Traineeships in Advanced Computing for High Energy Physics (TAC-HEP)

FPGA module training

Week-7

Lecture-13: 11/03/2025

Varun Sharma

University of Wisconsin – Madison, USA

Content

- Vivado/Vitis HLS Setup
  - HLS Pragmas: Interface

TAC-HEP 2025

#pragma HLS Interface

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

The INTERFACE pragma

- Supported for use on the top-level function,
- Can't be used for sub-functions
- Specifies how RTL ports are created from the function arguments during interface synthesis

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Interfaces

HLS supports memory, stream, and register interface paradigms where each paradigm follows a certain interface protocol and uses the adapter to communicate with the external world

- **Memory Paradigm (m_axi):** the data is accessed by the kernel through memory such as DDR, HBM, PLRAM/BRAM/URAM
- **Stream Paradigm (axis):** the data is streamed into the kernel from another streaming source, such as video processor or another kernel, and can also be streamed out of the kernel
- **Register Paradigm (s_axilite):** The data is accessed by the kernel through register interfaces and accessed by software as register reads/writes

Default Interfaces

C-Argument Type	Supported Paradigms	Default Paradigm	Default Interface Protocol		
			Input	Output	Inout
Scalar variable (pass by value)	Register	Register	ap_none	N/A	N/A
Array	Memory, Stream	Memory	ap_memory	ap_memory	ap_memory
Pointer	Memory, Stream, Register	Register	ap_none	ap_vld	ap_ovld
Reference	Register	Register	ap_none	ap_vld	ap_vld
`hls::stream`	Stream	Stream	ap_fifo	ap_fifo	N/A

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

mode = <mode>

- Can be broken into three categories
  1. Port-level Protocols
  2. AXI Interface Protocols
  3. Block-Level Control Ports

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

<mode>: Port-Level Protocols

- ***ap_none:*** No protocol. The interface is a data port
- ***ap_vld:*** Implements the data port with an associated valid port to indicate when the data is valid for reading or writing

- ***ap_ack:*** Implements the data port with an associated acknowledge port to acknowledge that the data was read or written
- ***ap_hs:*** Implements the data port with associated valid and acknowledge ports to provide a two-way handshake to indicate when the data is valid for reading and writing and to acknowledge that the data was read or written

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

<mode>: Port-Level Protocols

- ***ap_stable*:** No protocol. The interface is a data port. The HLS tool assumes the data port is always stable after reset, which allows internal optimizations to remove unnecessary registers
- ***ap_fifo*:** Implements the port with a standard FIFO interface using data I/O ports with associated active-Low FIFO empty and full ports

- *ap_bus*: Implements pointer and pass-by-reference ports as a bus interface.
- *ap_memory*: Implements array arguments as a standard RAM interface
- ***ap_ovld*:** Implements the output data port with an associated valid port to indicate when the data is valid for reading or writing

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

<mode>: AXI-Interface Protocols

- ***s_axilite***: Implements all ports as an AXI4-Lite interface. The tool produces an associated set of C driver files when exporting the generated RT for the HLS component
- ***m_axi*:** Implements all ports as an AXI4 interface
- ***m_axi_addr64*** command to specify either 32-bit (default) or 64-bit address ports and to control any address offset.
- ***axis***: Implements all ports as an AXI4-Stream interface

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

<mode>: Block-level Control Protocols

- ***ap_ctrl_chain*:** Implements a set of block-level control ports to **start** the design operation, **continue** operation & indicate when the design is **idle**, **done**, & **ready** for new input data
- ***ap_ctrl_none***: No block-level I/O protocol
- ***ap_ctrl_hs***: Implements a set of block-level control ports to start the design operation and to indicate when the design is idle, done, and ready for new input data

https://docs.amd.com/r/en-US/ug1399-vitis-hls/pragma-HLS-interface

Pragma HLS interface: Syntax

#pragma HLS interface mode=<mode> port=<name> direct_io=<value>**[OPTIONS]**

port=<name>: Specifies the name of the function argument, function return, or global variable which the INTERFACE pragma applies to

[OPTIONS]
bundle=<string>: Groups function arguments into AXI interface ports

register: An optional keyword to register the signal and any relevant protocol signals, and causes the signals to persist until at least the last cycle of the function execution.

- Ap_none, ap_ack, ap_vld, ap_ovld, ap_hs, ap_stable, axis, s_axilite

TAC-HEP 2025

Some examples

Example – 1

```
void example(int a, int b, int *c){

   #pragma HLS INTERFACE s_axilite port=a
   #pragma HLS INTERFACE s_axilite port=b
   #pragma HLS INTERFACE s_axilite port=c
   #pragma HLS INTERFACE s_axilite port=return

   *c = a + b;
}
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|       39|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |        0|      -|      150|      232|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|        -|    -|
|Register             |        -|      -|        -|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |        0|      0|      150|      271|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        0|      0|    ~0   |    ~0   |    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        0|      0|    ~0   |    ~0   |    0|
+---------------------+---------+-------+---------+---------+-----+
```

Example – 2

```
void example(int *input, int *output, int size) {
    #pragma HLS INTERFACE m_axi port=input depth=1024
    #pragma HLS INTERFACE m_axi port=output depth=1024
    #pragma HLS INTERFACE s_axilite port=size
    #pragma HLS INTERFACE s_axilite port=return


    for (int i = 0; i < size; i++) {
        output[i] = input[i] * 2;
    }
}
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+------+---------+---------+-----+
|DSP                  |        -|     -|        -|        -|    -|
|Expression           |        -|     -|        0|       60|    -|
|FIFO                 |        -|     -|        -|        -|    -|
|Instance             |        4|     -|     1098|     1264|    -|
|Memory               |        -|     -|        -|        -|    -|
|Multiplexer          |        -|     -|        -|      116|    -|
|Register             |        -|     -|      140|        -|    -|
+---------------------+---------+------+---------+---------+-----+
|Total                |        4|     0|     1238|     1440|    0|
+---------------------+---------+------+---------+---------+-----+
|Available SLR        |     1440|  2280|   788160|   394080|  320|
+---------------------+---------+------+---------+---------+-----+
|Utilization SLR (%)  |    ~0   |     0|    ~0   |    ~0   |    0|
+---------------------+---------+------+---------+---------+-----+
|Available            |     4320|  6840|  2364480|  1182240|  960|
+---------------------+---------+------+---------+---------+-----+
|Utilization (%)      |    ~0   |     0|    ~0   |    ~0   |    0|
+---------------------+---------+------+---------+---------+-----+
```

Example – 3

```
#include <hls_stream.h>

void example(hls::stream<int> &input, hls::stream<int> &output)
{
  #pragma HLS INTERFACE axis port=input
  #pragma HLS INTERFACE axis port=output
  #pragma HLS INTERFACE ap_ctrl_none port=return

  int data;
  if (input.read_nb(data)) { // Non-blocking read
     output.write(data * 2);
  }
}
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|        6|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |        -|      -|        -|        -|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|       24|    -|
|Register             |        -|      -|        3|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |        0|      0|        3|       30|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        0|      0|       ~0|       ~0|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        0|      0|       ~0|       ~0|    0|
+---------------------+---------+-------+---------+---------+-----+
```

Example – 4

```
void example(int input[256], int output[256]) {

   #pragma HLS INTERFACE bram port=input
   #pragma HLS INTERFACE bram port=output
   #pragma HLS INTERFACE s_axilite port=return

   for (int i = 0; i < 256; i++) {
      output[i] = input[i] * 2;
   }
}
```

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+------+---------+---------+-----+
|DSP                  |        -|     -|        -|        -|    -|
|Expression           |        -|     -|        0|       29|    -|
|FIFO                 |        -|     -|        -|        -|    -|
|Instance             |        0|     -|       36|       40|    -|
|Memory               |        -|     -|        -|        -|    -|
|Multiplexer          |        -|     -|        -|       39|    -|
|Register             |        -|     -|       30|        -|    -|
+---------------------+---------+------+---------+---------+-----+
|Total                |        0|     0|       66|      108|    0|
+---------------------+---------+------+---------+---------+-----+
|Available SLR        |     1440|  2280|   788160|   394080|  320|
+---------------------+---------+------+---------+---------+-----+
|Utilization SLR (%)  |        0|     0|       ~0|       ~0|    0|
+---------------------+---------+------+---------+---------+-----+
|Available            |     4320|  6840|  2364480|  1182240|  960|
+---------------------+---------+------+---------+---------+-----+
|Utilization (%)      |        0|     0|       ~0|       ~0|    0|
+---------------------+---------+------+---------+---------+-----+
```

Other examples

```
#pragma HLS interface ap_ctrl_none port=return
```

Turns off block-level I/O protocols, and is assigned to the function return value

```
#pragma HLS interface ap_vld register port=InData
```

The function argument *InData* is specified to use the *ap_vld* interface, and also indicates the input should be registered

```
#pragma HLS interface ap_memory port=lookup_table
```

This exposes the global variable *lookup_table* as a port on the RTL design, with an *ap_memory* interface

Reminder: Assignments

- Assignment-1 (13-02-2025)
- Assignment-2 (18-02-2025)
- Assignment-3 (27-02-2025)
- Assignment-4 (06-03-2025)

Uploaded to cernbox: https://cernbox.cern.ch/s/gmUqRDHTxDLqx4M

Send via email: **varun.sharma@cern.ch**

Submit in 2 weeks from date of assignment

TAC-HEP 2025

Questions?

Acknowledgements:
- https://docs.amd.com/r/en-US/ug1399-vitis-hls/HLS-Pragmas
- ug871-vivado-high-level-synthesis-tutorial.pdf

List of Available Pragmas

Type	Attributes
Kernel Optimization	• pragma HLS aggregate • pragma HLS alias • pragma HLS disaggregate • pragma HLS expression_balance • pragma HLS latency • pragma HLS performance • pragma HLS protocol • pragma HLS reset • pragma HLS top • pragma HLS stable
Function Inlining	• pragma HLS inline
Interface Synthesis	• pragma HLS interface • pragma HLS stream
Task-level Pipeline	• pragma HLS dataflow • pragma HLS stream
Pipeline	• pragma HLS pipeline • pragma HLS occurrence
Loop Unrolling	• pragma HLS unroll • pragma HLS dependence
Loop Optimization	• pragma HLS loop_flatten • pragma HLS loop_merge • pragma HLS loop_tripcount
Array Optimization	• pragma HLS array_partition • pragma HLS array_reshape
Structure Packing	• pragma HLS aggregate • pragma HLS dataflow
Resource Utilization	• pragma HLS allocation • pragma HLS bind_op • pragma HLS bind_storage • pragma HLS function_instantiate

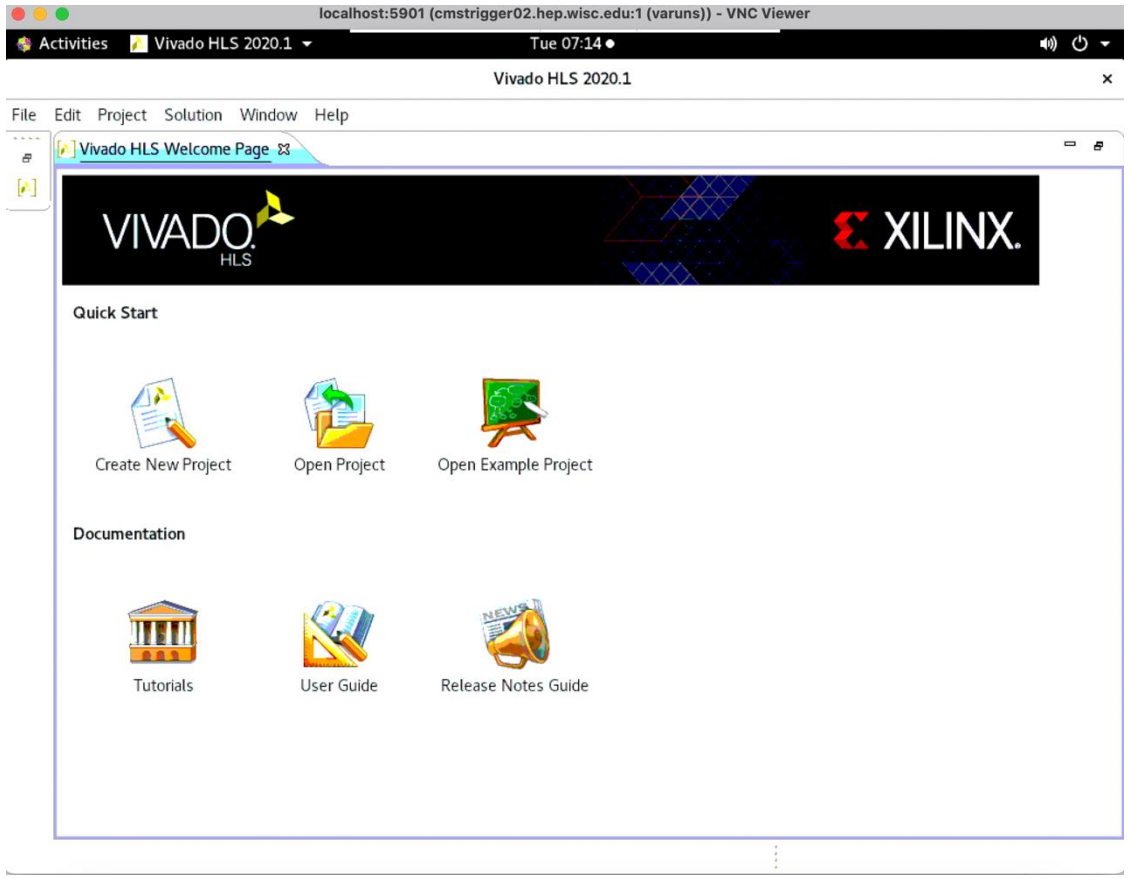
Reminder: HLS Setup

- ssh <username>@cmstrigger02-via-login -L5901:localhost:5901
  - Or whatever **:1** display number
  - Sometimes you may need to run vncserver -localhost -geometry 1024x768 again to start new vnc server

- **Connect to VNC server (remote desktop) client**
- **Open terminal**
  - source /opt/Xilinx/Vivado/2020.1/settings64.sh
  - cd /scratch/`whoami`
  - vivado_hls

OR

  - Source /opt/Xilinx/Vitis/2020.1/settings64.sh
  - Cd /scratch/`whoami`
  - vitis_hls



Jargons

- **ICs - Integrated chip:** assembly of hundreds of millions of transistors on a minor chip
- **PCB:** Printed Circuit Board
- **LUT - Look Up Table aka 'logic'** - generic functions on small bitwidth inputs. Combine many to build the algorithm
- **FF - Flip Flops** - control the flow of data with the clock pulse. Used to build the pipeline and achieve high throughput
- **DSP - Digital Signal Processor** - performs multiplication and other arithmetic in the FPGA
- **BRAM - Block RAM** - hardened RAM resource. More efficient memories than using LUTs for more than a few elements
- **PCIe or PCI-E - Peripheral Component Interconnect Express**: is a serial expansion bus standard for connecting a computer to one or more peripheral devices
- **InfiniBand** is a computer networking communications standard used in high-performance computing that features very high throughput and very low latency
- **HLS** - High Level Synthesis - compiler for C, C++, SystemC into FPGA IP cores
- **HDL** - Hardware Description Language - low level language for describing circuits
- **RTL** - Register Transfer Level - the very low level description of the function and connection of logic gates
- **FIFO** – First In First Out memory
- **Latency** - time between starting processing and receiving the result
  - Measured in clock cycles or seconds
- **II - Initiation Interval** - time from accepting first input to accepting next input