Korea Advanced Institute of Science and Technology

School of Electrical Engineering

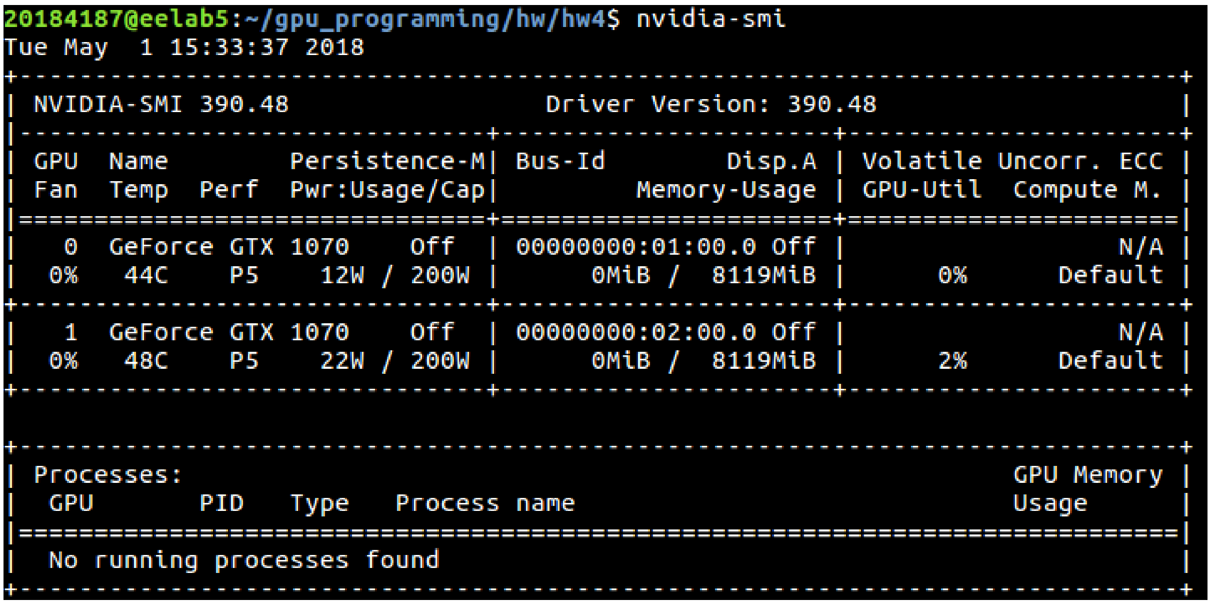
EE817 GPU Programming and Its Application Spring 2018

Student's Name: Dinh Vu

Student's ID: 20184187

Homework 4

The computer, used in my homework 4, contains NVIDIA GeForce GT 1070 based on Pascal GP104 architecture.



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvidia-smi
Tue May  1 15:33:37 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.48                 Driver Version: 390.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1070    Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   44C    P5    12W / 200W |      0MiB /  8119MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1070    Off  | 00000000:02:00.0 Off |                  N/A |
|  0%   48C    P5    22W / 200W |      0MiB /  8119MiB |      2%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

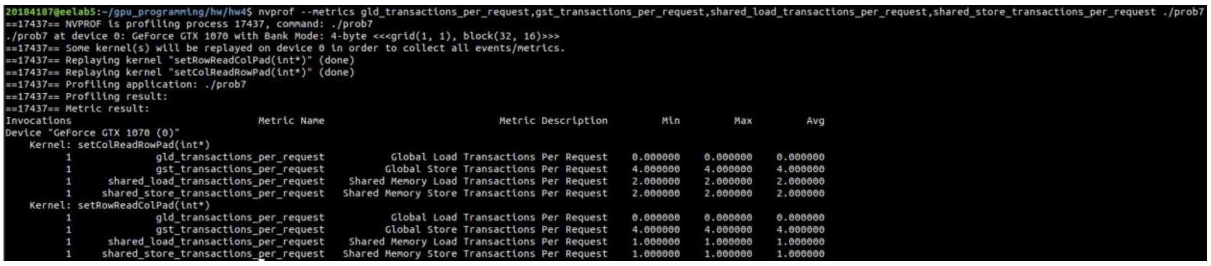
Figure 1. Graphic card information

1. Problem 7

The source code for problem 7 is prob7.cu file and the results are shown in Figure 1.1 and Table 1.1 below with thread block 32×16 using 4-byte access mode. There is not any load transaction from global memory, so Global Load Transactions Per Request equals to 0.

By padding IPAD = 2 elements in each row of the tile in the kernel setRowReadColPad, the even-column elements and the odd-column elements are distributed among even banks and odd banks, respectively. So, both writing by row-major and reading by column-major are conflict-free. It means the number of transactions for a request from shared memory is 1.00.

However, with the same padding, the kernel setColReadRowPad has a 2-way bank conflict in both reading and writing operation. The mapping from words to banks is illustrated at Figure 1.2 and Figure 1.3.

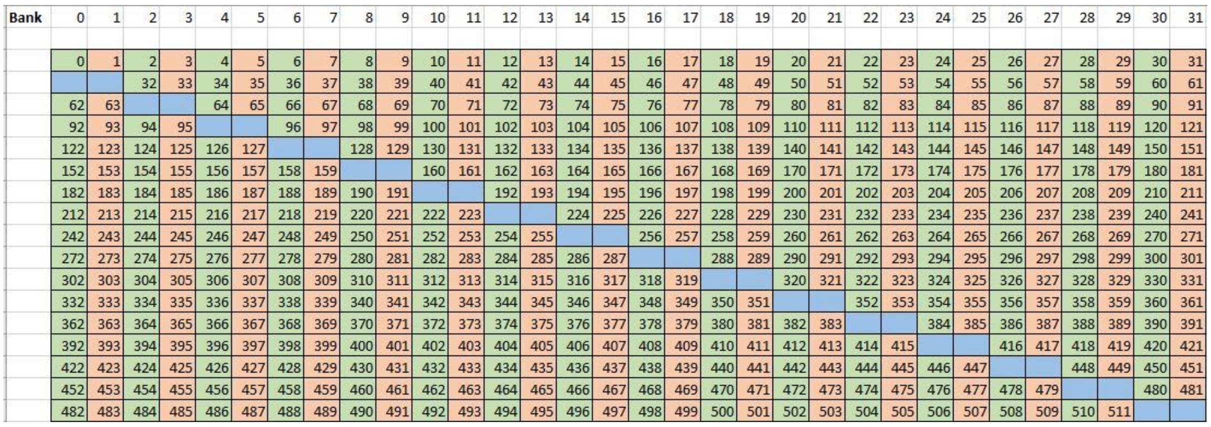


```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof --metrics gld_transactions_per_request,gst_transactions_per_request,shared_load_transactions_per_request,shared_store_transactions_per_request ./prob7
==17437== NVPROF is profiling process 17437, command: ./prob7
./prob7 at device 0: GeForce GTX 1070 with Bank Mode: 4-byte <<<grid(1, 1), block(32, 16)>>>
==17437== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics.
==17437== Replaying kernel "setRowReadColPad(int*)" (done)
==17437== Replaying kernel "setColReadRowPad(int*)" (done)
==17437== Profiling application: ./prob7
==17437== Profiling result:
==17437== Metric result:
Invocations                               Metric Name                        Metric Description         Min         Max         Avg
Device "GeForce GTX 1070 (0)"
    Kernel: setColReadRowPad(int*)
          1                  gld_transactions_per_request          Global Load Transactions Per Request    0.000000    0.000000    0.000000
          1                  gst_transactions_per_request         Global Store Transactions Per Request    4.000000    4.000000    4.000000
          1          shared_load_transactions_per_request      Shared Memory Load Transactions Per Request    2.000000    2.000000    2.000000
          1         shared_store_transactions_per_request     Shared Memory Store Transactions Per Request    2.000000    2.000000    2.000000
    Kernel: setRowReadColPad(int*)
          1                  gld_transactions_per_request          Global Load Transactions Per Request    0.000000    0.000000    0.000000
          1                  gst_transactions_per_request         Global Store Transactions Per Request    4.000000    4.000000    4.000000
          1          shared_load_transactions_per_request      Shared Memory Load Transactions Per Request    1.000000    1.000000    1.000000
          1         shared_store_transactions_per_request     Shared Memory Store Transactions Per Request    1.000000    1.000000    1.000000
```

Figure 1.1. Results of problem 7

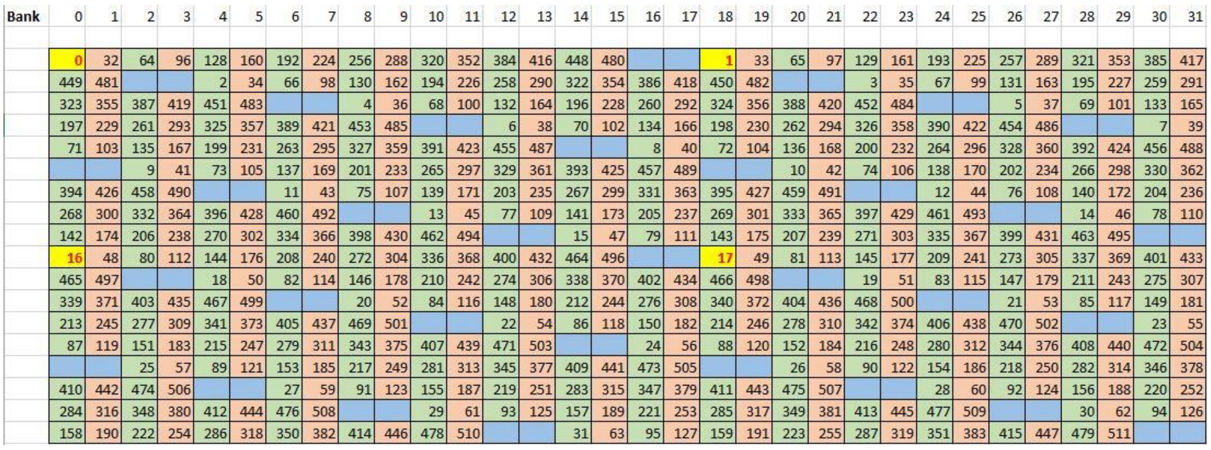
Table 1.1. Transaction Metrics

Kernel	setColReadRowPad	setRowReadColPad
Global Load Transactions Per Request	0.00	0.00
Global Store Transactions Per Request	4.00	4.00
Shared Memory Load Transactions Per Request	2.00	1.00
Shared Memory Store Transactions Per Request	2.00	1.00



Bank	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
			32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
	62	63			64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91
	92	93	94	95			96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121
	122	123	124	125	126	127			128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151
	152	153	154	155	156	157	158	159			160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181
	182	183	184	185	186	187	188	189	190	191			192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211
	212	213	214	215	216	217	218	219	220	221	222	223			224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241
	242	243	244	245	246	247	248	249	250	251	252	253	254	255			256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271
	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287			288	289	290	291	292	293	294	295	296	297	298	299	300	301
	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319			320	321	322	323	324	325	326	327	328	329	330	331
	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351			352	353	354	355	356	357	358	359	360	361
	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383			384	385	386	387	388	389	390	391
	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415			416	417	418	419	420	421
	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447			448	449	450	451
	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479			480	481
	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511		

Figure 1.2. setRowReadColPad with bank conflict-free

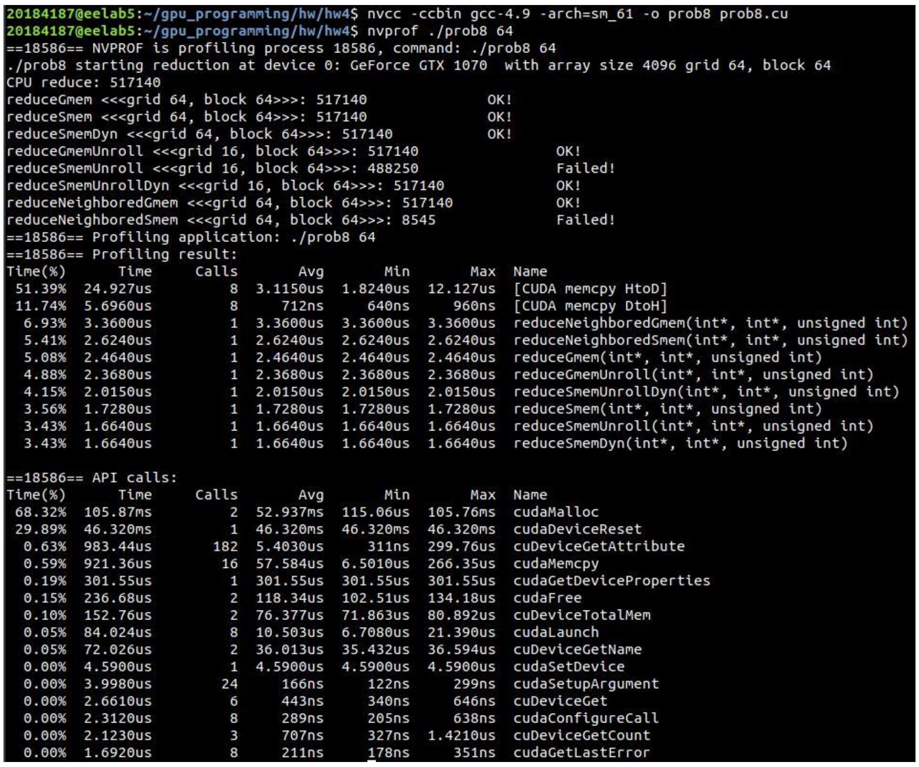


Bank	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	32	64	96	128	160	192	224	256	288	320	352	384	416	448	480			1	33	65	97	129	161	193	225	257	289	321	353	385	417
	449	481			2	34	66	98	130	162	194	226	258	290	322	354	386	418	450	482			3	35	67	99	131	163	195	227	259	291
	323	355	387	419	451	483			4	36	68	100	132	164	196	228	260	292	324	356	388	420	452	484			5	37	69	101	133	165
	197	229	261	293	325	357	389	421	453	485			6	38	70	102	134	166	198	230	262	294	326	358	390	422	454	486			7	39
	71	103	135	167	199	231	263	295	327	359	391	423	455	487			8	40	72	104	136	168	200	232	264	296	328	360	392	424	456	488
			9	41	73	105	137	169	201	233	265	297	329	361	393	425	457	489			10	42	74	106	138	170	202	234	266	298	330	362
	394	426	458	490			11	43	75	107	139	171	203	235	267	299	331	363	395	427	459	491			12	44	76	108	140	172	204	236
	268	300	332	364	396	428	460	492			13	45	77	109	141	173	205	237	269	301	333	365	397	429	461	493			14	46	78	110
	142	174	206	238	270	302	334	366	398	430	462	494			15	47	79	111	143	175	207	239	271	303	335	367	399	431	463	495		
	16	48	80	112	144	176	208	240	272	304	336	368	400	432	464	496			17	49	81	113	145	177	209	241	273	305	337	369	401	433
	465	497			18	50	82	114	146	178	210	242	274	306	338	370	402	434	466	498			19	51	83	115	147	179	211	243	275	307
	339	371	403	435	467	499			20	52	84	116	148	180	212	244	276	308	340	372	404	436	468	500			21	53	85	117	149	181
	213	245	277	309	341	373	405	437	469	501			22	54	86	118	150	182	214	246	278	310	342	374	406	438	470	502			23	55
	87	119	151	183	215	247	279	311	343	375	407	439	471	503			24	56	88	120	152	184	216	248	280	312	344	376	408	440	472	504
			25	57	89	121	153	185	217	249	281	313	345	377	409	441	473	505			26	58	90	122	154	186	218	250	282	314	346	378
	410	442	474	506			27	59	91	123	155	187	219	251	283	315	347	379	411	443	475	507			28	60	92	124	156	188	220	252
	284	316	348	380	412	444	476	508			29	61	93	125	157	189	221	253	285	317	349	381	413	445	477	509			30	62	94	126
	158	190	222	254	286	318	350	382	414	446	478	510			31	63	95	127	159	191	223	255	287	319	351	383	415	447	479	511		

Figure 1.3. setColReadRowPad with bank conflict at yellow elements

2. Problem 8

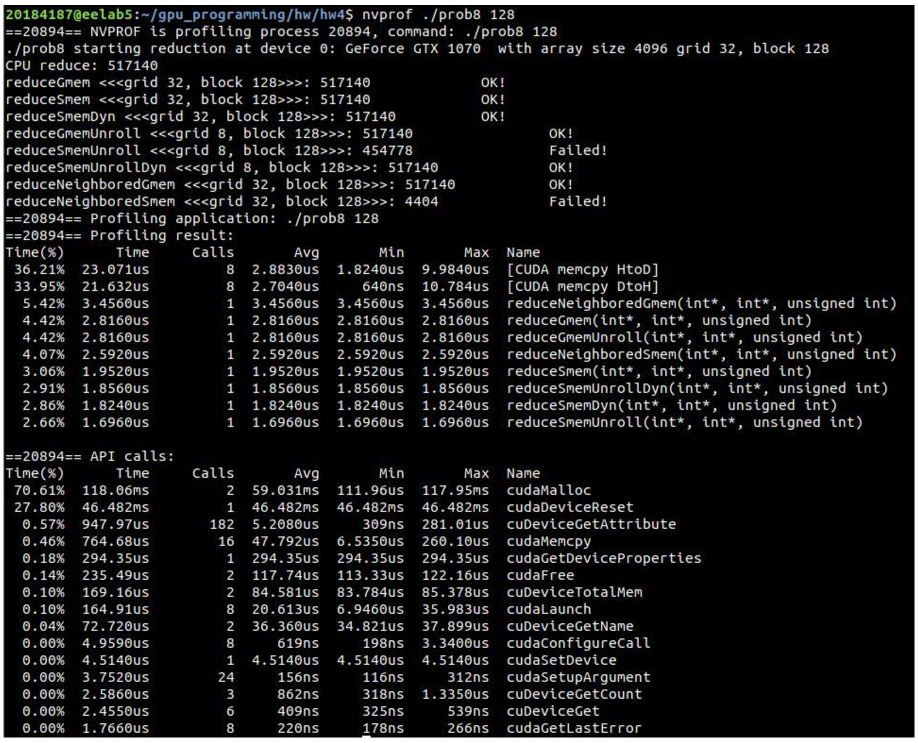
The source code for problem 8 is prob8.cu file and its results are display in Figure 2.1, Figure 2.2, Figure 2.3 and Figure 2.4 corresponding to block sizes of 64, 128, 512 and 1024. The summary of the results is typed in Table 2.1.



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvcc -ccbin gcc-4.9 -arch=sm_61 -o prob8 prob8.cu
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob8 64
==18586== NVPROF is profiling process 18586, command: ./prob8 64
./prob8 starting reduction at device 0: GeForce GTX 1070  with array size 4096 grid 64, block 64
CPU reduce: 517140
reduceGmem <<<grid 64, block 64>>>: 517140              OK!
reduceSmem <<<grid 64, block 64>>>: 517140              OK!
reduceSmemDyn <<<grid 64, block 64>>>: 517140           OK!
reduceGmemUnroll <<<grid 16, block 64>>>: 517140                OK!
reduceSmemUnroll <<<grid 16, block 64>>>: 488250                Failed!
reduceSmemUnrollDyn <<<grid 16, block 64>>>: 517140             OK!
reduceNeighboredGmem <<<grid 64, block 64>>>: 517140            OK!
reduceNeighboredSmem <<<grid 64, block 64>>>: 8545              Failed!
==18586== Profiling application: ./prob8 64
==18586== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 51.39%  24.927us         8  3.1150us  1.8240us  12.127us  [CUDA memcpy HtoD]
 11.74%  5.6960us         8     712ns     640ns     960ns  [CUDA memcpy DtoH]
  6.93%  3.3600us         1  3.3600us  3.3600us  3.3600us  reduceNeighboredGmem(int*, int*, unsigned int)
  5.41%  2.6240us         1  2.6240us  2.6240us  2.6240us  reduceNeighboredSmem(int*, int*, unsigned int)
  5.08%  2.4640us         1  2.4640us  2.4640us  2.4640us  reduceGmem(int*, int*, unsigned int)
  4.88%  2.3680us         1  2.3680us  2.3680us  2.3680us  reduceGmemUnroll(int*, int*, unsigned int)
  4.15%  2.0150us         1  2.0150us  2.0150us  2.0150us  reduceSmemUnrollDyn(int*, int*, unsigned int)
  3.56%  1.7280us         1  1.7280us  1.7280us  1.7280us  reduceSmem(int*, int*, unsigned int)
  3.43%  1.6640us         1  1.6640us  1.6640us  1.6640us  reduceSmemUnroll(int*, int*, unsigned int)
  3.43%  1.6640us         1  1.6640us  1.6640us  1.6640us  reduceSmemDyn(int*, int*, unsigned int)

==18586== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 68.32%  105.87ms         2  52.937ms  115.06us  105.76ms  cudaMalloc
 29.89%  46.320ms         1  46.320ms  46.320ms  46.320ms  cudaDeviceReset
  0.63%  983.44us       182  5.4030us     311ns  299.76us  cuDeviceGetAttribute
  0.59%  921.36us        16  57.584us  6.5010us  266.35us  cudaMemcpy
  0.19%  301.55us         1  301.55us  301.55us  301.55us  cudaGetDeviceProperties
  0.15%  236.68us         2  118.34us  102.51us  134.18us  cudaFree
  0.10%  152.76us         2  76.377us  71.863us  80.892us  cuDeviceTotalMem
  0.05%  84.024us         8  10.503us  6.7080us  21.390us  cudaLaunch
  0.05%  72.026us         2  36.013us  35.432us  36.594us  cuDeviceGetName
  0.00%  4.5900us         1  4.5900us  4.5900us  4.5900us  cudaSetDevice
  0.00%  3.9980us        24     166ns     122ns     299ns  cudaSetupArgument
  0.00%  2.6610us         6     443ns     340ns     646ns  cuDeviceGet
  0.00%  2.3120us         8     289ns     205ns     638ns  cudaConfigureCall
  0.00%  2.1230us         3     707ns     327ns  1.4210us  cuDeviceGetCount
  0.00%  1.6920us         8     211ns     178ns     351ns  cudaGetLastError
```

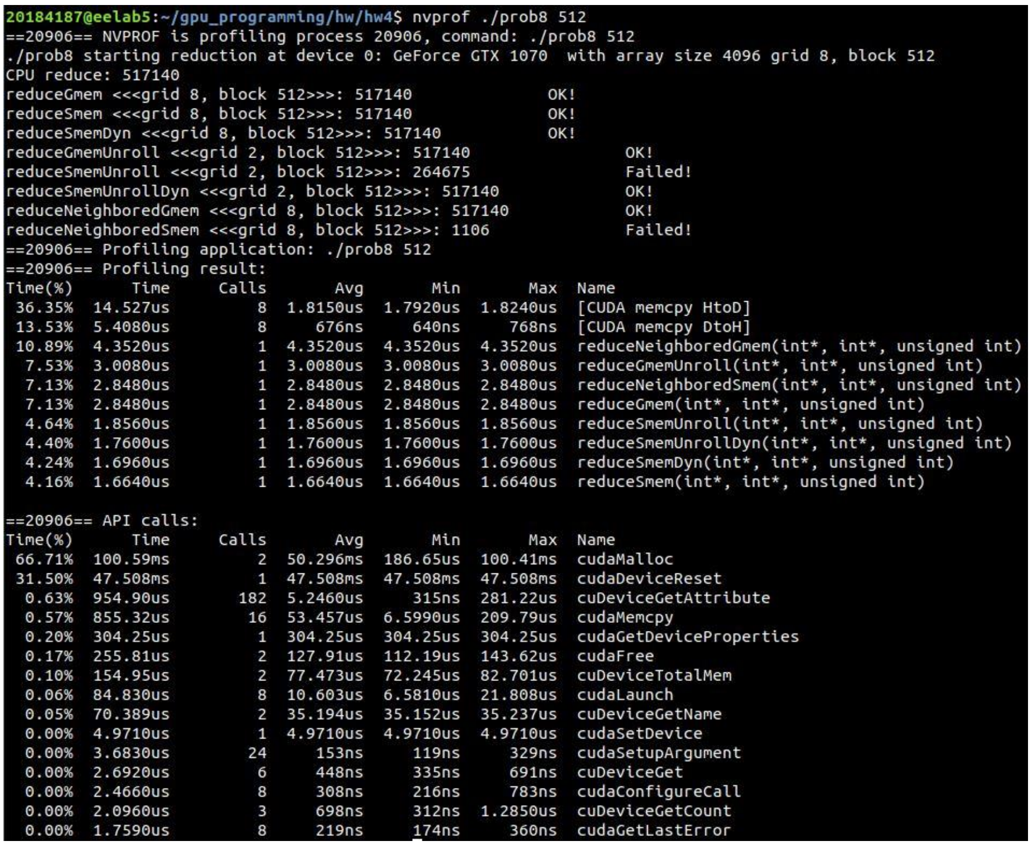
Figure 2.1. Test block size of 64



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob8 128
==20894== NVPROF is profiling process 20894, command: ./prob8 128
./prob8 starting reduction at device 0: GeForce GTX 1070  with array size 4096 grid 32, block 128
CPU reduce: 517140
reduceGmem <<<grid 32, block 128>>>: 517140             OK!
reduceSmem <<<grid 32, block 128>>>: 517140             OK!
reduceSmemDyn <<<grid 32, block 128>>>: 517140          OK!
reduceGmemUnroll <<<grid 8, block 128>>>: 517140                OK!
reduceSmemUnroll <<<grid 8, block 128>>>: 454778                Failed!
reduceSmemUnrollDyn <<<grid 8, block 128>>>: 517140             OK!
reduceNeighboredGmem <<<grid 32, block 128>>>: 517140           OK!
reduceNeighboredSmem <<<grid 32, block 128>>>: 4404             Failed!
==20894== Profiling application: ./prob8 128
==20894== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 36.21%  23.071us         8  2.8830us  1.8240us  9.9840us  [CUDA memcpy HtoD]
 33.95%  21.632us         8  2.7040us     640ns  10.784us  [CUDA memcpy DtoH]
  5.42%  3.4560us         1  3.4560us  3.4560us  3.4560us  reduceNeighboredGmem(int*, int*, unsigned int)
  4.42%  2.8160us         1  2.8160us  2.8160us  2.8160us  reduceGmem(int*, int*, unsigned int)
  4.42%  2.8160us         1  2.8160us  2.8160us  2.8160us  reduceGmemUnroll(int*, int*, unsigned int)
  4.07%  2.5920us         1  2.5920us  2.5920us  2.5920us  reduceNeighboredSmem(int*, int*, unsigned int)
  3.06%  1.9520us         1  1.9520us  1.9520us  1.9520us  reduceSmem(int*, int*, unsigned int)
  2.91%  1.8560us         1  1.8560us  1.8560us  1.8560us  reduceSmemUnrollDyn(int*, int*, unsigned int)
  2.86%  1.8240us         1  1.8240us  1.8240us  1.8240us  reduceSmemDyn(int*, int*, unsigned int)
  2.66%  1.6960us         1  1.6960us  1.6960us  1.6960us  reduceSmemUnroll(int*, int*, unsigned int)

==20894== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 70.61%  118.06ms         2  59.031ms  111.96us  117.95ms  cudaMalloc
 27.80%  46.482ms         1  46.482ms  46.482ms  46.482ms  cudaDeviceReset
  0.57%  947.97us       182  5.2080us     309ns  281.01us  cuDeviceGetAttribute
  0.46%  764.68us        16  47.792us  6.5350us  260.10us  cudaMemcpy
  0.18%  294.35us         1  294.35us  294.35us  294.35us  cudaGetDeviceProperties
  0.14%  235.49us         2  117.74us  113.33us  122.16us  cudaFree
  0.10%  169.16us         2  84.581us  83.784us  85.378us  cuDeviceTotalMem
  0.10%  164.91us         8  20.613us  6.9460us  35.983us  cudaLaunch
  0.04%  72.720us         2  36.360us  34.821us  37.899us  cuDeviceGetName
  0.00%  4.9590us         8     619ns     198ns  3.3400us  cudaConfigureCall
  0.00%  4.5140us         1  4.5140us  4.5140us  4.5140us  cudaSetDevice
  0.00%  3.7520us        24     156ns     116ns     312ns  cudaSetupArgument
  0.00%  2.5860us         3     862ns     318ns  1.3350us  cuDeviceGetCount
  0.00%  2.4550us         6     409ns     325ns     539ns  cuDeviceGet
  0.00%  1.7660us         8     220ns     178ns     266ns  cudaGetLastError
```

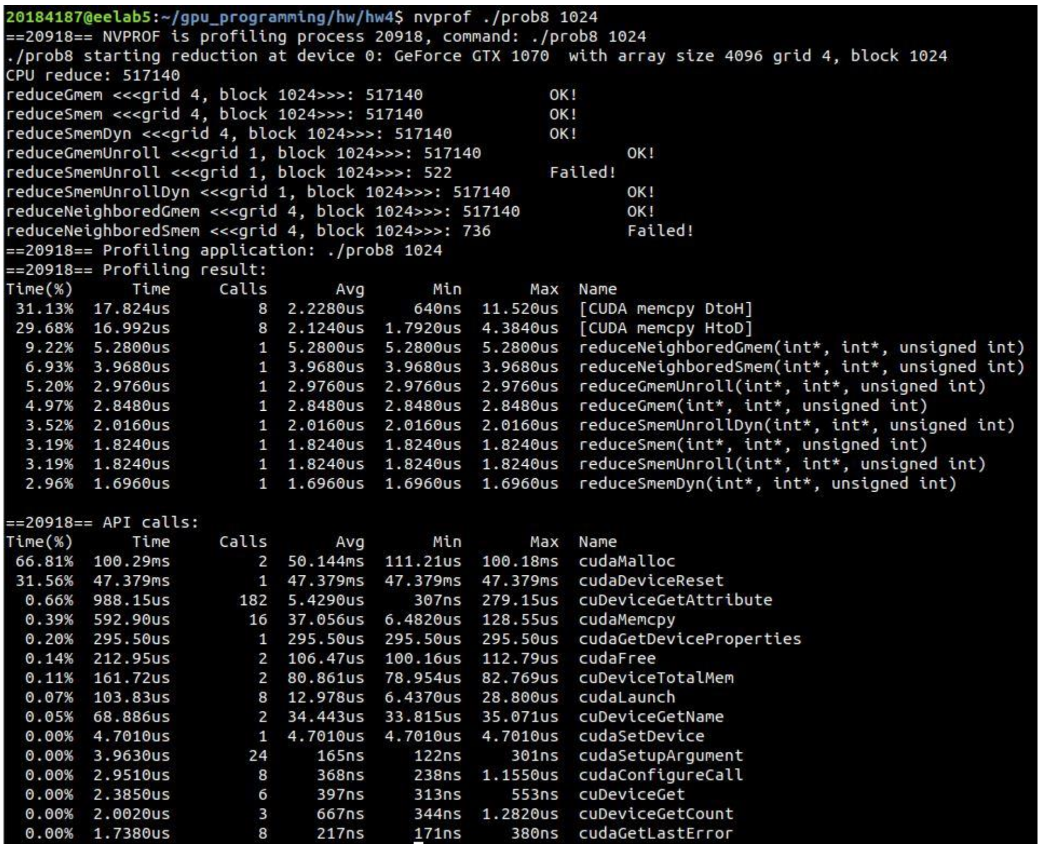
Figure 2.2. Test block size of 128



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob8 512
==20906== NVPROF is profiling process 20906, command: ./prob8 512
./prob8 starting reduction at device 0: GeForce GTX 1070  with array size 4096 grid 8, block 512
CPU reduce: 517140
reduceGmem <<<grid 8, block 512>>>: 517140              OK!
reduceSmem <<<grid 8, block 512>>>: 517140              OK!
reduceSmemDyn <<<grid 8, block 512>>>: 517140           OK!
reduceGmemUnroll <<<grid 2, block 512>>>: 517140                OK!
reduceSmemUnroll <<<grid 2, block 512>>>: 264675                Failed!
reduceSmemUnrollDyn <<<grid 2, block 512>>>: 517140             OK!
reduceNeighboredGmem <<<grid 8, block 512>>>: 517140            OK!
reduceNeighboredSmem <<<grid 8, block 512>>>: 1106              Failed!
==20906== Profiling application: ./prob8 512
==20906== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 36.35%  14.527us         8  1.8150us  1.7920us  1.8240us  [CUDA memcpy HtoD]
 13.53%  5.4080us         8     676ns     640ns     768ns  [CUDA memcpy DtoH]
 10.89%  4.3520us         1  4.3520us  4.3520us  4.3520us  reduceNeighboredGmem(int*, int*, unsigned int)
  7.53%  3.0080us         1  3.0080us  3.0080us  3.0080us  reduceGmemUnroll(int*, int*, unsigned int)
  7.13%  2.8480us         1  2.8480us  2.8480us  2.8480us  reduceNeighboredSmem(int*, int*, unsigned int)
  7.13%  2.8480us         1  2.8480us  2.8480us  2.8480us  reduceGmem(int*, int*, unsigned int)
  4.64%  1.8560us         1  1.8560us  1.8560us  1.8560us  reduceSmemUnroll(int*, int*, unsigned int)
  4.40%  1.7600us         1  1.7600us  1.7600us  1.7600us  reduceSmemUnrollDyn(int*, int*, unsigned int)
  4.24%  1.6960us         1  1.6960us  1.6960us  1.6960us  reduceSmemDyn(int*, int*, unsigned int)
  4.16%  1.6640us         1  1.6640us  1.6640us  1.6640us  reduceSmem(int*, int*, unsigned int)

==20906== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 66.71%  100.59ms         2  50.296ms  186.65us  100.41ms  cudaMalloc
 31.50%  47.508ms         1  47.508ms  47.508ms  47.508ms  cudaDeviceReset
  0.63%  954.90us       182  5.2460us     315ns  281.22us  cuDeviceGetAttribute
  0.57%  855.32us        16  53.457us  6.5990us  209.79us  cudaMemcpy
  0.20%  304.25us         1  304.25us  304.25us  304.25us  cudaGetDeviceProperties
  0.17%  255.81us         2  127.91us  112.19us  143.62us  cudaFree
  0.10%  154.95us         2  77.473us  72.245us  82.701us  cuDeviceTotalMem
  0.06%  84.830us         8  10.603us  6.5810us  21.808us  cudaLaunch
  0.05%  70.389us         2  35.194us  35.152us  35.237us  cuDeviceGetName
  0.00%  4.9710us         1  4.9710us  4.9710us  4.9710us  cudaSetDevice
  0.00%  3.6830us        24     153ns     119ns     329ns  cudaSetupArgument
  0.00%  2.6920us         6     448ns     335ns     691ns  cuDeviceGet
  0.00%  2.4660us         8     308ns     216ns     783ns  cudaConfigureCall
  0.00%  2.0960us         3     698ns     312ns  1.2850us  cuDeviceGetCount
  0.00%  1.7590us         8     219ns     174ns     360ns  cudaGetLastError
```

Figure 2.3. Test block size of 512



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob8 1024
==20918== NVPROF is profiling process 20918, command: ./prob8 1024
./prob8 starting reduction at device 0: GeForce GTX 1070  with array size 4096 grid 4, block 1024
CPU reduce: 517140
reduceGmem <<<grid 4, block 1024>>>: 517140             OK!
reduceSmem <<<grid 4, block 1024>>>: 517140             OK!
reduceSmemDyn <<<grid 4, block 1024>>>: 517140          OK!
reduceGmemUnroll <<<grid 1, block 1024>>>: 517140               OK!
reduceSmemUnroll <<<grid 1, block 1024>>>: 522          Failed!
reduceSmemUnrollDyn <<<grid 1, block 1024>>>: 517140            OK!
reduceNeighboredGmem <<<grid 4, block 1024>>>: 517140           OK!
reduceNeighboredSmem <<<grid 4, block 1024>>>: 736              Failed!
==20918== Profiling application: ./prob8 1024
==20918== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 31.13%  17.824us         8  2.2280us     640ns  11.520us  [CUDA memcpy DtoH]
 29.68%  16.992us         8  2.1240us  1.7920us  4.3840us  [CUDA memcpy HtoD]
  9.22%  5.2800us         1  5.2800us  5.2800us  5.2800us  reduceNeighboredGmem(int*, int*, unsigned int)
  6.93%  3.9680us         1  3.9680us  3.9680us  3.9680us  reduceNeighboredSmem(int*, int*, unsigned int)
  5.20%  2.9760us         1  2.9760us  2.9760us  2.9760us  reduceGmemUnroll(int*, int*, unsigned int)
  4.97%  2.8480us         1  2.8480us  2.8480us  2.8480us  reduceGmem(int*, int*, unsigned int)
  3.52%  2.0160us         1  2.0160us  2.0160us  2.0160us  reduceSmemUnrollDyn(int*, int*, unsigned int)
  3.19%  1.8240us         1  1.8240us  1.8240us  1.8240us  reduceSmem(int*, int*, unsigned int)
  3.19%  1.8240us         1  1.8240us  1.8240us  1.8240us  reduceSmemUnroll(int*, int*, unsigned int)
  2.96%  1.6960us         1  1.6960us  1.6960us  1.6960us  reduceSmemDyn(int*, int*, unsigned int)

==20918== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 66.81%  100.29ms         2  50.144ms  111.21us  100.18ms  cudaMalloc
 31.56%  47.379ms         1  47.379ms  47.379ms  47.379ms  cudaDeviceReset
  0.66%  988.15us       182  5.4290us     307ns  279.15us  cuDeviceGetAttribute
  0.39%  592.90us        16  37.056us  6.4820us  128.55us  cudaMemcpy
  0.20%  295.50us         1  295.50us  295.50us  295.50us  cudaGetDeviceProperties
  0.14%  212.95us         2  106.47us  100.16us  112.79us  cudaFree
  0.11%  161.72us         2  80.861us  78.954us  82.769us  cuDeviceTotalMem
  0.07%  103.83us         8  12.978us  6.4370us  28.800us  cudaLaunch
  0.05%  68.886us         2  34.443us  33.815us  35.071us  cuDeviceGetName
  0.00%  4.7010us         1  4.7010us  4.7010us  4.7010us  cudaSetDevice
  0.00%  3.9630us        24     165ns     122ns     301ns  cudaSetupArgument
  0.00%  2.9510us         8     368ns     238ns  1.1550us  cudaConfigureCall
  0.00%  2.3850us         6     397ns     313ns     553ns  cuDeviceGet
  0.00%  2.0020us         3     667ns     344ns  1.2820us  cuDeviceGetCount
  0.00%  1.7380us         8     217ns     171ns     380ns  cudaGetLastError
```

Figure 2.4. Test block size of 1024

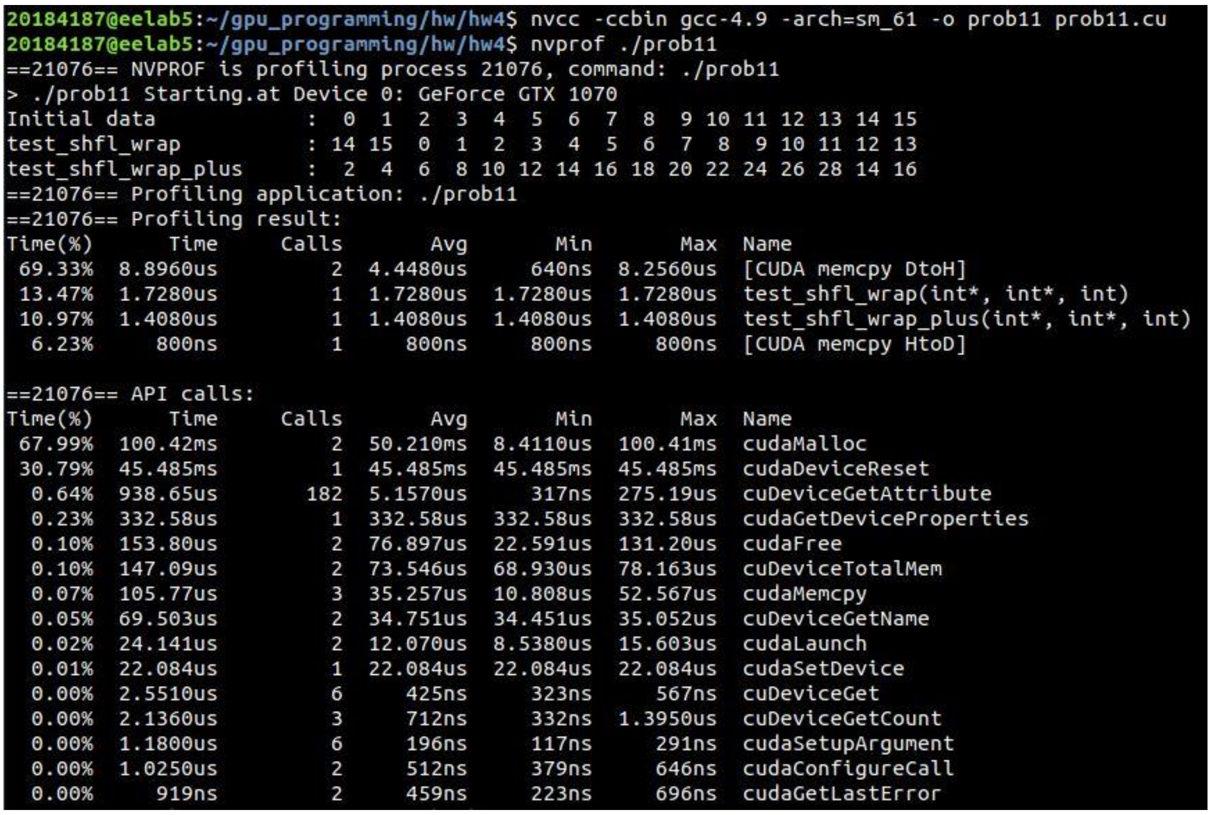
Table 2.1. Performance on various block sizes

Kernel	**Execution time in µs**			
	64 threads	**128 threads**	**512 threads**	**1024 threads**
reduceGmem	2.4640	2.8160	2.8480	2.8480
reduceSmem	1.7280	1.9520	1.6640	1.8240
reduceSmemDyn	1.6640	1.8240	1.6960	1.6960
reduceGmemUnroll	2.3680	2.8160	3.0080	2.9760
reduceSmemUnroll	1.6640	1.6960	1.8560	1.8240
reduceSmemUnrollDyn	2.0150	1.8560	1.7600	2.0160
reduceNeighboredGmem	3.3600	3.4560	4.3520	5.2800
reduceNeighboredSmem	2.6240	2.5920	2.8480	3.9680

In general, with Pascal architecture, the best execution configuration is 64 threads per block, though reduceSmem, reduceSmemDyn and reduceSmemUnrollDyn achieve the best performance with 512 threads per block. The execution time with block size of 1024 is always highest.

3. Problem 11

For problem 11, the source code is prob11.cu file and the results, depicted in Figure 3.1 below. Based on the kernel test_shfl_wrap, the kernel test_shfl_wrap_plus increases the current thread's value by the value of the thread, which is two indexes greater. Because the width of the shuffle operation is still 16, so the first two initial threads warp around to the bottom two threads. The operation of the kernel test_shfl_wrap_plus, presented in Figure 3.2.



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvcc -ccbin gcc-4.9 -arch=sm_61 -o prob11 prob11.cu
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob11
==21076== NVPROF is profiling process 21076, command: ./prob11
> ./prob11 Starting.at Device 0: GeForce GTX 1070
Initial data            :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
test_shfl_wrap          : 14 15  0  1  2  3  4  5  6  7  8  9 10 11 12 13
test_shfl_wrap_plus     :  2  4  6  8 10 12 14 16 18 20 22 24 26 28 14 16
==21076== Profiling application: ./prob11
==21076== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 69.33%  8.8960us         2  4.4480us     640ns  8.2560us  [CUDA memcpy DtoH]
 13.47%  1.7280us         1  1.7280us  1.7280us  1.7280us  test_shfl_wrap(int*, int*, int)
 10.97%  1.4080us         1  1.4080us  1.4080us  1.4080us  test_shfl_wrap_plus(int*, int*, int)
  6.23%     800ns         1     800ns     800ns     800ns  [CUDA memcpy HtoD]

==21076== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 67.99%  100.42ms         2  50.210ms  8.4110us  100.41ms  cudaMalloc
 30.79%  45.485ms         1  45.485ms  45.485ms  45.485ms  cudaDeviceReset
  0.64%  938.65us       182  5.1570us     317ns  275.19us  cuDeviceGetAttribute
  0.23%  332.58us         1  332.58us  332.58us  332.58us  cudaGetDeviceProperties
  0.10%  153.80us         2  76.897us  22.591us  131.20us  cudaFree
  0.10%  147.09us         2  73.546us  68.930us  78.163us  cuDeviceTotalMem
  0.07%  105.77us         3  35.257us  10.808us  52.567us  cudaMemcpy
  0.05%  69.503us         2  34.751us  34.451us  35.052us  cuDeviceGetName
  0.02%  24.141us         2  12.070us  8.5380us  15.603us  cudaLaunch
  0.01%  22.084us         1  22.084us  22.084us  22.084us  cudaSetDevice
  0.00%  2.5510us         6     425ns     323ns     567ns  cuDeviceGet
  0.00%  2.1360us         3     712ns     332ns  1.3950us  cuDeviceGetCount
  0.00%  1.1800us         6     196ns     117ns     291ns  cudaSetupArgument
  0.00%  1.0250us         2     512ns     379ns     646ns  cudaConfigureCall
  0.00%     919ns         2     459ns     223ns     696ns  cudaGetLastError
```

Figure 3.1. The results of problem 11

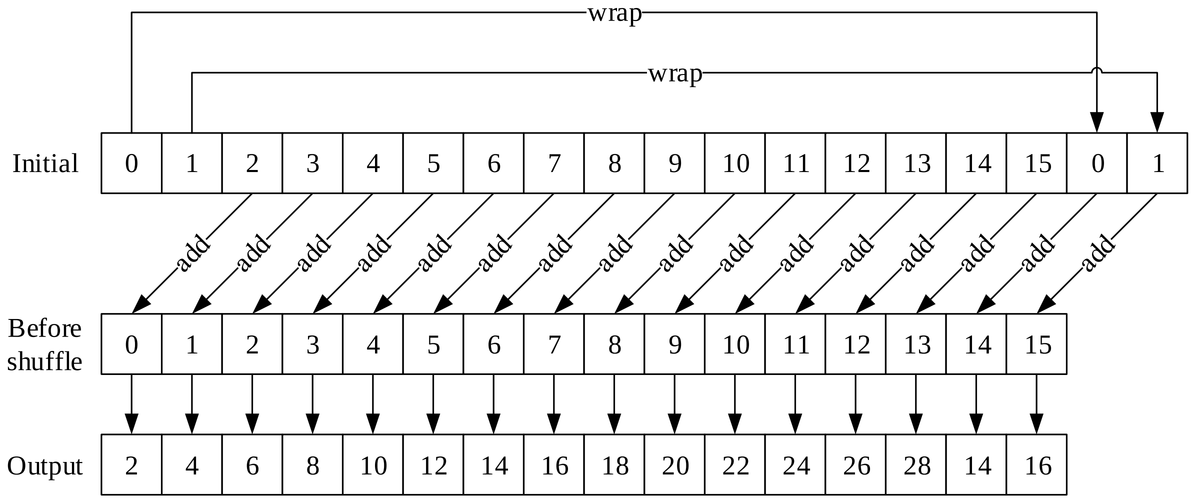
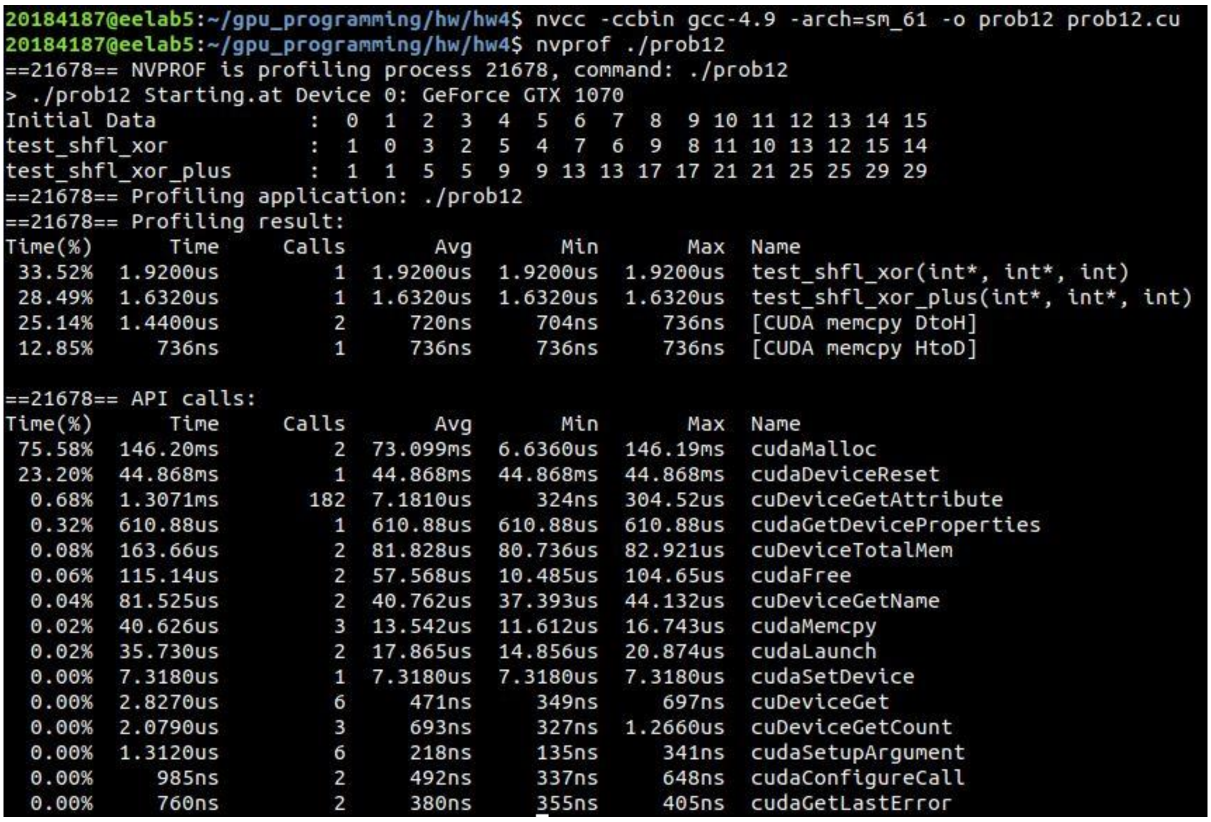


Figure 3.2. The operation of the kernel test_shfl_wrap_plus

4. Problem 12

The prob12.cu file is the source code of problem 12 and its results are shown in Figure 4.1. Based on the kernel test_shfl_xor, the kernel named test_shfl_xor_plus simply performs __shfl_xor function by passing one as the mask and increases the current thread's value with the received value. Every even thread adds the value of the odd thread above it and every odd thread receives the value of the even thread below it.



```
20184187@eelab5:~/gpu_programming/hw/hw4$ nvcc -ccbin gcc-4.9 -arch=sm_61 -o prob12 prob12.cu
20184187@eelab5:~/gpu_programming/hw/hw4$ nvprof ./prob12
==21678== NVPROF is profiling process 21678, command: ./prob12
> ./prob12 Starting.at Device 0: GeForce GTX 1070
Initial Data            :  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
test_shfl_xor           :  1  0  3  2  5  4  7  6  9  8 11 10 13 12 15 14
test_shfl_xor_plus      :  1  1  5  5  9  9 13 13 17 17 21 21 25 25 29 29
==21678== Profiling application: ./prob12
==21678== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 33.52%  1.9200us         1  1.9200us  1.9200us  1.9200us  test_shfl_xor(int*, int*, int)
 28.49%  1.6320us         1  1.6320us  1.6320us  1.6320us  test_shfl_xor_plus(int*, int*, int)
 25.14%  1.4400us         2     720ns     704ns     736ns  [CUDA memcpy DtoH]
 12.85%     736ns         1     736ns     736ns     736ns  [CUDA memcpy HtoD]

==21678== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 75.58%  146.20ms         2  73.099ms  6.6360us  146.19ms  cudaMalloc
 23.20%  44.868ms         1  44.868ms  44.868ms  44.868ms  cudaDeviceReset
  0.68%  1.3071ms       182  7.1810us     324ns  304.52us  cuDeviceGetAttribute
  0.32%  610.88us         1  610.88us  610.88us  610.88us  cudaGetDeviceProperties
  0.08%  163.66us         2  81.828us  80.736us  82.921us  cuDeviceTotalMem
  0.06%  115.14us         2  57.568us  10.485us  104.65us  cudaFree
  0.04%  81.525us         2  40.762us  37.393us  44.132us  cuDeviceGetName
  0.02%  40.626us         3  13.542us  11.612us  16.743us  cudaMemcpy
  0.02%  35.730us         2  17.865us  14.856us  20.874us  cudaLaunch
  0.00%  7.3180us         1  7.3180us  7.3180us  7.3180us  cudaSetDevice
  0.00%  2.8270us         6     471ns     349ns     697ns  cuDeviceGet
  0.00%  2.0790us         3     693ns     327ns  1.2660us  cuDeviceGetCount
  0.00%  1.3120us         6     218ns     135ns     341ns  cudaSetupArgument
  0.00%     985ns         2     492ns     337ns     648ns  cudaConfigureCall
  0.00%     760ns         2     380ns     355ns     405ns  cudaGetLastError
```

Figure 4.1. The results of problem 12

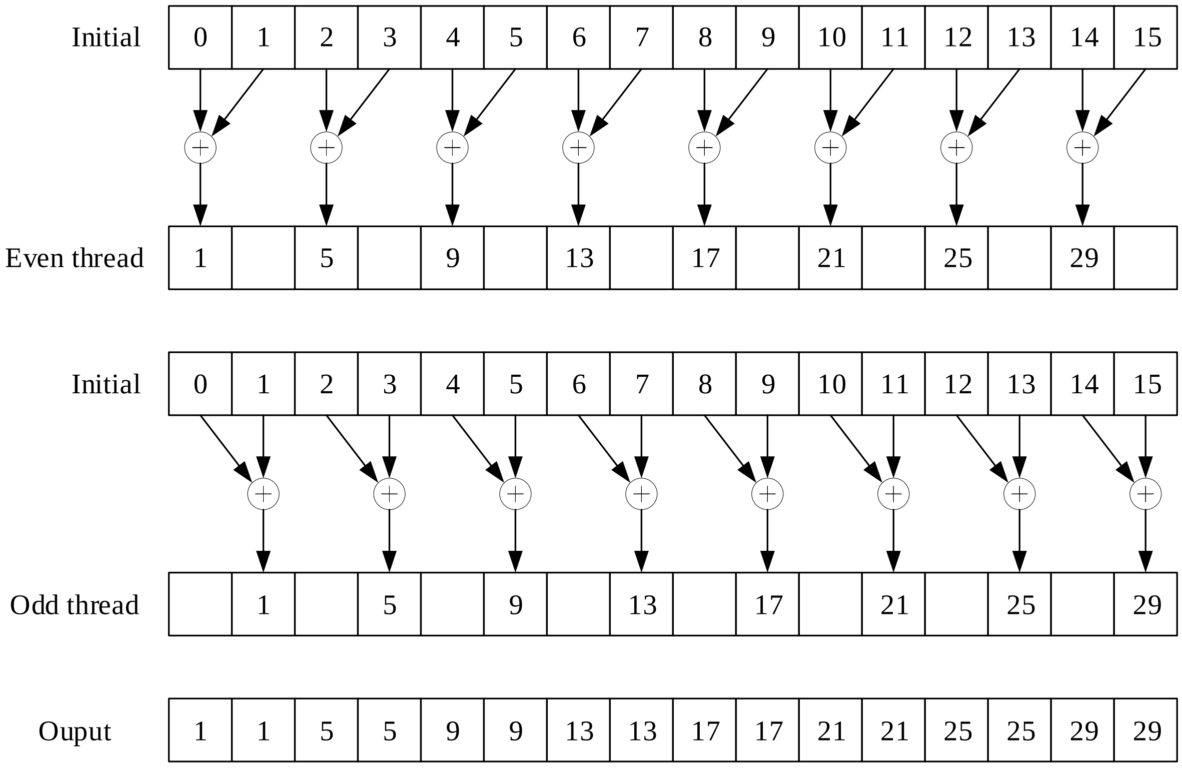


Figure 4.2. The operation of the kernel test_shfl_xor_plus