

An Introduction to Vitis HLS



Agenda

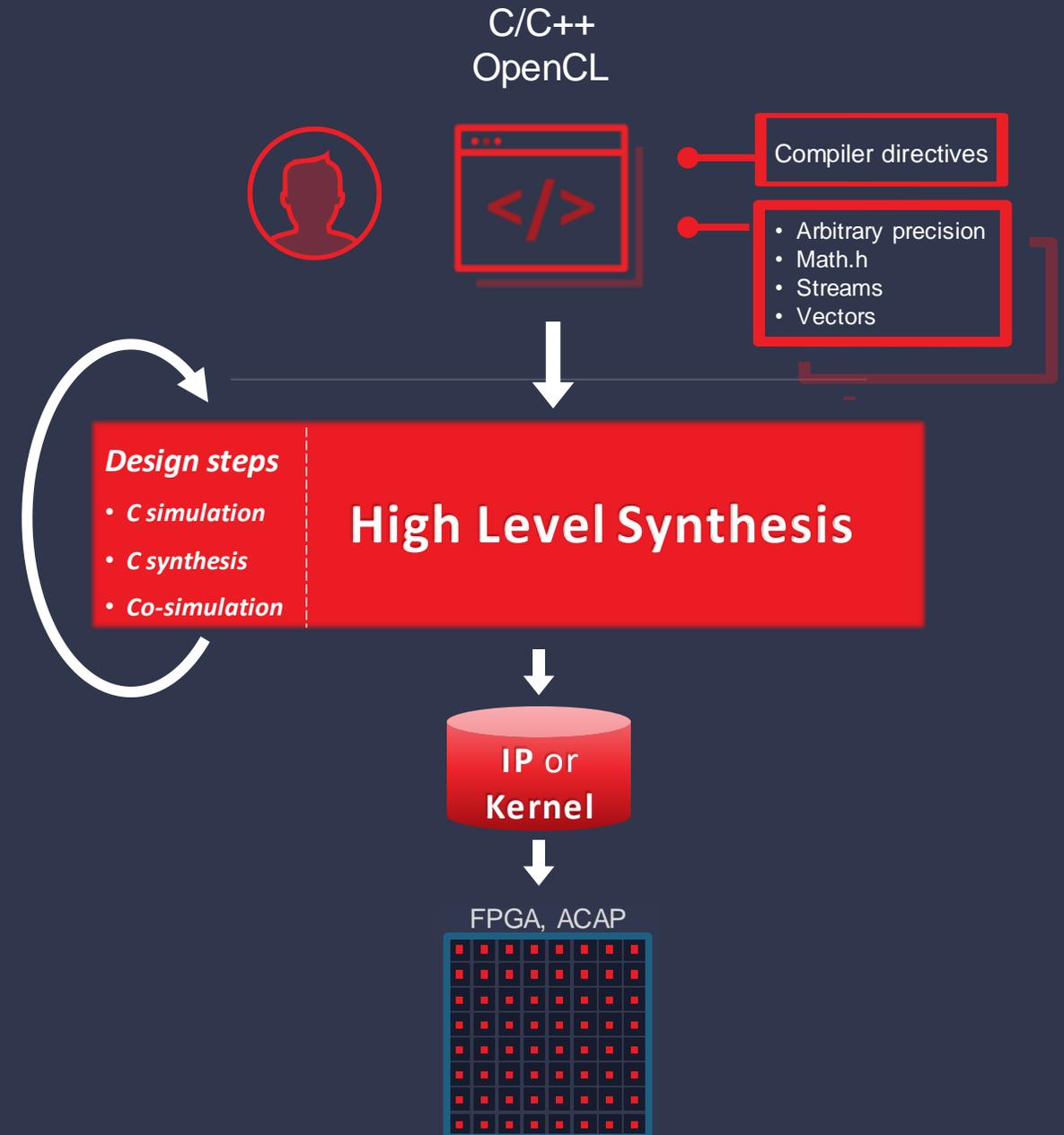
- High-Level Synthesis (HLS) Flows
- Technical Overview + Demo Examples
- Resources

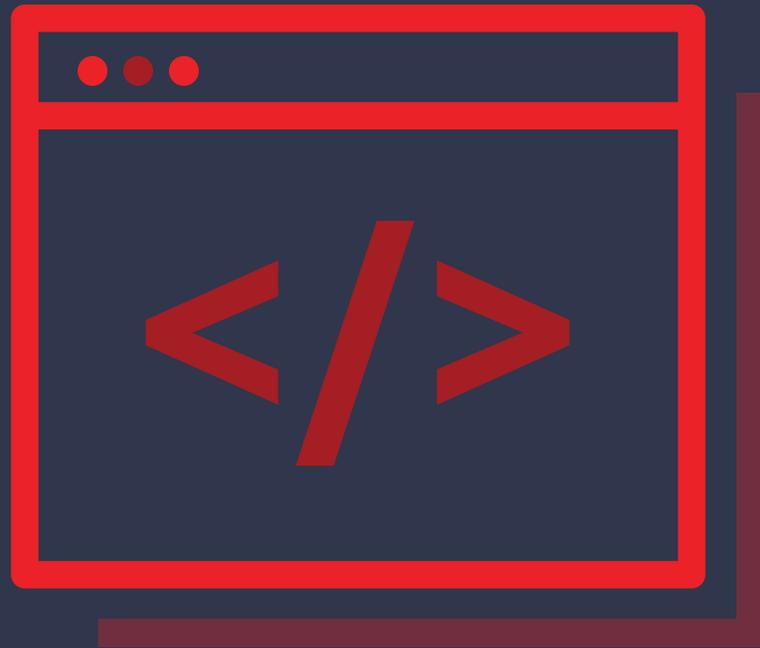
Vitis HLS

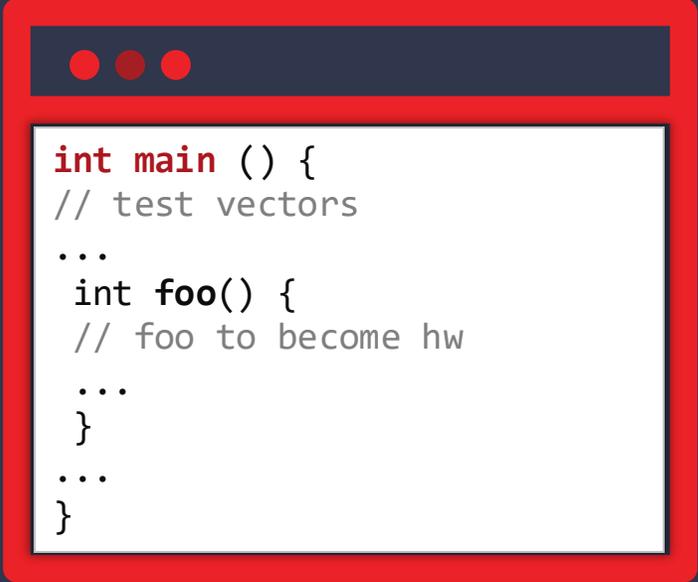
> Abstracted C-based Design Entry

> Higher Productivity !

- >> Concise code
- >> Fast C simulation
- >> Automated Simulation of Generated RTL
- >> Optimized Libraries







```
int main () {  
  // test vectors  
  ...  
  int foo() {  
    // foo to become hw  
    ...  
  }  
  ...  
}
```

Applications for High-Level Synthesis



Surveillance, AI

- Classification
- Recognition

Robotics

- Drones
- Micro-controller, AI



Aerospace and Defense

- Radar, Sonar
- Signals Intelligence

Communications

- LTE MIMO receiver
- Advanced wireless antenna positioning



Industrial, Scientific, Medical

- Ultrasound systems
- Motor controllers

Audio, Video, Broadcast

- 3D cameras
- Video transport

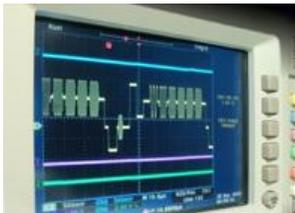


Automotive

- Infotainment
- Driver assistance / AI

Consumer

- 3D television
- eReaders



Test & Measurement

- Communications instruments
- Semiconductor ATE

Computing & Storage

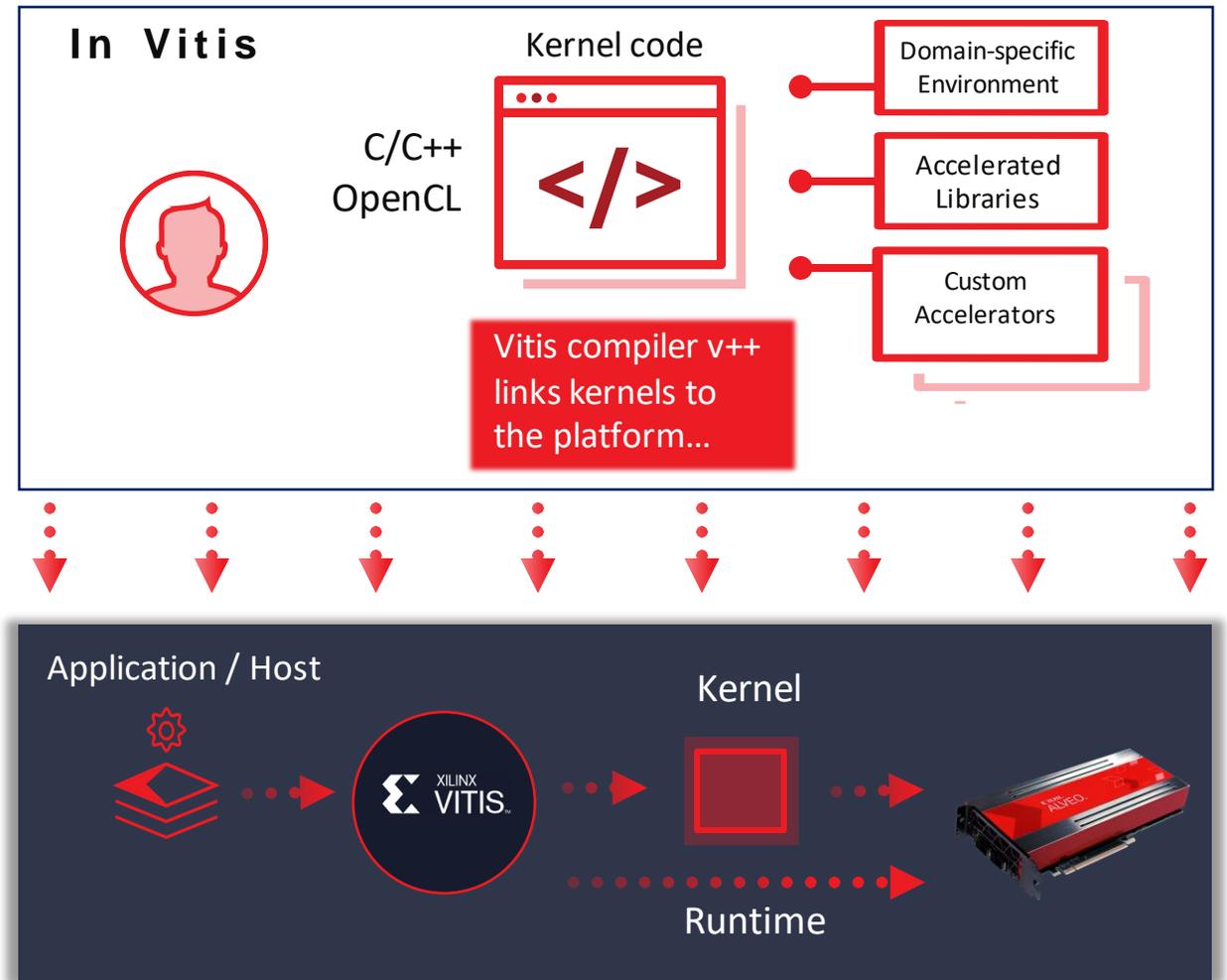
- High performance computing
- Database acceleration



HLS in Vitis Flow

HLS compiles C-based Kernels

- ✓ v++ performs all the compiles and links
 - ✓ HLS is automatically invoked
- ✓ No necessary direct interaction with HLS
- ✓ HLS reports imported in Vitis Analyzer
- ✓ Full application can be C-based



HLS in Vivado Flow

HLS exports RTL IP...

- ✓ User runs HLS directly
- ✓ Typically block assembly done in IPI
- ✓ Design entry is C/C++
- ✓ Can invoke Vivado waveform viewer

Vitis HLS

Synthesis Summary Report of digitree

General Information

Date: Tue May 26 2020 1 (08:00:00) (Mon May 10 2020 10:00:00)

Version: 2020.1 (Build 20200510)

Project: test_3-nn.prj

Solution: test_solution (Vivado IP Flow Target)

Product Family: xrtexplus

Target Device: xrtexplus-fsgd2104-2L

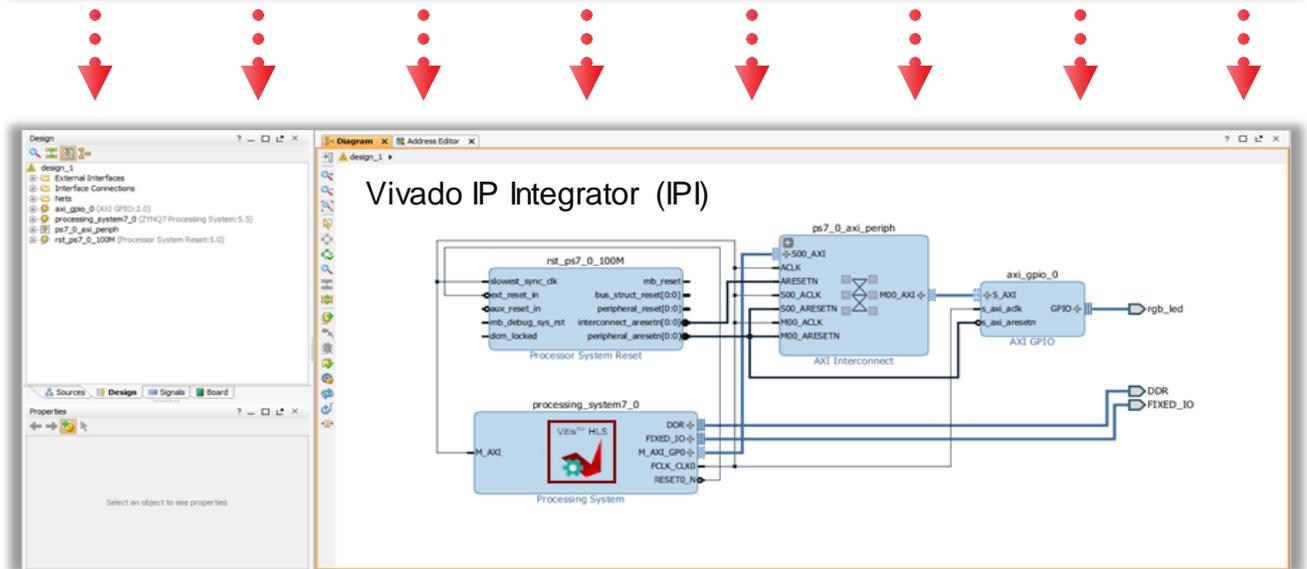
Performance: Test: Estimate

Type	Latency	Latency (absolute)	Iteration Latency	Interval	Count	Min	Max	FF	LUT	Slack
1	1	1	1	1	1	1	1	553	4573	1.41
1	1	1	1	1	1	1	1	310	517	-
10	18000	yes	-	-	-	-	-	-	-	-

Compiler Directives

Accelerated Libraries

HLS exports an IP compatible with IP Integrator



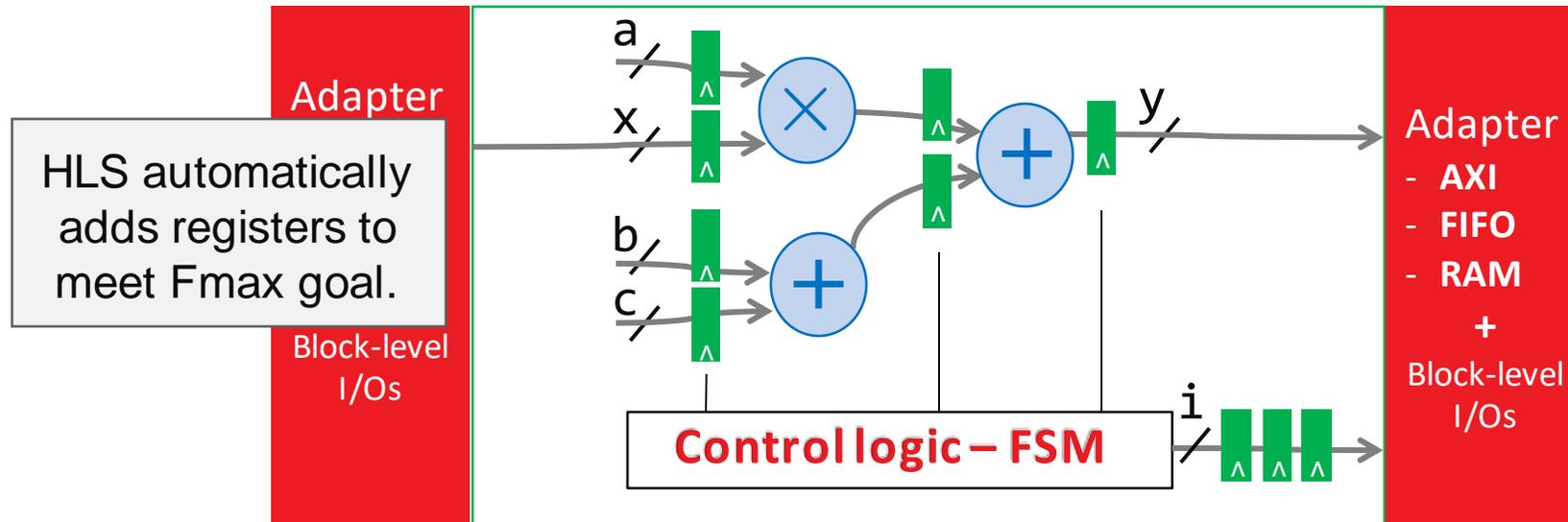
Agenda

- High-Level Synthesis (HLS) Flows
- Technical Overview + Demo Examples
- Resources

Automatic Interface and Control Logic

- > Simple C code quickly become a kernel or an IP...

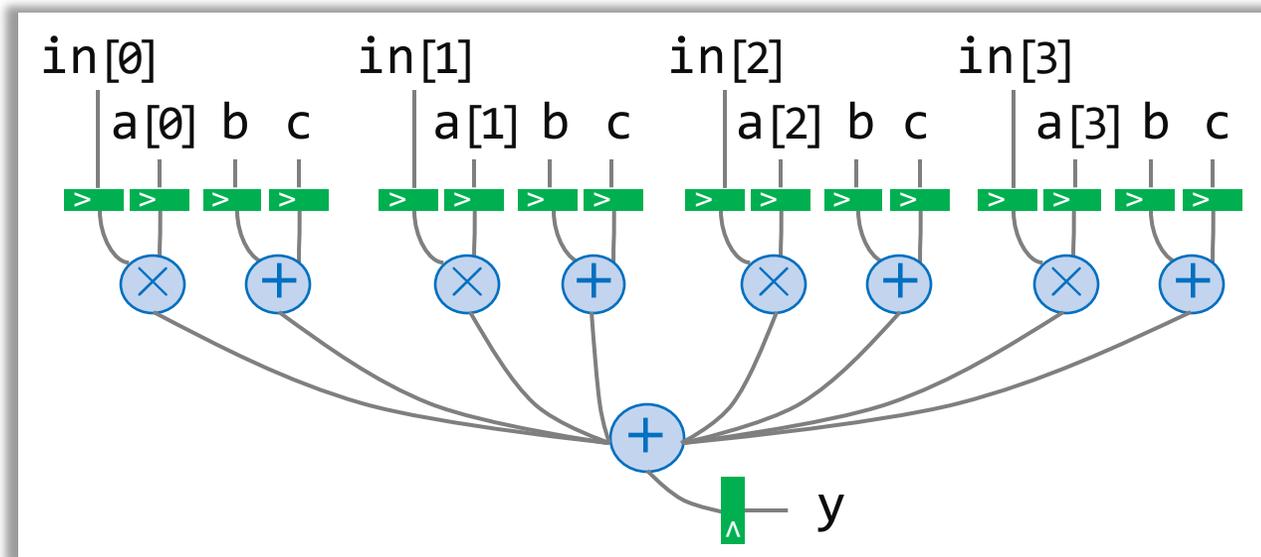
```
void f(int in[4], int out[4]) {  
    int a,b,c,x,y;  
    for(int i = 0; i < 4; i++) {  
        x = in[i]; y = a*x + b + c; out[i] = y;  
    }  
}
```



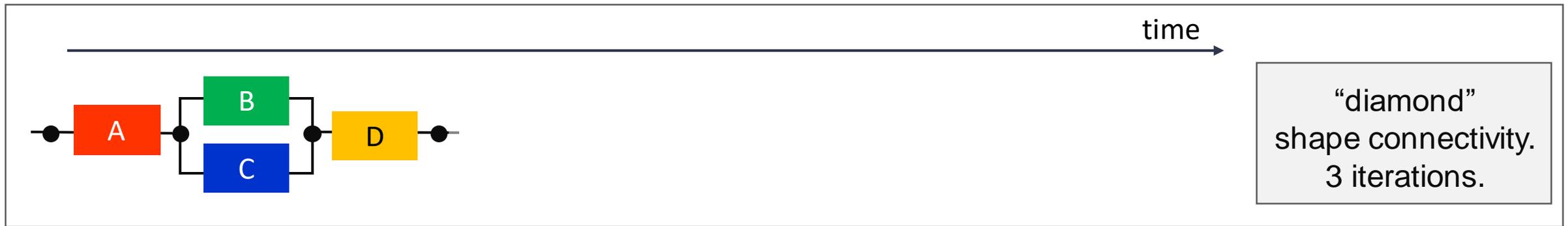
Design Space Exploration via Pragmas

> Pragmas change the circuit topology...

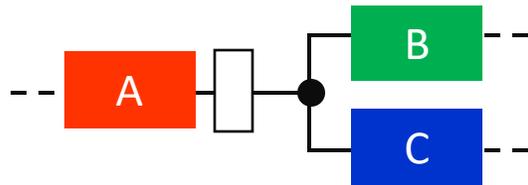
```
void f(int in[4], int &y, int a[4], int b, int c) {  
  #pragma HLS ARRAY_PARTITION variable=in dim=1 complete  
  #pragma HLS ARRAY_PARTITION variable=a dim=1 complete  
  #pragma HLS PIPELINE  
  for(int i = 0; i < 4; i++)  
    y += a[i] * in[i] + b + c;  
}
```



Task Parallelism with HLS

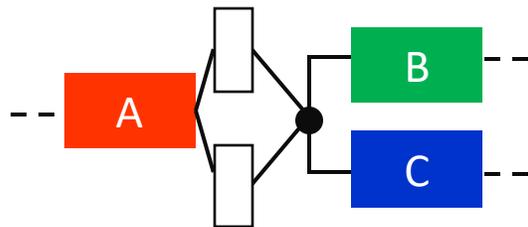


Default versus ping-pong buffers



By default...

As **A** produces its buffer, **B** and **C** wait...
Then **B** and **C** consume and **A** waits...



With **ping-pong** buffers...

A produces in one buffer, **B** and **C** read the other.
Next invocation, buffers switch roles: tasks can work continuously

The dataflow pragma in HLS automates memory expansion to enable task parallelism



Demo #1 – Task Parallelism

Vectorized Data Types...

```
// Vectorization means that the compiler detects that
// independent instructions can be executed as SIMD instructions.
// So, something like this...
for(i=0; i<N; i++){
    a[i] = a[i] + b[i];
}
// ... becomes "vectorized" as... (using vector notation)
for (i=0; i<(N-N%VF); i+=VF){
    a[i:i+VF] = a[i:i+VF] + b[i:i+VF];
}

// 1 operation that can be done on VF elements of the array
// at the same time and does this N/VF times instead of doing
// the single operation N times...
```

Vectorized Data Types in Vitis HLS

> Vitis HLS supports the C++14 `vector_size` attribute

>> Simply using C++...

```
// vector_size specifies size in bytes
typedef float float16 __attribute__((vector_size(64)));
```

Custom vector type float16 based on C++ attribute

> ... and also supports arbitrary precision types via `hls_vector.h`

>> Examples

```
#include "hls_vector.h"
using float16 = hls::vector<float, 16>;
```

Same as above using `hls::vector`

```
#include "hls_vector.h"
using quad = hls::vector<ap_int<18>, 4>;
```

Vector of four 18-bit signed variables

Vectorized Data Types – Operations

> Initialization

```
hls::vector<int, 4> x;           // uninitialized
hls::vector<int, 4> y = 10;      // scalar initialized
hls::vector<int, 4> z = {0, 1, 2, 3}; // initializer list (must have 4 elements)
```

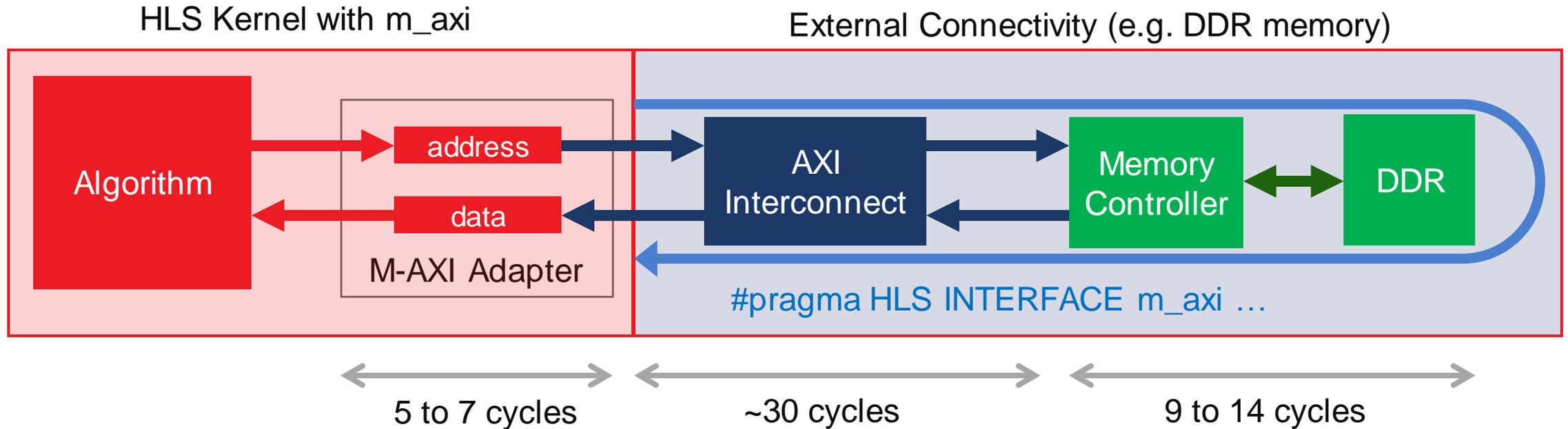
> Access

```
myvec[i] = ...; // reference to an element
... = myvec[i]; // value of an element
```

> Recommendations

- >> Use `hls::vector<T, N>` with `N` as a power of 2 for a better alignment that guarantees smaller initiation interval (II)
- >> Use the `__attribute__((no_ctor))` for better II when using dataflow

Optimizing Interfaces



> Efficient Pipeline

- >> Adapt the latency parameter of the interface for efficiency

> Loop Bursts

- >> Segmented into smaller bursts by the adapter (and that's okay!)
- >> Adapter will pipeline for you (independent state machine!)



Demo #2 – Vector types and AXI Interfaces

Agenda

- High-Level Synthesis (HLS) Flows
- Technical Overview + Demo Examples
- Resources

Resources – HLS

> HTML and PDF User Guides

> Basic examples

>> Github examples accessible from Vitis HLS

> Tutorials and complete examples

>> Github libraries: [Vitis Libraries](#)

>> Vitis examples: [Vitis Accel Examples](#)

> Forums

>> Monitored by Xilinx support staff

The screenshot shows the Xilinx website's documentation page for 'Introduction to Vitis HLS'. The page title is 'Vitis Unified Software Development Platform 2020.1 Documentation'. The navigation bar includes 'XILINX', 'Developers', 'Support', and 'Forums'. A 'User Guide' label is highlighted in a red box. The page content includes a search bar, a breadcrumb trail (Home / Vitis HLS / Using Vitis HLS / Introduction to Vitis HLS), a table of contents for 'Using Vitis HLS', and the main article text. The article text describes the Vitis HLS tool's purpose and its role in the application acceleration flow. A sidebar on the right lists 'On this page' topics such as 'Basics of High-Level Synthesis', 'Scheduling and Binding Example', and 'Extracting Control Logic and Implementing I/O Ports Example'.

The screenshot shows the Xilinx Forum page. The navigation bar includes 'Solutions', 'Products', 'Support', and 'XILINX'. A 'Forum' label is highlighted in a red box. The page features a 'Discussions' section with a 'Post a Question' button. The discussions list includes 'AXI Tutorials', 'HLS Key Documents and Getting Started FAQ', 'Vivado HLS csin error', 'Basic I/O Concepts', 'AXI4-Stream Video Port stuck with TREADY low', and 'ROHC on Vivado HLS'. The right sidebar shows 'Acceleration' topics like 'Vitis Acceleration, SDAccel, SDSoc' and 'High-Level Synthesis (HLS)'. Below that is a 'Top Kudoed Posts' section with a table of subjects and kudos.

SUBJECT	KUDOS
Re: Missing constructor for streams with non-zero ...	2
Re: LUT as Distributed RAM over-utilized in Top Le...	2
Re: Create IP AXI4-Lite	2
vitis_hls breaks stream depth pragma	2

Summary

- Vitis HLS used both in Vitis and Vivado
- C based entry boosts productivity
- Get started with examples and tutorials