Name: Nirmal Kumar Sedhumadhavan

ECE 786 Programming Assignment 1

CUDA Programming and GPGPU Simulator

Part- A: CUDA Program

CUDA Program for Single Quantum Bit Gate Simulation with different memory management methods.

CUDA Kernel Function:

```
__global__ void matrix_multiply(const float *input, float *output, const float
*Umatrix, int size, int qbit)
{
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int mask = 1 << qbit;
  int index = i ^ mask;
  if (i <= (size - mask))
  {
    if((i/mask) % 2 != 1)
    {
     output[i] = Umatrix[0] * input[i] + Umatrix[1] * input[index];
     output[index] = Umatrix[2] * input[i] + Umatrix[3] * input[index];
    }
  }
}
```

Explanation:

The implemented kernel function matrix_multiply takes input vector (float *input), unitary matrix vector (float *Umatrix), qbit position (int qbit), size of the input vector(int size), output vector (float *output) as arguments.

int i = blockIdx.x * blockDim.x + threadIdx.x;

Inside the function, index for the threads is calculated based on the thread hierarchy. Each thread processes 2 elements in the output vector.

mask = 1 << qbit; //Calculates the value of 2 to the power of qbit(2^{qbit})
index = i ^ mask; //Calcultes the position of the 2^{nd} element
if (i <= (size - mask))
{
if((i/mask) % 2 != 1) // Check if the 2^{nd} element is already calculated
{
output[i] = Umatrix[0] * input[i] + Umatrix[1] * input[index]; //Update the 1^{st} element
output[index] = Umatrix[2] * input[i] + Umatrix[3] * input[index];//Update the 2^{nd} element
}
}

The above Kernel function is used for two different memory management methods which are discrete/separate CPU and GPU memories and Unified Virtual Memory with 256 threads in each thread block**.**

Timing Report:

Input (Taken from more examples of Program Assignment 1)	Time taken with Separate Memories (microseconds)	Time taken with Unified Virtual Memory (microseconds)
4-qubit quantum output state after applying a single-qubit gate on qubit 1	26	29
7-qubit quantum output state after applying a single-qubit gate on qubit 4	31	30

Part-B: GPGPU Sim

Ran the above CUDA code (for separate CPU and GPU memory management method) with the GPGPU sim and observed the statistics.
quamsimV1.cu is run with input taken from more examples of Program Assignment 1 (7-qubit quantum output state after applying a single-qubit gate on qubit 4) and below are the statistics.

1) IPC of the program
gpu_sim_cycle = 5779
gpu_sim_insn = 5990
gpu_ipc = gpu_sim_insn / gpu_sim_cycle
gpu_ipc = 1.0365

2) Data Cache miss rate
L1D_total_cache_accesses = 64
L1D_total_cache_misses = 40
L1D_total_cache_miss_rate = L1D_total_cache_misses / L1D_total_cache_accesses
L1D_total_cache_miss_rate = 0.625