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

FPGA module training

Week-8

Lecture-15: 20/03/2025

Varun Sharma

University of Wisconsin – Madison, USA

Content

So far

- HLS Pragmas:
  - Interface
  - Array Partition

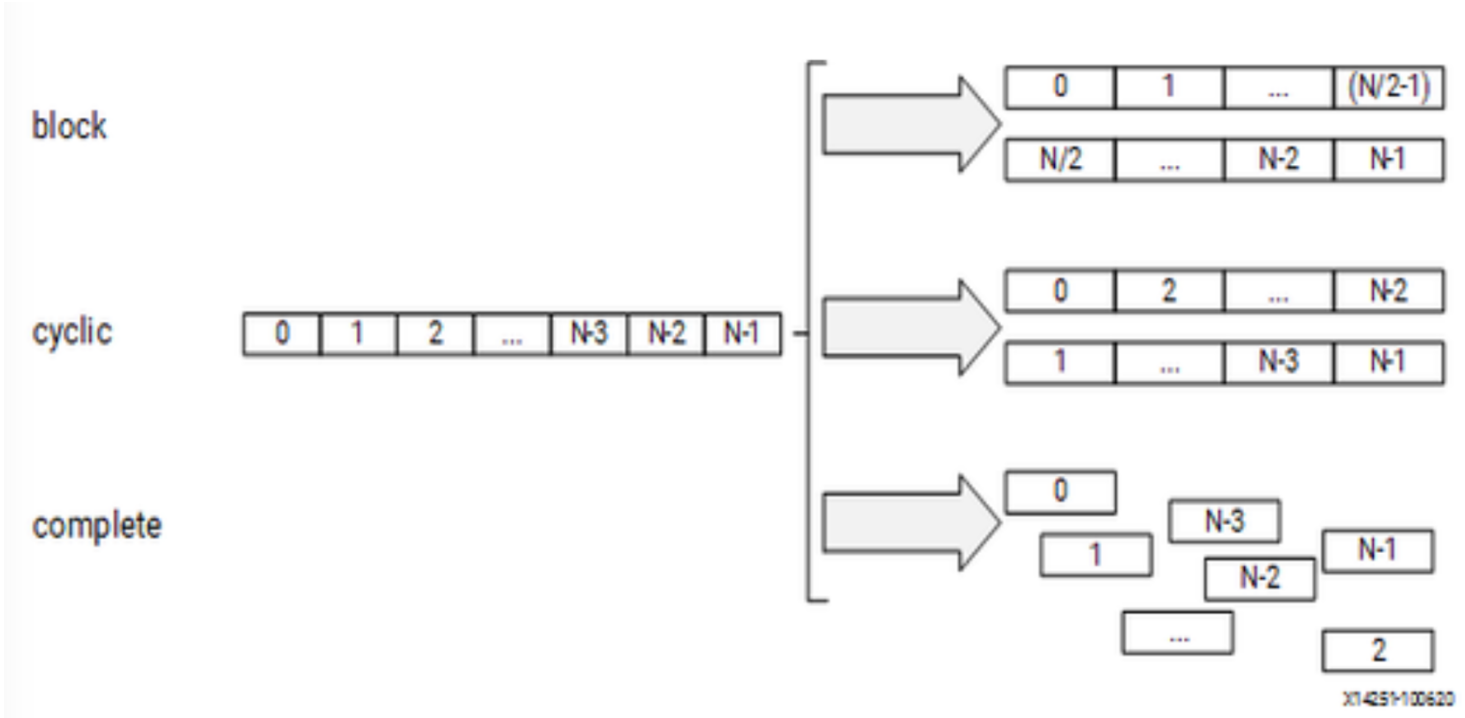
Today

- HLS Pragmas:
  - Array Reshape
  - Pipeline

Pragma HLS array_partition

#pragma HLS array_partition variable=<name> <type> factor=<int> dim=<int>

Three ***types*** of array partitioning available:



A Comparison

	Default	Restructured	Pragma HLS Partition		
			Complete	Block factor = 4	Cyclic factor = 4
Estimated Clock (ns)	3.390	2.534	4.339	3.142	3.998
Latency (cycle)	187	126	63	187	187
Latency (μs)	4.675	3.150	1.575	4.675	4.675
Resources (FF)	35	50	19	42	37
Resources (LUT)	164	152	920	296	284
Resource (Others)	0	0	0	0	0

TAC-HEP 2025

#pragma HLS array_rehape

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

Pragma HLS array_reshape

#pragma HLS array_reshape variable=<name> <type> factor=<int> dim=<int>

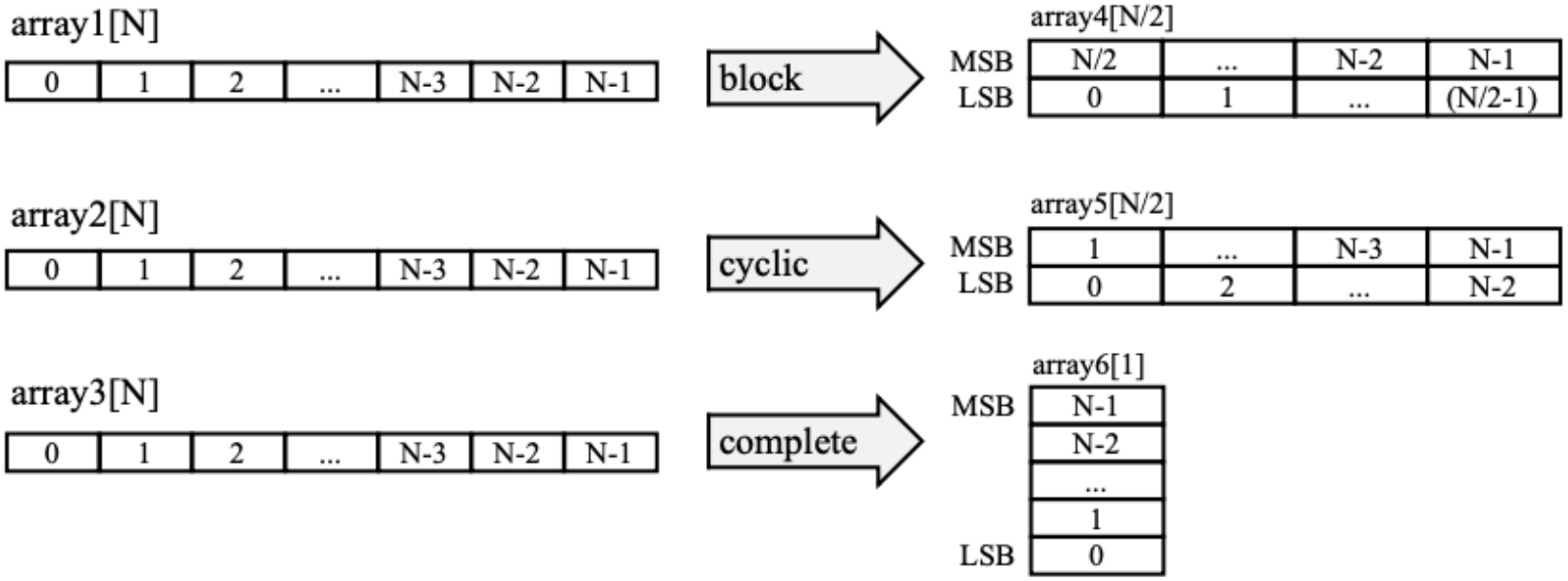
- ARRAY_RESHAPE pragma reforms the array with vertical remapping and concatenating elements of arrays by increasing bit-widths
- Reduces the number of block RAM consumed while providing parallel access to the data
- Pragma creates a new array with fewer elements but with greater bit-width, allowing more data to be accessed in a single clock cycle

Pragma HLS array_reshape

#pragma HLS array_reshape variable=<name> <type> factor=<int> dim=<int>

Cyclic: Cyclic partitioning creates smaller arrays by interleaving elements from the original array
Block: Block partitioning creates smaller arrays from consecutive N-blocks of the original array
Complete: Complete partitioning decomposes the array into individual elements

- For a 1-D array, this corresponds to resolving a memory into individual registers



Pragma HLS array_reshape: Complete

#pragma HLS array_reshape variable=<name> <type> factor=<int> dim=<int>

```
#include "example.h"

dout_t example(din_t mem[N]) {

#pragma HLS ARRAY_RESHAPE variable=mem complete

   dout_t total = 0;

   for (int i = 2; i < N; ++i)
      total += mem[i] + mem[i - 1] + mem[i - 2];

   return total;
}
```

```
Timing:
  * Summary:
  +--------+----------+----------+------------+
  |  Clock |  Target  | Estimated| Uncertainty|
  +--------+----------+----------+------------+
  |ap_clk  | 25.00 ns | 5.988 ns |   3.12 ns  |
  +--------+----------+----------+------------+

Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+---------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
  |   min   |   max   |    min   |    max   | min | max |   Type  |
  +---------+---------+----------+----------+-----+-----+---------+
  |       63|       63| 1.575 us | 1.575 us |   63|   63|   none  |
  +---------+---------+----------+----------+-----+-----+---------+
```

```
* Loop:
+----------+---------+---------+----------+-----------+-----------+------+----------+
|          |  Latency (cycles) | Iteration|  Initiation Interval  | Trip |          |
| Loop Name|   min   |   max   |  Latency |  achieved |   target  | Count| Pipelined|
+----------+---------+---------+----------+-----------+-----------+------+----------+
|- Loop 1  |       62|       62|         1|          -|          -|    62|    no    |
+----------+---------+---------+----------+-----------+-----------+------+----------+
```

```
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|     5897|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |        -|      -|        -|        -|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|       33|    -|
|Register             |        -|      -|       19|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |        0|      0|       19|     5930|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        0|      0|    ~0   |        1|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        0|      0|    ~0   |    ~0   |    0|
+---------------------+---------+-------+---------+---------+-----+
```

TAC-HEP 2025

array_partition vs array_reshape

PARTITION vs RESHAPE

	Pragma HLS PARTITION			Pragma HLS RESHAPE		
	Complete	Block factor = 4	Cyclic factor = 4	Complete	Block factor = 4	Cyclic factor = 4
Estimated Clock (ns)	4.339	3.142	3.998	5.988	2.969	3.117
Latency (cycle)	63	187	187	63	187	187
Latency (μs)	1.575	4.675	4.675	1.575	4.675	4.675
Resources (FF)	19	42	37	19	46	42
Resources (LUT)	920	296	284	5930	488	476
Resource (Others)	0	0	0	0	0	0

PARTITION vs RESHAPE: Expression

PARTITION

* Expression:

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
i_fu_1056_p2	+	0	0	15	7	1
ret_V_1_fu_1040_p2	+	0	0	16	9	9
ret_V_fu_1026_p2	+	0	0	15	8	8
total_V_fu_1050_p2	+	0	0	17	10	10
icmp_ln9_fu_938_p2	icmp	0	0	11	7	8
ap_condition_486	or	0	0	2	1	1
Total		0	0	76	42	37

* Multiplexer:

Name	LUT	Input Size	Bits	Total Bits
agg_result_V_0_reg_656	9	2	10	20
ap_NS_fsm	15	3	1	3
ap_phi_mux_phi_ln215_2_phi_fu_811_p124	269	63	7	441
ap_phi_mux_phi_ln215_phi_fu_681_p124	269	63	7	441
i_0_reg_667	9	2	7	14
Total	571	133	32	919

RESHAPE

* Expression:

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln215_1_fu_154_p2	+	0	0	15	3	6
add_ln215_fu_103_p2	+	0	0	15	2	6
i_fu_211_p2	+	0	0	15	1	7
ret_V_1_fu_195_p2	+	0	0	16	9	9
ret_V_fu_144_p2	+	0	0	15	8	8
total_V_fu_205_p2	+	0	0	17	10	10
sub_ln215_1_fu_121_p2	-	0	0	16	9	9
sub_ln215_2_fu_172_p2	-	0	0	16	9	9
sub_ln215_fu_80_p2	-	0	0	16	9	9
icmp_ln9_fu_58_p2	icmp	0	0	11	7	8
lshr_ln215_1_fu_131_p2	lshr	0	0	1915	448	448
lshr_ln215_2_fu_182_p2	lshr	0	0	1915	448	448
lshr_ln215_fu_90_p2	lshr	0	0	1915	448	448
Total		0	0	5897	1411	1425

* Multiplexer:

Name	LUT	Input Size	Bits	Total Bits
agg_result_V_0_reg_36	9	2	10	20
ap_NS_fsm	15	3	1	3
i_0_reg_47	9	2	7	14
Total	33	7	18	37

PARTITION vs RESHAPE: Interface

PARTITION

* Summary:

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	example	return value
ap_rst	in	1	ap_ctrl_hs	example	return value
ap_start	in	1	ap_ctrl_hs	example	return value
ap_done	out	1	ap_ctrl_hs	example	return value
ap_idle	out	1	ap_ctrl_hs	example	return value
ap_ready	out	1	ap_ctrl_hs	example	return value
ap_return	out	10	ap_ctrl_hs	example	return value
mem_0_V	in	7	ap_none	mem_0_V	pointer
mem_1_V	in	7	ap_none	mem_1_V	pointer
mem_2_V	in	7	ap_none	mem_2_V	pointer
mem_3_V	in	7	ap_none	mem_3_V	pointer
mem_4_V	in	7	ap_none	mem_4_V	pointer
mem_5_V	in	7	ap_none	mem_5_V	pointer
mem_6_V	in	7	ap_none	mem_6_V	pointer
mem_7_V	in	7	ap_none	mem_7_V	pointer
mem_8_V	in	7	ap_none	mem_8_V	pointer
mem_9_V	in	7	ap_none	mem_9_V	pointer
mem_10_V	in	7	ap_none	mem_10_V	pointer
mem_11_V	in	7	ap_none	mem_11_V	pointer
mem_12_V	in	7	ap_none	mem_12_V	pointer
mem_13_V	in	7	ap_none	mem_13_V	pointer
mem_14_V	in	7	ap_none	mem_14_V	pointer
mem_15_V	in	7	ap_none	mem_15_V	pointer
mem_16_V	in	7	ap_none	mem_16_V	pointer
mem_17_V	in	7	ap_none	mem_17_V	pointer
mem_18_V	in	7	ap_none	mem_18_V	pointer
mem_19_V	in	7	ap_none	mem_19_V	pointer
mem_20_V	in	7	ap_none	mem_20_V	pointer
mem_21_V	in	7	ap_none	mem_21_V	pointer
mem_22_V	in	7	ap_none	mem_22_V	pointer
mem_23_V	in	7	ap_none	mem_23_V	pointer
mem_24_V	in	7	ap_none	mem_24_V	pointer
mem_25_V	in	7	ap_none	mem_25_V	pointer
mem_26_V	in	7	ap_none	mem_26_V	pointer
mem_27_V	in	7	ap_none	mem_27_V	pointer
mem_28_V	in	7	ap_none	mem_28_V	pointer
mem_29_V	in	7	ap_none	mem_29_V	pointer
mem_30_V	in	7	ap_none	mem_30_V	pointer
mem_31_V	in	7	ap_none	mem_31_V	pointer
mem_32_V	in	7	ap_none	mem_32_V	pointer

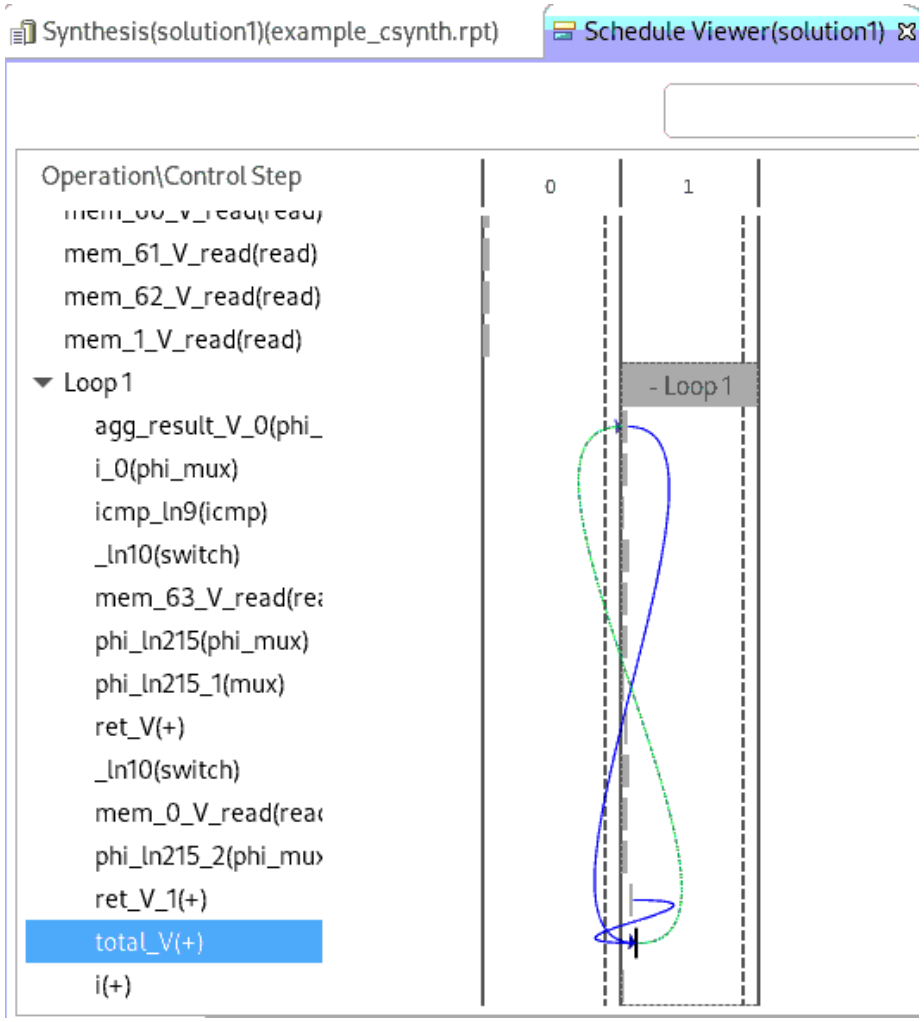
RESHAPE

* Summary:

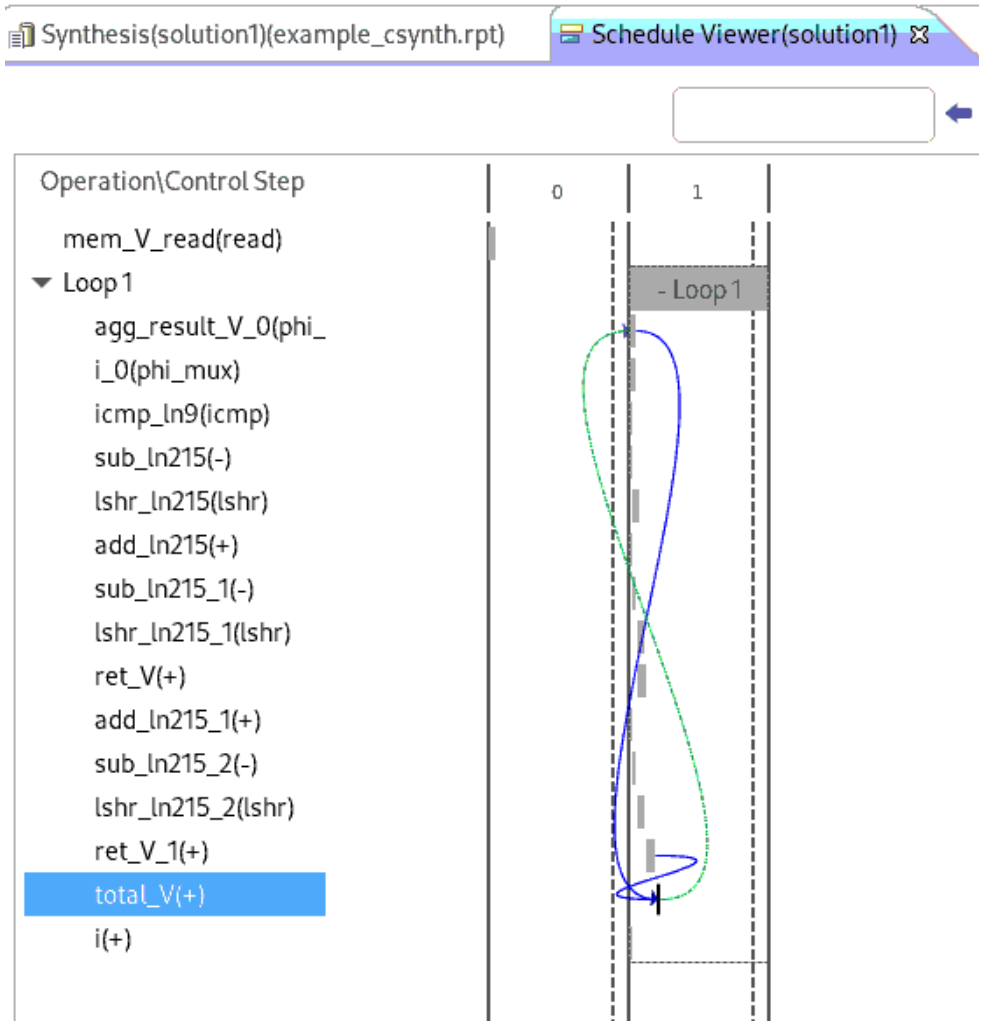
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	example	return value
ap_rst	in	1	ap_ctrl_hs	example	return value
ap_start	in	1	ap_ctrl_hs	example	return value
ap_done	out	1	ap_ctrl_hs	example	return value
ap_idle	out	1	ap_ctrl_hs	example	return value
ap_ready	out	1	ap_ctrl_hs	example	return value
ap_return	out	10	ap_ctrl_hs	example	return value
mem_V	in	448	ap_none	mem_V	pointer

PARTITION vs RESHAPE

PARTITION



RESHAPE



Pragma HLS array_reshape: Block

```
#include "example.h"

dout_t example(din_t mem[N]) {

#pragma HLS ARRAY_RESHAPE variable=mem block factor=4

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

Timing:

* Summary:

Clock	Target	Estimated	Uncertainty
ap_clk	25.00 ns	2.969 ns	3.12 ns

Latency:

* Summary:

Latency (cycles) min	Latency (cycles) max	Latency (absolute) min	Latency (absolute) max	Interval min	Interval max	Pipeline Type
187	187	4.675 us	4.675 us	187	187	none

* Loop:

Loop Name	Latency (cycles) min	Latency (cycles) max	Iteration Latency	Initiation Interval achieved	Initiation Interval target	Trip Count	Pipelined
- Loop 1	186	186	3	-	-	62	no

* Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	428	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	60	-
Register	-	-	46	-	-
Total	0	0	46	488	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

Pragma HLS array_reshape: Cyclic

```
#include "example.h"

dout_t example(din_t mem[N]) {

#pragma HLS ARRAY_RESHAPE variable=mem cyclic factor=4

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

```
Timing:
  * Summary:
  +--------+----------+----------+------------+
  |  Clock |  Target  | Estimated| Uncertainty|
  +--------+----------+----------+------------+
  |ap_clk  | 25.00 ns | 3.117 ns |   3.12 ns  |
  +--------+----------+----------+------------+

Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+---------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
  |   min   |   max   |    min   |    max   | min | max |   Type  |
  +---------+---------+----------+----------+-----+-----+---------+
  |      187|      187| 4.675 us | 4.675 us |  187|  187|   none  |
  +---------+---------+----------+----------+-----+-----+---------+
```

```
* Loop:
+----------+---------+---------+----------+-----------+-----------+------+----------+
|          |  Latency (cycles) | Iteration|  Initiation Interval  | Trip |          |
| Loop Name|   min   |   max   |  Latency | achieved |   target  | Count| Pipelined|
+----------+---------+---------+----------+-----------+-----------+------+----------+
|- Loop 1  |      186|      186|         3|          -|          -|    62|    no    |
+----------+---------+---------+----------+-----------+-----------+------+----------+
```

```
* Summary:
+--------------------+---------+-------+---------+---------+-----+
|        Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+--------------------+---------+-------+---------+---------+-----+
|DSP                 |        -|      -|        -|        -|    -|
|Expression          |        -|      -|        0|      416|    -|
|FIFO                |        -|      -|        -|        -|    -|
|Instance            |        -|      -|        -|        -|    -|
|Memory              |        -|      -|        -|        -|    -|
|Multiplexer         |        -|      -|        -|       60|    -|
|Register            |        -|      -|       42|        -|    -|
+--------------------+---------+-------+---------+---------+-----+
|Total               |        0|      0|       42|      476|    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|
+--------------------+---------+-------+---------+---------+-----+
```

TAC-HEP 2025

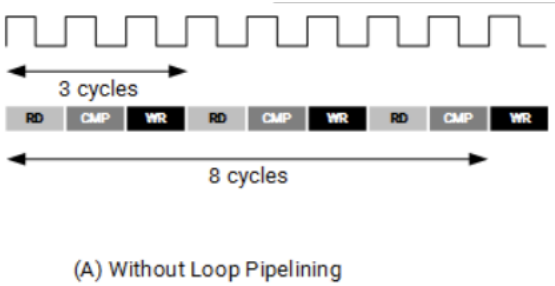
#pragma HLS Pipeline

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

Pragma HLS Pipeline

#pragma HLS pipeline II=<int>

- The PIPELINE pragma reduces the initiation interval (II) for a function or loop by allowing the concurrent execution of operations
- A pipelined function or loop can process new inputs every <N> clock cycles
- If HLS can't create a design with the specified II, it issues a warning and creates a design with the lowest possible II



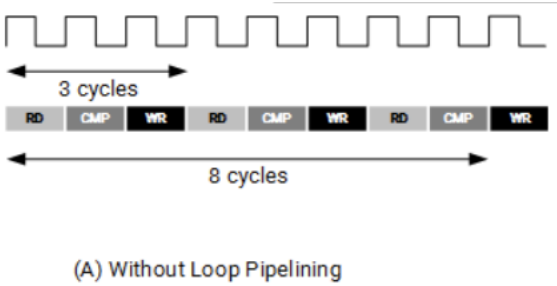
(A) Without Loop Pipelining

```
void func(input, output){
...
  for(i=0; i>=N; i++){
#pragma HLS pipeline II=2
    op_read;
    op_compute;
    op_write;
  }
...
}
```

Pragma HLS Pipeline

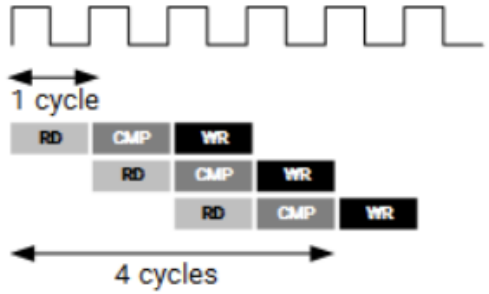
#pragma HLS pipeline II=<int>

- The PIPELINE pragma reduces the initiation interval (II) for a function or loop by allowing the concurrent execution of operations
- A pipelined function or loop can process new inputs every <N> clock cycles
- If HLS can't create a design with the specified II, it issues a warning and creates a design with the lowest possible II



Without Loop pipelining

```
void func(input, output){
...
  for(i=0; i>=N; i++){
#pragma HLS pipeline II=2
    op_read;
    op_compute;
    op_write;
  }
...
}
```



With Loop pipelining

Pragma HLS Pipeline: Example

```
#include "example.h"

dout_t example(din_t mem[N]) {

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

Pragma HLS Pipeline: Example

```
#include "example.h"

dout_t example(din_t mem[N]) {

    #pragma HLS pipeline II=2

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

Pragma HLS Pipeline: Example

```
#include "example.h"

dout_t example(din_t mem[N]) {

    #pragma HLS pipeline II=1

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

```
Timing:
  * Summary:
  +--------+----------+----------+------------+
  |  Clock |  Target  | Estimated| Uncertainty|
  +--------+----------+----------+------------+
  |ap_clk  | 25.00 ns | 2.534 ns |   3.12 ns  |
  +--------+----------+----------+------------+

Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+---------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
  |   min   |   max   |    min   |    max   | min | max |   Type  |
  +---------+---------+----------+----------+-----+-----+---------+
  |      126|      126| 3.150 us | 3.150 us |  126|  126|   none  |
  +---------+---------+----------+----------+-----+-----+---------+
```

```
* Loop:
+----------+---------+---------+----------+----------+----------+------+----------+
|          |  Latency (cycles) | Iteration|  Initiation Interval  | Trip |          |
| Loop Name|   min   |   max   |  Latency |  achieved |   target | Count| Pipelined|
+----------+---------+---------+----------+----------+----------+------+----------+
|- Loop 1  |      124|      124|         3|         2|         1|    62|    yes   |
+----------+---------+---------+----------+----------+----------+------+----------+
```

```
* Summary:
+--------------------+---------+-------+---------+---------+-----+
|        Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+--------------------+---------+-------+---------+---------+-----+
|DSP                 |        -|      -|        -|        -|    -|
|Expression          |        -|      -|        0|      106|    -|
|FIFO                |        -|      -|        -|        -|    -|
|Instance            |        -|      -|        -|        -|    -|
|Memory              |        -|      -|        -|        -|    -|
|Multiplexer         |        -|      -|        -|       78|    -|
|Register            |        -|      -|       46|        -|    -|
+--------------------+---------+-------+---------+---------+-----+
|Total               |        0|      0|       46|      184|    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|
+--------------------+---------+-------+---------+---------+-----+
```

Pragma HLS Pipeline: Example

```
#include "example.h"

dout_t example(din_t mem[N]) {

    #pragma HLS pipeline II=2

    dout_t total = 0;

    for (int i = 2; i < N; ++i)
        total += mem[i] + mem[i - 1] + mem[i - 2];

    return total;
}
```

```
Timing:
  * Summary:
  +--------+----------+----------+------------+
  |  Clock |  Target  | Estimated| Uncertainty|
  +--------+----------+----------+------------+
  |ap_clk  | 25.00 ns | 2.534 ns |   3.12 ns  |
  +--------+----------+----------+------------+

Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+---------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
  |   min   |   max   |    min   |    max   | min | max |   Type  |
  +---------+---------+----------+----------+-----+-----+---------+
  |      126|      126| 3.150 us | 3.150 us |  126|  126|   none  |
  +---------+---------+----------+----------+-----+-----+---------+
```

```
* Loop:
+----------+---------+---------+----------+----------+----------+------+----------+
|          |  Latency (cycles) | Iteration|  Initiation Interval  | Trip |          |
| Loop Name|   min   |   max   |  Latency |  achieved |   target  | Count| Pipelined|
+----------+---------+---------+----------+----------+----------+------+----------+
|- Loop 1  |      124|      124|         3|         2|         2|    62|    yes   |
+----------+---------+---------+----------+----------+----------+------+----------+
```

```
* Summary:
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|      106|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |        -|      -|        -|        -|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|       78|    -|
|Register             |        -|      -|       46|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |        0|      0|       46|      184|    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|
+---------------------+---------+-------+---------+---------+-----+
```

Pragma HLS Pipeline: Example

```
#include "example.h"

dout_t example(din_t mem[N]) {

   #pragma HLS pipeline II=3

   dout_t total = 0;

   for (int i = 2; i < N; ++i)
      total += mem[i] + mem[i - 1] + mem[i - 2];

   return total;
}
```

+ Timing:
  * Summary:

Clock	Target	Estimated	Uncertainty
ap_clk	25.00 ns	2.534 ns	3.12 ns

+ Latency:
  * Summary:

Latency (cycles) min	Latency (cycles) max	Latency (absolute) min	Latency (absolute) max	Interval min	Interval max	Pipeline Type
188	188	4.700 us	4.700 us	188	188	none

* Loop:

Loop Name	Latency (cycles) min	Latency (cycles) max	Iteration Latency	Initiation Interval achieved	Initiation Interval target	Trip Count	Pipelined
|- Loop 1	186	186	3	3	3	62	yes

* Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	104	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	66	-
Register	-	-	37	-	-
Total	0	0	37	170	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

A Comparison

	Default	Restructured	Pragma HLS Pipeline		
			II = 1	II = 2	II = 3
Estimated Clock (ns)	3.390	2.534	2.534	2.534	2.534
Latency (cycle)	187	126	126	126	188
Latency (μs)	4.675	3.150	3.150	3.150	4.7
Resources (FF)	35	50	46	46	37
Resources (LUT)	164	152	184	184	170
Resource (Others)	0	0	0	0	0

Reminder: Assignments

- Assignment-1 (13-02-2025)
- Assignment-2 (18-02-2025)
- Assignment-3 (27-02-2025)
- Assignment-4 (18-03-2025)
- Assignment-5 (18-03-2025)

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

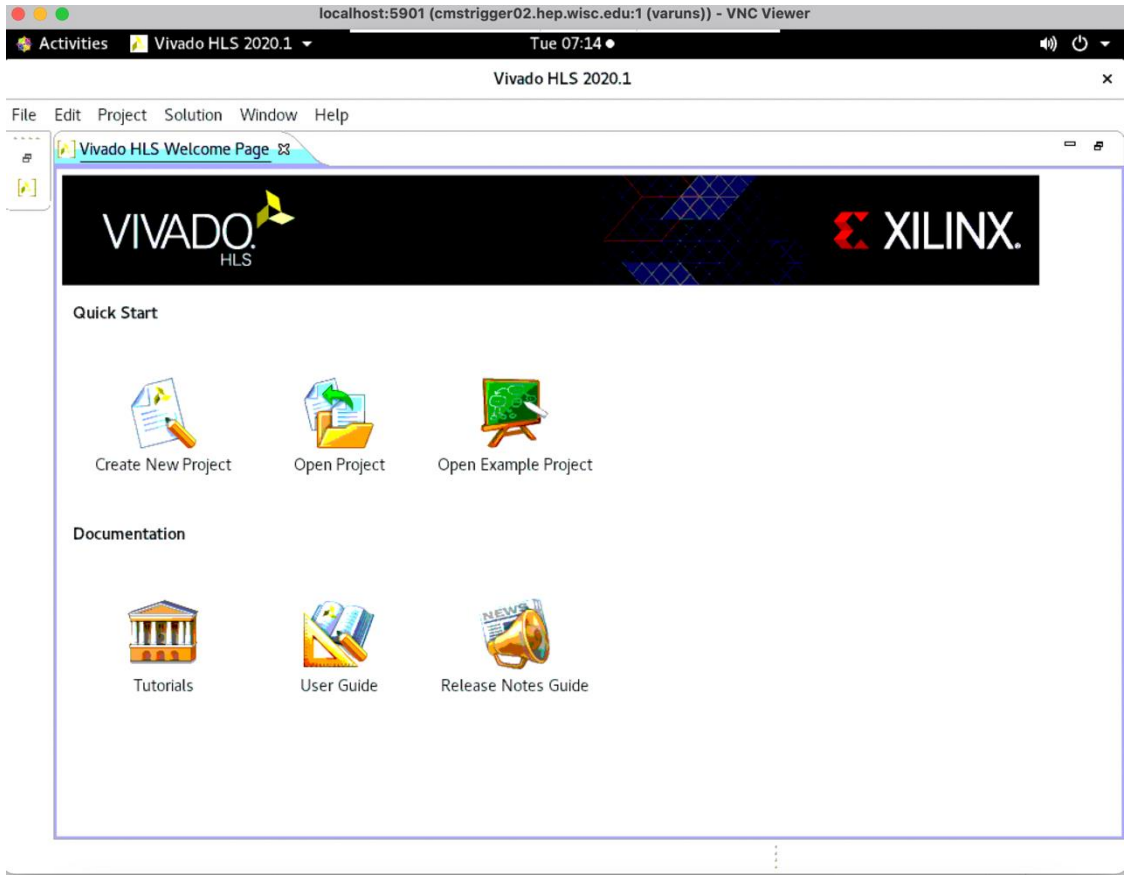
Reminder: HLS Setup

- 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