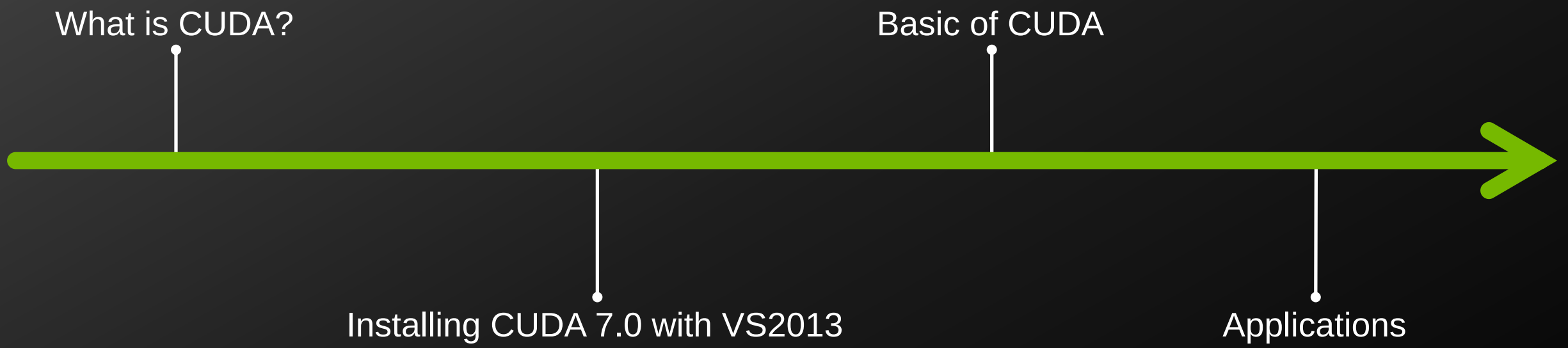
Parallel Programming Using Compute Unified Device Architecture

ISL Lab Seminar
Han-Sol Kang

INDEX


What is CUDA?
Basic of CUDA
Installing CUDA 7.0 with VS2013
Applications

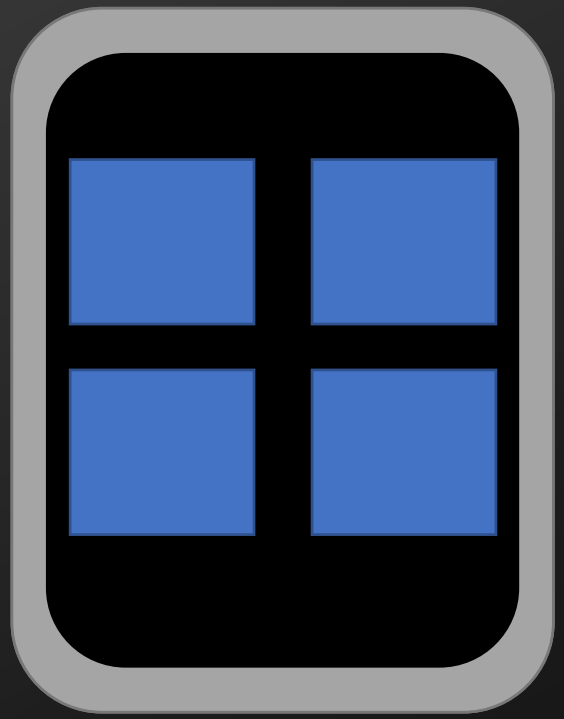
What is CUDA?

CUDA® is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

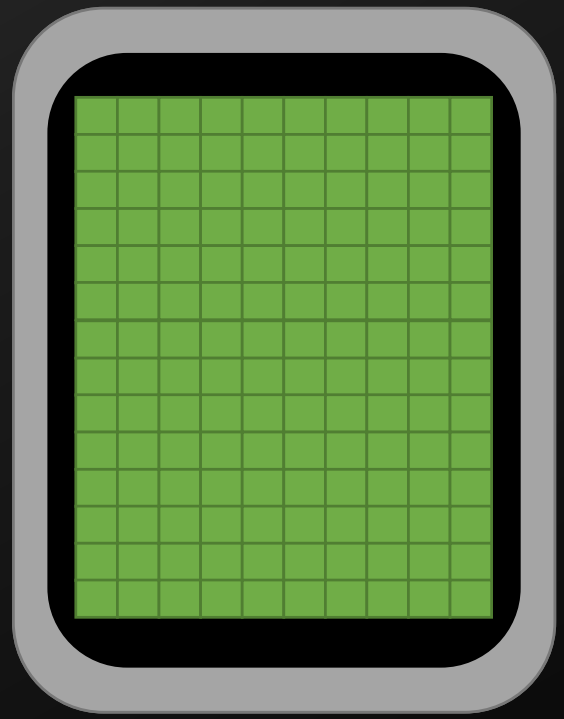
* http://www.nvidia.com/object/cuda_home_new.html#sthash.1HqbToYO.dpuf

What is CUDA?

CPU Vs. GPU



CPU
Multiple Cores

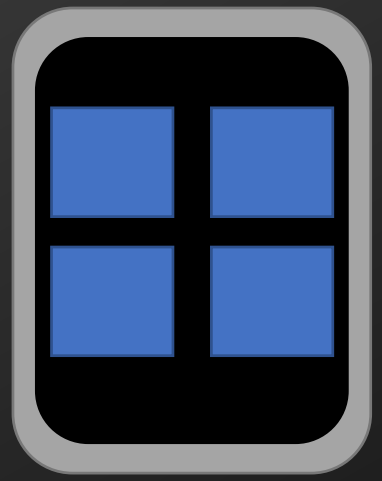


GPU
Thousand of Cores

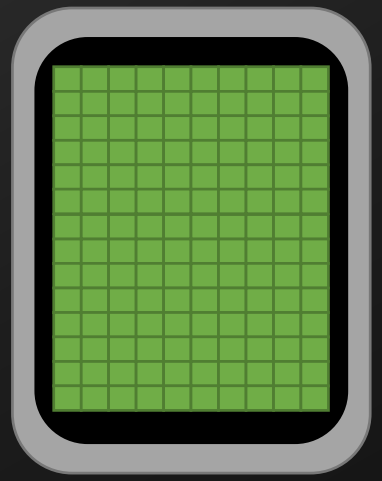
Image System Laboratory

What is CUDA?

CPU Vs. GPU



CPU
Multiple Cores



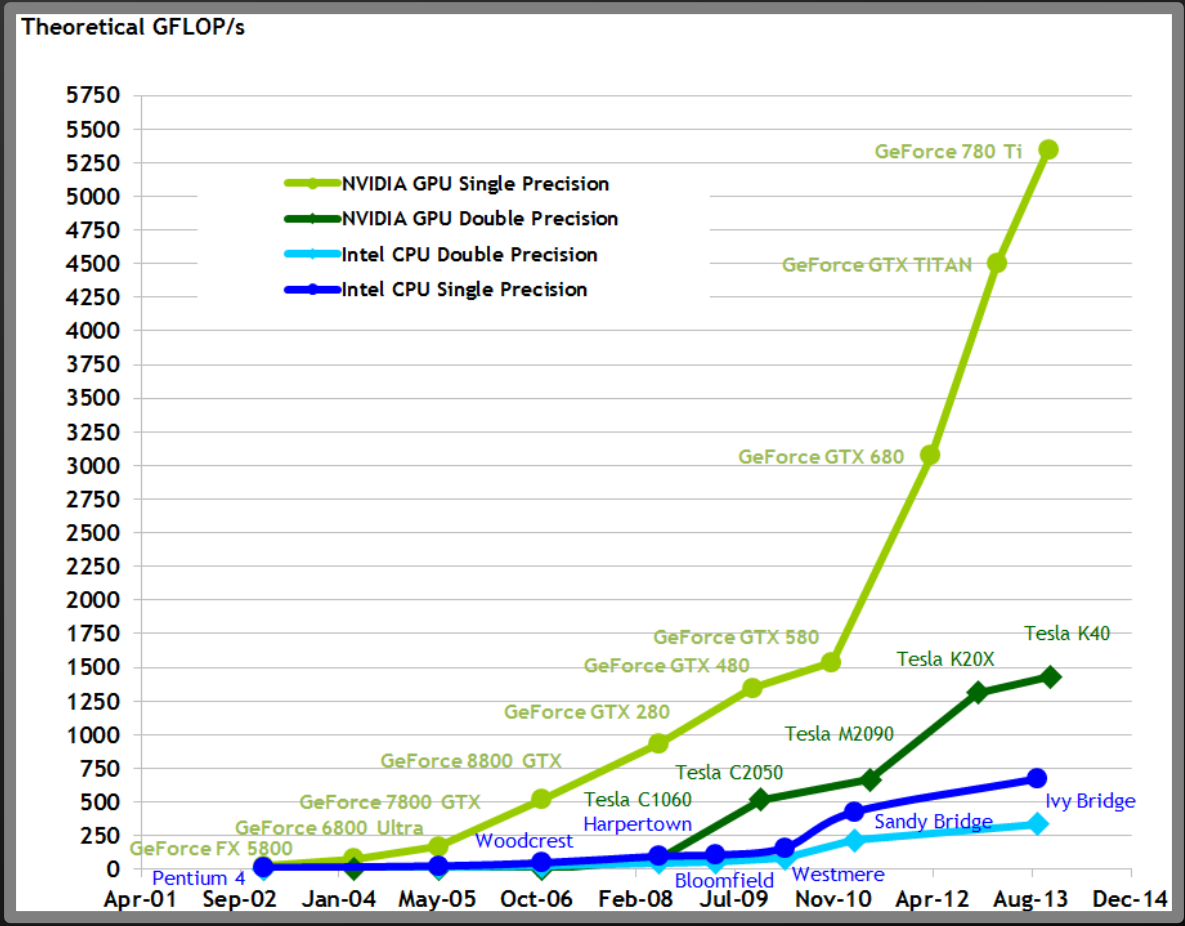
GPU
Thousand of Cores

GPGPU : General Purpose Graphic Processing Unit

ISL Image System Laboratory

What is CUDA

Floating point Operation per second for the CPU and GPU



* http://docs.nvidia.com/cuda/cuda-c-programming-guide/

ISL Image System Laboratory

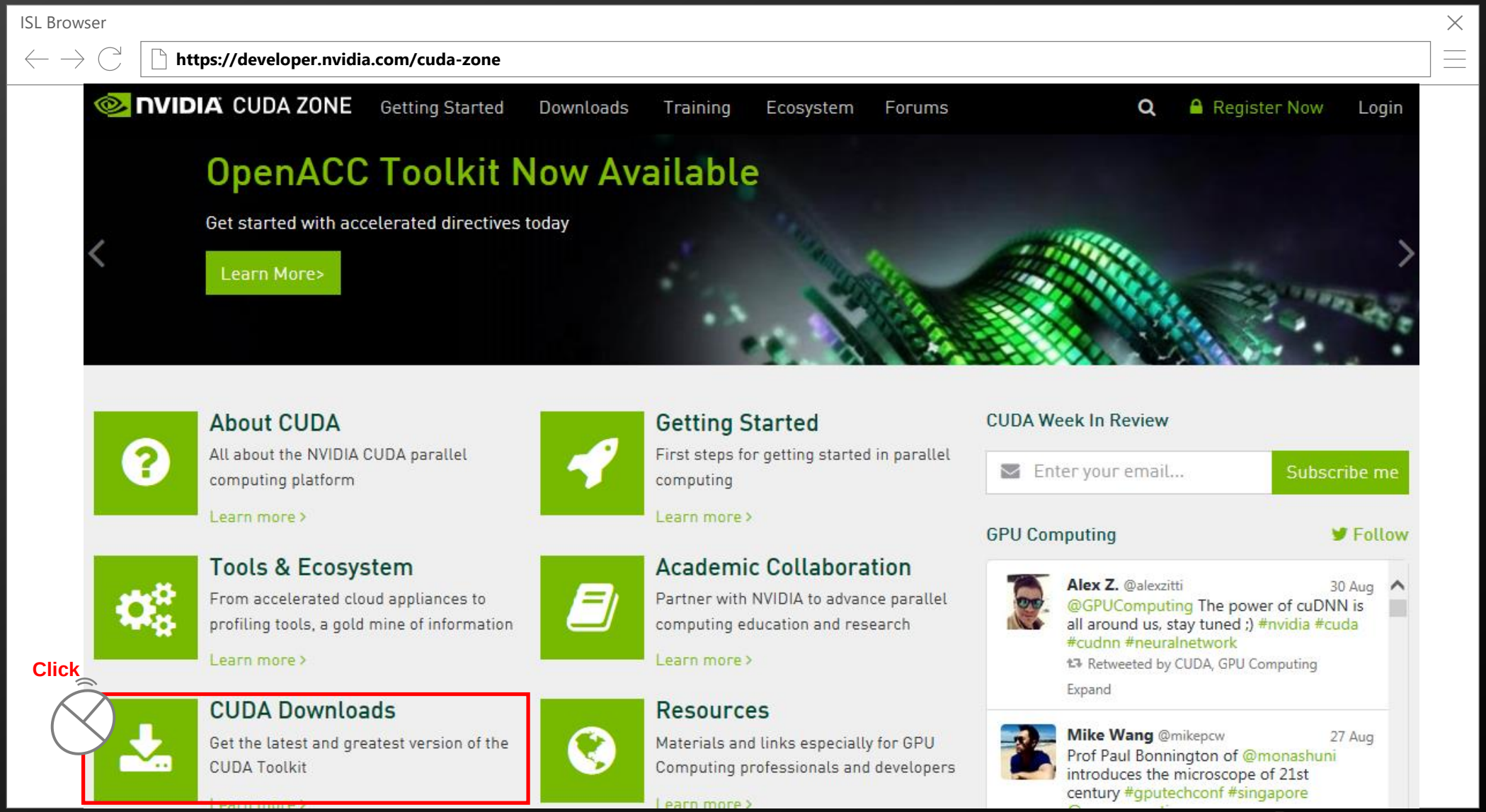
What is CUDA

GPGPU Programming Models *

	Controllable Hardware	Platform Dependency	Features
NVIDIA CUDA	GPU	NVIDA Geforce8 ↑	Runtime API /Device API
OpenCL	Heterogeneous CPU/GPU	Platform Independent	C/Runtime/API
OpenACC Directives for Accelerators			Compiler hints, “directives”
C++ AMP Accelerated Massive Parallelism with Microsoft Visual C++		Window 7 ↑ Visual Studio 2012 ↑ DirectX 11.0 ↑	Lambda /parallel loop statement

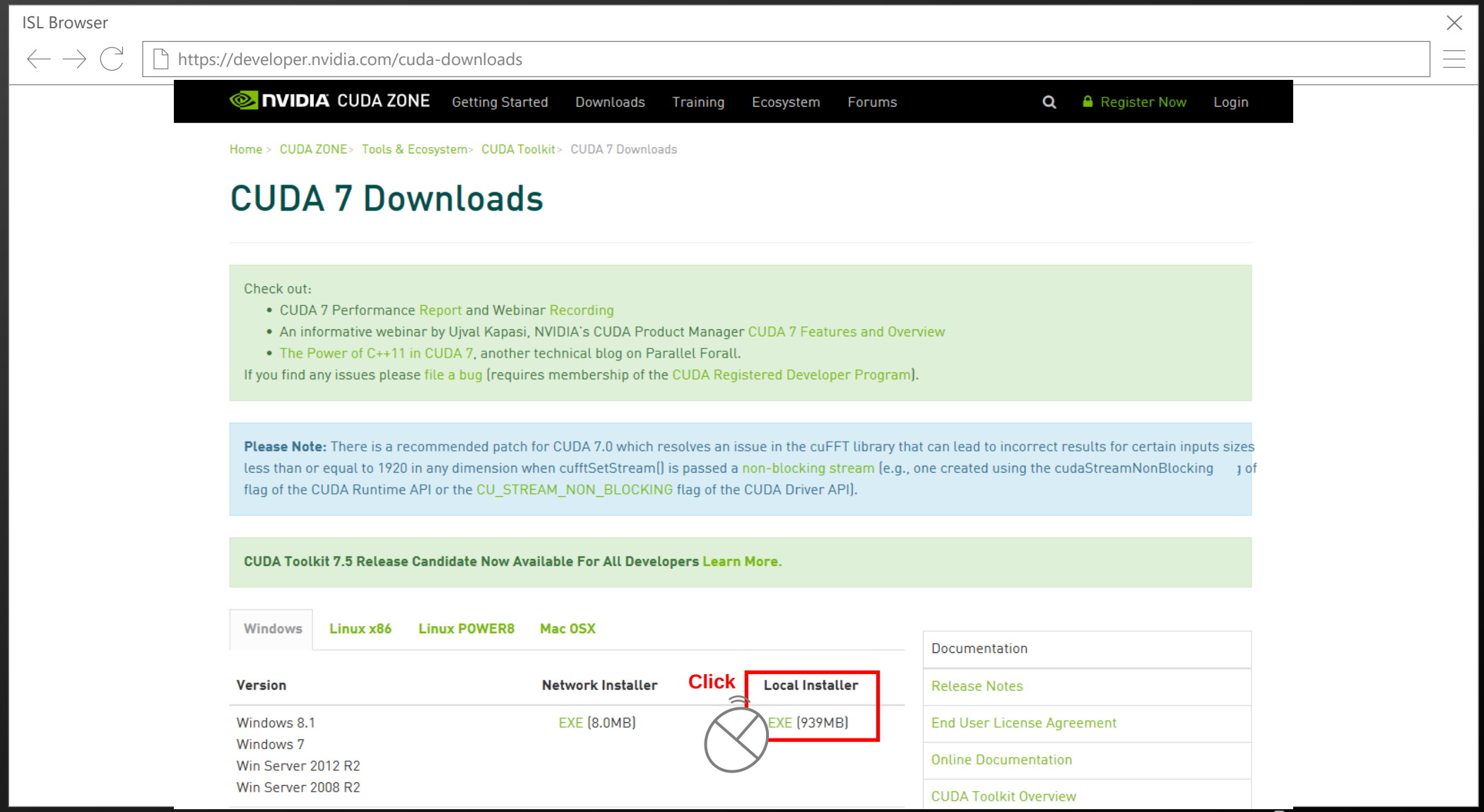
* 이현진, et al. “GPGPU 프로그래밍 모델의 기술 동향." *한국정보처리학회 춘계학술대회* 20.1 (2013): 389-391.

Installing CUDA 7.0 with VS 2013



ISL Image System Laboratory

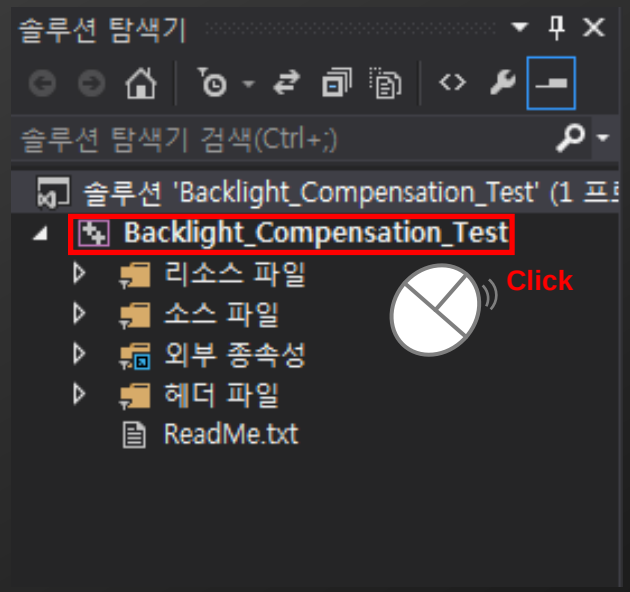
Installing CUDA 7.0 with VS 2013



ISL Image System Laboratory

Installing CUDA 7.0 with VS 2013

Visual Studio Settings



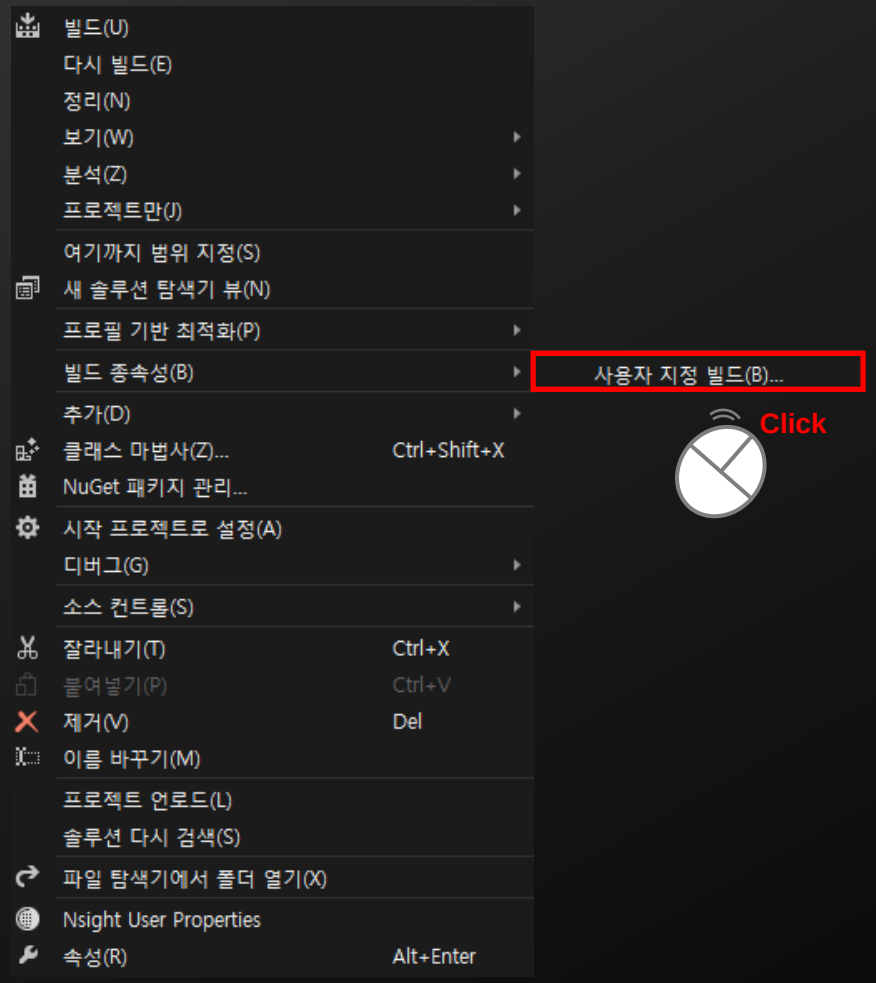


Image System Laboratory

Installing CUDA 7.0 with VS 2013

Visual Studio Settings

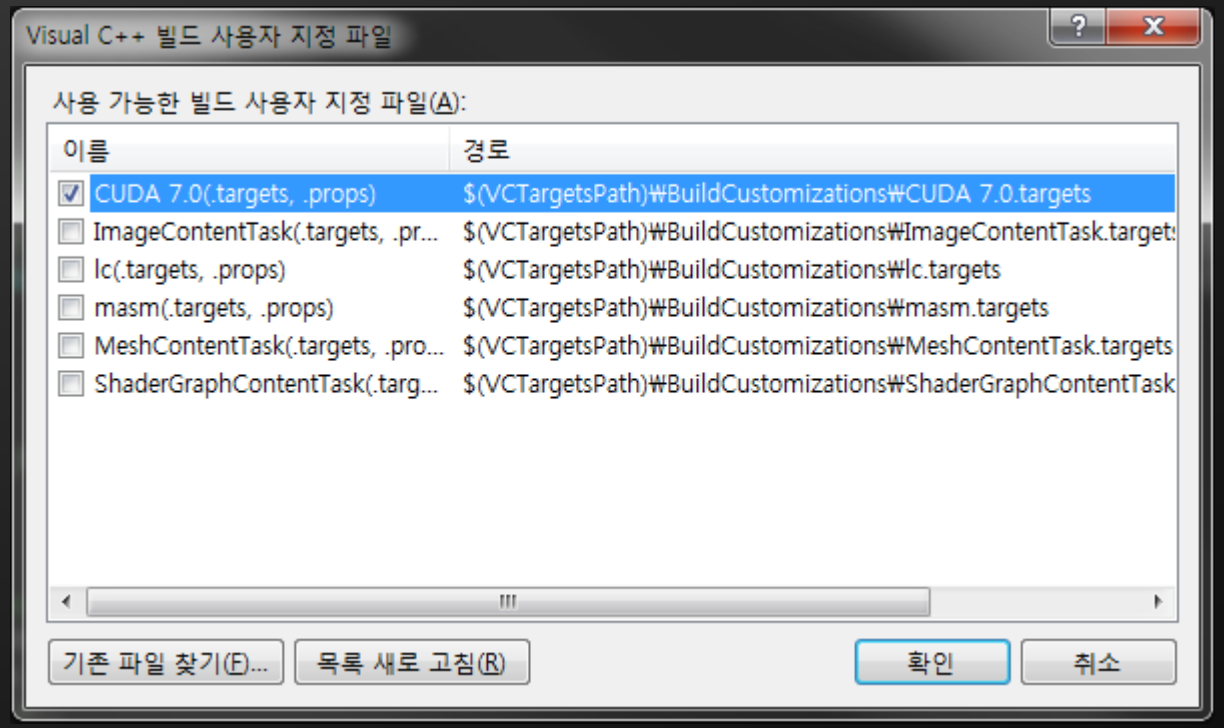
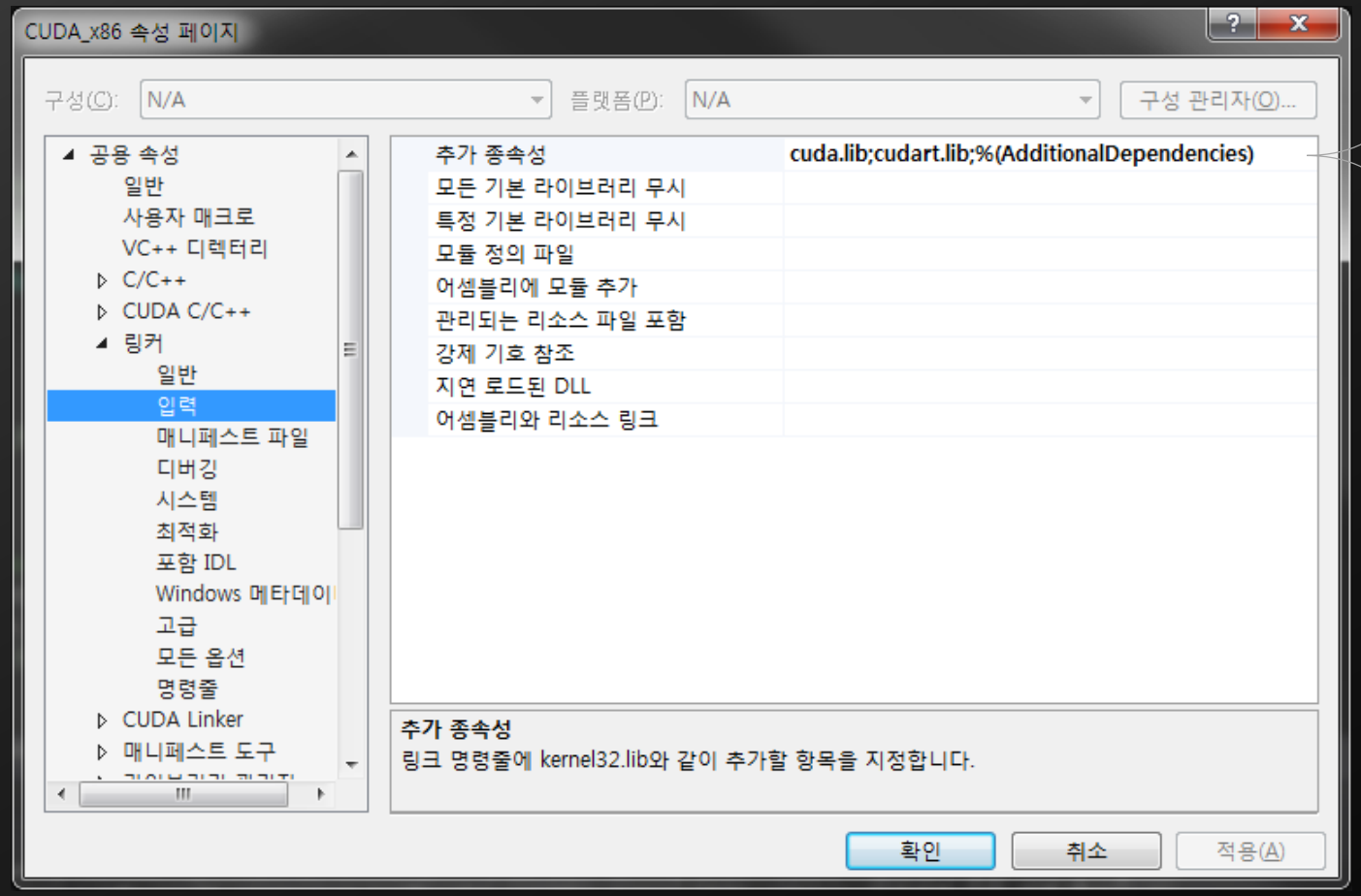


Image System Laboratory

Installing CUDA 7.0 with VS 2013

Visual Studio Settings



cublas.lib
cublas_device.lib
cuda.lib
cudadevrt.lib
cudart.lib
cudart_static.lib
cufft.lib
cufftw.lib
curand.lib
cusparse.lib
nppc.lib
nppi.lib
npps.lib
nvblas.lib
nvcuvid.lib
nvrtc.lib
OpenCL.lib

ISL Image System Laboratory

Installing CUDA 7.0 with VS 2013

Unsupported Features

- 32-bit x86 Linux Systems
- Red Hat Enterprise Linux 5 and CentOS5
- Tesla Architecture

- Certain CUDA Features on 32-bit and 32-on-64-bit Windows Systems
  - Running 32-bit applications on Tesla and Quadro products
  - Using the Thrust library form 32-bit applications
  - 32-bit version of the CUDA Toolkit scientific libraries, including cuBlLAS, cuSPARSE, cuFFT, cuRAND and NPP
  - 32-bit version of the CUDA samples

ISL Image System Laboratory

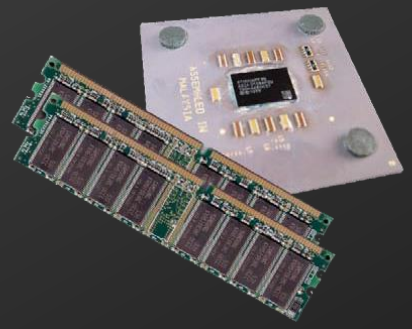
NVIDIA

CU

Basic of CUDA

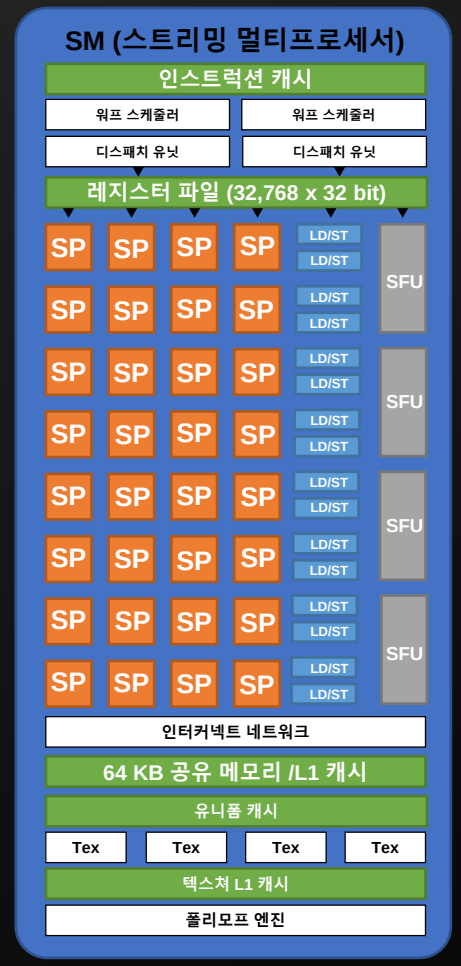
Terminology

Host : The CPU and its memory



Device : The GPU and its memory

Fermi Architecture



DA

ISL Image System Laboratory

Basic of CUDA

Directives

__global__	Host to Device Return type is void, Not allow Recursion
__device__	Device to Device
__host__	Host to Host

Graphic Card Memory Allocation

```
cudaError_t cudaMalloc(void** devPtr, size_t cout)

cudaError_t cudaFree(void* devPtr)

cudaError_t cudaMemcpy(void* dst, const void* src, size_t cout, cudaMemcpyKind kind)
```


ISL Image System Laboratory

Basic of CUDA

Thread, Block, Grid

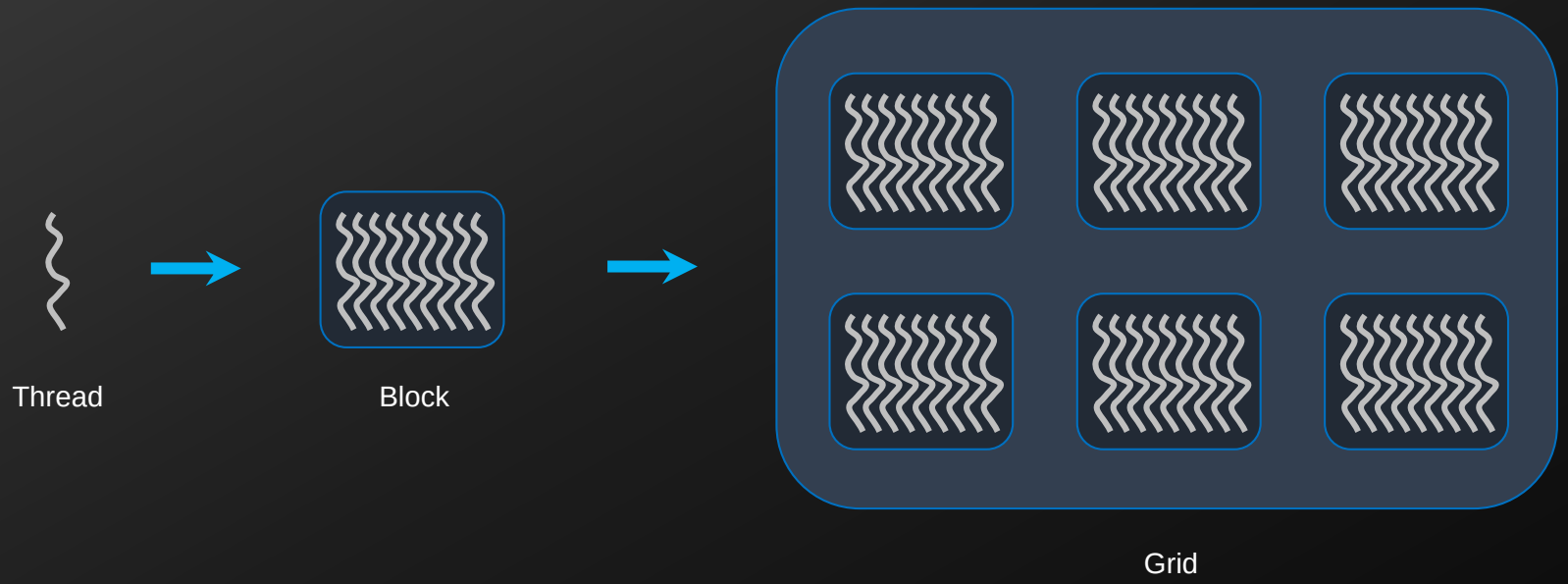


Image System Laboratory

Basic of CUDA

Kernel Function

```
__global__ void kernel << <Dg, Db, Ns, S >> > ();
```

Dg(Dimensions of the grid) : type dim3

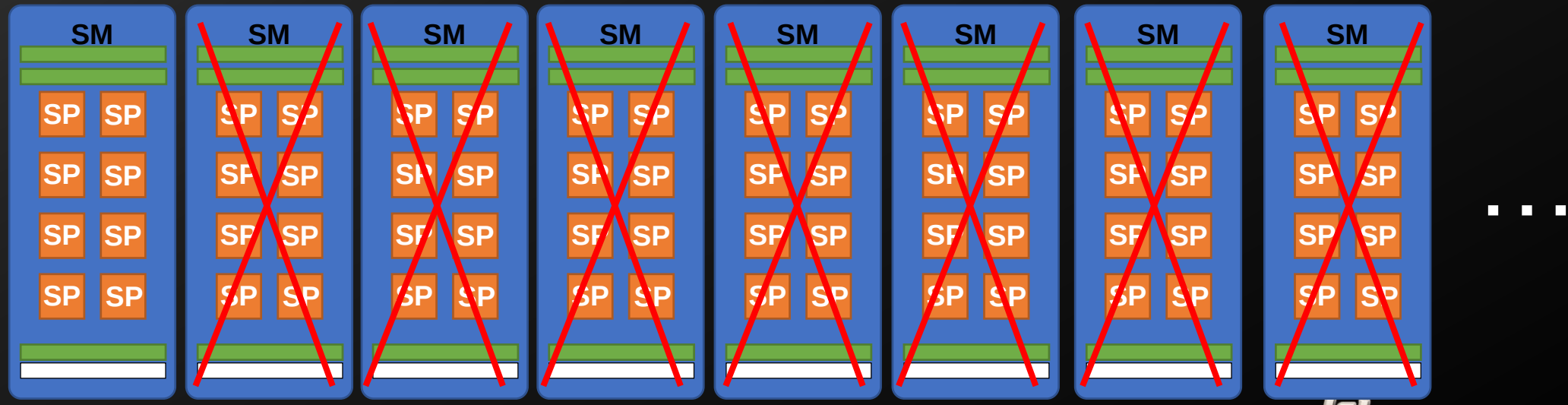
Db(Dimensions of the block) : type dim3

Ns(Number of bytes shared memory dynamically allocated /block) : type size_t

S(associated cudaStream_t)

ex) `__global__ void kernel << <1, 512 >> > (int a, int, b, int c);`

G80



ISL Image System Laboratory

Basic of CUDA

Kernel Function

```
__global__ void kernel << <Dg, Db, Ns, S >> > ();
```

Dg(Dimensions of the grid) : type dim3

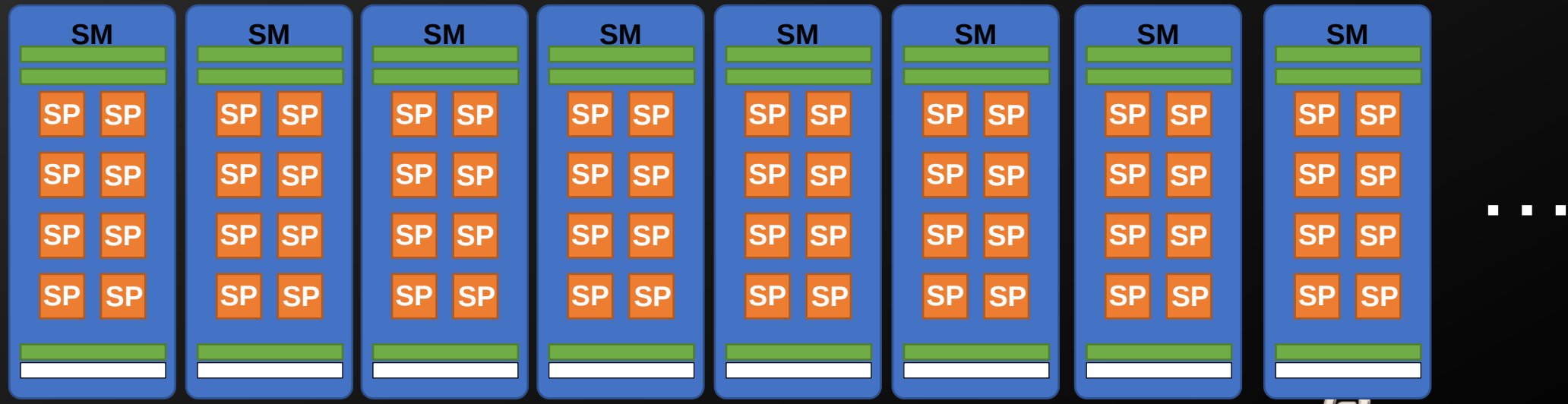
Db(Dimensions of the block) : type dim3

Ns(Number of bytes shared memory dynamically allocated /block) : type size_t

S(associated cudaStream_t)

ex) `__global__ void kernel << <8, 64 >> > (int a, int, b, int c);`

G80



ISL Image System Laboratory

Basic of CUDA

Example

```
__global__ void VectorAdd(int *a, int *b, int *c, int n)
{
        int i = threadIdx.x;
        if (i < n)
        {
                c[i] = a[i] + b[i];
        }
}
```

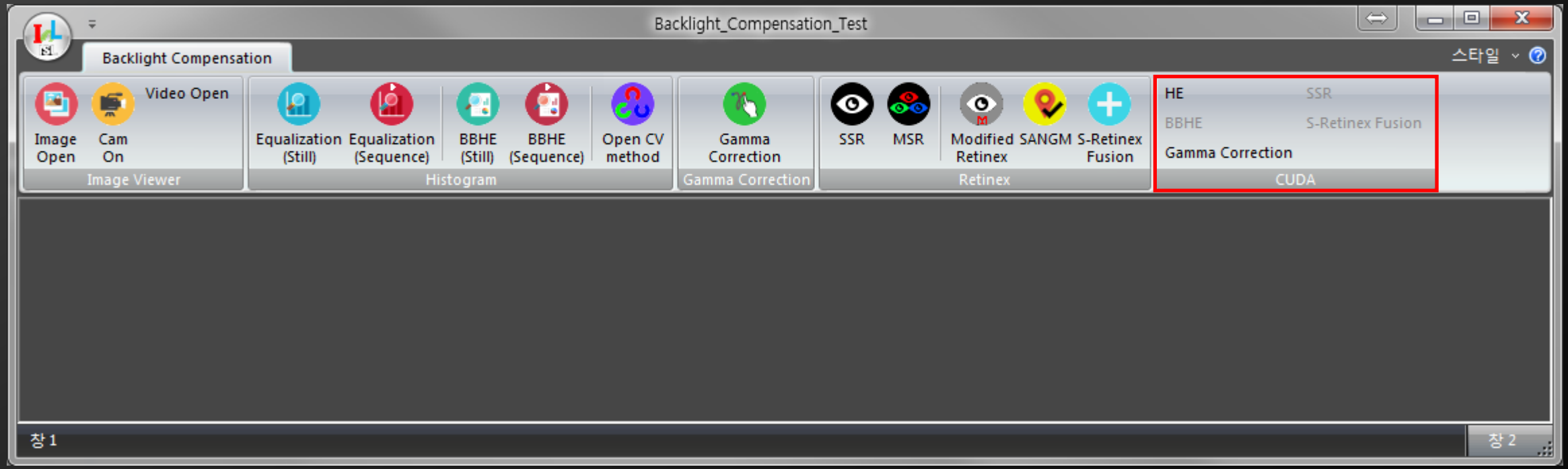
```
int main()
{
        int *a, *b, *c;
        int *d_a, *d_b, *d_c;

        a = (int *)malloc(SIZE*sizeof(int));
        b = (int *)malloc(SIZE*sizeof(int));
        c = (int *)malloc(SIZE*sizeof(int));

        cudaMalloc(&d_a, SIZE*sizeof(int));
        cudaMalloc(&d_b, SIZE*sizeof(int));
        cudaMalloc(&d_c, SIZE*sizeof(int));
        for (int i = 0; i < SIZE; ++i)
        {
                a[i] = i;
                b[i] = i;
                c[i] = 0;
        }

        cudaMemcpy(d_a, a, SIZE*sizeof(int), cudaMemcpyHostToDevice);
        cudaMemcpy(d_b, b, SIZE*sizeof(int), cudaMemcpyHostToDevice);
        cudaMemcpy(d_c, c, SIZE*sizeof(int), cudaMemcpyHostToDevice);

        VectorAdd <<< 1, SIZE >>>(d_a, d_b, d_c, SIZE);
        cudaMemcpy(c, d_c, SIZE*sizeof(int), cudaMemcpyDeviceToHost);
        for (int i = 0; i < 10; ++i)
        {
                printf("c[%d] = %d\n", i, c[i]);

        }

        free(a);
        free(b);
        free(c);

        cudaFree(d_a);
        cudaFree(d_b);
        cudaFree(d_c);
        return 0;

}
```

```
c[0] = 0
c[1] = 2
c[2] = 4
c[3] = 6
c[4] = 8
c[5] = 10
c[6] = 12
c[7] = 14
c[8] = 16
c[9] = 18
계속하려면 아무 키나 누르십시오 . . .
```


ISL Image System Laboratory

Applications

Backlight Compensation Test Program



ISL Image System Laboratory

Applications

Classes of BCT

- **CHistManager**
  - **Calc_Histogram** `void Calc_Histogram(Mat Image, float* fHist, int colorcode);`
  - **Calc_Hist_Eq** `void Calc_Hist_Eq(Mat Image, Mat& Output, float* fHist, float* Trans, int colorcode, int nStart, int nEnd);`
  - **Calc_Hist_Eq3** `void Calc_Hist_Eq3(Mat Image, Mat& Output, float* fHist, float* Trans, int colorcode, int nStart, int nEnd);`
  - **Hist_Visual** `void Hist_Visual(Mat Image, Mat& Output, int colorcode);`
- **CAvgStdGamma**
  - **Calc_Avg_Std** `void Calc_Avg_Std(Mat Image, float &average, float &sigma);`
  - **Gamma_Correction** `void Gamma_Correction(Mat Image, Mat& Output, float gamma);`

Image System Laboratory

Applications

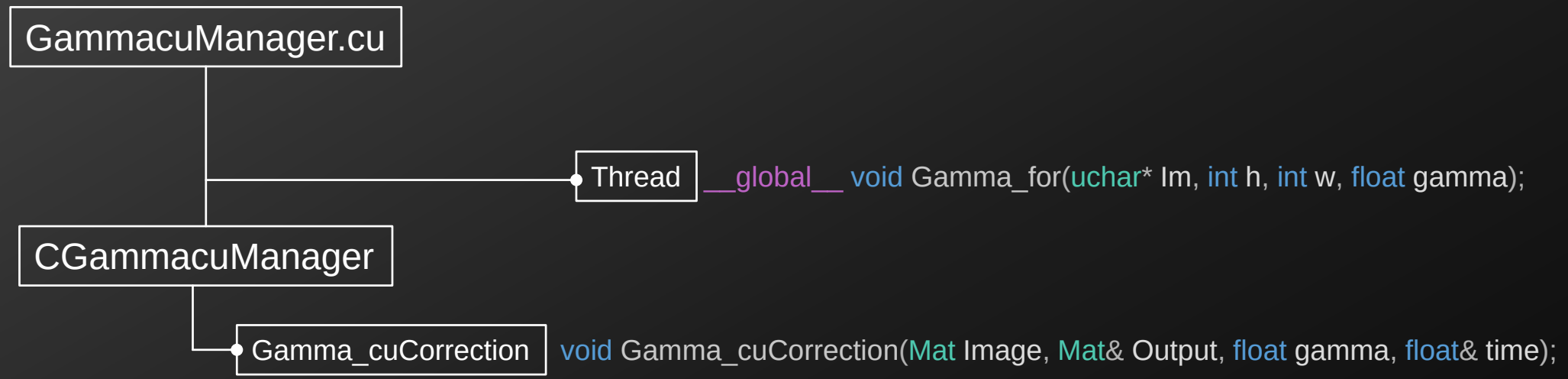
Classes of BCT

- CRetinexManager
  - Calc_SSR — `void Calc_SSR(Mat Image, Mat& Output);`
  - Calc_MSR — `void Calc_MSR(Mat Image, Mat& Output, float Rweight, float Gweight, float Bweight);`
  - Calc_Sretinex — `void Calc_Sretinex(Mat Image, Mat& Output, float a, float b, float t);`
  - Calc_SANGM — `void Calc_SANGM(Mat Image, Mat& Output);`
  - Calc_SRFA — `void Calc_SRFA(Mat Image, Mat& Output);`

ISL Image System Laboratory

Applications

Classes of BCT



Applications

Classes of BCT

- HistuManager.cu
  - Thread1 — `__global__ void cuHistCalc(uchar* Im, float* fHist, float h, float w);`
  - Thread2 — `__global__ void cuHistEq(uchar* Im, float* fHist, float* Trans, float h, float w, int nStart, int nEnd);`
- CHistcuManager
  - Calc_Hist_Eq — `void Calc_Hist_Eq(Mat Image, Mat& Output, float* fHist, float* Trans, int nStart, int nEnd);`
  - Calc_Histogram — `void Calc_Histogram(Mat Image, float* fHist);`

ISL Image System Laboratory

CU Applications

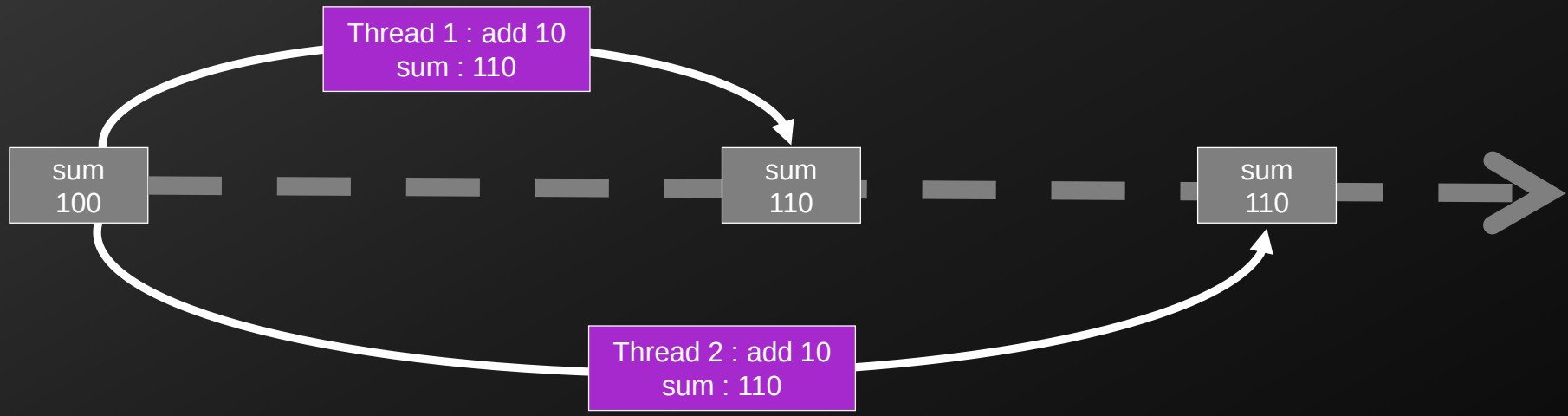
Example – Histogram

```
void CHistcuManager::Calc_Histogram(Mat Image, float* fHist)
{
		float h = Image.size().height;
		float w = Image.size().width;
		uchar *Im = Image.ptr<uchar>(0);

		uchar *d_Im;
		float *d_fHist;
		size_t SIZE = h*w;

		cudaMalloc((void**)&d_Im, SIZE*sizeof(uchar));
		cudaMalloc((void**)&d_fHist, (size_t)256 * sizeof(float));

		cudaMemcpy(d_Im, Im, SIZE*sizeof(uchar), cudaMemcpyHostToDevice);
		cudaMemcpy(d_fHist, fHist, (size_t)256 * sizeof(float), cudaMemcpyHostToDevice);
		cuHistCalc <<< 65535, 512 >>> (d_Im, d_fHist, h, w);

		cudaMemcpy(Im, d_Im, SIZE*sizeof(uchar), cudaMemcpyDeviceToHost);
		cudaMemcpy(fHist, d_fHist, (size_t)256 * sizeof(float), cudaMemcpyDeviceToHost);
		cudaFree(d_Im);
		cudaFree(d_fHist);
}
```

```
__global__ void cuHistCalc(uchar* Im, float* fHist, float h, float w)
{
		int i = blockDim.x*blockIdx.x + threadIdx.x;

		if (i < h*w)
		{
			atomicAdd(&(fHist[Im[i]]), 1);
		}
}
```

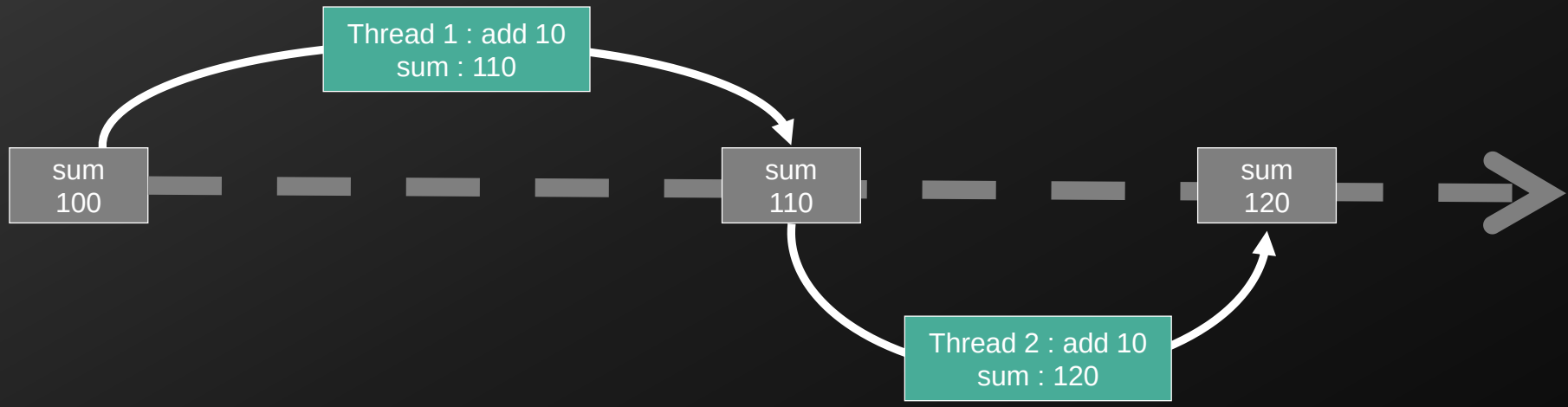
Applications

Race Condition



Applications

Race Condition



Applications

Atomic Operation

Arithmetic

atomicAdd()	computes (old + val), and stores the result back to memory at the same address
atomicSub()	computes (old - val), and stores the result back to memory at the same address
atomicExch()	computes swap(old, val), and stores the result back to memory at the same address
atomicMin()	computes Min(old, val), and stores the result back to memory at the same address
atomicMax()	computes Max(old, val), and stores the result back to memory at the same address
atomicInc()	computes ((old >= val) ? val : (old-1)), and stores the result back to memory at the same address
atomicDec()	computes (((old == 0) | (old > val)) ? val : (old-1)), and stores the result back to memory at the same address
atomicCAS()	computes (old == compare ? val : old), and stores the result back to memory at the same address

Image System Laboratory

Applications

Atomic Operation

Bit

atomicAnd()	(old & val), and stores the result back to memory at the same address
atomicOr()	(old | val), and stores the result back to memory at the same address
atomicXor()	(old ^ val), and stores the result back to memory at the same address

CU Applications

Example – Gamma Correction

```
void CGammacuManager::Gamma_cuCorrection(Mat Image, Mat& Output, float gamma)
{
		float elapsed_time_ms = 0.0f;
		cudaEvent_t start, stop;

		cudaEventCreate(&start);
		cudaEventCreate(&stop);
		cudaEventRecord(start, 0);

		cvtColor(Image, Image, CV_RGB2YCrCb);		//YCrCb
		split(Image, m_vecChannel);

		int h = Image.size().height;
		int w = Image.size().width;
		uchar* Im = m_vecChannel[0].ptr<uchar>(0);
		uchar* dev_Im;
		cudaMalloc(&dev_Im, h*w*sizeof(uchar));
		cudaMemcpy(dev_Im, Im, h*w*sizeof(uchar), cudaMemcpyHostToDevice);


		Gamma_for <<<65535, 512>>> (dev_Im, h, w, gamma);

		cudaMemcpy(Im, dev_Im, h*w*sizeof(uchar), cudaMemcpyDeviceToHost);
		merge(m_vecChannel, Output);
		cvtColor(Output, Output, CV_YCrCb2RGB);
		cudaFree(dev_Im);
}
```

```
__global__ void Gamma_for(uchar* Im, int h, int w, float gamma)
{
		const int idx = blockIdx.x * blockDim.x + threadIdx.x;

		float inv_gamma = 1.f / gamma;
		if ( (idx < w*h) )
		{
			Im[idx] = Clipping(pow((Im[idx] / 255.f), inv_gamma) * 255.f + 0.5f);
		}

}
```


DA
ISL Image System Laboratory

Applications

Gamma Correction Results

기준영상
(1920 x 1080)

Trial	Entire (unit : s) Normal Function	CUDA Function
1	0.102687	0.014429
2	0.101979	0.015645
3	0.102959	0.015179
4	0.109468	0.014846
5	0.104883	0.014513
6	0.102613	0.014487
7	0.101135	0.01484
8	0.101176	0.014149
9	0.104109	0.014452
10	0.101946	0.01418
11	0.102001	0.015072
12	0.101744	0.015184
13	0.102265	0.0143
14	0.100986	0.014198
15	0.101596	0.014725
16	0.101665	0.014718
17	0.102031	0.014338
18	0.101583	0.01496
19	0.102474	0.014493
20	0.101204	0.0014437
21	0.100953	0.014219
22	0.101229	0.014535
23	0.101547	0.014537
24	0.10177	0.014708
25	0.10201	0.014421
26	0.101805	0.014247
27	0.101574	0.01591
28	0.101128	0.014228
29	0.103729	0.014163
30	0.100847	0.01403
Average	0.102236533	0.014171657

Q & A

Thank you for your attention