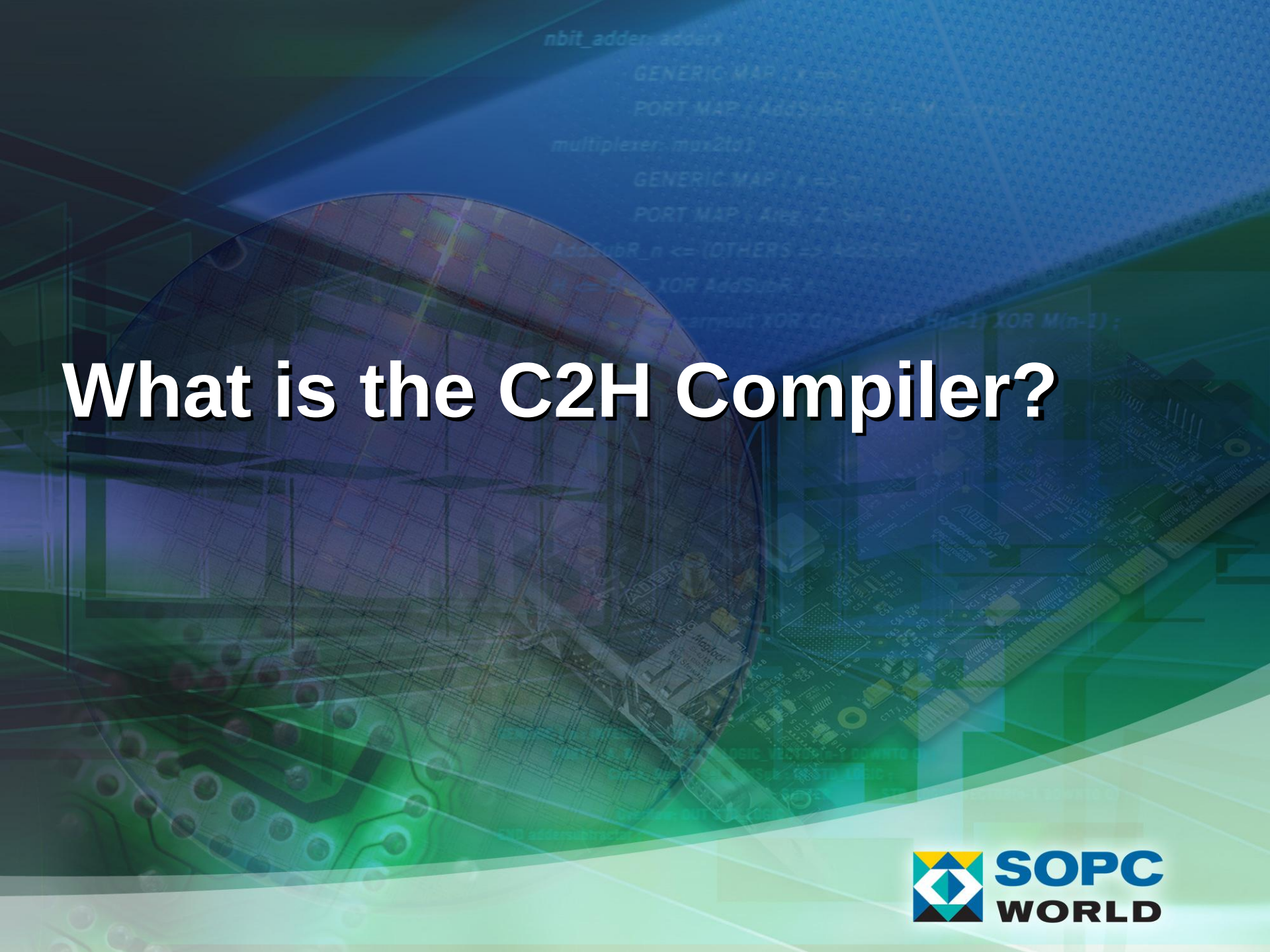
Improving Software Performance Using Hardware Acceleration

SOPC WORLD

Agenda

- What is the C-to-Hardware Acceleration (C2H) Compiler?
- How does it work?
- C2H Compiler “under the hood”
- C-based industry partnership
- Summary
- Q&A

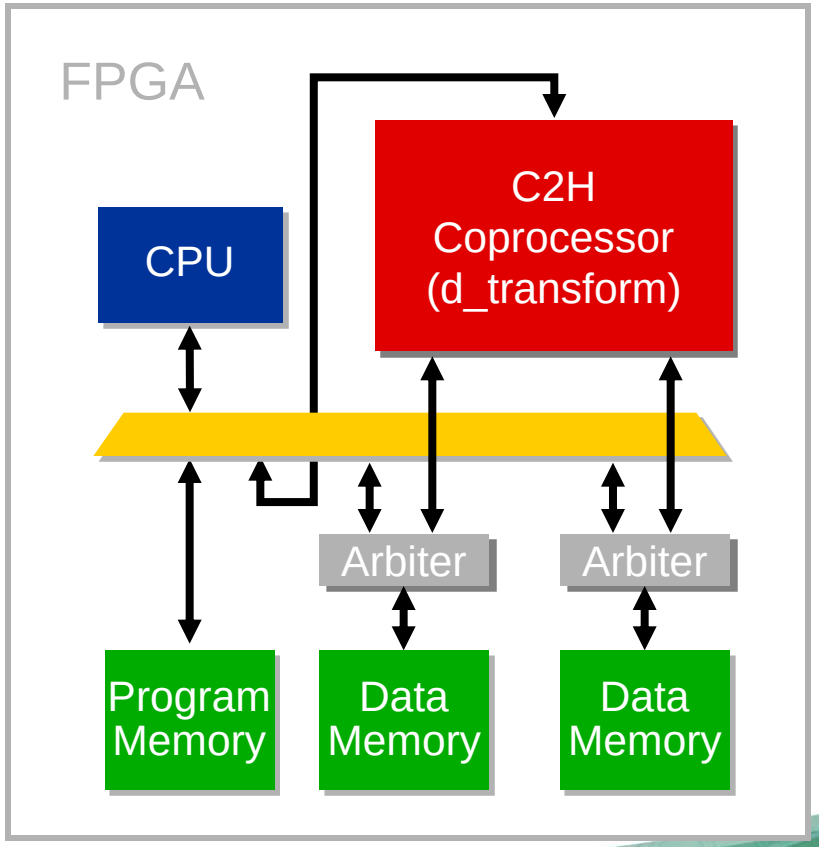
What is the C2H Compiler?



SOPC WORLD

What Is the C2H Compiler?

- A productivity tool to automatically create hardware accelerators from ANSI C code



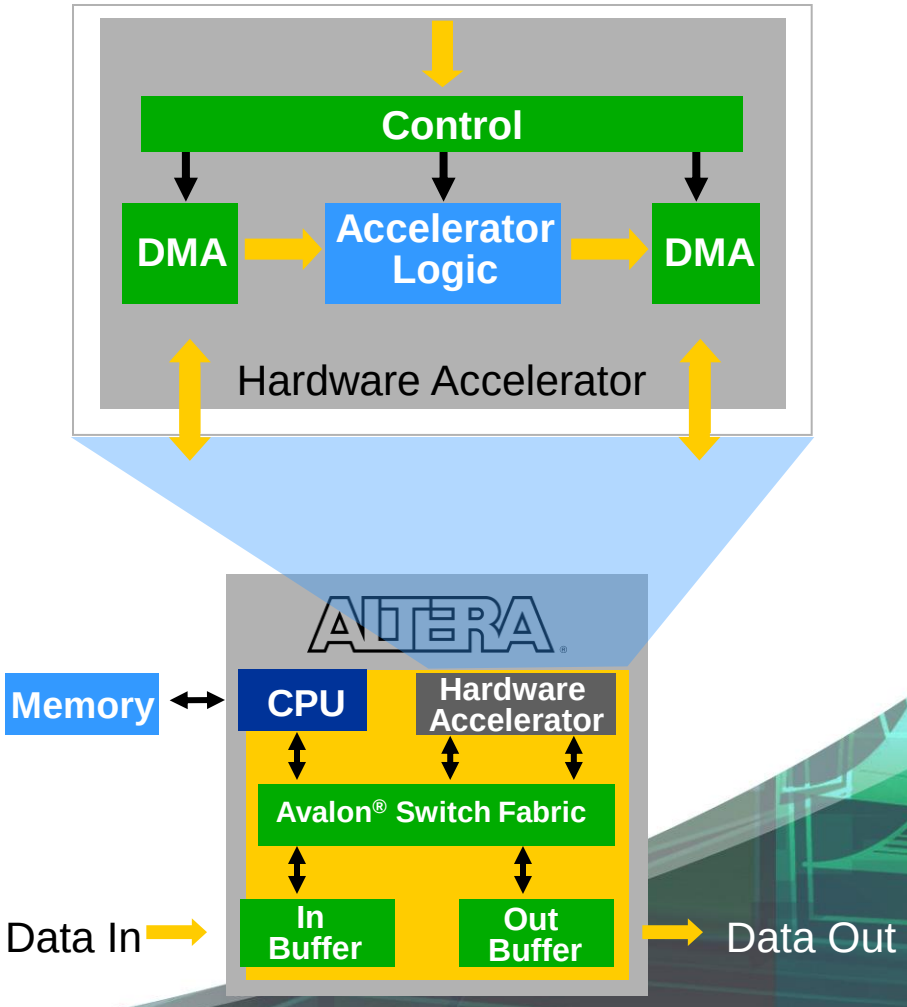

```
main ()
{ …variable declarations…
  init();

  while (!error && got_data())
  {
    do_user_interface();
    gather_statistics();
    if (got_new_data())
      d_transform(in_buf, out_buf);
    check_for_errors();
  }
  cleanup();
}
```


SOPC WORLD

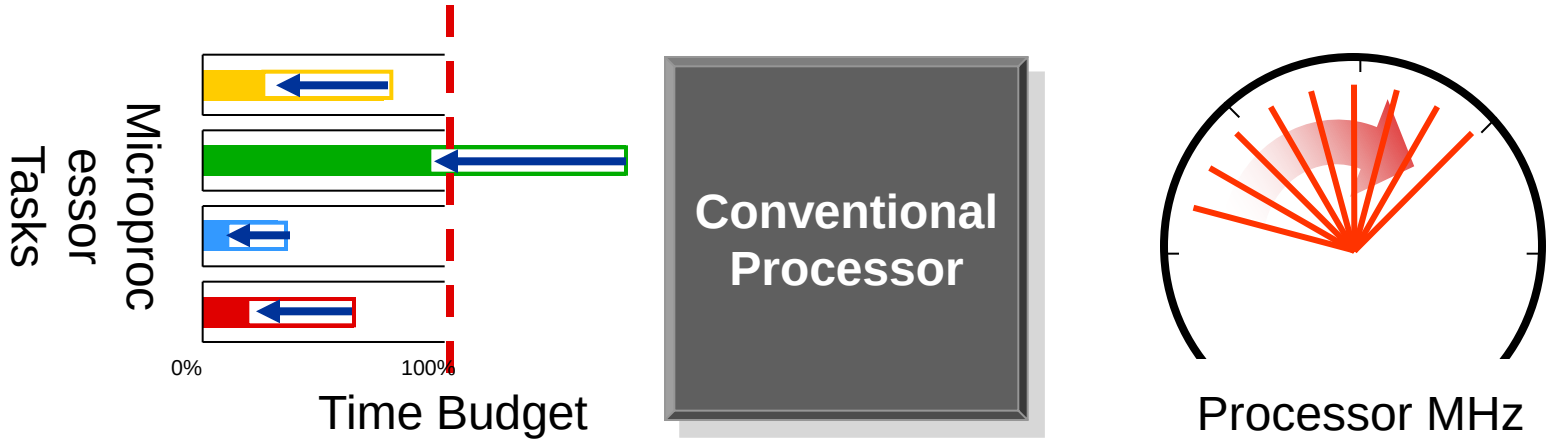
Nios II C2H Compiler Overview

- ***Compiles*** performance-critical C code into fast hardware accelerators in FPGA
- ***Integrates*** accelerators into Nios II system
- ***Links*** accelerators into Nios II software build process
- ***Boosts*** performance by 10X or more for many applications



SOPC WORLD

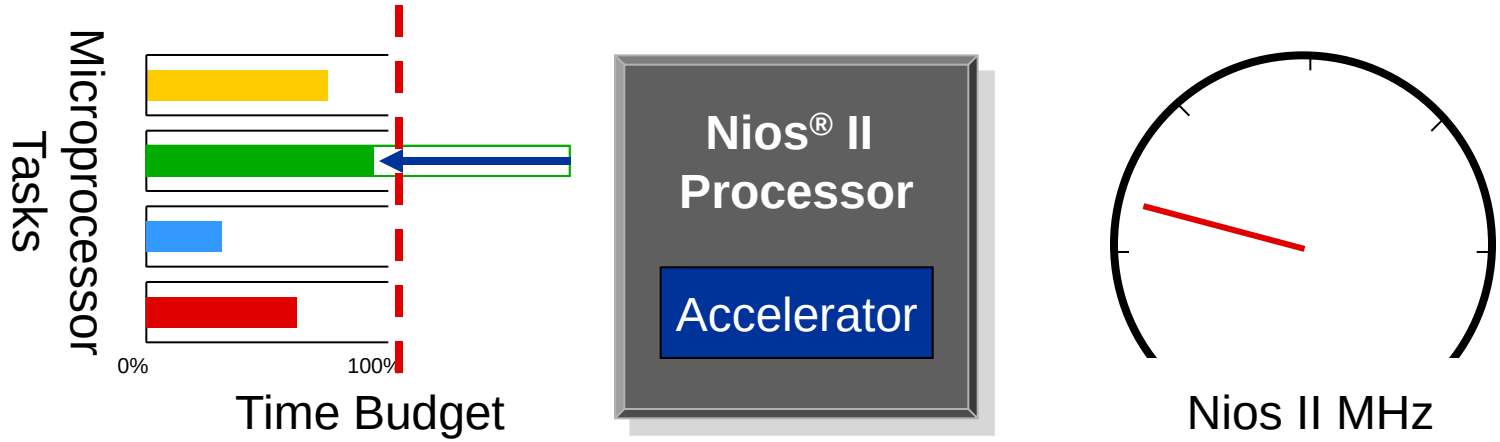
Traditional Processor Acceleration



Potential Issues With Power, Board Layout, Memory Speed, Device Cost, and Availability

SOPC WORLD

Accelerate Only What Is Needed

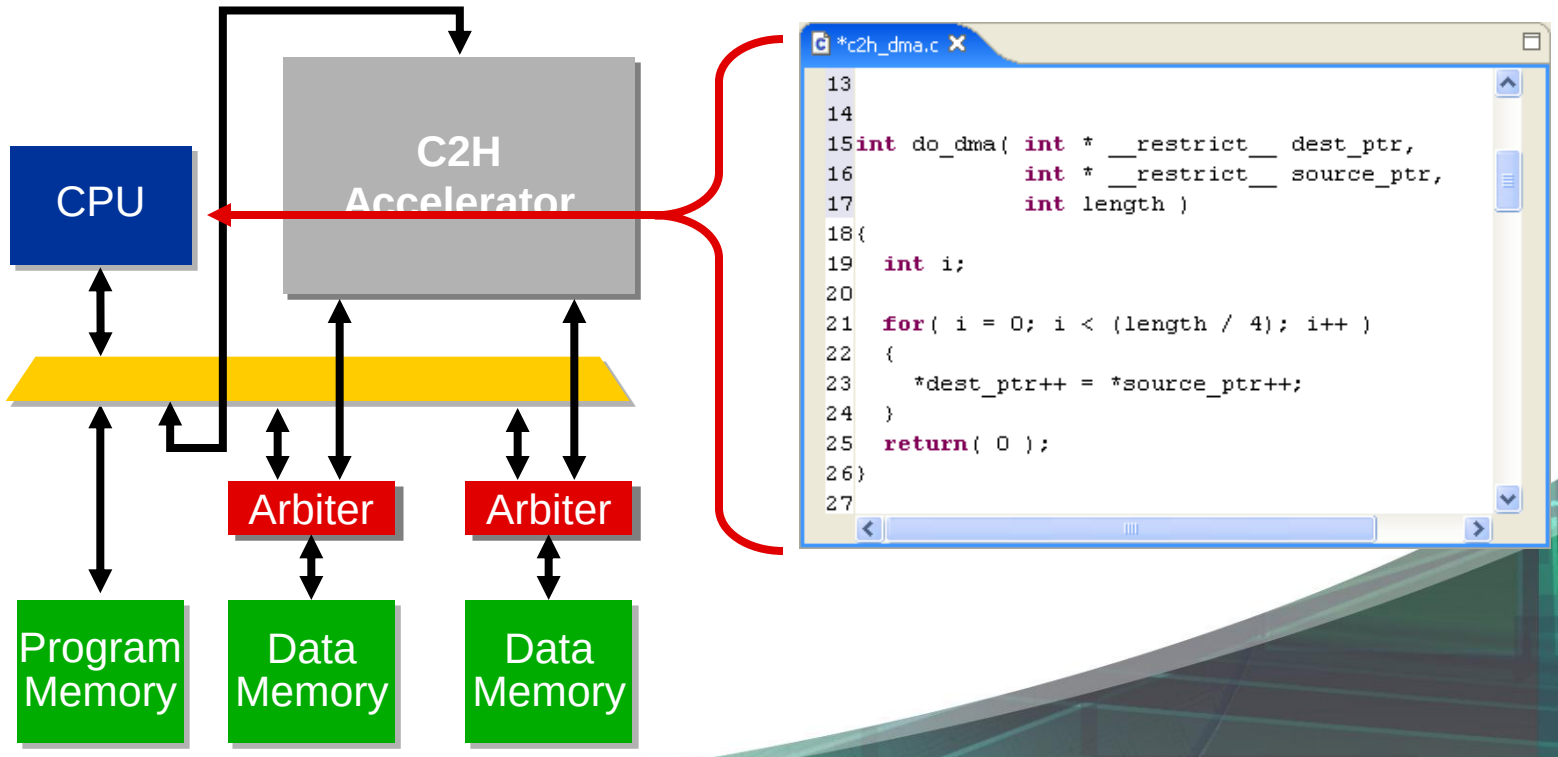


Transfer Processing to Hardware
Highly Effective, Minimal Impact to System

SOPC WORLD

What Does C2H Do?

- Generates a custom hardware accelerator from an ANSI C function.



SOPC WORLD

Hand-Crafted Accelerator vs. C2H Compiler

Standard Flow

- Profile code
- Identify bottlenecks
- Repartition memory
- Create/modify an accelerator
  - HDL
    - Design
    - Simulate
  - C wrapper
    - Write function
    - Test
    - Re-build C code
  - Import into SOPC Builder
- Repeat based upon results

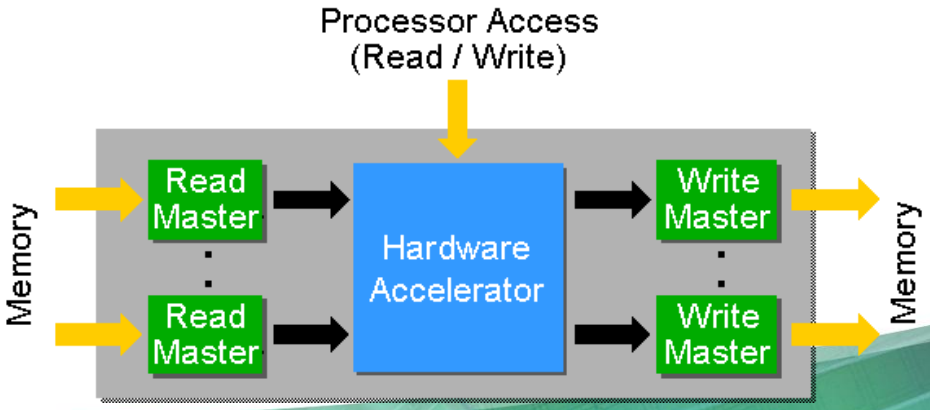
C2H Compiler Flow

- Profile code
- Identify bottlenecks
- Repartition memory
- Select code
  - Right-click to accelerate
  - Build
- Repeat based upon results

SOPC WORLD

Anatomy of an Accelerator

- Multiple master ports to memory
  - Handles pointer dereferencing
  - Pipelined logic
  - Matched to memory latency
  - Sustain *extremely* high memory bandwidth
- Slave port for CPU control
  - Registers for parameter passing and synchronization
- Software wrapper
  - Replaces function call



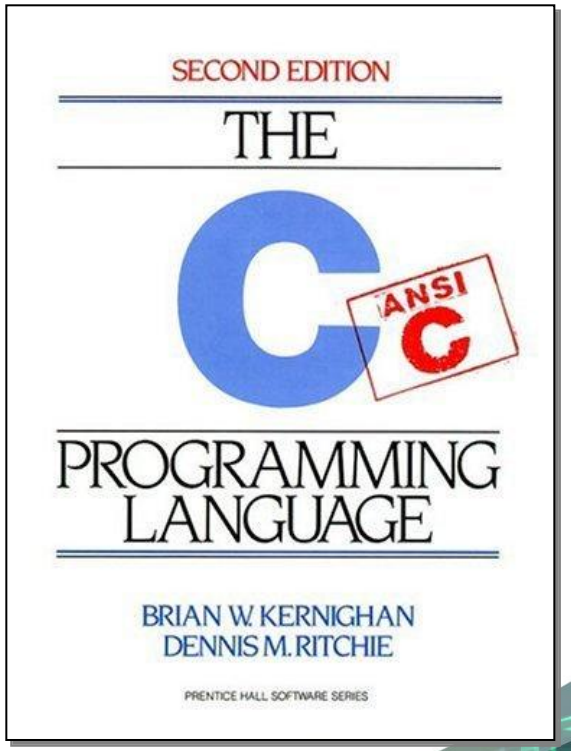
C2H Compiler Key Facts

- C2H Compiler accelerates ANSI C code
  - No new language to learn
- Aimed at accelerating a critical function
  - Not whole software design
- Relies on Nios II processor
  - Built into system with SOPC Builder
- Used within Nios II Integrated Development Environment (IDE)
  - Build your application software
  - “Point and accelerate” critical functions

SOPC WORLD

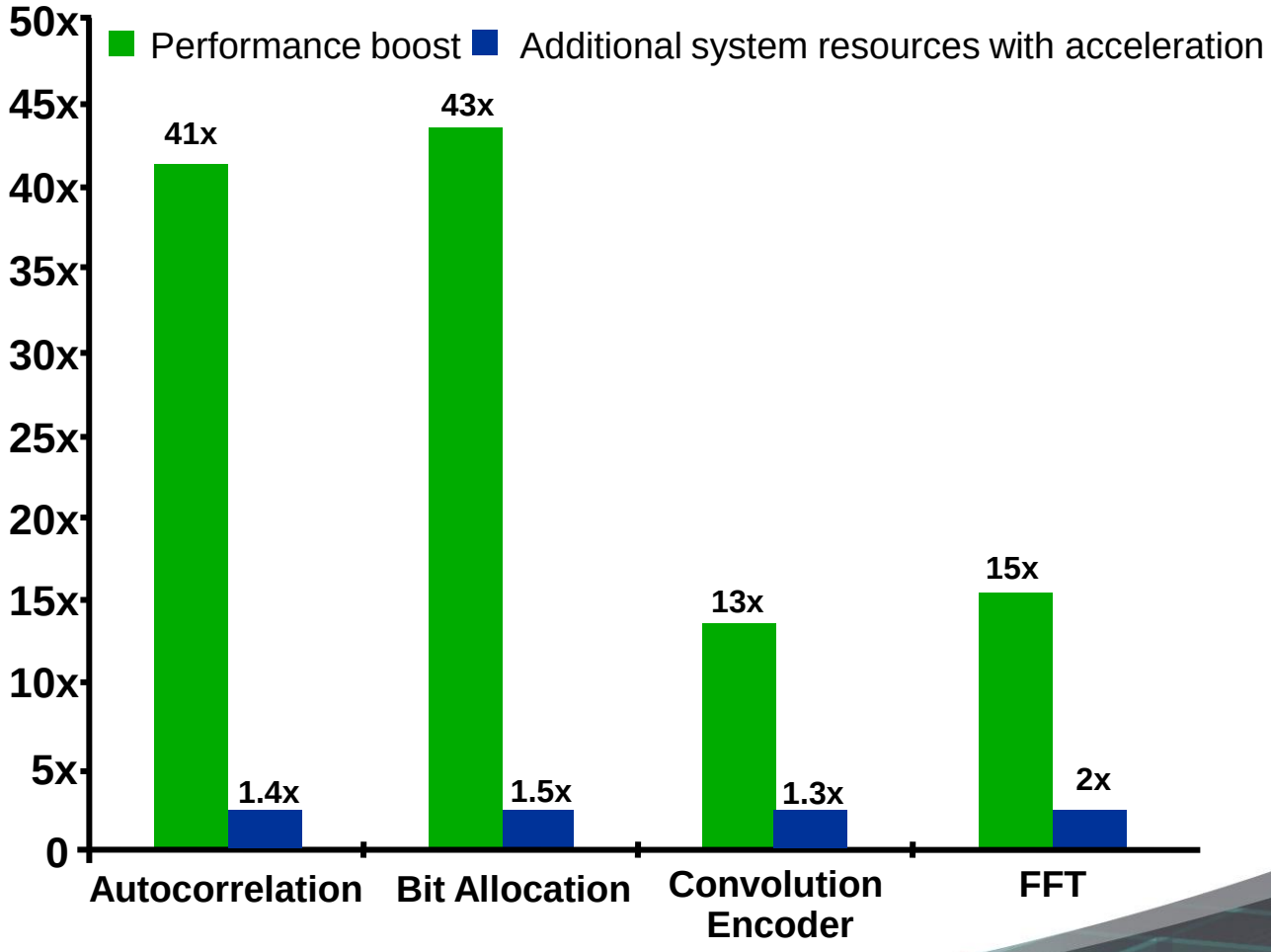
ANSI C Language Support

- All data types
- All operators
- All control-flow constructs
- All looping constructs
- Macros
- Function calls
- Pointer and array access



SOPC WORLD

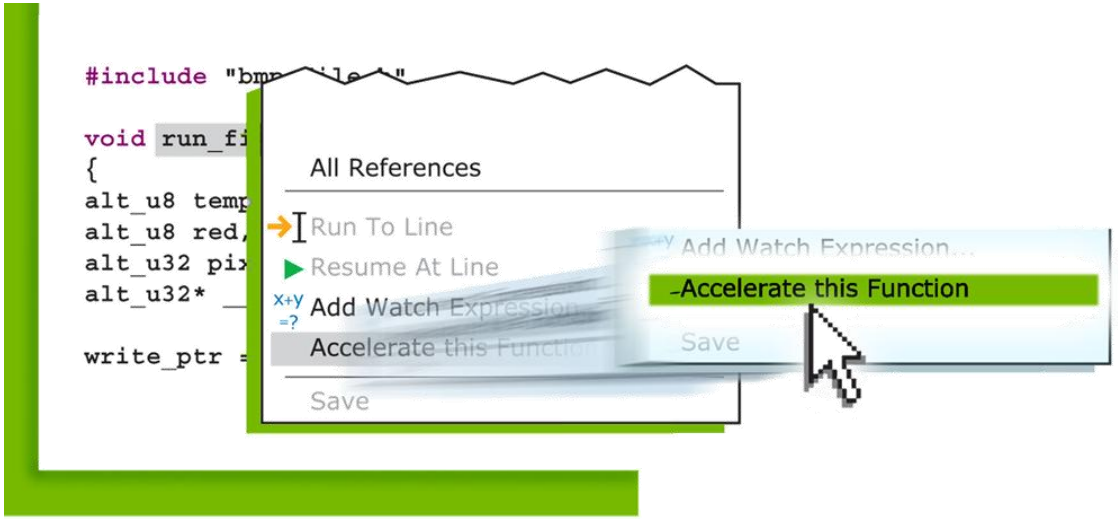
Dramatic Performance Boost



SOPC WORLD

Advanced Productivity Tool

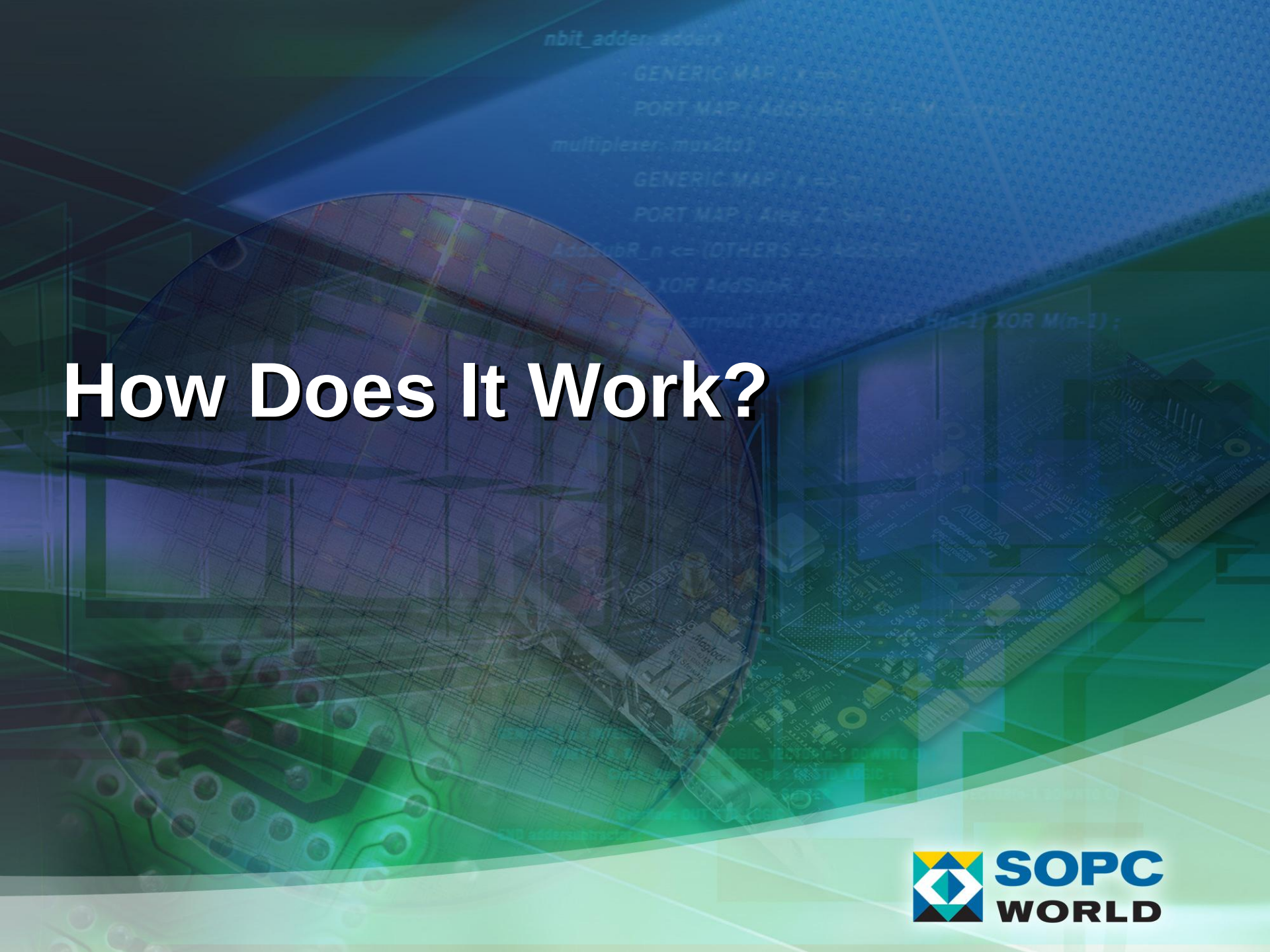
- Automates creation and integration of hardware accelerators
- Intuitive user interface streamlines C acceleration
- Fully integrated with Nios II IDE
- Tool license costs $2,995



Right-Click to Accelerate

SOPC WORLD

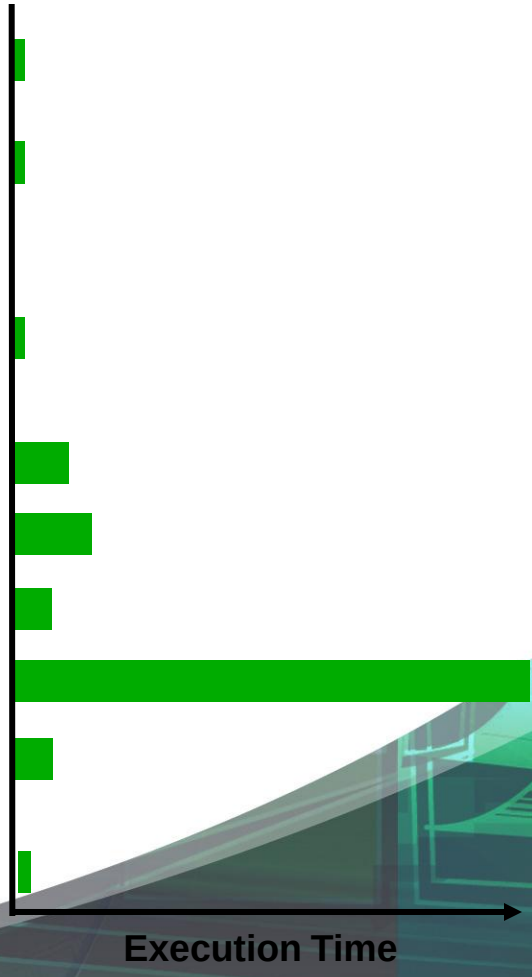
How Does It Work?



Step 1: Identify Software Bottlenecks

```
main ()
{ …variable declarations…
  init();

  while (!error && got_data())
  {
    do_user_interface();
    gather_statistics();
    if (got_new_data())
      d_transform(in_buf, out_buf);
    check_for_errors();
  }
  cleanup();
}
```

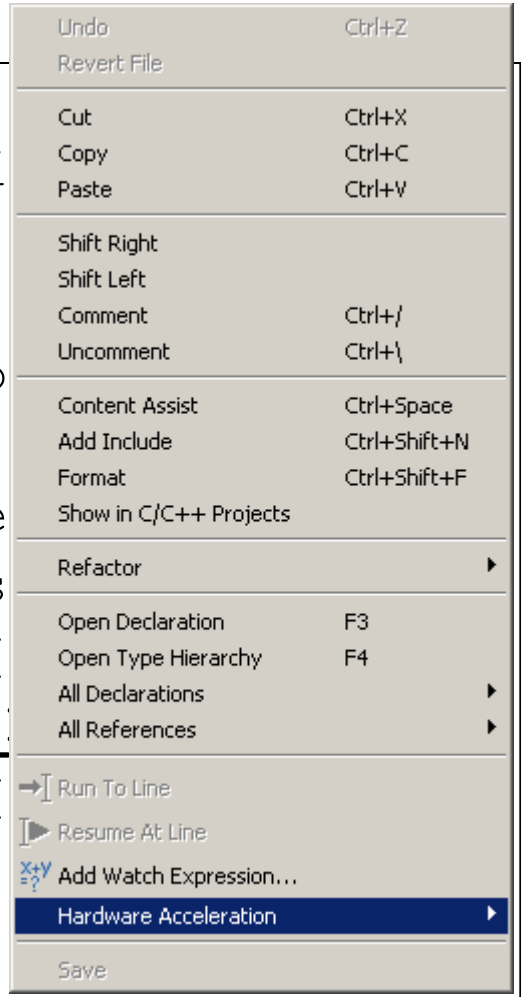


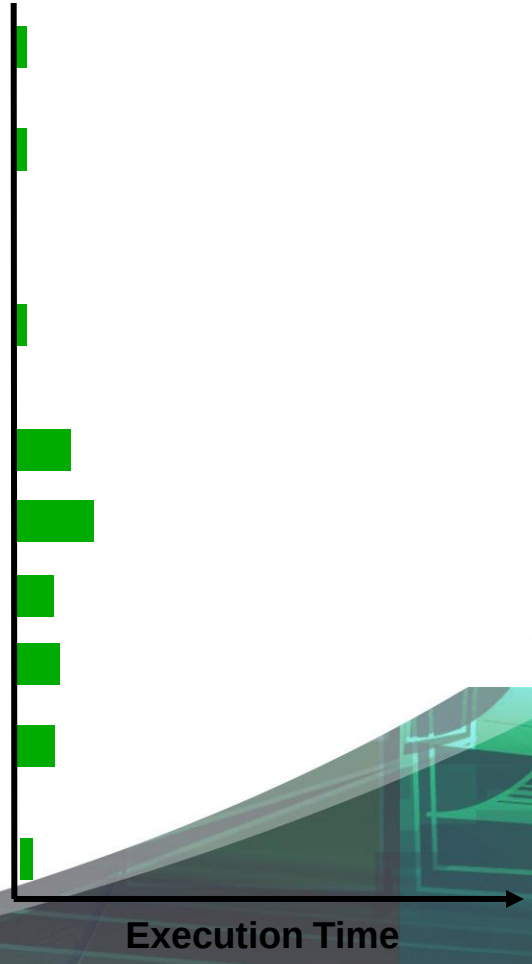
SOPC WORLD

Step 2: Right-Click to *Accelerate*

```
main ()
{ …variable declarati
  init();

  while (!error && go
  {
    do_user_interface
    gather_statistics
    if (got_new_data(
    d_transform (in_
    check_for_errors(
  }
  cleanup();
}
```

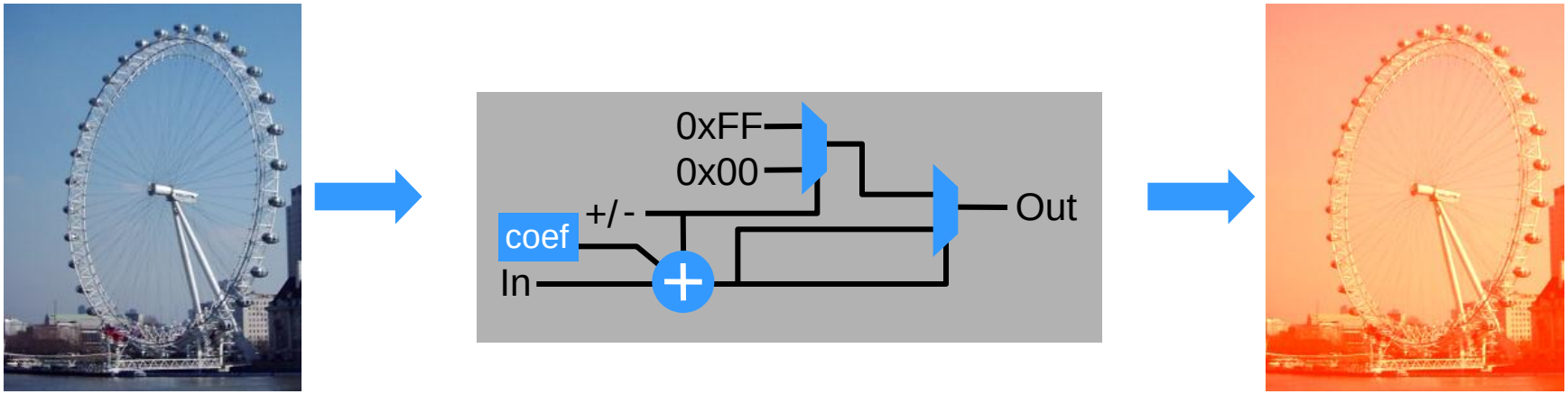




SOPC WORLD

Color Filtering Demonstration

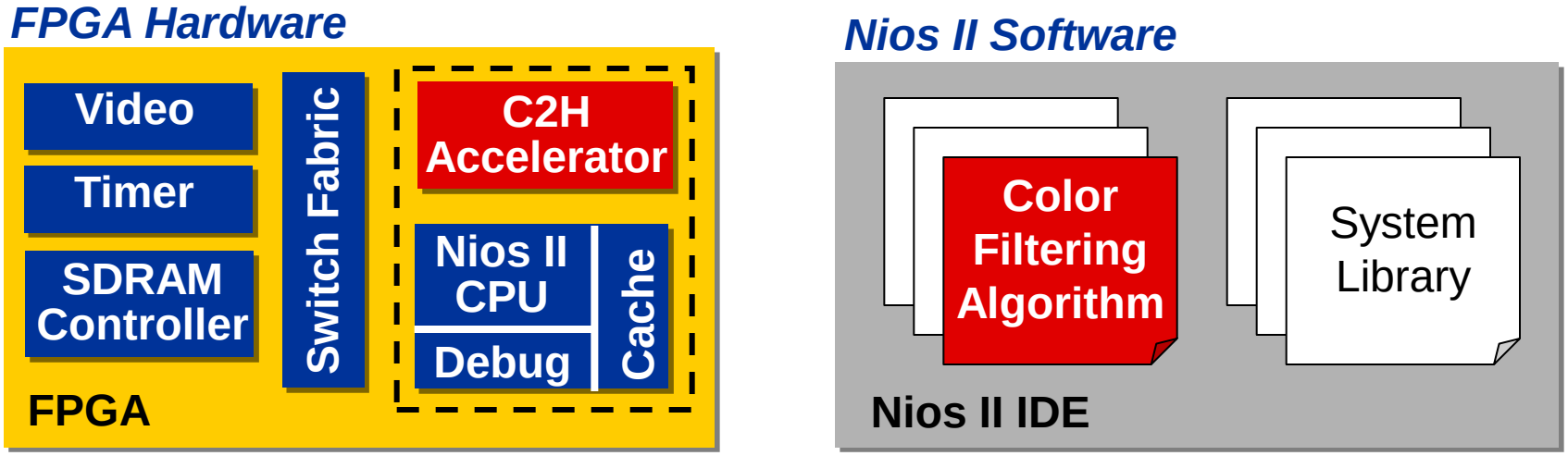
input.bmp | **Color Filtering (RGB)** | **output.bmp**



- Demo
  - Image color filtering
  - Algorithm applied to every pixel (RGB)
- Critical path
  - Color-filtering algorithm
  - Accelerated via C2H Compiler
- Timer measurement
  - Indicates time taken by software vs. C2H acceleration

SOPC WORLD

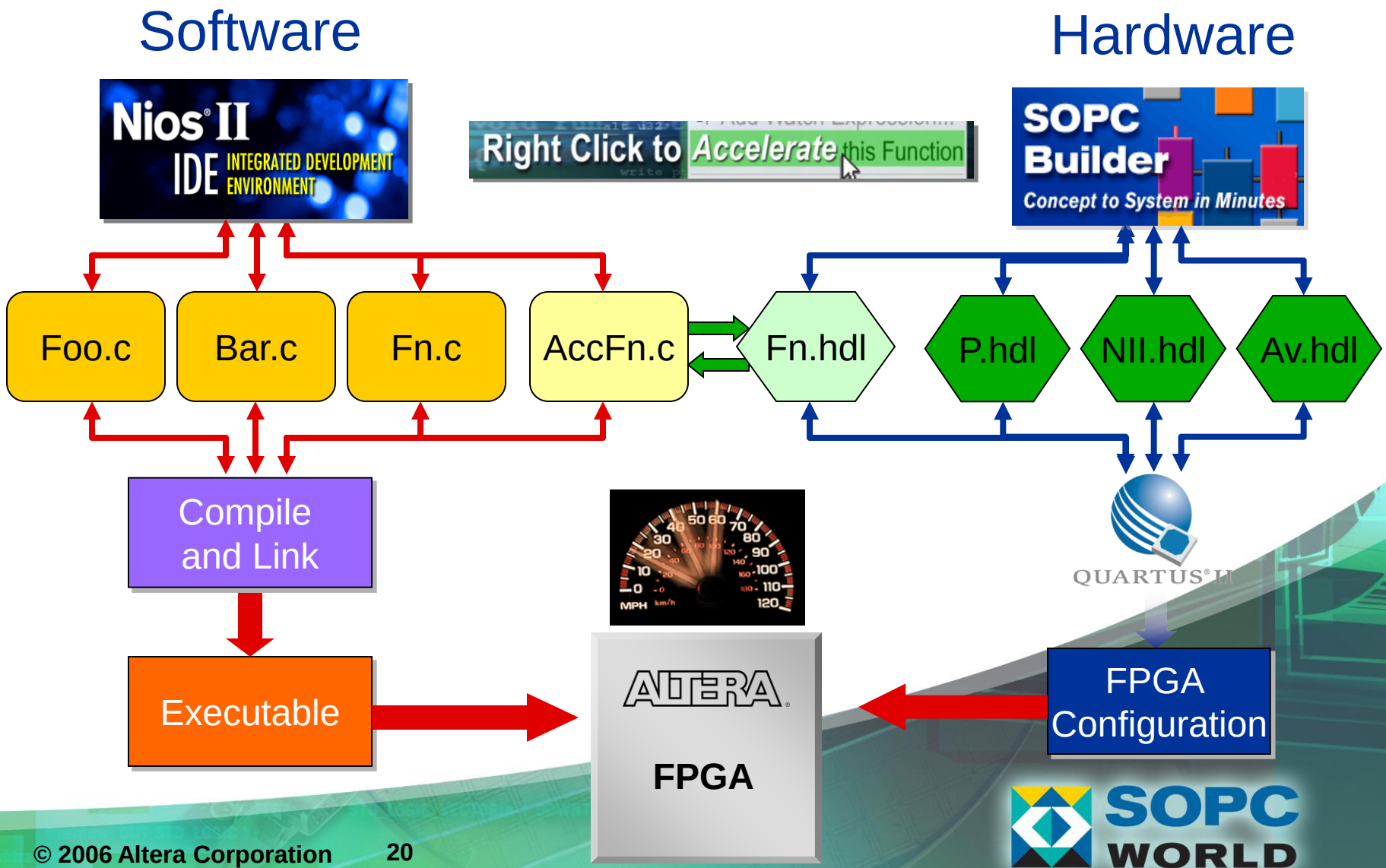
Step by Step: Building the System



- Base system hardware
  - Already built in SOPC Builder
- C2H Compiler demonstration
  1. Create software in Nios II IDE
  2. Accelerate system performance with C2H Compiler

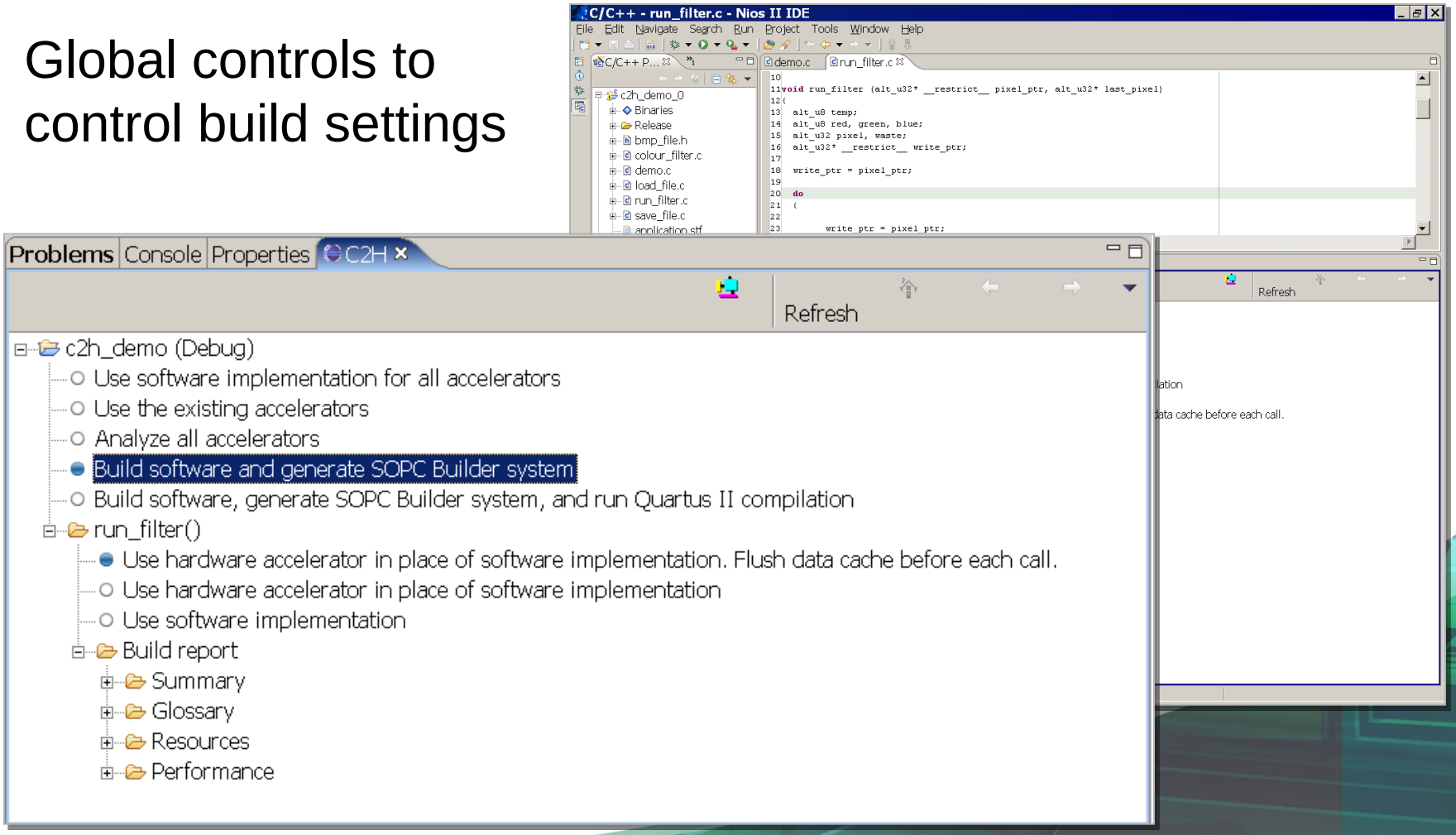
SOPC WORLD

C2H Compiler Integration



C2H Compiler Controls

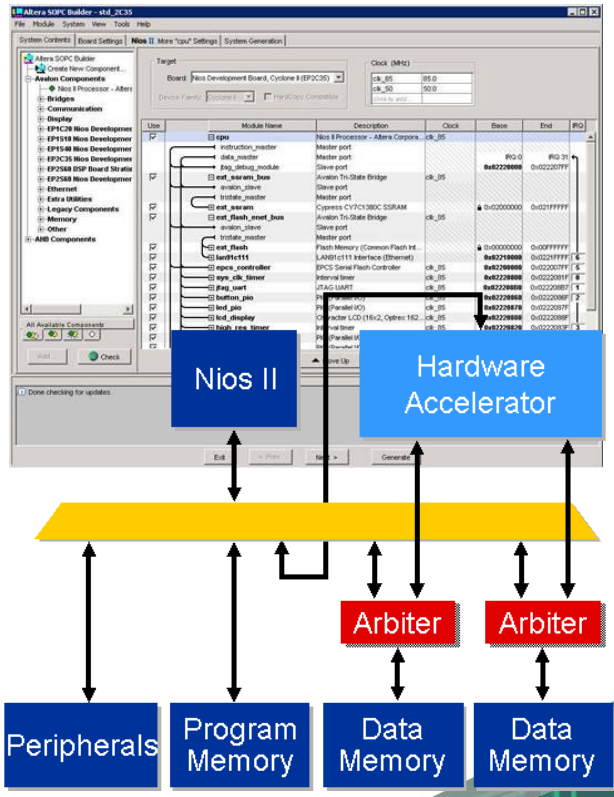
- Global controls to control build settings



SOPC WORLD

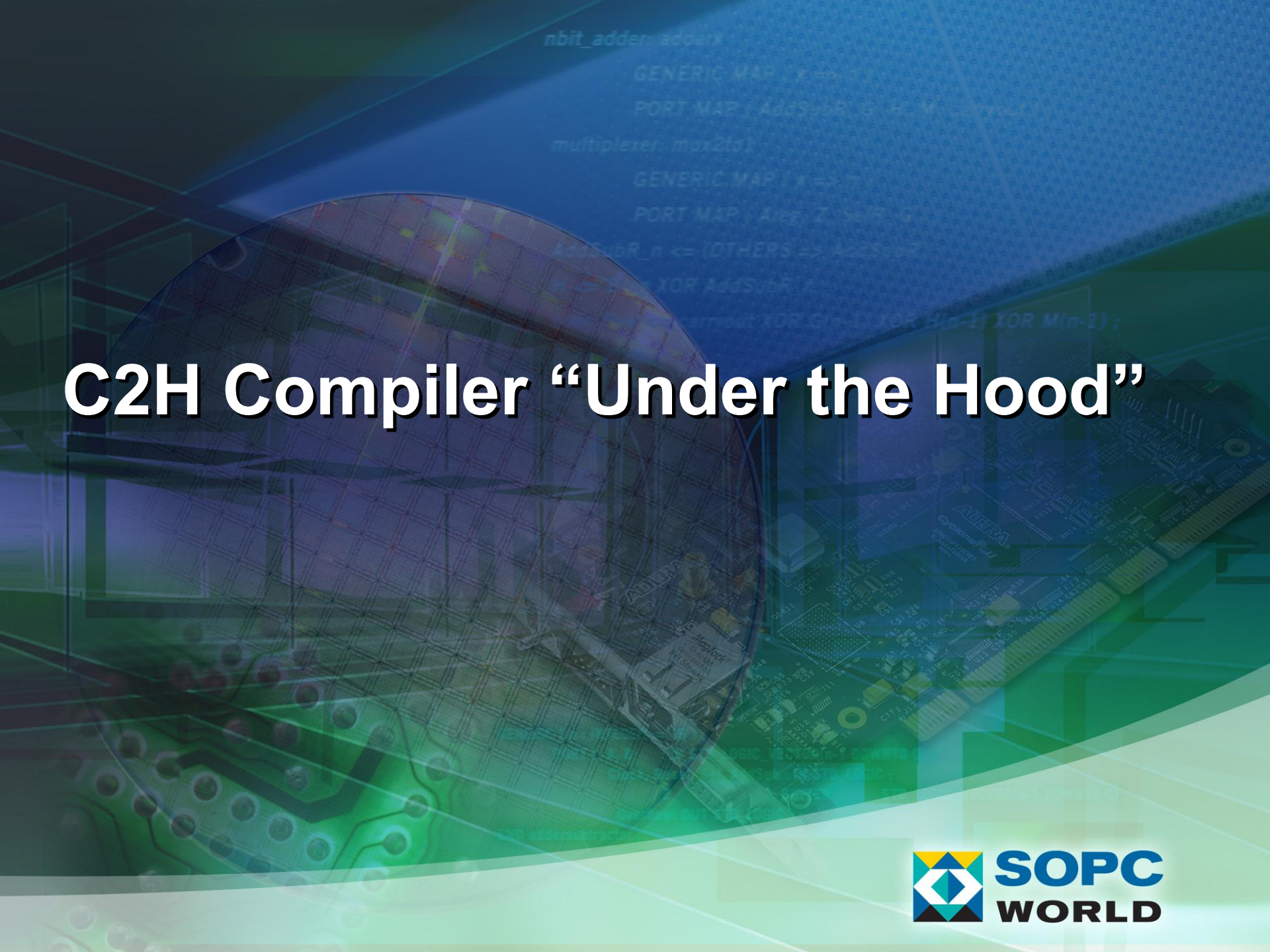
"Secret Sauce" Behind C2H Compiler

- SOPC Builder
  - Connects CPU, accelerator, and memory
  - Understands pipeline depth for scheduling
- Avalon interconnect fabric
  - Resolves pointer dereferencing
  - High bandwidth interconnect
  - Parallel memory access



SOPC WORLD

C2H Compiler “Under the Hood”



SOPC WORLD

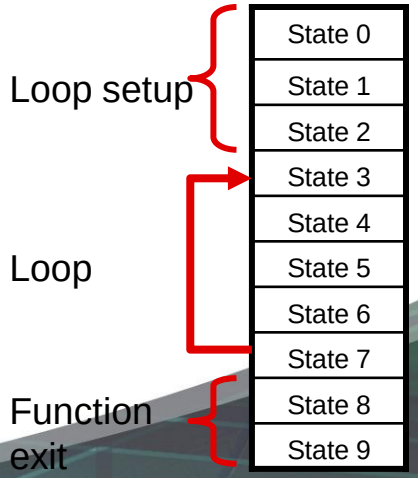
Controlling Logic Generation

- "What you type is what you get"
- Every arithmetic operator you type
  - Makes equivalent hardware unit in accelerator
    - Examples: *, +, %, >>
  - Some arithmetic requires zero logic (wires)
    - Examples: *y = x << 3, x &= 0xf;*
- Control-flow statements generate control state machine
- Pointer and array references create pipelined master interfaces to memory

Example: Arithmetic Hardware

```
long long MAC
  (int *a, int *b, int len)
{
   long long result = 0;
   while (len > 0) {
      result += *a++ * *b++;
      len--;
   }
   return result;
}
```

- 1 - 32x32=64 multiplier
- 3 - 32-bit incrementers
- 1 - 64-bit adder
- 1 - 32-bit comparator
- 2 - read-masters
- Nominal control logic



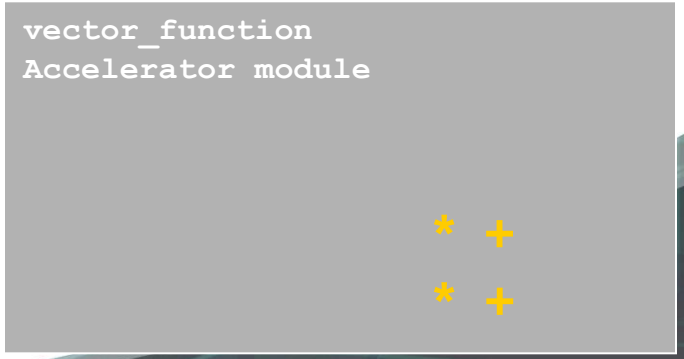
SOPC WORLD

Example: Controlling Implementation

```
void vector_function ()
{
  long long x, y;
  int *a,*b,*c,*d;
  int len;
   <… Various set-up …>

  while (<loop-condition>) {
     <…loop overhead…>
     while (len-- > 0) {
       x += *a++ * *b++;
       y += *c++ * *d++;
     }
     <…result-saving…>
  }
}
```

- 1 accelerator unit created
  - Incorporates all arithmetic
- 2 *s typed
  - Creates 2 multipliers
- Control flow
  - X and Y computed in parallel
  - Y does not need to wait for X
- Faster unit, more hardware



SOPC WORLD

Example: Controlling Implementation

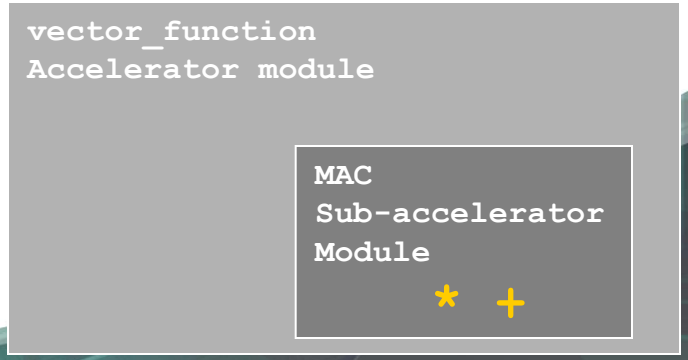
```
void vector_function ()
{
  long long x, y;
  int *a,*b,*c,*d;
  int len;
   <… Various set-up …>

  while (<loop-condition>) {
     <…loop overhead…>

     x = MAC (a, b, len);
     y = MAC (c, d, len);

     <…result-saving…>
  }
}
```

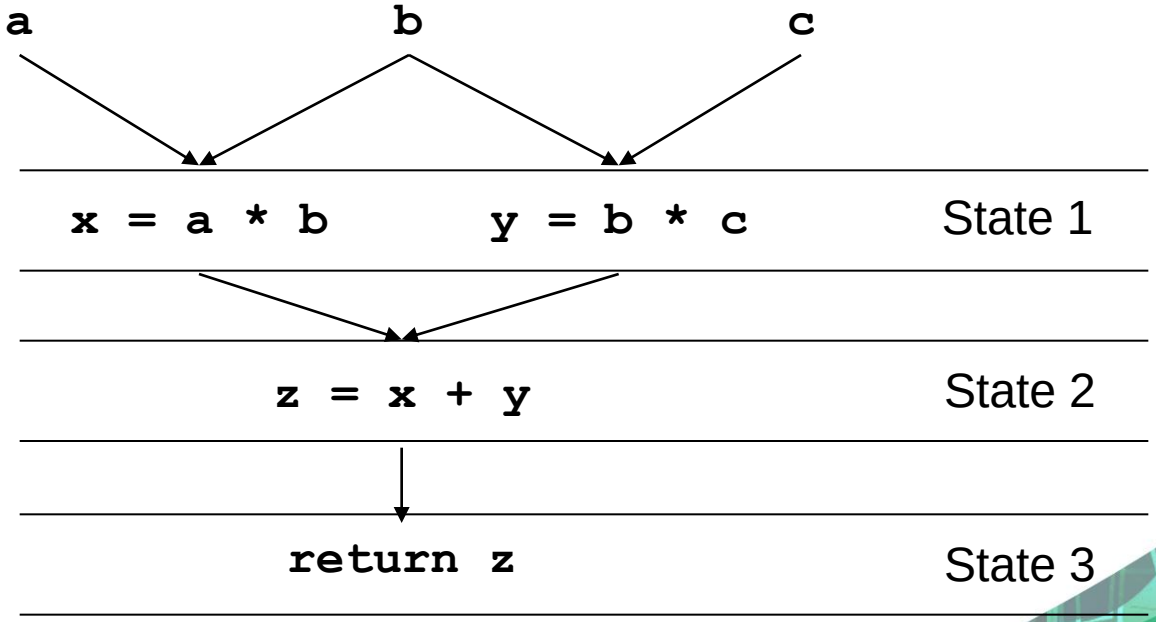
- 2 accelerator units created
  - vector_function
  - MAC
- MAC unit is *shared*
  - Only 1 multiplier (MAC)
  - Used twice
- Control flow
  - Y esult *waits* for X result



SOPC WORLD

Basic Data Dependencies

```
int foo(int a, int b, int c)
{
  int x ,y, z;
  x = a * b;
  y = b * c;
  z = x + y;
  return z;
}
```

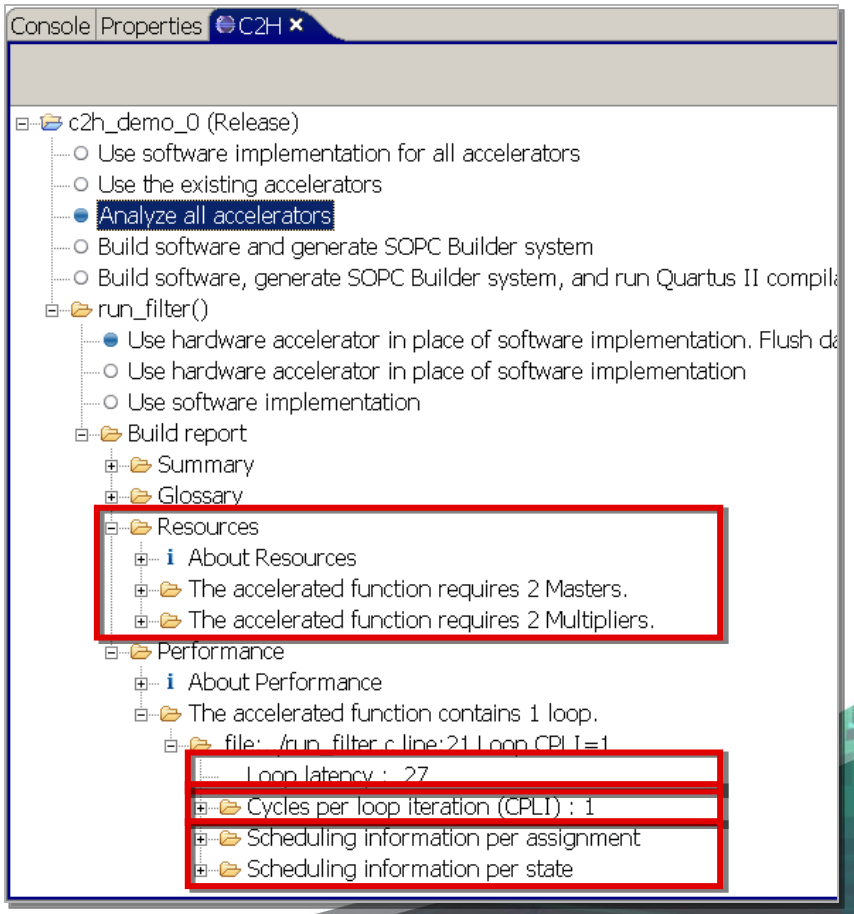


States assigned directly from the dependency graph

SOPC WORLD

Performance/Resource Report

- Accelerator data
  - Hardware used
  - Loop latency
  - Clocks per loop iteration
  - Algorithm scheduling



SOPC WORLD

Sequential Operation

- Software calls hardware accelerators sequentially
  - Processor must wait for accelerators to finish
  - Reduces benefit of dedicated hardware

```
main ()
{ …variable declarations…
  init();
  while (!error && got_data())
  {
    do_user_interface();
    gather_statistics();
    if (got_new_data())
      edge_detect(in_buf1, out_buf1);
      image_rotate(in_buf2, out_buf2);
    check_for_errors();
  }
  cleanup();
}
```

Concurrent Operation

- Software calls accelerators as *threads*
  - Each accelerator runs concurrently
  - True multitasking system (CPU, hardware accelerators)

```
main ()
{ …variable declarations…
  init();
  message_queue* msg;

  thread_start (&edge_detect, msg);
  thread_start (&image_rotate, msg);
  thread_start (&gather_statistics, msg);

  while (!msg->done()) { sleep();}
}
```

C-based Industry Partnership

SOPC WORLD

Best of Altera and Partner Technologies

- C language design tools available today from industry-leading suppliers

- C-based design initiative
  - Open application program interface (API) for Altera® system generation tools
    - SOPC Builder
    - C2H Compiler
  - Technology sharing and joint development
    - "Right-click to accelerate" for ESL partner tools from within Nios II IDE

SOPC WORLD

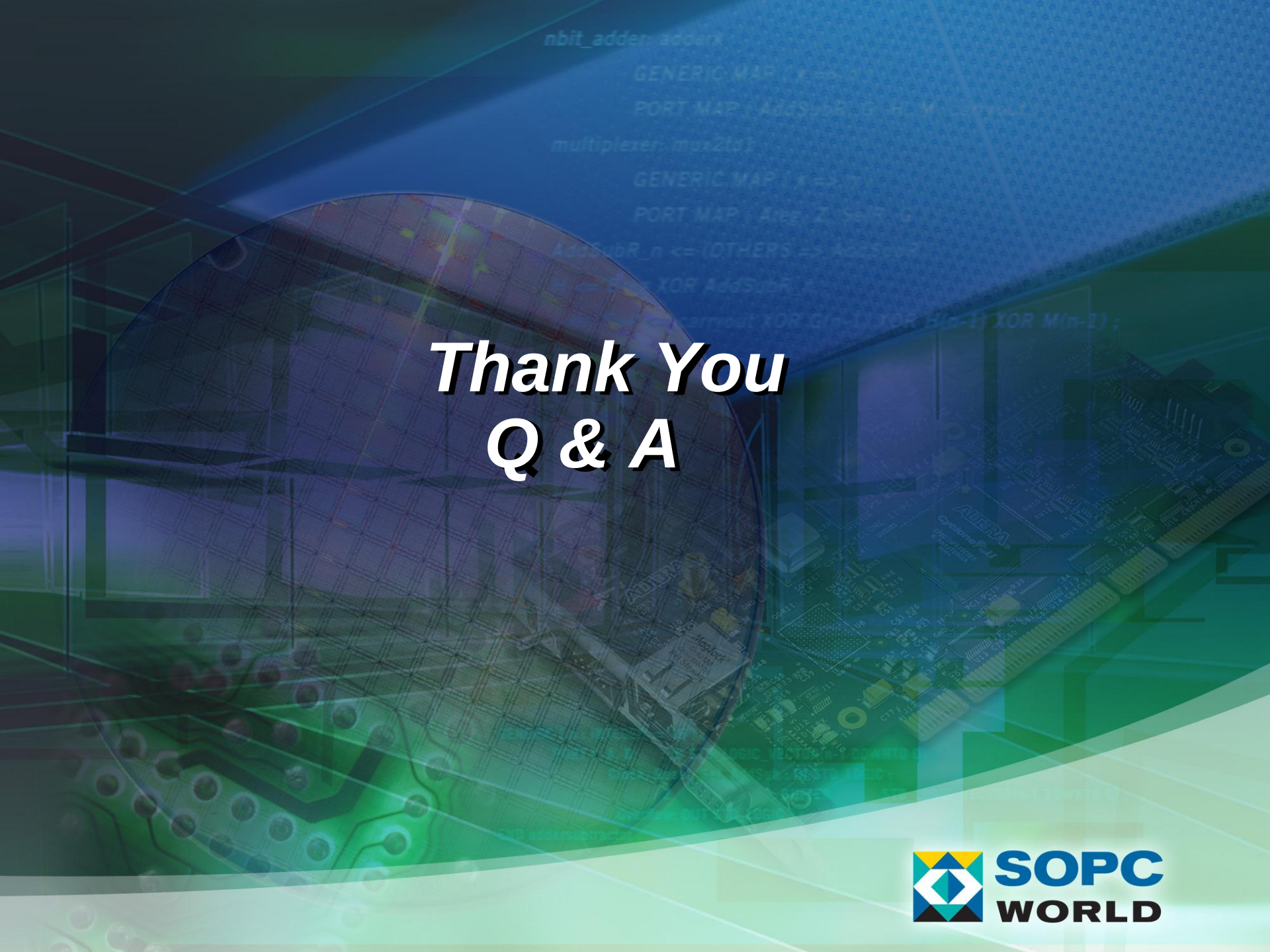
Summary

Nios II C2H Compiler Offers

- Performance
  - Boosts overall FPGA computing performance
  - No increase in clock frequency
- Productivity
  - Generates and adds hardware accelerators automatically
  - Results in substantial time savings
  - Supports existing tool chain

Altera Is Driving Innovation
With System-Level Design Tools

SOPC WORLD


Thank You
Q & A
SOPC
WORLD