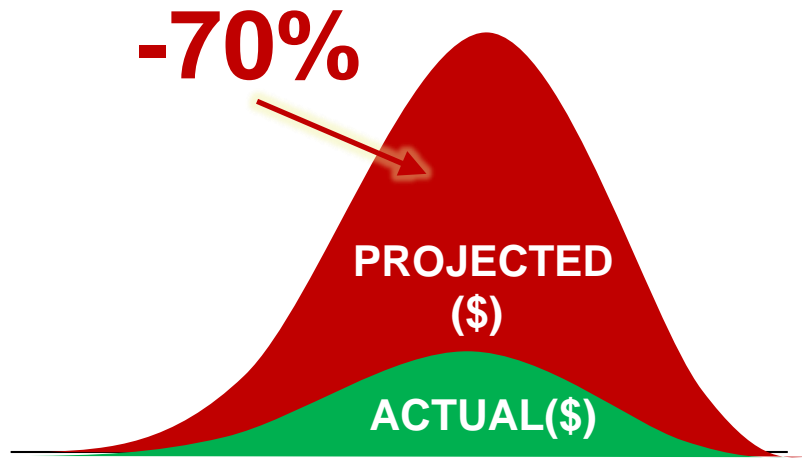
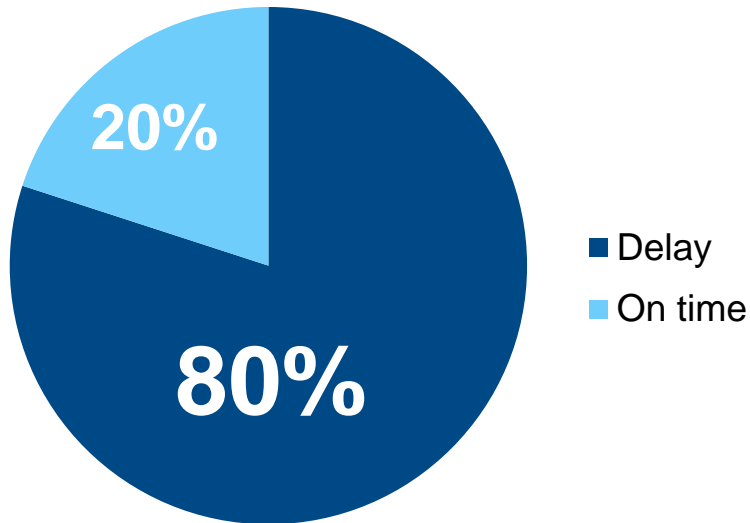
# Industry Trends

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



# Developing FPGAs Takes Too Long!

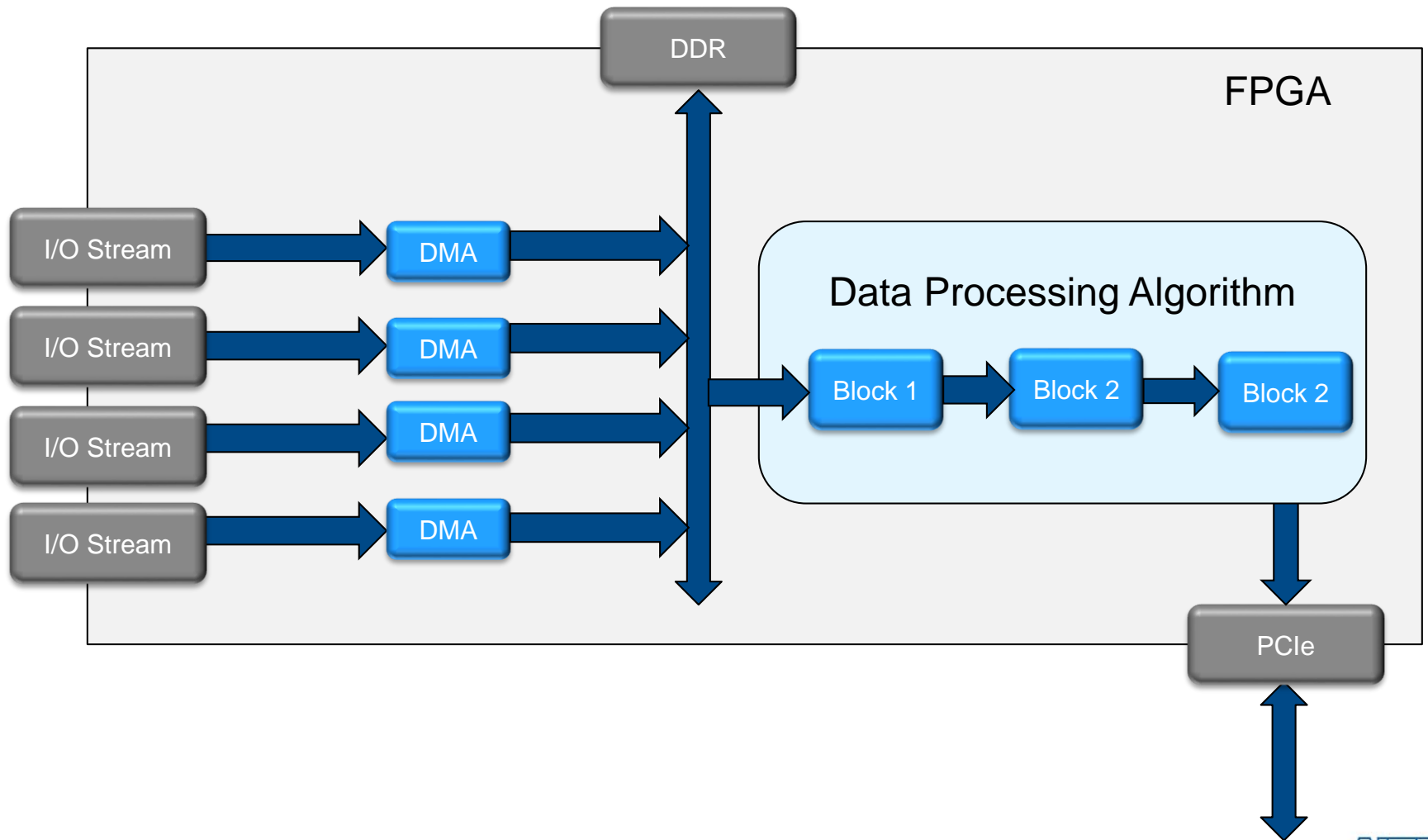
- 80% of customers missed their production deadline
  - Driven by change in requirements and delays in their FPGA development
- Production delay can cost them as much as 70% of their potential volume and revenue



- Ramifications of not making improvements:

***Delays → up to 70% in Revenue Loss***

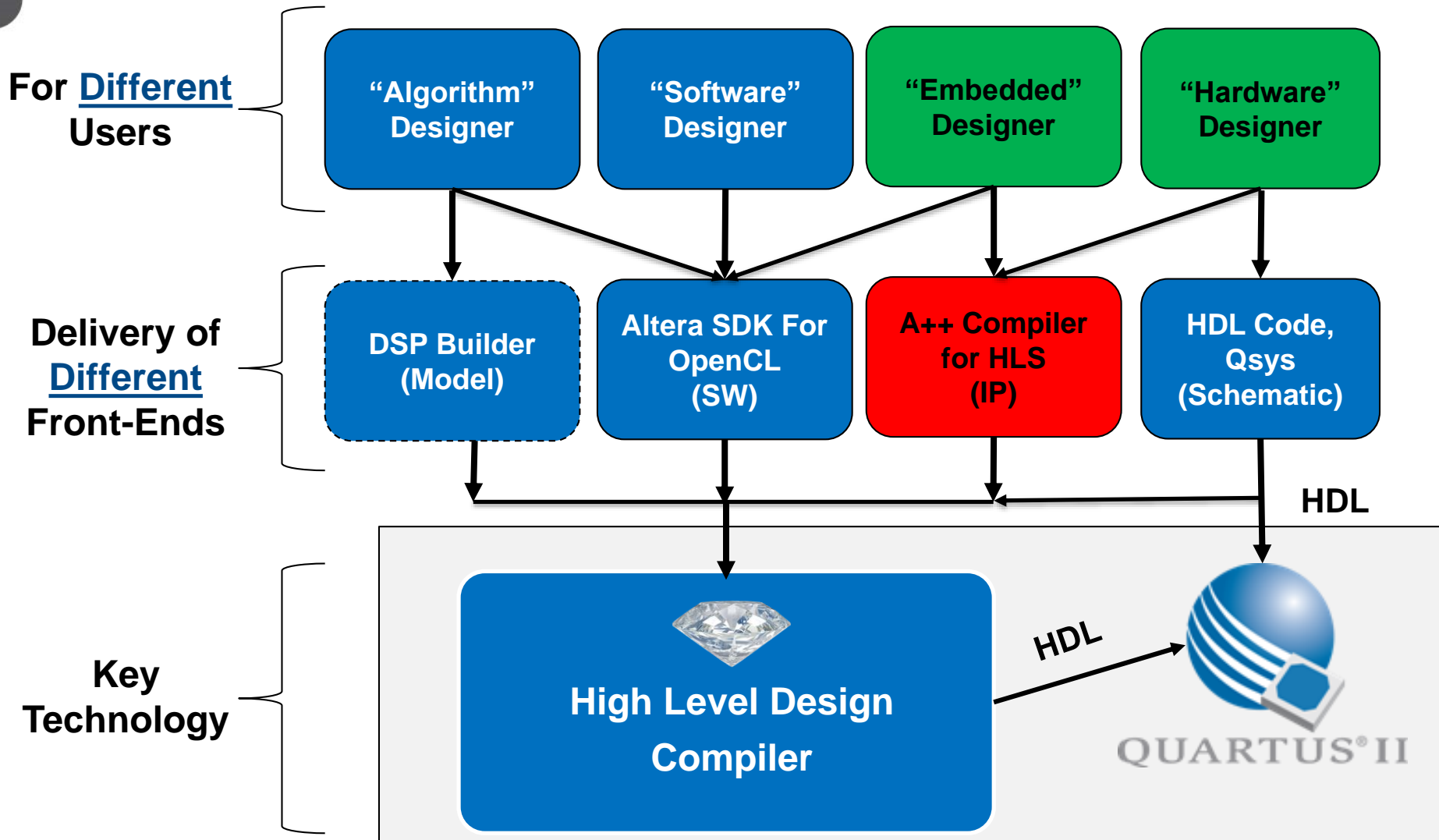
# What Do FPGA Developers Build?



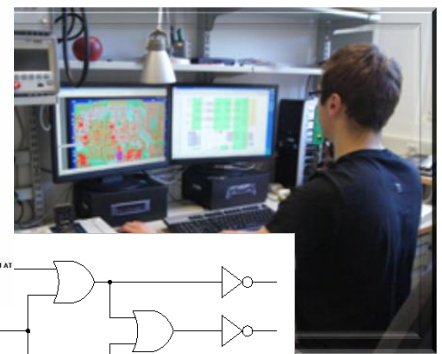
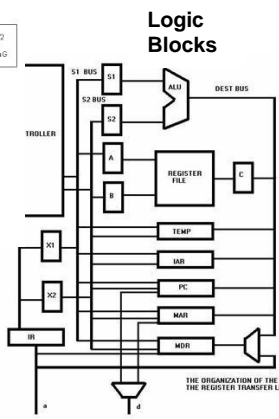
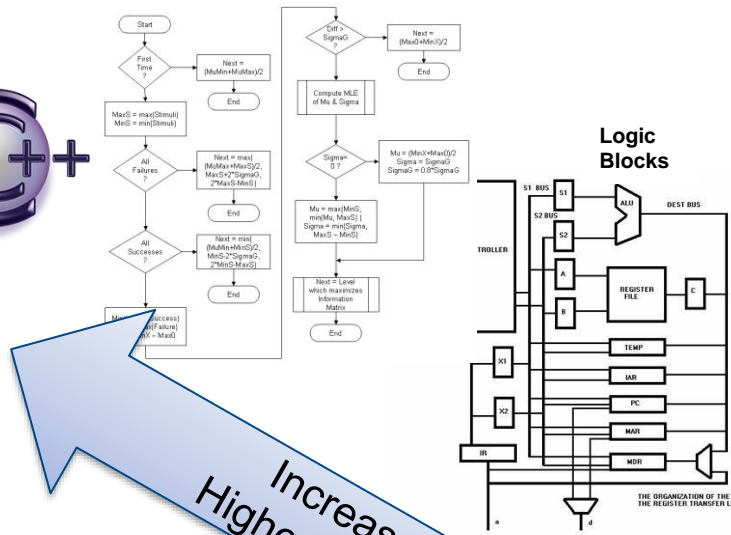


# What Tools Do They Use?

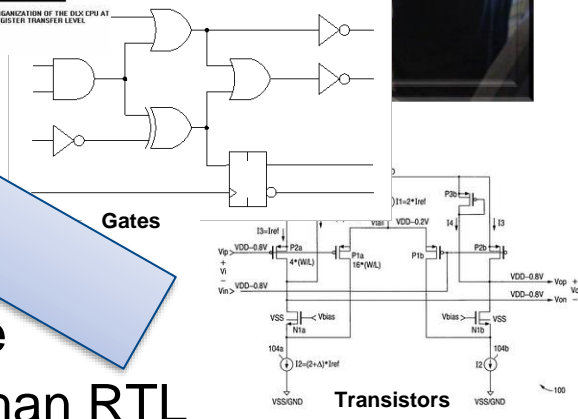
## Different Objectives and Requirements



# So, Why HLS?

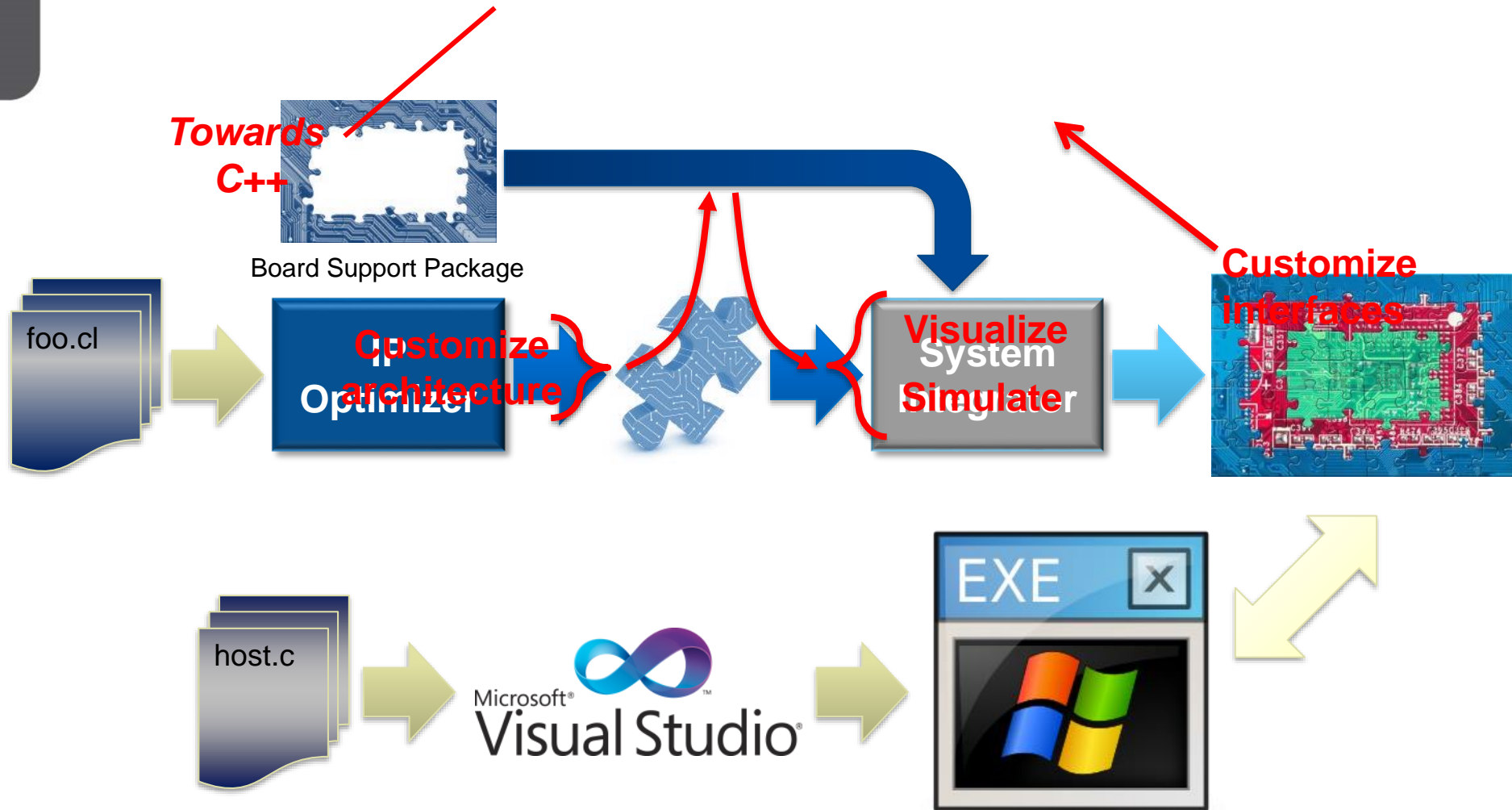


Increase Productivity  
Higher Level of Abstraction



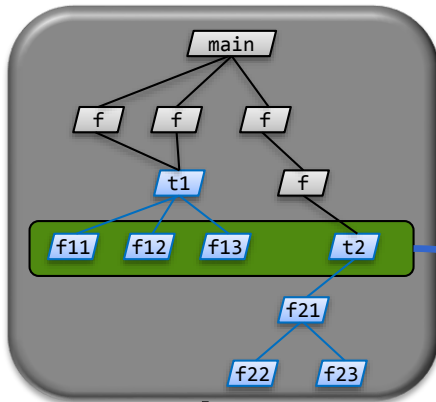
- ⌂ Debugging software is much faster than hardware
- ⌂ Many functions are easier to specify in software than RTL
- ⌂ Simulation of RTL takes thousands times longer than software
  - Creating test benches takes a long time
  - Updates to code requires changes to testbench to verify
  - Simulation of 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?

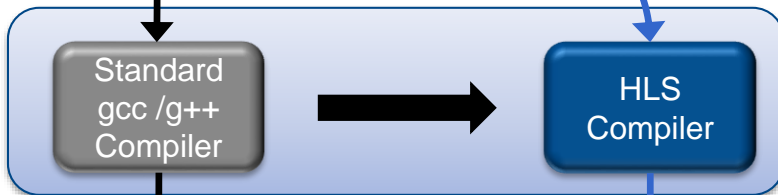


# HLS Use Model

## C/C++ Code

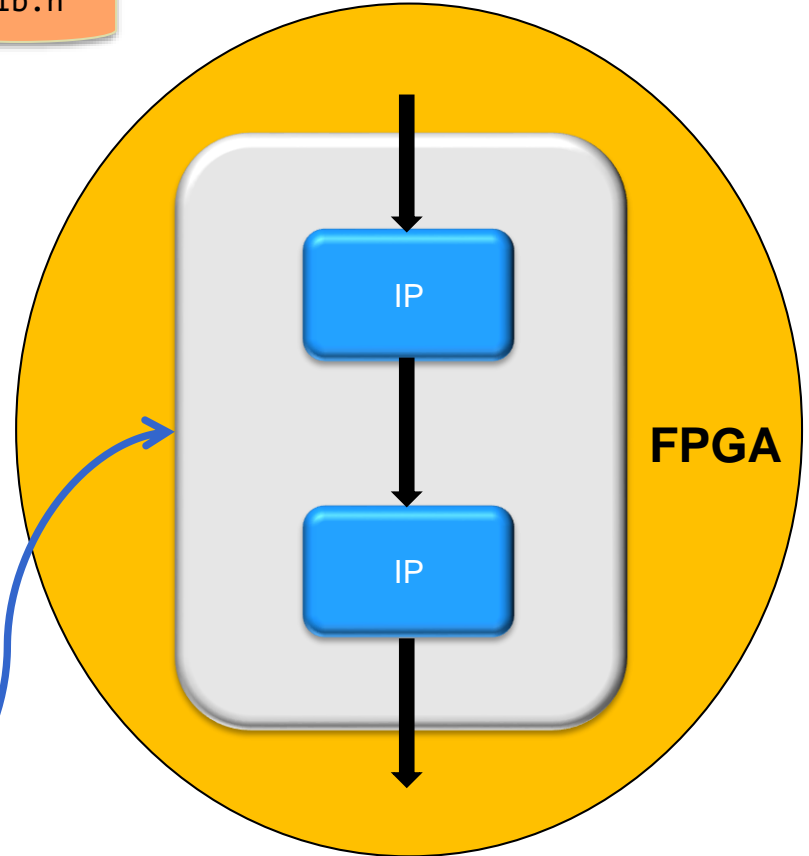
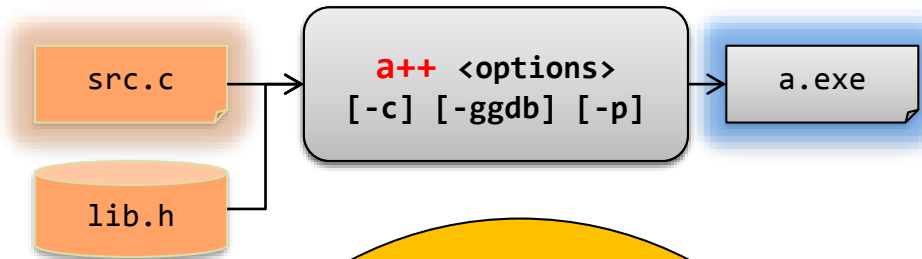
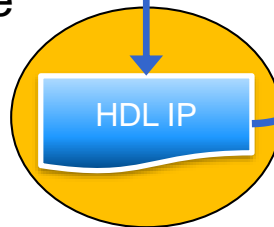


Directives



100% Makefile compatible

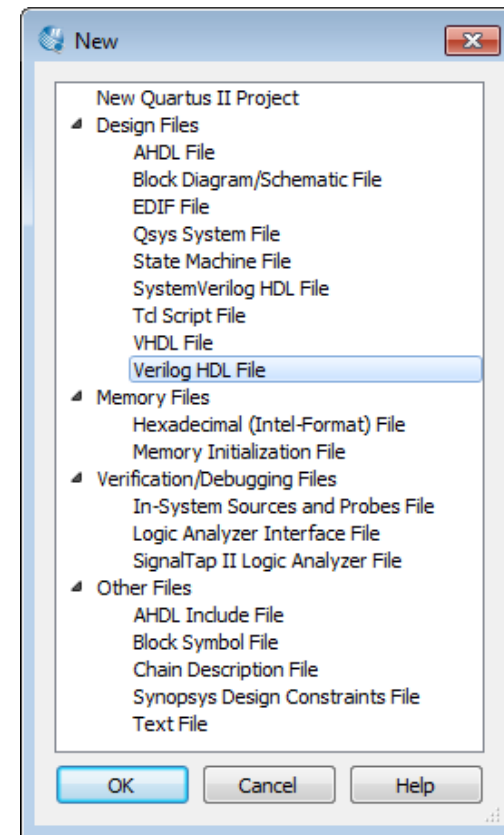
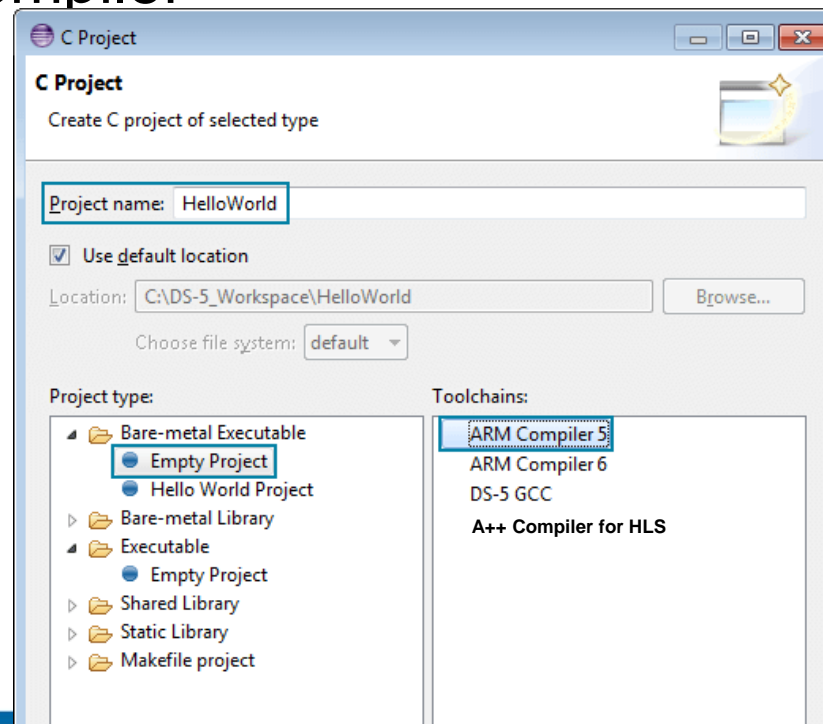
EXE



Quartus II Ecosystem

# 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

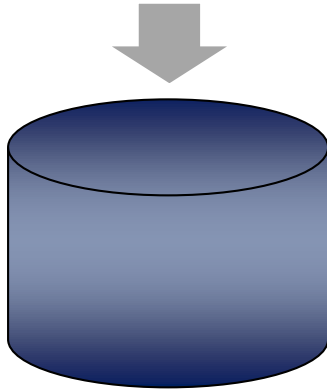
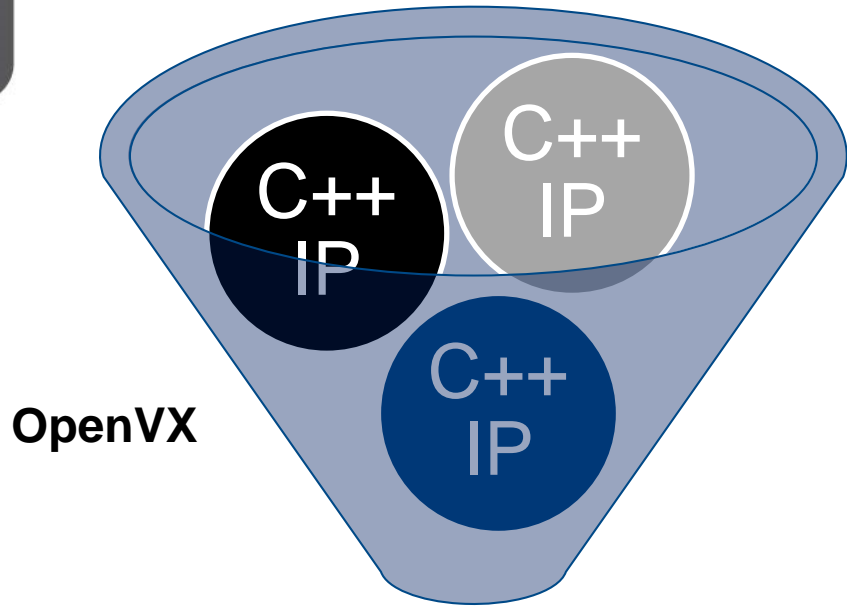


## **A++ Compiler** *for HLS*

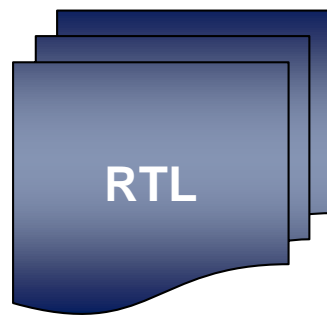
- |  |   |  |
|--|---|--|
| Targets CPU, GPU and FPGAs                       | → | Targets FPGAs  |
| Target user is HW or SW                          | → | Target user is HW  |
| Implements FPGA in software development flow     | → | Implements FPGA in traditional FPGA development flow                 |
| 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



A++ Compiler for HLS

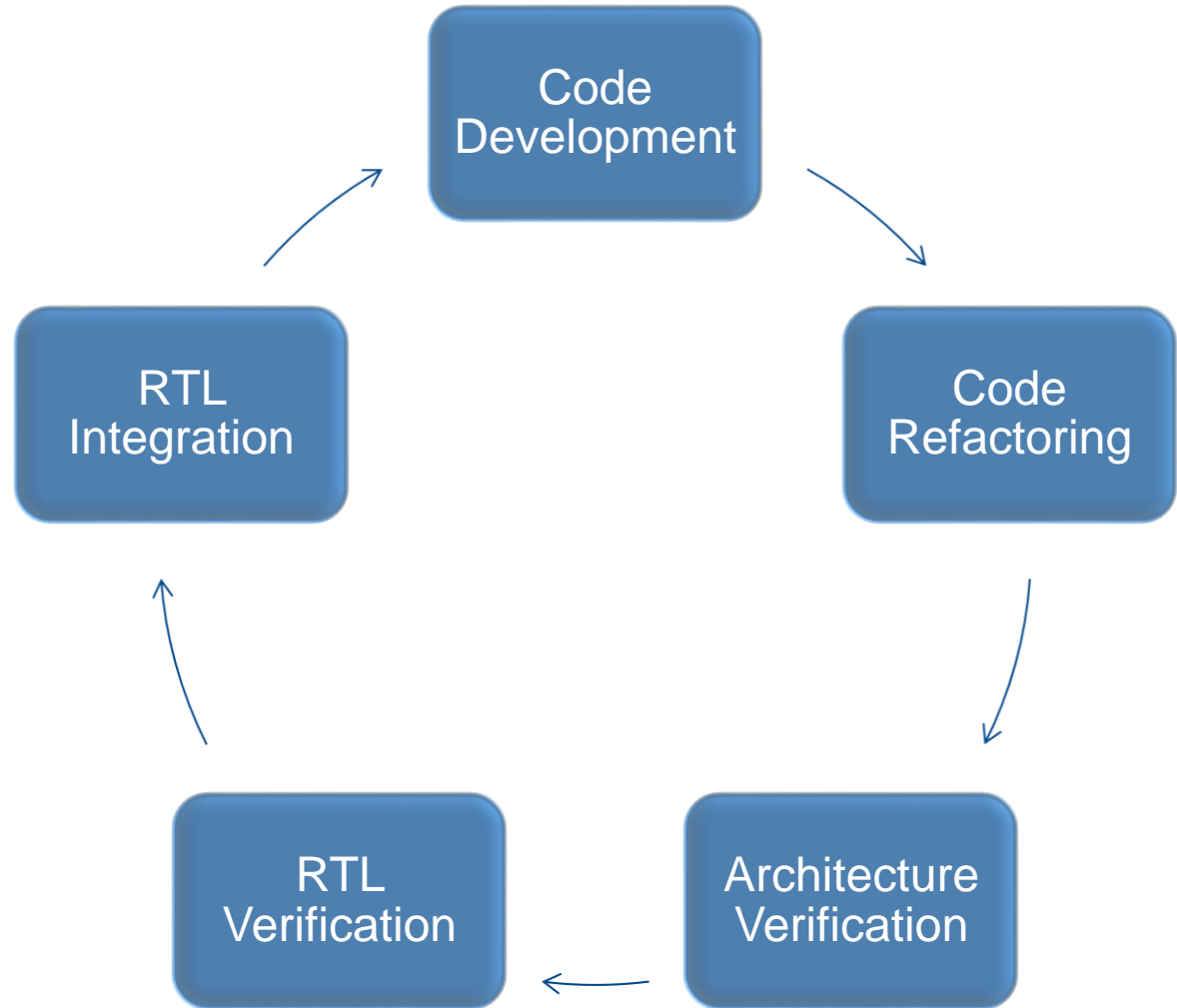
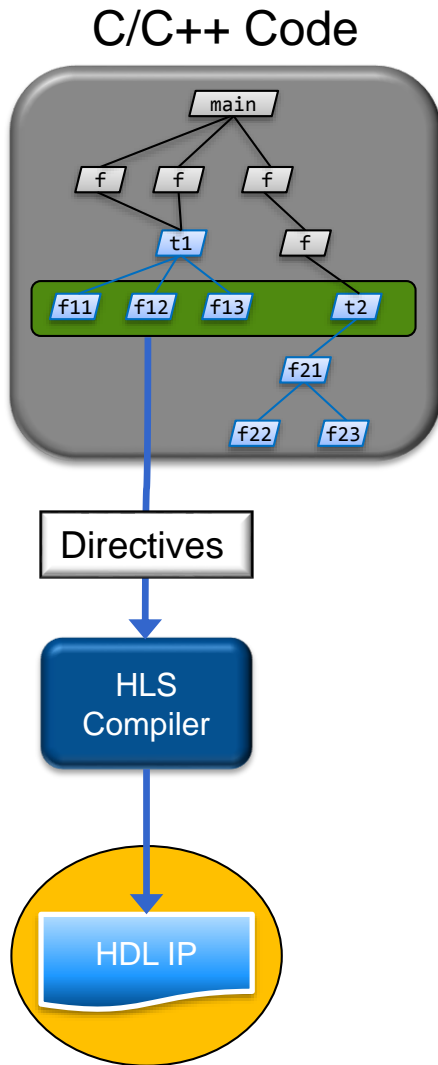


The screenshot shows the Qsys software interface with several tabs at the top: "Contents", "Address Map", "Clock Settings", "Project Settings", "Instance Parameters", and "System". The main area displays a list of components and their connections. The components are organized into groups:

- clk**: Clock Source, Clock Input, Reset Input, Clock Output, Reset Output.
- master\_0**: JTAG to Avalon Master Bridge, Clock Input, Reset Input, Avalon Memory Mapped Master, Reset Output.
- sys\_0**: nios\_sys, Clock Input, Reset Input, onchip\_memory2\_0..., Clock Input, onchip\_memory2\_0..., Avalon Memory Mapped Slave, mm\_bridge\_0\_m0, Avalon Memory Mapped Master.
- address\_span\_exten...**: Address Span Extender, Clock Input, Reset Input, windowed\_slave, Avalon Memory Mapped Slave, expanded\_master, Avalon Memory Mapped Master, cntl, Avalon Memory Mapped Slave.
- address\_span\_exten...**: Address Span Extender, Clock Input, Reset Input, windowed\_slave, Avalon Memory Mapped Slave, expanded\_master, Avalon Memory Mapped Master, cntl, Avalon Memory Mapped Slave.
- hps\_sys**: Hard Processor System, Reset Output, Clock Output, f2h\_axi\_clock, Clock Input, f2h\_axi\_slave, AXI Slave, h2f\_axi\_clock, Clock Input, h2f\_axi\_master, AXI Master, f2h\_sdram0\_data, AXI Slave, f2h\_sdram0\_clock, Clock Input, memory, Conduit, hps\_io, Conduit.

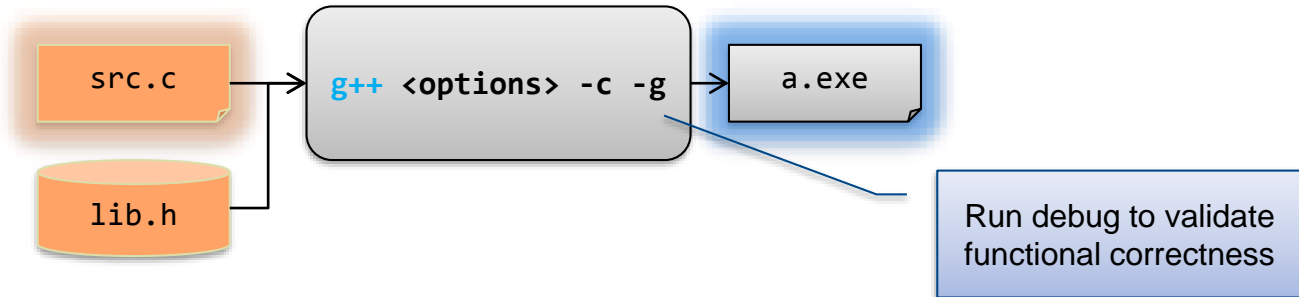
The "Connections" column shows a network of lines connecting the various components.

# HLS Development Phases

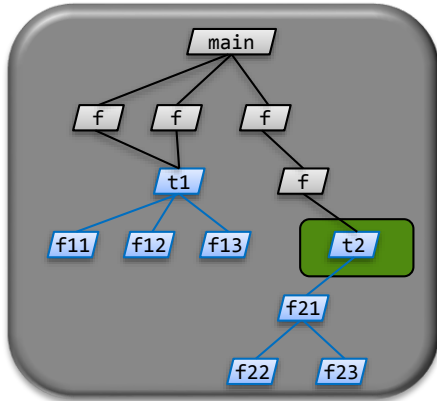


# Code Development

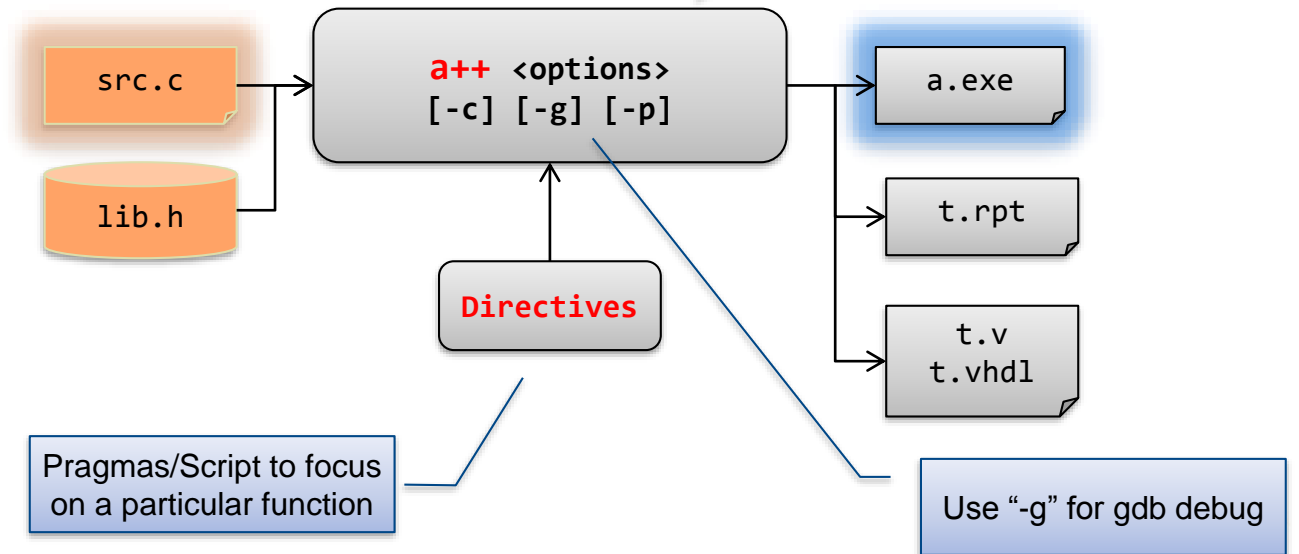
## Develop with C/C++:



## C/C++ Code

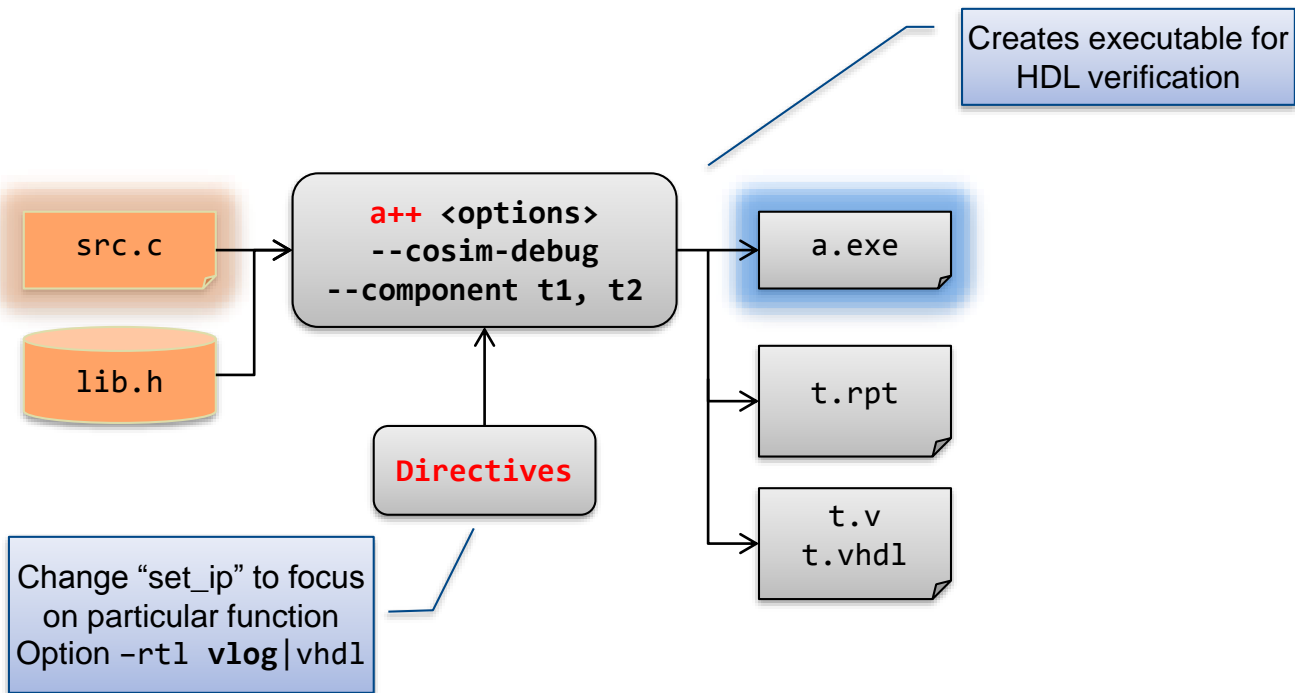
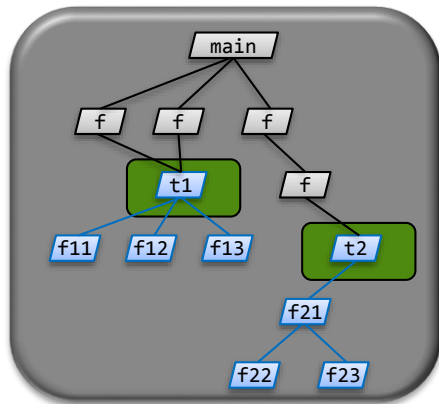


## Run A++ Compiler for HLS:



# 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;
}

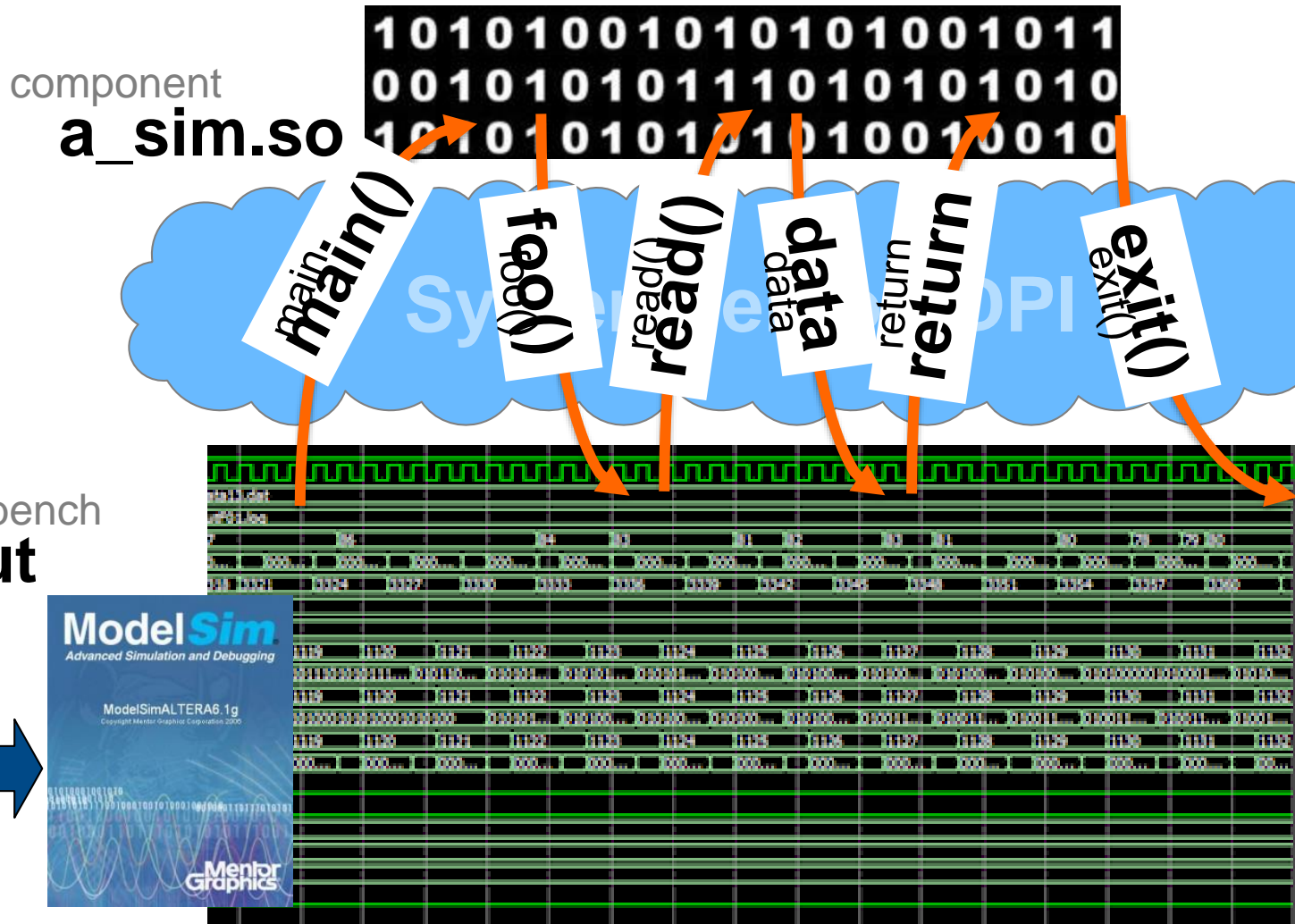
int component accelerate(int x, int y){
    return impl(x,y);
}

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



# 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 FPGA developers to increase their productivity and time to market
  - Simulating C code is thousands of times 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