Variants of Mersenne Twister Suitable for Graphic Processors

Mutsuo Saito[1], Makoto Matsumoto[2]

[1]Hiroshima University, [2]University of Tokyo

August 16, 2010

This study is granted in part by JSPS Grant-In-Aid #21654004, #19204002, #21654017, and JSPS Core-to-Core Program No.18005.

Introduction

Graphic Processing Unit(GPU)

- Hardware (chip) specialized for graphic processing
- A GPU contains hundreds of "CPUs" (very restricted in ability)
- High performance for parallel processing (over 100GFLOPS)
- 3D Game Machines massively use GPUs $\Rightarrow$ low price

Introduction

Graphic Processing Unit(GPU)

- Hardware (chip) specialized for graphic processing
- A GPU contains hundreds of "CPUs" (very restricted in ability)
- High performance for parallel processing (over 100GFLOPS)
- 3D Game Machines massively use GPUs $\Rightarrow$ low price

General Purpose computing on GPU (GPGPU)

- Use GPUs for non-graphic computations
- Cheap supercomputers (of TFLOPS) use a grid of GPUs
  price $\sim$ 10,000 US dollars
- Parallelism of GPUs is suitable for some Monte Carlo simulations
  (if the problem is partitionable into pieces, e.g. 3D simulation)
- Needs of pseudorandom number generators (PRNGs) for GPUs

Purpose of Study

- Design efficient PRNGs taking advantage of GPUs:
  Mersenne Twister for Graphic Processors (MTGP).
- This time, we designed for NVIDIA's CUDA-enabled GPU:
  GeForce GT* series. (CUDA=a developping environment for GPU.)
- The codes work for any GT* GPU, and the generated sequence is reproducible and independent of GPUs.
- Dynamic Creator for MTGP: produces parameter sets for MTGP generators, according to the users' specification.
  Convenient for a large grid of GPUs.

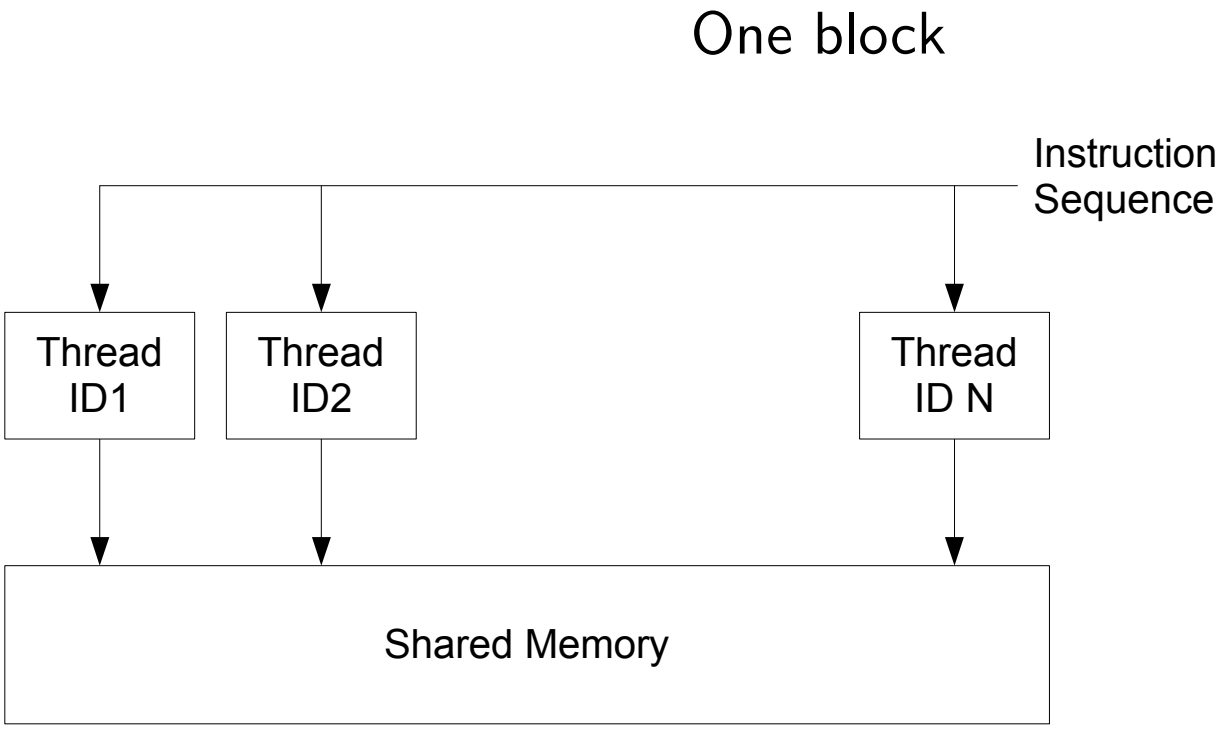
GeForce GPUs from NVIDIA: processes

We mainly explain software level only (hardware: complicated).

- A process is called a thread. This is a smallest unit of a program.
- A block consists of many (but at most 512) threads, which may run in parallel (physically). No ordering among the threads is assured. (Thus, the threads are similar to the processes in a multi-process OS, but they may run physically in paralell.)
- A GPU can run several blocks in paralell (physically).
  Eg. GTX-260 GPU can ran 54 blocks at the same time
  (depend on consumed memory, etc.).
- Each block has its own memory in the GPU chip, called shared memory. Size of memory is 16KByte.
  This is accessible from threads in the block, but inaccesible from other blocks (so no collision between blocks for shared memory).

Many threads and one shared memory in one block

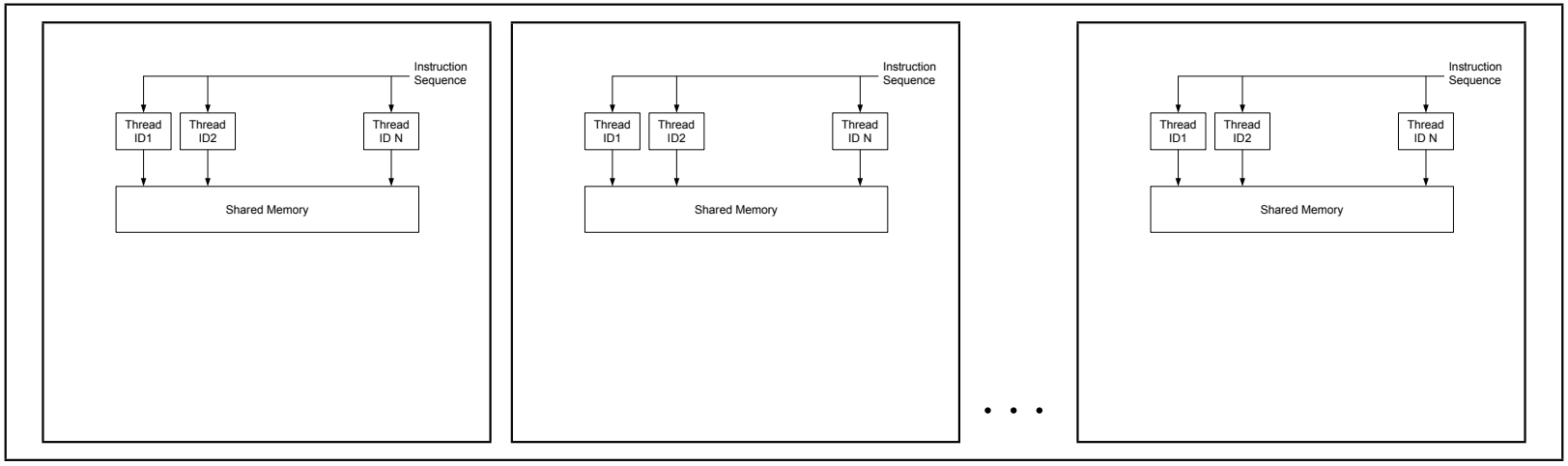
The following is a picture of one block. A GPU may ran 54 blocks in parallel (with 27 core hardwares in GPU).



54 blocks in one GPU

A GPU may ran 54 blocks in parallel (with 27 core hardwares in GPU).

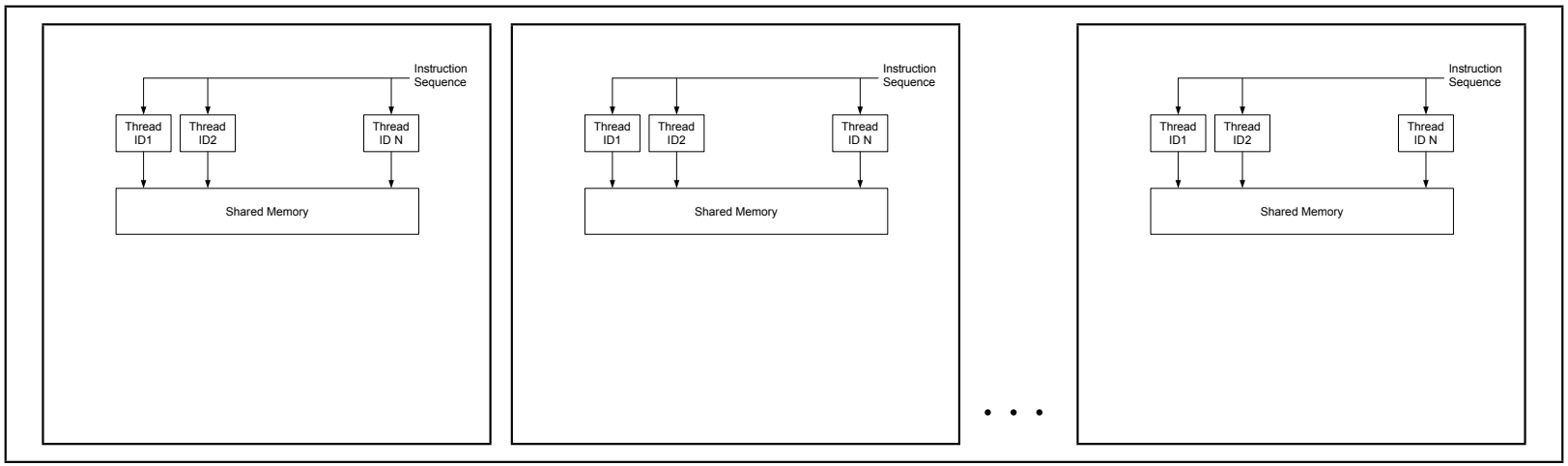
A GPU chip



54 blocks in one GPU

A GPU may ran 54 blocks in parallel (with 27 core hardwares in GPU).

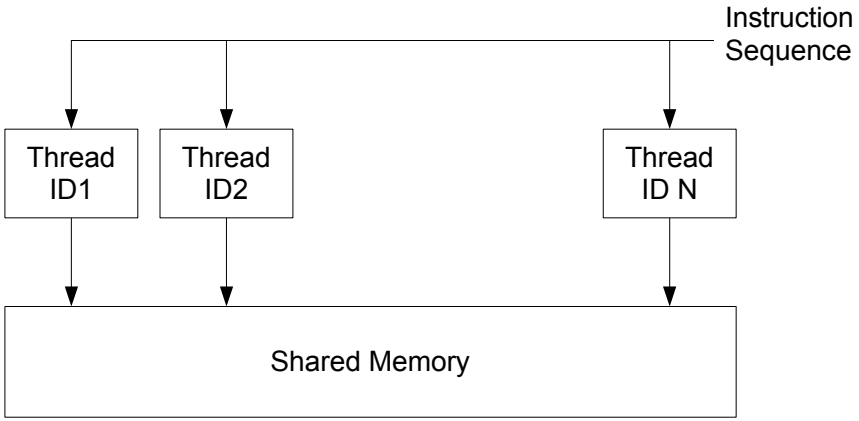
A GPU chip



⇳ 448-bit data bus

Device memory (outside GPU chip)

Restriction on threads in a block



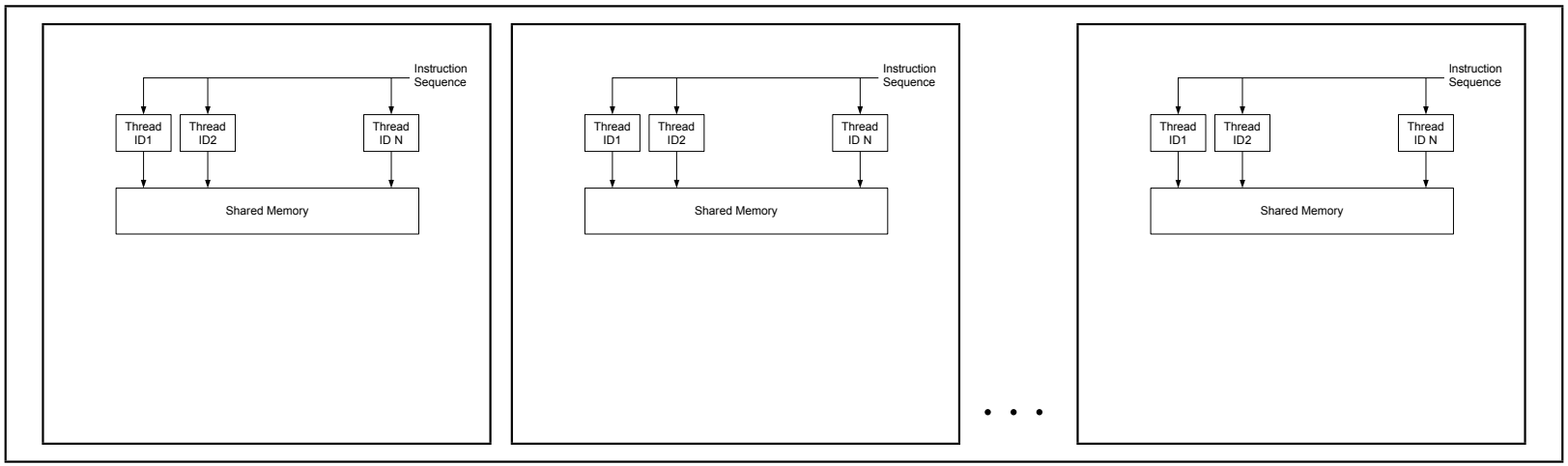
- Every thread in a block gets one same instruction sequence. Thus, every thread does the same operation, except for:
- Each thread has its own ID number (consecutive), and acts on the shared memory with address shifted by the ID.
- Thus, two threads in one block do not access one same address of shared memory, which avoids collision of access. Typically, 32 threads can run "physically simultaneously" in one block, and 512 threads can run "logically" in paralell in one block.

GeForce GPUs from NVIDIA: memory

- Specialized memory chips, called device memory, are equipped outside the GPU. Size: for GTX260, 896Mbyte. Data bus 448-bit, transfer 112Gbyte/sec. (Cf. typical CPU's memory: transfer 26Gbyte/sec.)
- Blocks running in a GPU can access the device memory. Blocks can exchange information only via the device memory.
- But typically, every block is assigned its own part in the device memory, so access collision can possibly be avoided.
- Similarly to the shared memory, each thread in one block does the same operation on the device memory assigned for the block, with the address shifted according to the thread ID.

GPU and Device Memory

A GPU chip



⇕ 448-bit data bus

Device memory (896Mbyte, outside GPU chip)

PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).

PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).
  - Necessity of same recursion

PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).
  - Necessity of same recursion
    $\Leftarrow$ Threads get the same instructions

PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).
  - Necessity of same recursion
    $\Leftarrow$ Threads get the same instructions
  - Possibility of distinct parameters

PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).
  - Necessity of same recursion
    $\Leftarrow$ Threads get the same instructions
  - Possibility of distinct parameters
    $\Leftarrow$ Store the parameters in the shared (or device) memory

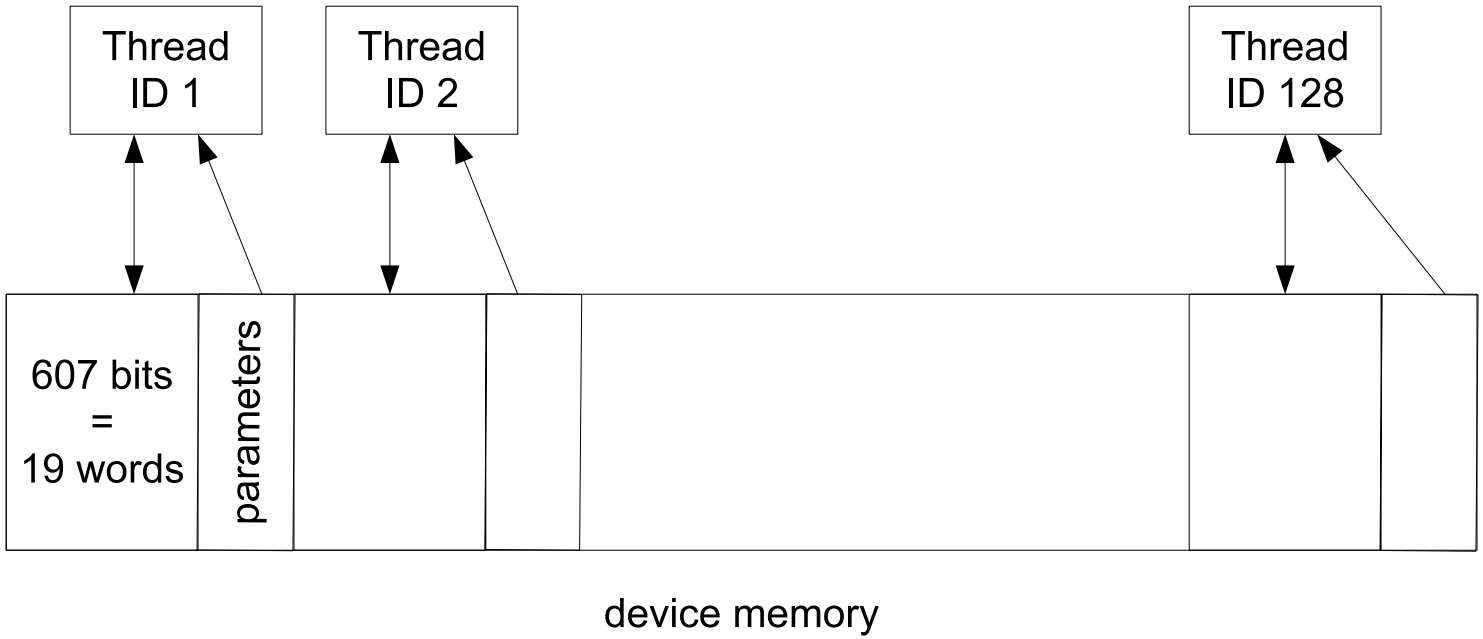
PRNGs for GPUs : Naive

Most naive idea: one generator for one thread:

- For each thread, prepare one generator
  (say, of same recursion with distinct parameters).
  - Necessity of same recursion
    $\Leftarrow$ Threads get the same instructions
  - Possibility of distinct parameters
    $\Leftarrow$ Store the parameters in the shared (or device) memory
- Example: SDK-MT (sample program from NVIDIA).
  32 blocks $\times$ 128 $=$ 4096 threads.
  SDK-MT prepares 4096 distinct parameter sets
  of MT607 $=$ Mersenne Twister PRNG with 607-bit state space.
  Each thread uses its own MT607.

PRNGs for GPUs : Naive=SDK-MT

SDK-MT: 1 block has
128 threads, each
processes one MT607

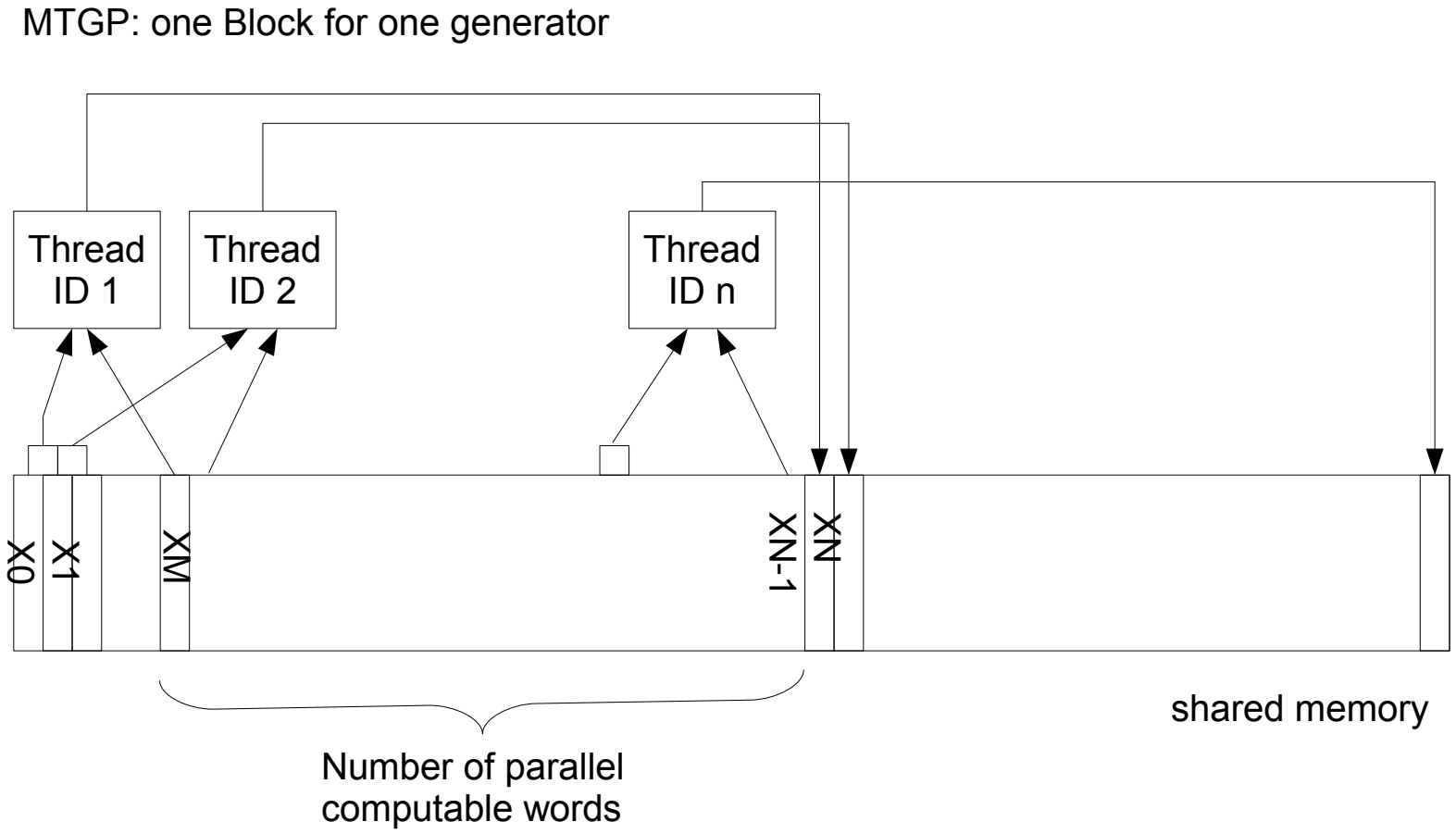


PRNGs for GPUs : MTGP

Strategy in MTGP:

- One generator for one block. Threads in one block process one large generator, with state space of p =11213 to 44497 dimensions. (These numbers are Mersenne exponents(MEXP), i.e. p with $2^p - 1$ being prime.)
- The state space is accomodated in the shared memory.
- In the state space, a large part can be computed in parallel. Select a recursion permitting this.

PRNGs for GPUs : MTGP



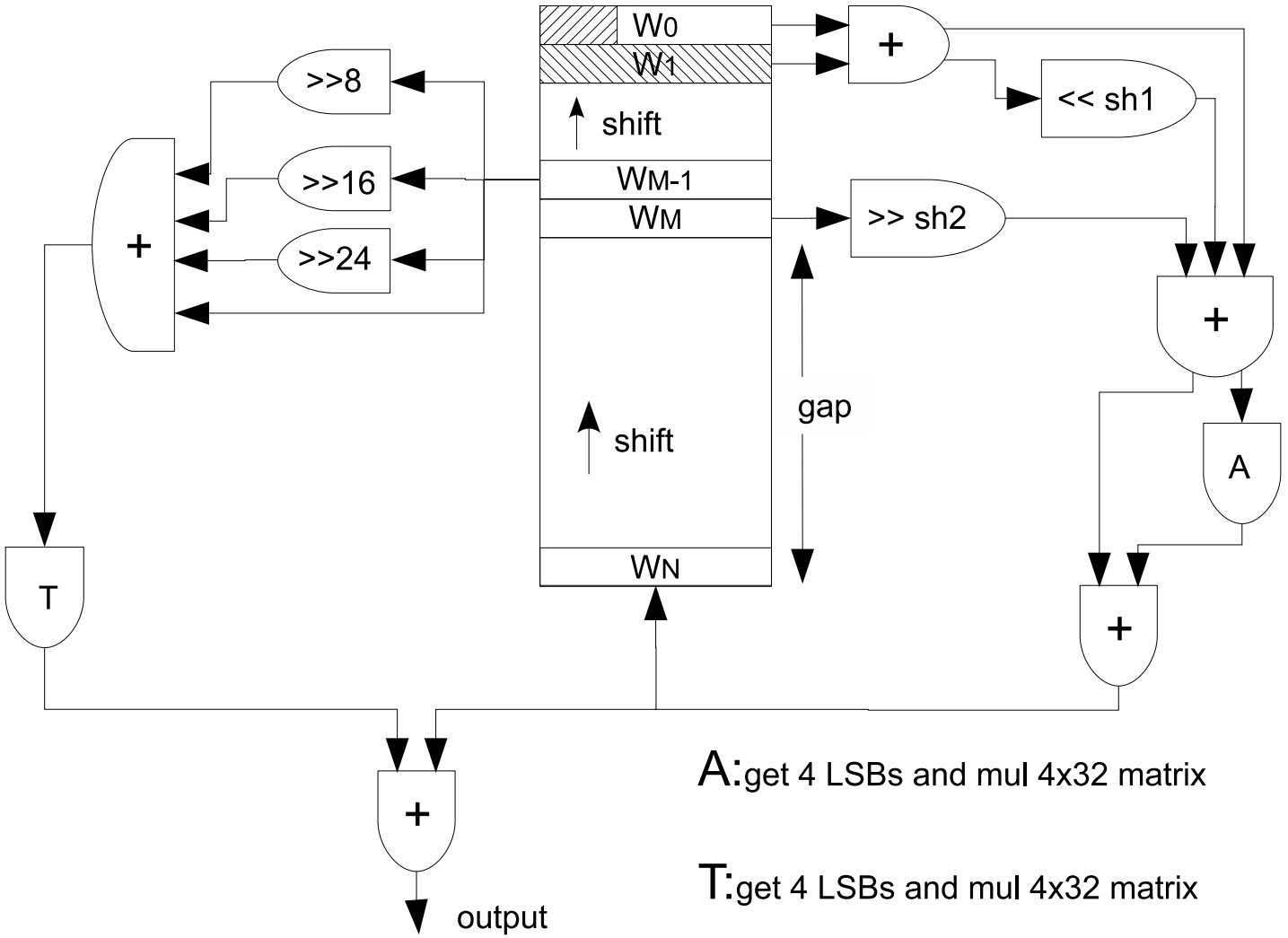
Thread $i+1$ processes recursion $x_{N+i} = f(x_{M+i}, x_{1+i}, x_i)$.
The gap $n = N - M$ is the number of parallely computable words.

This parallelism is classical but efficient

This type of parallelization for Shift Register sequence is common since 1980's. Its merit compared to SDK-MT is:

- SDK-MT's consumption of memory counted in bit is $(607 + \text{parameter size}) \times$ the number of threads.
- MTGP's consumption is $32\times$ the number of threads
- If the state spaces of SDK-MT are kept in the shared memory (16KByte), then the number of parallel threads is small: (16KByte)/(size of working space for MT607)< 100
- The period of generated sequence: SDK-MT has period $2^{607} - 1$, while MTGP has period $2^{11213} - 1$ and higher dimensional equidistribution property (explain later).

Circuit-like description of MTGP



The size of "gap"=the max number of parallel threads workable on one state space

Spec of designed MTGP

- We distribute versions with period $2^{11213} - 1$, $2^{23209} - 1$ and $2^{44497} - 1$.
- The "gap" (i.e. the number of parallel computable words) is 256, 512, and 1024, respectively.
- We list 128 distinct parameter sets for each period. Thus, 128 different MTGPs for each period.
- 32-bit integer, 32-bit floating point, 64-bit integer, 64-bit floating point are supported as the output.

Comparison of SDK-MT and MTGP

- CUDA SDK: cuda SDK MerseneTwister sample
  - period: $2^{607} - 1$
  - use 4096 parameter sets (=4096 different MT607s)
  - =32 blocks, one block has 128 threads
- MTGP:
  - period: $2^{11213} - 1$
  - use 108 parameter sets (=108 different MTGP11213s)
  - 108 blocks, one block has 256 threads

Comparison of speed

The time (ms) required for 5×10^7 generations.

	SDK MT	MTGP		
	single[0,1)	32 bit int	single[1,2)	single[0,1)
GT 120 (4-core)	50.2ms	32.5ms	32.8ms	33.9ms
GTX 260 (27-core)	18.6ms	4.6ms	4.8ms	4.9ms

Dimension of equidistribution

Definition

A sequence of v-bit integers with period $P = 2^p - 1$

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{P-1}, \mathbf{x}_P = \mathbf{x}_0, \ldots$$

is said to be k-dimensionally equidistributed if the multi set (i.e. counted with multiplicity)

$$\{(\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+k-1}) | i = 0, \ldots, P-1\}$$

is uniformly distributed over all possible kv-bit patterns (we permit one time lack of all zero pattern).

dimension of equidistribution to v-bit accuracy

Definition

A periodic sequence of $b(=32)$-bit integers is k-dimensionally equidistributed to v-bit accuracy if the most significant v-bit-integer sequence is k-dimensionally equidistributed.

The dimension of equidistribution to v-bit accuracy $k(v)$ is max such k. Larger is better.

For $P = 2^p - 1$, there is a bound $k(v) \leq \lfloor p/v \rfloor$.

The dimension defect $d(v)$ at v is the difference $d(v) := \lfloor p/v \rfloor - k(v)$,
The total dimension defect Δ is their sum over v: $\Delta := \sum_{v=1}^{b} d(v)$.

dimension of equidistribution

$k(v)$ and $d(v)$ of MTGP23209 ID=0

v	$k(v)$	$d(v)$	v	$k(v)$	$d(v)$	v	$k(v)$	$d(v)$	v	$k(v)$	$d(v)$
1	23209	0	9	2578	0	17	1355	10	25	726	202
2	11604	0	10	2320	0	18	1268	21	26	725	167
3	7736	0	11	2109	0	19	1200	21	27	725	134
4	5801	1	12	1933	1	20	1125	35	28	725	103
5	4641	0	13	1785	0	21	1054	51	29	725	75
6	3867	1	14	1657	0	22	926	128	30	725	48
7	3315	0	15	1547	0	23	925	84	31	725	23
8	2900	1	16	1450	0	24	924	43	32	725	0

Δ is 1149.

c.f. Δ of MT19937 is 6750.

Dynamic Creator for MTGP

Dynamic Creator is a parameter-set generator for Mersenne Twister, which is intended for a large scale parallel simulation.
We released MTGP Dynamic Creator (MTGPDC):

- ID is any 32-bit integer, embedded in the recursion formula.
- runs on CPU.
- searches for parameter sets that assure the maximal period.
- searches for output functions to have high $k(v)$ $(v = 1, \ldots, 32)$.

Speed of MTGPDC

Seconds required to search one recursion and
to search one good output function
for several Mersenne Exponent (MEXP).

CPU time (sec.) for recursion and output parameter search

	MEXP	3217	4423	11213	23209	44497
	samples	3000	3000	1500	1500	750
re	min	0	0	4	24	143
cur	max	90	191	3318	10146	49987
sion	average	11.2	25.0	338.1	1404.7	6529.4
out	min	10	15	76	379	946
put	max	25	40	253	1040	3893
	average	21.7	34.1	213.7	910.0	3236.4

We used SIS (Harase-Saito-M 2009) algorithm for computing $k(v)$.

Conclusion

Proposed MTGP: pseudorandom number generator for GPUs.

- Run on GPUs, taking advantage of parallelism and memory hierarchy of GPUs.
- Merits in speed, period, and dimensions of equidistribution.

Conclusion

Proposed MTGP: pseudorandom number generator for GPUs.

- Run on GPUs, taking advantage of parallelism and memory hierarchy of GPUs.
- Merits in speed, period, and dimensions of equidistribution.

Proposed Dynamic creator for MTGP.

- 2^{32} different parameter sets of recursion (and output function) of MTGP.
- Run on CPU.

Conclusion

Proposed MTGP: pseudorandom number generator for GPUs.

- Run on GPUs, taking advantage of parallelism and memory hierarchy of GPUs.
- Merits in speed, period, and dimensions of equidistribution.

Proposed Dynamic creator for MTGP.

- 2^{32} different parameter sets of recursion (and output function) of MTGP.
- Run on CPU.

32-bit integer, 32-bit floating point, 64-bit integer, 64-bit floating point versions are downloadable from our home page:
`http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MTGP/index.html`

Conclusion

Proposed MTGP: pseudorandom number generator for GPUs.

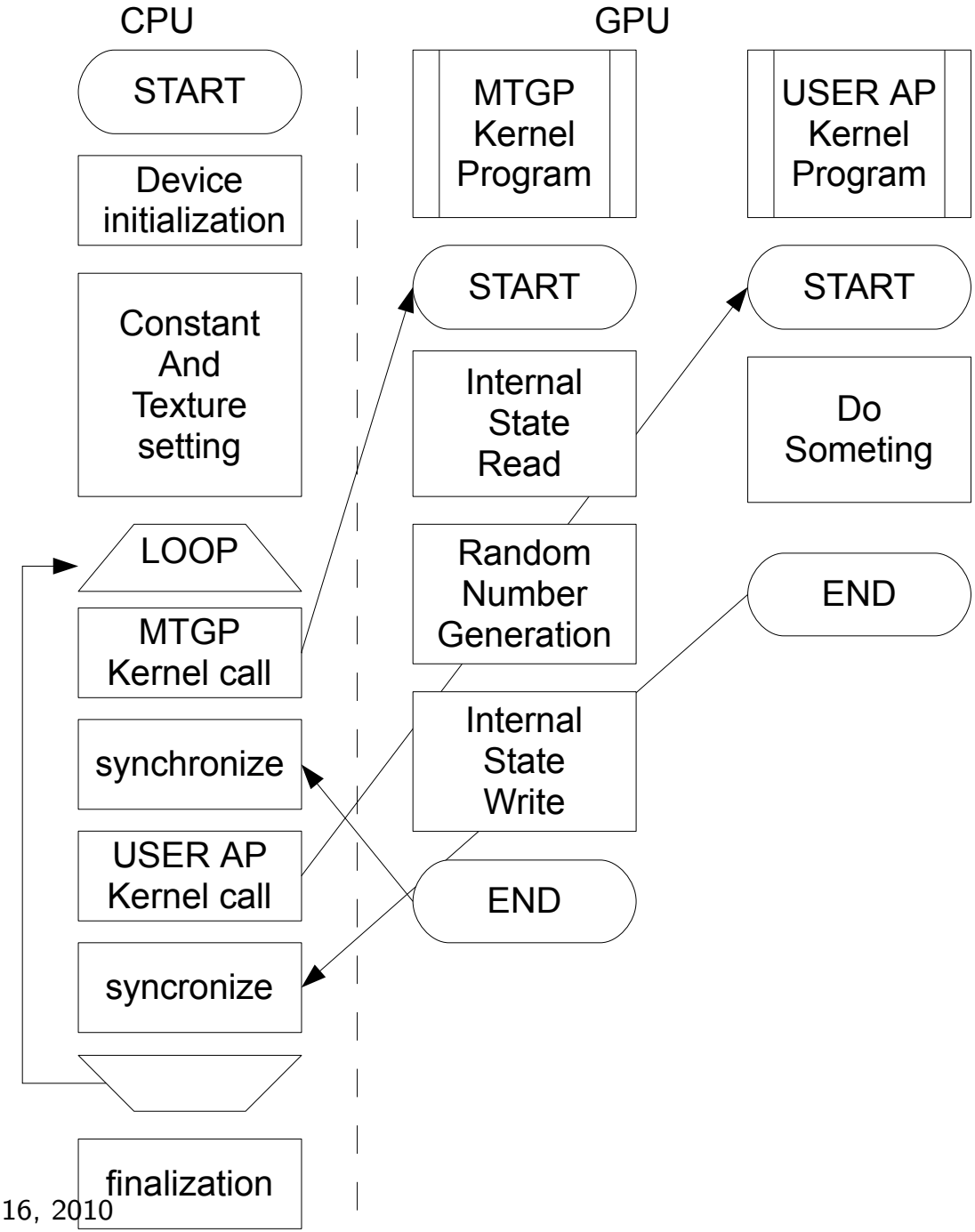
- Run on GPUs, taking advantage of parallelism and memory hierarchy of GPUs.
- Merits in speed, period, and dimensions of equidistribution.

Proposed Dynamic creator for MTGP.

- 2^{32} different parameter sets of recursion (and output function) of MTGP.
- Run on CPU.

32-bit integer, 32-bit floating point, 64-bit integer, 64-bit floating point versions are downloadable from our home page:
`http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MTGP/index.html`

Thank you for listening.


CPU
GPU
START
Device initialization
Constant And Texture setting
LOOP
MTGP Kernel call
synchronize
USER AP Kernel call
syncronize
finalization
MTGP Kernel Program
START
Internal State Read
Random Number Generation
Internal State Write
END
USER AP Kernel Program
START
Do Someting
END