HLS LAB #B Cholesky algorithm

110061638 呂政和

I. Introduction of Cholesky algorithm

在線性代數中，cholesky decomposition 是指將一個正定的 Hermiton 矩陣分解成一個下三角矩陣，與其共軛轉置之乘積，利用這種方法能提高代數的運算效率，在使用 monte carlo 上也十分有用。

方程式可以從下圖進行分析。

If we write out the equation

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix} = \begin{pmatrix} L_{11}^2 & & \text{(symmetric)} \\ L_{21}L_{11} & L_{21}^2 + L_{22}^2 & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}$$

we obtain the following:

$$\mathbf{L} = \begin{pmatrix} \sqrt{A_{11}} & 0 & 0 \\ A_{21}/L_{11} & \sqrt{A_{22} - L_{21}^2} & 0 \\ A_{31}/L_{11} & \left(A_{32} - L_{31}L_{21}\right)/L_{22} & \sqrt{A_{33} - L_{31}^2 - L_{32}^2} \end{pmatrix}$$

and therefore the following formulas for the entries of **L**:

$$L_{j,j} = (\pm)\sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2},$$

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right) \quad \text{for } i > j.$$

因此，要計算只需利用其的左、上方元素的值。計算通常是以以下其中一種順序進行。

- Cholesky–Banachiewicz 演算法從矩陣 L 的左上角開始，依行進行計算
- Cholesky–Crout 演算法從矩陣 L 的左上角開始，依列進行計算。

若有需要，整個矩陣可以逐個元素計算得出，無論使用何種順序讀取。

經典 cholesky decomposition 的一個變形是 LDL 分解，即 A=LDL*，其中 L 是一個下三角矩陣，D 是一個對角矩陣。

該分解步驟如下：A=LDL*=L$D^{0.5}$($D^{0.5}$)*L*=L$D^{0.5}$(L$D^{0.5}$)*

以下舉一個實對稱矩陣作為範例：

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 6 & 1 & 0 \\ -8 & 5 & 3 \end{pmatrix} \begin{pmatrix} 2 & 6 & -8 \\ 0 & 1 & 5 \\ 0 & 0 & 3 \end{pmatrix}$$

下面為其 LDL^T分解：

$$\begin{pmatrix} 4 & 12 & -16 \\ 12 & 37 & -43 \\ -16 & -43 & 98 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -4 & 5 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 9 \end{pmatrix} \begin{pmatrix} 1 & 3 & -4 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

II. Run this design in CPU

透過利用 CPU 來計算：不同 size 下所需的運算時間，可以從下圖看到從 128*128 到 512*512 所需的時間會是 128*128 最少，512*512 最大。

```
[g110061638@ic22 cpu_src]$ ./test -M 128 -N 128 -seed 12
INFO: Matrix Row M: 128
INFO: Matrix Col N: 128
INFO: Finish CPU execution
INFO: CPU execution time is:300 us
errA = 0
dataAN = 128
dataAM = 128
--------------
INFO: Result correct
--------------
```

圖一：128*128 所需時間為 300us

```
[g110061638@ic22 cpu_src]$ ./test -M 256 -N 256 -seed 12
INFO: Matrix Row M: 256
INFO: Matrix Col N: 256
INFO: Finish CPU execution
INFO: CPU execution time is:2509 us
errA = 0
dataAN = 256
dataAM = 256
--------------
INFO: Result correct
```

圖二：256*256 所需時間為 2509us

```
[g110061638@ic22 cpu_src]$ g++ cpu_cholesky.cpp test.cpp matrixUtility.hpp -std=c++0x -O3 -o test
[g110061638@ic22 cpu_src]$ ./test -M 512 -N 512 -seed 12
INFO: Matrix Row M: 512
INFO: Matrix Col N: 512
INFO: Finish CPU execution
INFO: CPU execution time is:21390 us
errA = 0
dataAN = 512
dataAM = 512
-------------
INFO: Result correct
```

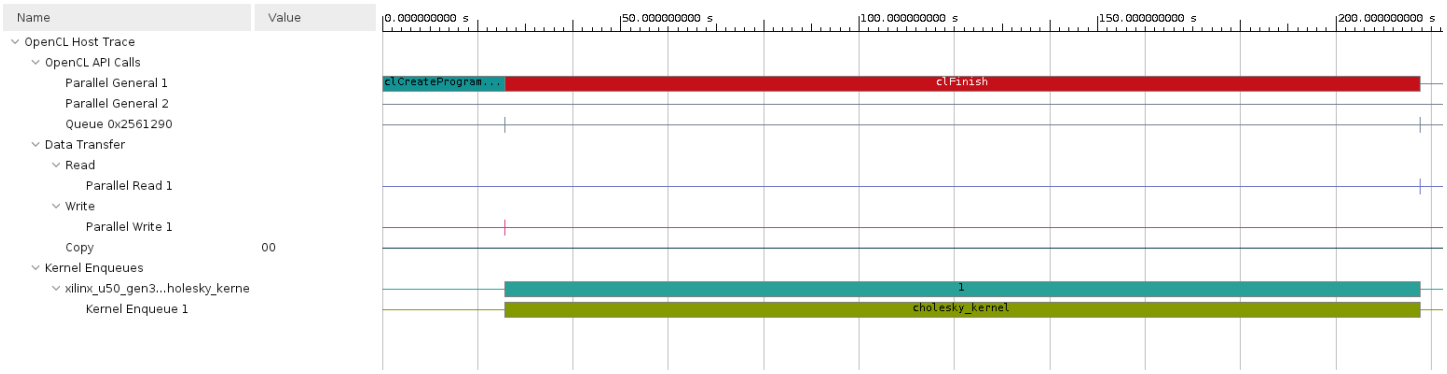
圖三：512*512 需時間為 21390us

這邊特別注意的是，在 cholesky 的 code 中，他的最大值可以設定到 2048*2048 去進行運算，但可以從下圖看到，運算會出現 error，這是因為學校 server 的限制所導致。

```
[g110061638@ic22 cpu_src]$ ./test -M 1024 -N 1024 -seed 12
INFO: Matrix Row M: 1024
INFO: Matrix Col N: 1024
Segmentation fault
```

III. Module 1 BaseLine

在這個 LAB 中，我不是用 vitis GUI 介面來完成，而是利用 terminal 來見環境並跑軟體與硬體的模擬，可以從產生的 run summary 看到執行的結果如下圖：



其中，圖上 OpenCL API Calls 說明所有的 OpenCL API Calls 會記錄在這項

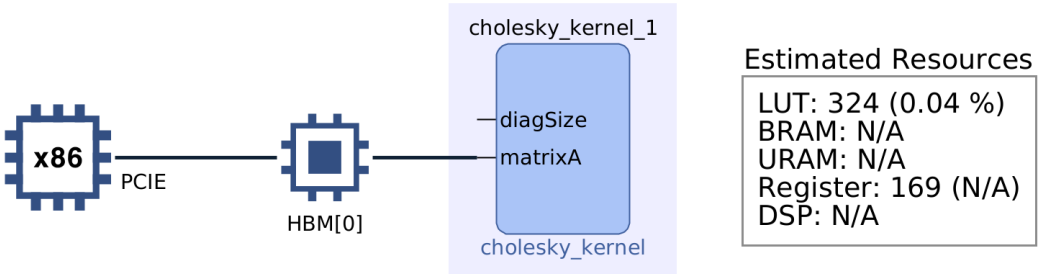
Data Transfer：這欄會記錄 DMA transfers from the host to the memory。

The data transfer from the host to the device appear under Write as they are written by the host, and the transfers from device to host appear under Read.

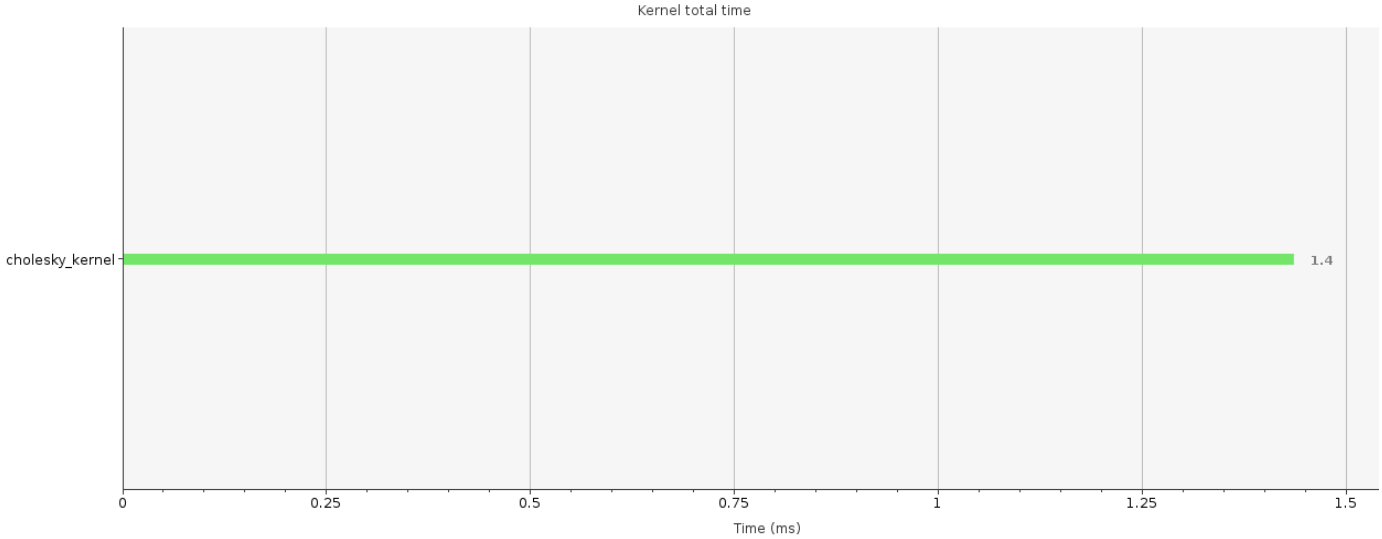
Kernel Enqueues：The kernels enqueued by the host program are shown here.

Multiple kernels can be scheduled to be executed at the same time, and they are traced from the point they are scheduled to run until the end of the kernel execution. Multiple entries would be shown in different rows depending on the number of overlapping kernel executions.

System diagram:



Profile Summary:



Kernel	Enqueues	Total time	Min time	Avg time	Max time
Cholesky_kernel	1	1.436	1.436	1.436	1.436

Performance estimation:

```
Timing:
  * Summary:
  +--------+---------+----------+------------+
  |  Clock |  Target | Estimated| Uncertainty|
  +--------+---------+----------+------------+
  |ap_clk  |  3.33 ns|  2.433 ns|     0.90 ns|
  +--------+---------+----------+------------+

Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+---------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
  |   min   |   max   |    min   |    max   | min | max |   Type  |
  +---------+---------+----------+----------+-----+-----+---------+
  |        ?|        ?|         ?|         ?|    ?|    ?|       no|
  +---------+---------+----------+----------+-----+-----+---------+
```

Utilization estimation:

== Utilization Estimates

* Summary:

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	294	-
FIFO	-	-	-	-	-
Instance	4	20	3561	3128	0
Memory	32	-	0	0	0
Multiplexer	-	-	-	2144	-
Register	-	-	1071	-	-
Total	36	20	4632	5566	0
Available SLR	1344	2976	871680	435840	320
Utilization SLR (%)	2	~0	~0	1	0
Available	2688	5952	1743360	871680	640
Utilization (%)	1	~0	~0	~0	0

Interface:

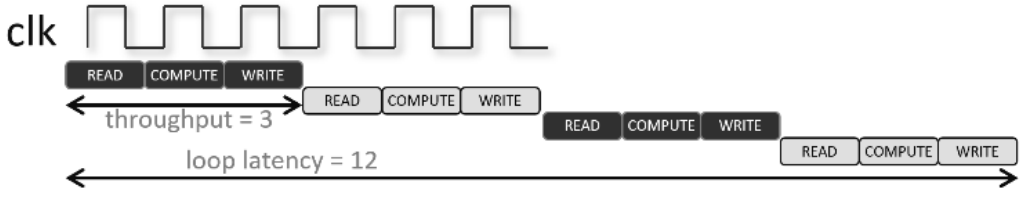
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_control_AWVALID	in	1	s_axi	control	scalar
s_axi_control_AWREADY	out	1	s_axi	control	scalar
s_axi_control_AWADDR	in	6	s_axi	control	scalar
s_axi_control_WVALID	in	1	s_axi	control	scalar
s_axi_control_WREADY	out	1	s_axi	control	scalar
s_axi_control_WDATA	in	32	s_axi	control	scalar
s_axi_control_WSTRB	in	4	s_axi	control	scalar
s_axi_control_ARVALID	in	1	s_axi	control	scalar
s_axi_control_ARREADY	out	1	s_axi	control	scalar
s_axi_control_ARADDR	in	6	s_axi	control	scalar
s_axi_control_RVALID	out	1	s_axi	control	scalar
s_axi_control_RREADY	in	1	s_axi	control	scalar
s_axi_control_RDATA	out	32	s_axi	control	scalar
s_axi_control_RRESP	out	2	s_axi	control	scalar
s_axi_control_BVALID	out	1	s_axi	control	scalar
s_axi_control_BREADY	in	1	s_axi	control	scalar
s_axi_control_BRESP	out	2	s_axi	control	scalar
ap_local_block	out	1	ap_ctrl_chain	cholesky_kernel	return value
ap_clk	in	1	ap_ctrl_chain	cholesky_kernel	return value
ap_rst_n	in	1	ap_ctrl_chain	cholesky_kernel	return value
interrupt	out	1	ap_ctrl_chain	cholesky_kernel	return value
m_axi_gmem_AWVALID	out	1	m_axi	gmem	pointer
m_axi_gmem_AWREADY	in	1	m_axi	gmem	pointer
m_axi_gmem_AWADDR	out	64	m_axi	gmem	pointer
m_axi_gmem_AWID	out	1	m_axi	gmem	pointer
m_axi_gmem_AWLEN	out	8	m_axi	gmem	pointer
m_axi_gmem_AWSIZE	out	3	m_axi	gmem	pointer
m_axi_gmem_AWBURST	out	2	m_axi	gmem	pointer
m_axi_gmem_AWLOCK	out	2	m_axi	gmem	pointer
m_axi_gmem_AWCACHE	out	4	m_axi	gmem	pointer
m_axi_gmem_AWPROT	out	3	m_axi	gmem	pointer
m_axi_gmem_AWQOS	out	4	m_axi	gmem	pointer
m_axi_gmem_AWREGION	out	4	m_axi	gmem	pointer
m_axi_gmem_AWUSER	out	1	m_axi	gmem	pointer
m_axi_gmem_WVALID	out	1	m_axi	gmem	pointer

m_axi_gmem_WREADY	in	1	m_axi	gmem	pointer
m_axi_gmem_WDATA	out	64	m_axi	gmem	pointer
m_axi_gmem_WSTRB	out	8	m_axi	gmem	pointer
m_axi_gmem_WLAST	out	1	m_axi	gmem	pointer
m_axi_gmem_WID	out	1	m_axi	gmem	pointer
m_axi_gmem_WUSER	out	1	m_axi	gmem	pointer
m_axi_gmem_ARVALID	out	1	m_axi	gmem	pointer
m_axi_gmem_ARREADY	in	1	m_axi	gmem	pointer
m_axi_gmem_ARADDR	out	64	m_axi	gmem	pointer
m_axi_gmem_ARID	out	1	m_axi	gmem	pointer
m_axi_gmem_ARLEN	out	8	m_axi	gmem	pointer
m_axi_gmem_ARSIZE	out	3	m_axi	gmem	pointer
m_axi_gmem_ARBURST	out	2	m_axi	gmem	pointer
m_axi_gmem_ARLOCK	out	2	m_axi	gmem	pointer
m_axi_gmem_ARCACHE	out	4	m_axi	gmem	pointer
m_axi_gmem_ARPROT	out	3	m_axi	gmem	pointer
m_axi_gmem_ARQOS	out	4	m_axi	gmem	pointer
m_axi_gmem_ARREGION	out	4	m_axi	gmem	pointer
m_axi_gmem_ARUSER	out	1	m_axi	gmem	pointer
m_axi_gmem_RVALID	in	1	m_axi	gmem	pointer
m_axi_gmem_RREADY	out	1	m_axi	gmem	pointer
m_axi_gmem_RDATA	in	64	m_axi	gmem	pointer
m_axi_gmem_RLAST	in	1	m_axi	gmem	pointer
m_axi_gmem_RID	in	1	m_axi	gmem	pointer
m_axi_gmem_RUSER	in	1	m_axi	gmem	pointer
m_axi_gmem_RRESP	in	2	m_axi	gmem	pointer
m_axi_gmem_BVALID	in	1	m_axi	gmem	pointer
m_axi_gmem_BREADY	out	1	m_axi	gmem	pointer
m_axi_gmem_BRESP	in	2	m_axi	gmem	pointer
m_axi_gmem_BID	in	1	m_axi	gmem	pointer
m_axi_gmem_BUSER	in	1	m_axi	gmem	pointer

IV. Module 2 Pipeline

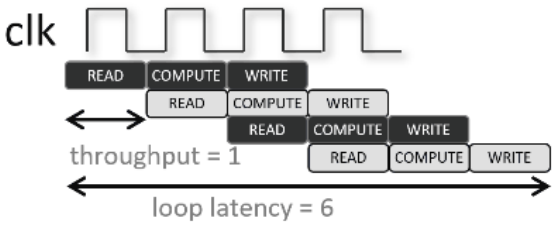
This module is meant to focus on the pipeline pragma and go through the description below. The kernel source code with the loops annotated with the pragma will produce the same results as in module 1, that's because since simple loops and inner loops (for nested loops) are automatically pipelined by the tool. Before pipeline:

```
void F (...)  {
...
lbl: for (i=0;i=<4;i++) {
       op_READ;
       op_COMPUTE;
       op_WRITE;
}
...
```

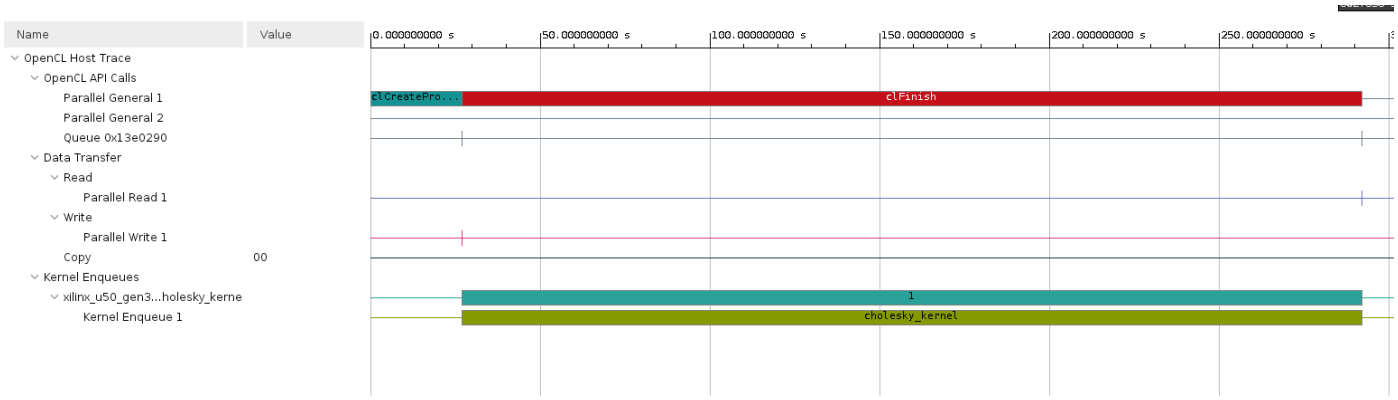


After pipeline:

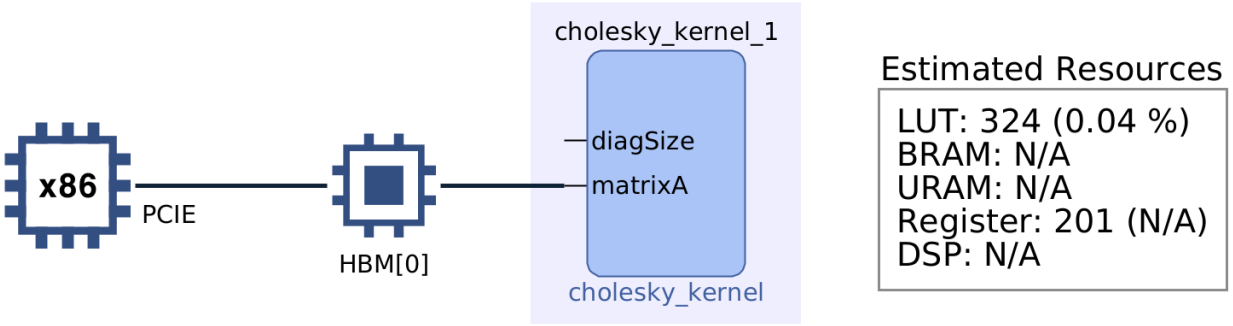
```
void F (...)  {
...
lbl: for (i=0;i=<4;i++) {
# PRAGMA HLS PIPELINE
      op_READ;
      op_COMPUTE;
      op_WRITE;
}
...
```



Run summary:



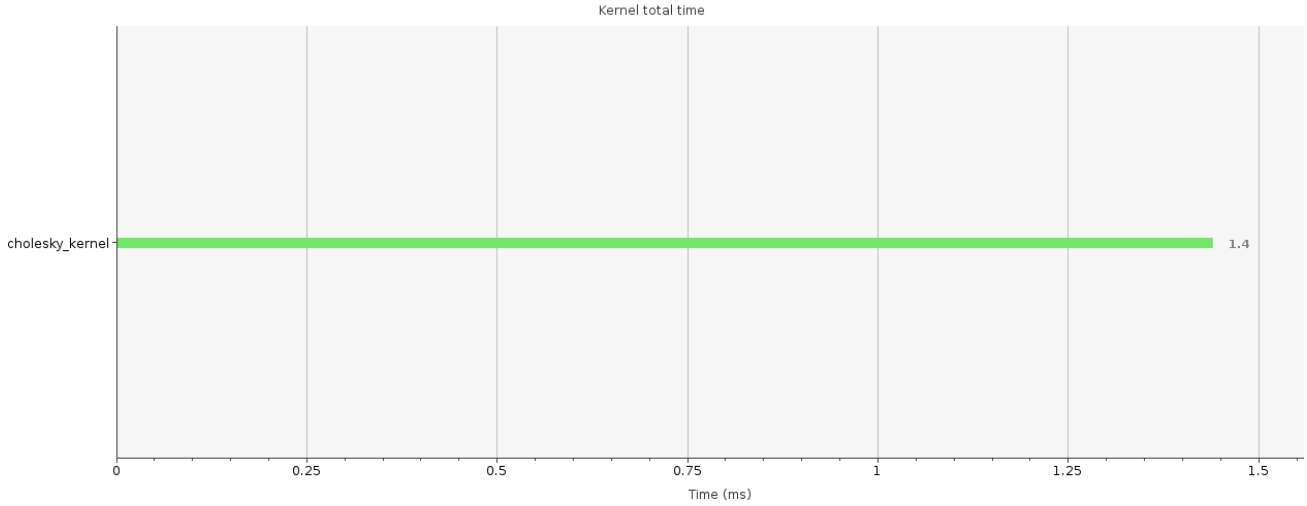
System diagram:



Estimated Resources

LUT:	324 (0.04 %)
BRAM:	N/A
URAM:	N/A
Register:	201 (N/A)
DSP:	N/A

Profile Summary:



Kernel	Enqueues	Total time	Min time	Avg time	Max time
Cholesky_kernel	1	1.436	1.436	1.436	1.436

Performance estimation:

```
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  |  3.33 ns|  2.433 ns|     0.90 ns|
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+---------+
    |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
    |   min   |   max   |    min   |    max   | min | max |   Type  |
    +---------+---------+----------+----------+-----+-----+---------+
    |        ?|        ?|         ?|         ?|    ?|    ?|       no|
    +---------+---------+----------+----------+-----+-----+---------+
```

Utilization estimation:

== Utilization Estimates

* Summary:

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	294	-
FIFO	-	-	-	-	-
Instance	4	20	3561	3128	0
Memory	32	-	0	0	0
Multiplexer	-	-	-	2144	-
Register	-	-	1135	-	-
Total	36	20	4696	5566	0
Available SLR	1344	2976	871680	435840	320
Utilization SLR (%)	2	~0	~0	1	0
Available	2688	5952	1743360	871680	640
Utilization (%)	1	~0	~0	~0	0

Interface:

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_control_AWVALID	in	1	s_axi	control	scalar
s_axi_control_AWREADY	out	1	s_axi	control	scalar
s_axi_control_AWADDR	in	6	s_axi	control	scalar
s_axi_control_WVALID	in	1	s_axi	control	scalar
s_axi_control_WREADY	out	1	s_axi	control	scalar
s_axi_control_WDATA	in	32	s_axi	control	scalar
s_axi_control_WSTRB	in	4	s_axi	control	scalar
s_axi_control_ARVALID	in	1	s_axi	control	scalar
s_axi_control_ARREADY	out	1	s_axi	control	scalar
s_axi_control_ARADDR	in	6	s_axi	control	scalar
s_axi_control_RVALID	out	1	s_axi	control	scalar
s_axi_control_RREADY	in	1	s_axi	control	scalar
s_axi_control_RDATA	out	32	s_axi	control	scalar
s_axi_control_RRESP	out	2	s_axi	control	scalar
s_axi_control_BVALID	out	1	s_axi	control	scalar
s_axi_control_BREADY	in	1	s_axi	control	scalar
s_axi_control_BRESP	out	2	s_axi	control	scalar
ap_local_block	out	1	ap_ctrl_chain	cholesky_kernel	return value
ap_clk	in	1	ap_ctrl_chain	cholesky_kernel	return value
ap_rst_n	in	1	ap_ctrl_chain	cholesky_kernel	return value
interrupt	out	1	ap_ctrl_chain	cholesky_kernel	return value

m_axi_gmem0_AWVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_AWADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_AWID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_AWSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_WDATA	out	64	m_axi	gmem0	pointer
m_axi_gmem0_WSTRB	out	8	m_axi	gmem0	pointer
m_axi_gmem0_WLAST	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_ARADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_ARID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_ARSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RDATA	in	64	m_axi	gmem0	pointer
m_axi_gmem0_RLAST	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RUSER	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_BRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BUSER	in	1	m_axi	gmem0	pointer

V. Module 3 datatype

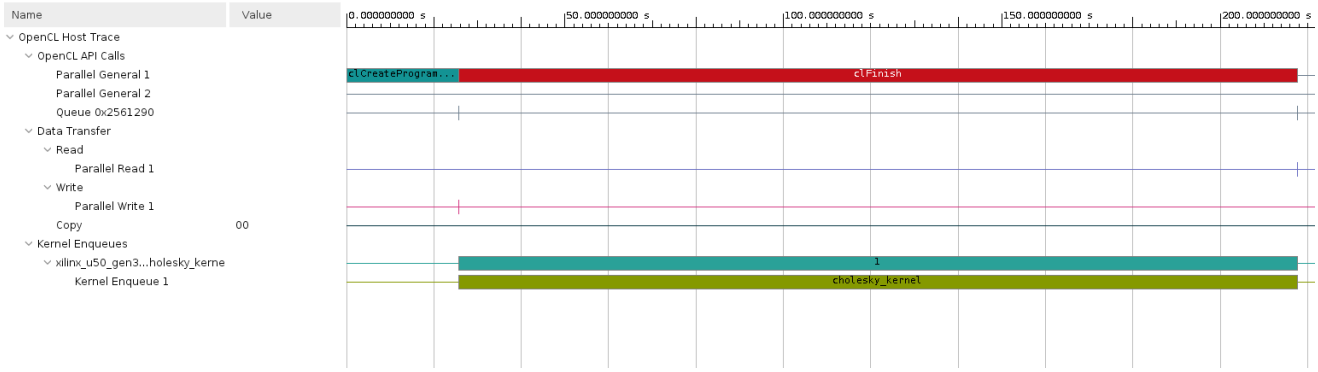
In this module both the kernel and host code are modified to use 32-bit floating point data types(float) instead of the 64-bit floating point(double) to show the performance and Xilinx utilization beneficial impact of downsizing data types.

Kernel Resources Used (regular floating point versus double):

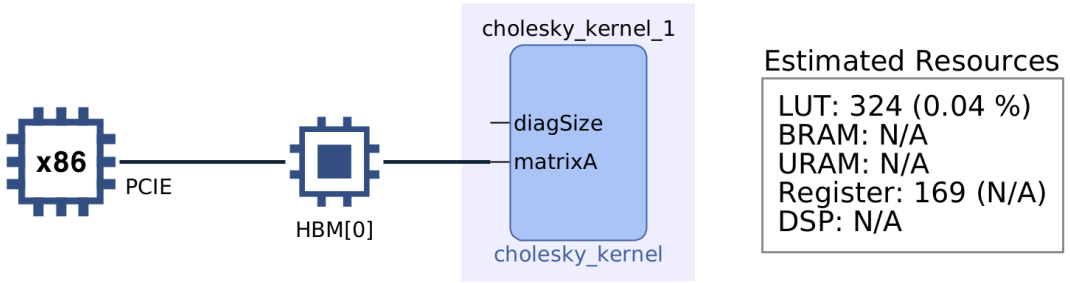
Name	LUT	LUTAsMem	REG	BRAM	DSP
Kernel with double	10190	799	10191	514	18
Kernel with float	5071	746	5124	258	12

As expected, the resource utilization comes down across for both logic and storage

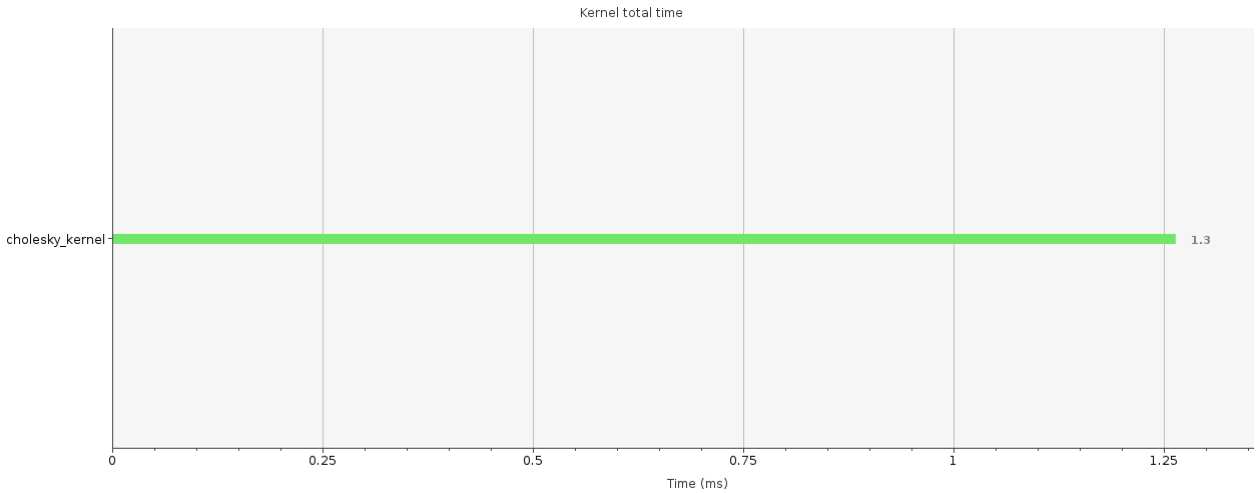
Run summary:



System diagram:



Profile Summary:



Kernel	Enqueues	Total time	Min time	Avg time	Max time
Cholesky_kernel	1	1.264	1.264	1.264	1.264

Performance estimation:

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  |  3.33 ns|  2.433 ns|     0.90 ns|
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+---------+
    |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
    |   min   |   max   |    min   |    max   | min | max |   Type  |
    +---------+---------+----------+----------+-----+-----+---------+
    |        ?|        ?|         ?|         ?|    ?|    ?|       no|
    +---------+---------+----------+----------+-----+-----+---------+
```

Utilization estimation:

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|     294|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        2|    14|     2496|    2388|    0|
|Memory               |       16|     -|        0|       0|    0|
|Multiplexer          |        -|     -|        -|    1820|    -|
|Register             |        -|     -|      930|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |       18|    14|     3426|    4502|    0|
+---------------------+---------+------+---------+--------+-----+
|Available SLR        |     1344|  2976|   871680|  435840|  320|
+---------------------+---------+------+---------+--------+-----+
|Utilization SLR (%)  |        1|    ~0|       ~0|       1|    0|
+---------------------+---------+------+---------+--------+-----+
|Available            |     2688|  5952|  1743360|  871680|  640|
+---------------------+---------+------+---------+--------+-----+
|Utilization (%)      |       ~0|    ~0|       ~0|      ~0|    0|
+---------------------+---------+------+---------+--------+-----+
```

Interface:

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_control_AWVALID	in	1	s_axi	control	scalar
s_axi_control_AWREADY	out	1	s_axi	control	scalar
s_axi_control_AWADDR	in	6	s_axi	control	scalar
s_axi_control_WVALID	in	1	s_axi	control	scalar
s_axi_control_WREADY	out	1	s_axi	control	scalar
s_axi_control_WDATA	in	32	s_axi	control	scalar
s_axi_control_WSTRB	in	4	s_axi	control	scalar
s_axi_control_ARVALID	in	1	s_axi	control	scalar
s_axi_control_ARREADY	out	1	s_axi	control	scalar
s_axi_control_ARADDR	in	6	s_axi	control	scalar
s_axi_control_RVALID	out	1	s_axi	control	scalar
s_axi_control_RREADY	in	1	s_axi	control	scalar
s_axi_control_RDATA	out	32	s_axi	control	scalar
s_axi_control_RRESP	out	2	s_axi	control	scalar
s_axi_control_BVALID	out	1	s_axi	control	scalar
s_axi_control_BREADY	in	1	s_axi	control	scalar
s_axi_control_BRESP	out	2	s_axi	control	scalar
ap_local_block	out	1	ap_ctrl_chain	cholesky_kernel	return value
ap_clk	in	1	ap_ctrl_chain	cholesky_kernel	return value
ap_rst_n	in	1	ap_ctrl_chain	cholesky_kernel	return value
interrupt	out	1	ap_ctrl_chain	cholesky_kernel	return value
m_axi_gmem0_AWVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_AWADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_AWID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_AWSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_WDATA	out	32	m_axi	gmem0	pointer
m_axi_gmem0_WSTRB	out	4	m_axi	gmem0	pointer
m_axi_gmem0_WLAST	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_ARADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_ARID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_ARSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RDATA	in	32	m_axi	gmem0	pointer
m_axi_gmem0_RLAST	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RUSER	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_BRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BUSER	in	1	m_axi	gmem0	pointer

VI. Module 4 dataflow

The **DATAFLOW** pragma enables task-level pipelining, allowing functions and loops to overlap in their operation, increasing the concurrency of the register transfer level(RTL) implementation, and increasing the overall throughput of the design.

When the **DATAFLOW** pragma is specified, the HLS tool analyzes the data flow between sequential functions or loops and creates channels (based on ping pong RAMs or FIFOs) that allow consumer functions or loops to start operation before the producer functions or loops have completed. This allows functions or loops to operate in parallel, which decreases latency and improves the throughput of the RTL.

Specifies `DATAFLOW` optimization within the loop wr_loop_j.

```
wr_loop_j: for (int j = 0; j < TILE_PER_ROW; ++j) {
#pragma HLS DATAFLOW
   wr_buf_loop_m: for (int m = 0; m < HEIGHT; ++m) {
      wr_buf_loop_n: for (int n = 0; n < WIDTH; ++n) {
      #pragma HLS PIPELINE
      // should burst WIDTH in WORD beat
         outFifo >> tile[m][n];
      }
   }
   wr_loop_m: for (int m = 0; m < HEIGHT; ++m) {
      wr_loop_n: for (int n = 0; n < WIDTH; ++n) {
      #pragma HLS PIPELINE
         outx[HEIGHT*TILE_PER_ROW*WIDTH*i+TILE_PER_ROW*WIDTH*m+WIDTH*j+n] = tile[m][n];
      }
   }
}
```

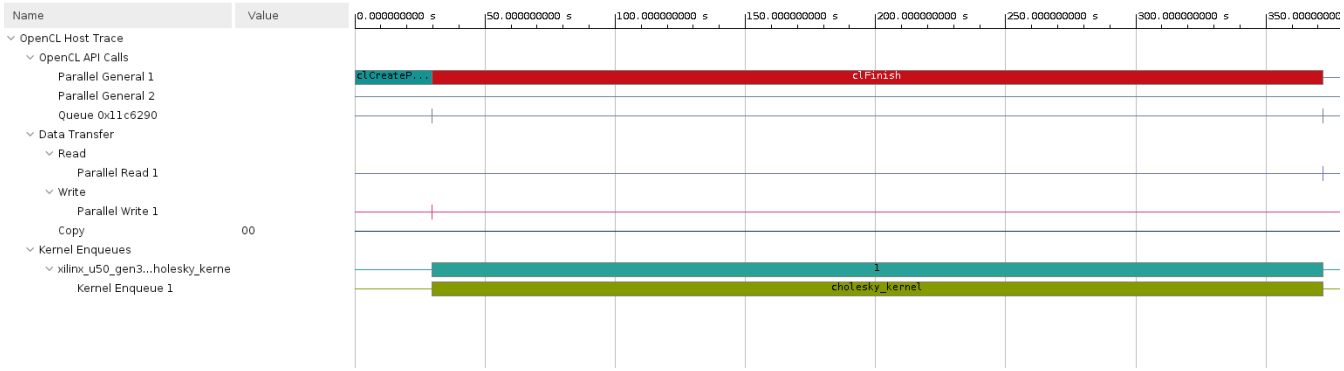
上圖為範例，在 function 中加入#pragma HLS DATAFLOW 讓 HLS 能優化其迴圈。

在 module4 中，新設了一個變數 NCU，去表示平行運算的數量，NCU 會被傳到 chol_col_wrapper 中，以下程式會呼叫 chol_col 16 次

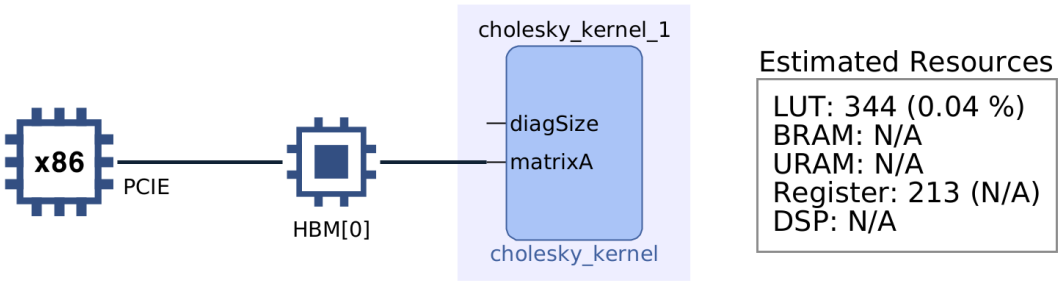
```
template <typename T, int N, int NCU>
void chol_col_wrapper(int n, T dataA[NCU][(N + NCU - 1) / NCU][N], T dataj[NCU][N], T tmp1, int j)
{
#pragma HLS DATAFLOW

Loop_row:
    for (int num = 0; num < NCU; num++)
    {
#pragma HLS unroll factor = NCU
        chol_col<T, N, NCU>(n, dataA[num], dataj[num], tmp1, num, j);
    }
}
```

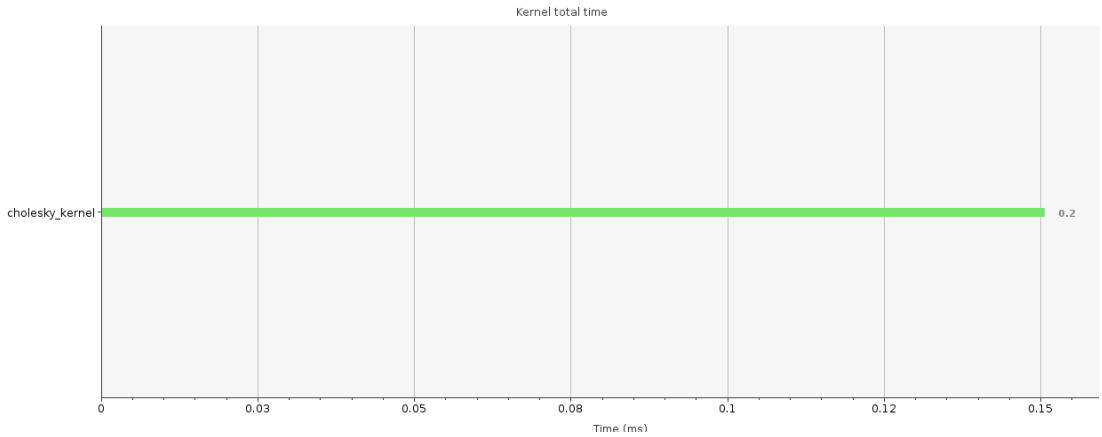
Run summary:



System diagram:



Profile summary:



Kernel	Enqueues	Total time	Min time	Avg time	Max time
Cholesky_kernel	1	0.151	0.151	0.151	0.151

Performance estimation:

```
================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  |  3.33 ns|  2.433 ns|     0.90 ns|
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+---------+
    |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
    |   min   |   max   |    min   |    max   | min | max |   Type  |
    +---------+---------+----------+----------+-----+-----+---------+
    |        ?|        ?|         ?|         ?|    ?|    ?|       no|
    +---------+---------+----------+----------+-----+-----+---------+
```

Utilization estimation:

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+------+---------+--------+-----+
|         Name        | BRAM_18K|  DSP |    FF   |   LUT  | URAM|
+---------------------+---------+------+---------+--------+-----+
|DSP                  |        -|     -|        -|       -|    -|
|Expression           |        -|     -|        0|      88|    -|
|FIFO                 |        -|     -|        -|       -|    -|
|Instance             |        4|   196|    81560|   60228|    0|
|Memory               |       32|     -|        0|       0|   64|
|Multiplexer          |        -|     -|        -|    3189|    -|
|Register             |        -|     -|      631|       -|    -|
+---------------------+---------+------+---------+--------+-----+
|Total                |       36|   196|    82191|   63505|   64|
+---------------------+---------+------+---------+--------+-----+
|Available SLR        |     1344|  2976|   871680|  435840|  320|
+---------------------+---------+------+---------+--------+-----+
|Utilization SLR (%)  |        2|     6|        9|      14|   20|
+---------------------+---------+------+---------+--------+-----+
|Available            |     2688|  5952|  1743360|  871680|  640|
+---------------------+---------+------+---------+--------+-----+
|Utilization (%)      |        1|     3|        4|       7|   10|
+---------------------+---------+------+---------+--------+-----+
```

Interface:

== Interface

* Summary:

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
s_axi_control_AWVALID	in	1	s_axi	control	scalar
s_axi_control_AWREADY	out	1	s_axi	control	scalar
s_axi_control_AWADDR	in	6	s_axi	control	scalar
s_axi_control_WVALID	in	1	s_axi	control	scalar
s_axi_control_WREADY	out	1	s_axi	control	scalar
s_axi_control_WDATA	in	32	s_axi	control	scalar
s_axi_control_WSTRB	in	4	s_axi	control	scalar
s_axi_control_ARVALID	in	1	s_axi	control	scalar
s_axi_control_ARREADY	out	1	s_axi	control	scalar
s_axi_control_ARADDR	in	6	s_axi	control	scalar
s_axi_control_RVALID	out	1	s_axi	control	scalar
s_axi_control_RREADY	in	1	s_axi	control	scalar
s_axi_control_RDATA	out	32	s_axi	control	scalar
s_axi_control_RRESP	out	2	s_axi	control	scalar
s_axi_control_BVALID	out	1	s_axi	control	scalar
s_axi_control_BREADY	in	1	s_axi	control	scalar
s_axi_control_BRESP	out	2	s_axi	control	scalar
ap_local_block	out	1	ap_ctrl_chain	cholesky_kernel	return value
ap_clk	in	1	ap_ctrl_chain	cholesky_kernel	return value
ap_rst_n	in	1	ap_ctrl_chain	cholesky_kernel	return value
interrupt	out	1	ap_ctrl_chain	cholesky_kernel	return value
m_axi_gmem0_AWVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_AWADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_AWID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_AWLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_AWSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_AWCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_AWQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_AWUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_WDATA	out	64	m_axi	gmem0	pointer
m_axi_gmem0_WSTRB	out	8	m_axi	gmem0	pointer
m_axi_gmem0_WLAST	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_WUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARVALID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARREADY	in	1	m_axi	gmem0	pointer
m_axi_gmem0_ARADDR	out	64	m_axi	gmem0	pointer
m_axi_gmem0_ARID	out	1	m_axi	gmem0	pointer
m_axi_gmem0_ARLEN	out	8	m_axi	gmem0	pointer
m_axi_gmem0_ARSIZE	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARBURST	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARLOCK	out	2	m_axi	gmem0	pointer
m_axi_gmem0_ARCACHE	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARPROT	out	3	m_axi	gmem0	pointer
m_axi_gmem0_ARQOS	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARREGION	out	4	m_axi	gmem0	pointer
m_axi_gmem0_ARUSER	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_RDATA	in	64	m_axi	gmem0	pointer
m_axi_gmem0_RLAST	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RUSER	in	1	m_axi	gmem0	pointer
m_axi_gmem0_RRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BVALID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BREADY	out	1	m_axi	gmem0	pointer
m_axi_gmem0_BRESP	in	2	m_axi	gmem0	pointer
m_axi_gmem0_BID	in	1	m_axi	gmem0	pointer
m_axi_gmem0_BUSER	in	1	m_axi	gmem0	pointer

Result Summary:

Module	CPU	Module1	Moduel2	Module3	Module4
Exe. Time	21461	793950	793732	536784	11698
Speed up(cpu)	1	0.03x	0.03x	0.04x	1.83x
Speed up	N/A	1	1	1.48x	68x

心得：這次的 Lab.實驗需要我們自己按照 github 上面的 Tutorial 來進行操作，但實際操作起來還是碰到非常多的問題，例如：我一開始有遇到在 make file 時，需要輸入”export LC_ALL = “C””的這個指令來解決，但在學校的伺服器中並沒有 export 這個指令，請教助教後，將這項指令加在 makefile 裡面就可以解決了，也遇到過 vitis 的版本問題，這時候也是在 makefile 裡面加上-version 2.0.1 就可以解決，最後我在按照 github 執行過程中，並不能像上面一樣使用 vitis GUI 來實驗，因為所給的壓縮檔中並沒有 cholesky_host 檔，有請教助教後，發現 test.cpp 應該就是 host 檔，但我是選擇第二個實驗方法，直接在 terminal 中 make run，就可以解決這個問題，在經過這個 Lab 之後，對於 vitis 的操作有了更深的了解，也清楚知道 baseline、pipeline、datatype、dataflow 之間的差異與 vitis 能帶來的加速方便，相信在之後的課程中會對 vitis 有更多的認識。

Github：https://github.com/hank871116/HLS_LAB_B_5