SHAKE-128 PTX 구현

정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

이전 진행상황

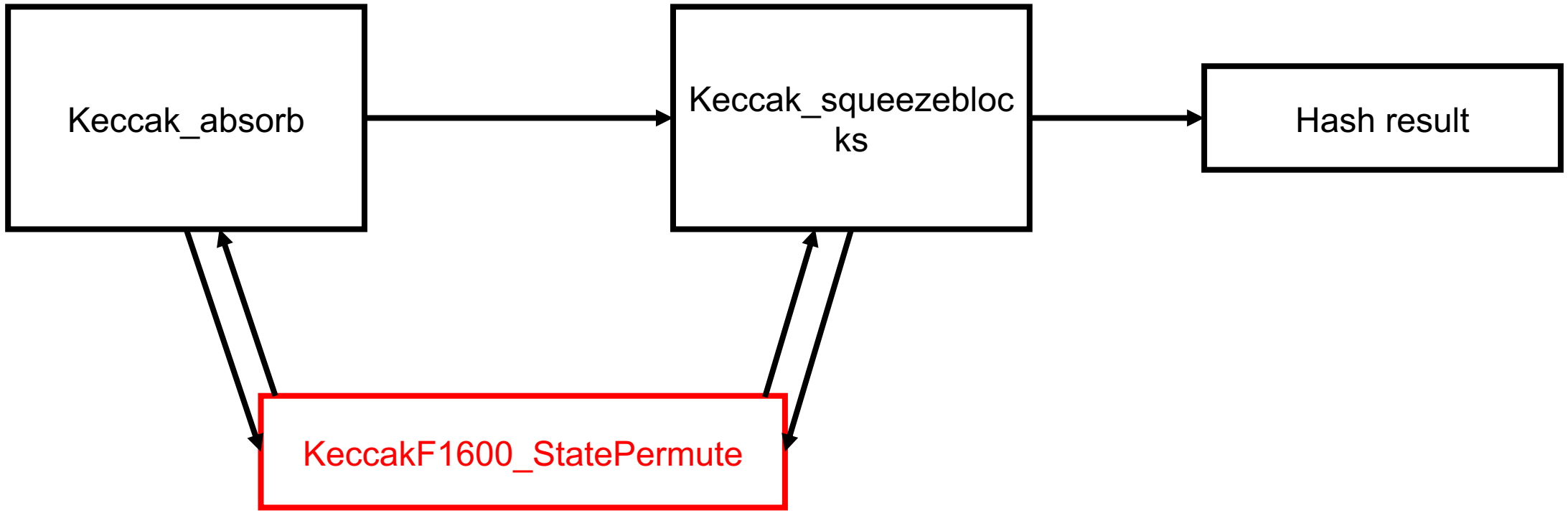
- SHAKE-128 GPU 구현을 시도
- Coarse Grain으로 구현
  - 적정 수준의 블록, 스레드 사용이 있어야 성능 향상
  - 스레드의 경우, 32스레드 이상을 사용
- **PTX 어셈블리**를 사용한 Coarse Grain 시도

PTX 구현

- PTX는 Parallel Thread Execution의 약자
- GPU 병렬 프로그래밍에서 사용 가능한 어셈블리어
- 기본적인 문법
  - (명령어).(자료형) (대상레지스터), (입력1), (입력2);
- 인라인 어셈블리 지원
  - C언어의 인라인 어셈블리와 마찬가지로, C언어 중간에 삽입하는 어셈블리

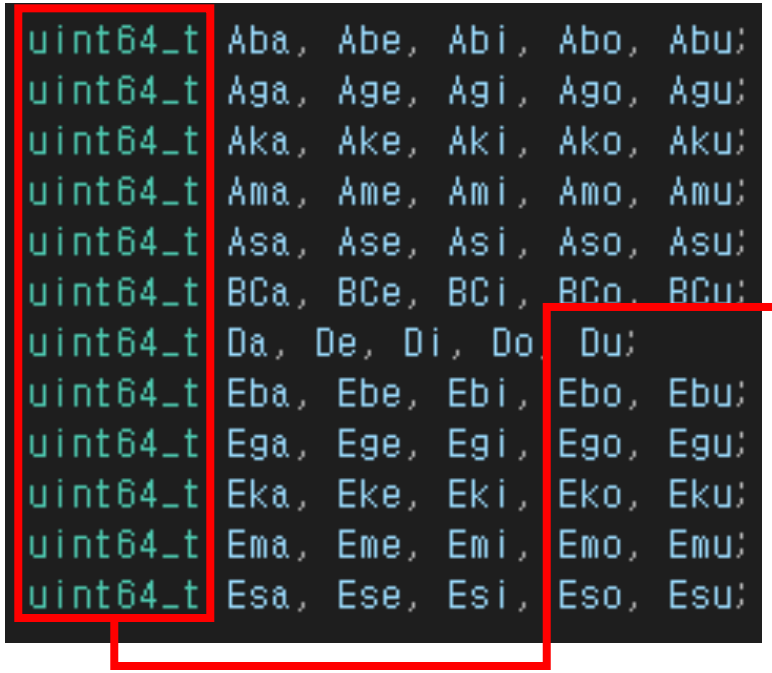
PTX 구현

- Keccak 내부 함수의 Permutation 부분에서 PTX 어셈블리 적용



문제점 해결

- XOR 연산자의 문법 오류
- 알 수 없는 오류가 발생
  - XOR.u64 %0, %1, %2 = %0 <- %1 ^ %2



- Uint64_t = unsigned long long
  - stdint.h에 정의됨
- 64-bit

문제점 해결

- PTX 어셈블리는 16종류의 자료형을 제공
- XOR 연산자는 b(n)형의 지정자를 필요로 함

기본 자료형	자료형 지정자
정수	.s8, .s16, .s32, .s64
부호 없는 정수	.u8, .u16, .u32, .u64
부동소수점	.f16, .f16x2, .f32, .f64
비트	.b8, .b16, .b32, .b64
서술형	.pred

문제점 해결

어셈블리

```
"xor.b64 %25, %0, %5;₩n₩t"   // prepareTheta
"xor.b64 %25, %25, %10;₩n₩t"
"xor.b64 %25, %25, %15;₩n₩t"
"xor.b64 %25, %25, %20;₩n₩t" // BCa = Aba ^ Aga ^ Aka ^ Ama ^ Asa;
"xor.b64 %26, %1, %6;₩n₩t"
"xor.b64 %26, %26, %11;₩n₩t"
"xor.b64 %26, %26, %16;₩n₩t"
"xor.b64 %26, %26, %21;₩n₩t" // BCe = Abe ^ Age ^ Ake ^ Ame ^ Ase;
"xor.b64 %27, %2, %7;₩n₩t"
"xor.b64 %27, %27, %12;₩n₩t"
"xor.b64 %27, %27, %17;₩n₩t"
"xor.b64 %27, %27, %22;₩n₩t" // BCi = Abi ^ Agi ^ Aki ^ Ami ^ Asi;
"xor.b64 %28, %3, %8;₩n₩t"
"xor.b64 %28, %28, %13;₩n₩t"
"xor.b64 %28, %28, %18;₩n₩t"
"xor.b64 %28, %28, %23;₩n₩t" // BCo = Abo ^ Ago ^ Ako ^ Amo ^ Aso;
"xor.b64 %29, %4, %9;₩n₩t"
"xor.b64 %29, %29, %14;₩n₩t"
"xor.b64 %29, %29, %19;₩n₩t"
"xor.b64 %29, %29, %24;₩n₩t" // BCu = Abu ^ Agu ^ Aku ^ Amu ^ Asu;
```

=

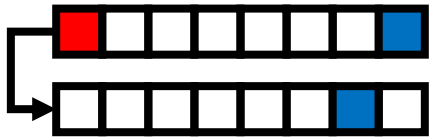
C

```
//    prepareTheta
BCa = Aba ^ Aga ^ Aka ^ Ama ^ Asa;
BCe = Abe ^ Age ^ Ake ^ Ame ^ Ase;
BCi = Abi ^ Agi ^ Aki ^ Ami ^ Asi;
BCo = Abo ^ Ago ^ Ako ^ Amo ^ Aso;
BCu = Abu ^ Agu ^ Aku ^ Amu ^ Asu;
```

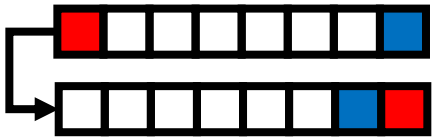
문제점 해결

- Rotation 연산 구현
  - Shift: 비트를 한 방향으로 이동, 반대쪽 방향에 0 입력
  - Rotation: 비트를 한 방향으로 이동, 반대쪽 방향으로 비트가 이동
- AVR에는 Rotation 명령어가 지원
- PTX에는 Rotation이 없으므로 Shift를 통해 간접적으로 구현

Shift



Rotation



AVR 명령어

ROL – Rotate Left trough Carry

ROR – Rotate Right through Carry

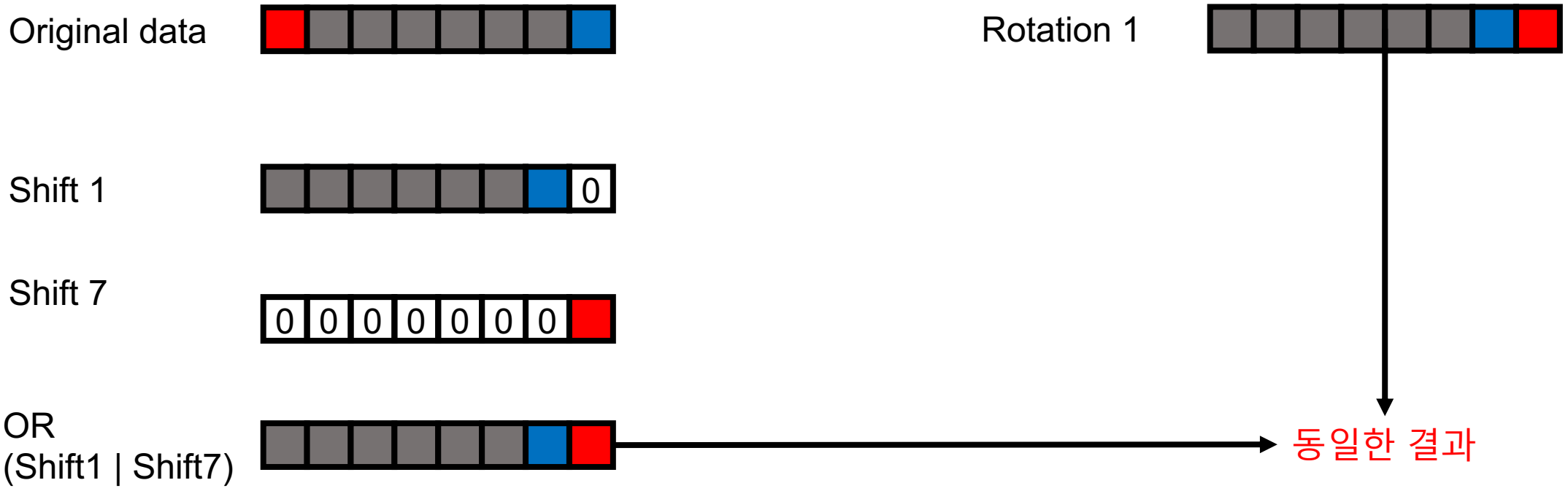
PTX 명령어

9.7.7.8. Logic and Shift Instructions: shl

9.7.7.9. Logic and Shift Instructions: shr

문제점 해결

- 2 Shift + 1 OR = 1 Rotation



문제점 해결

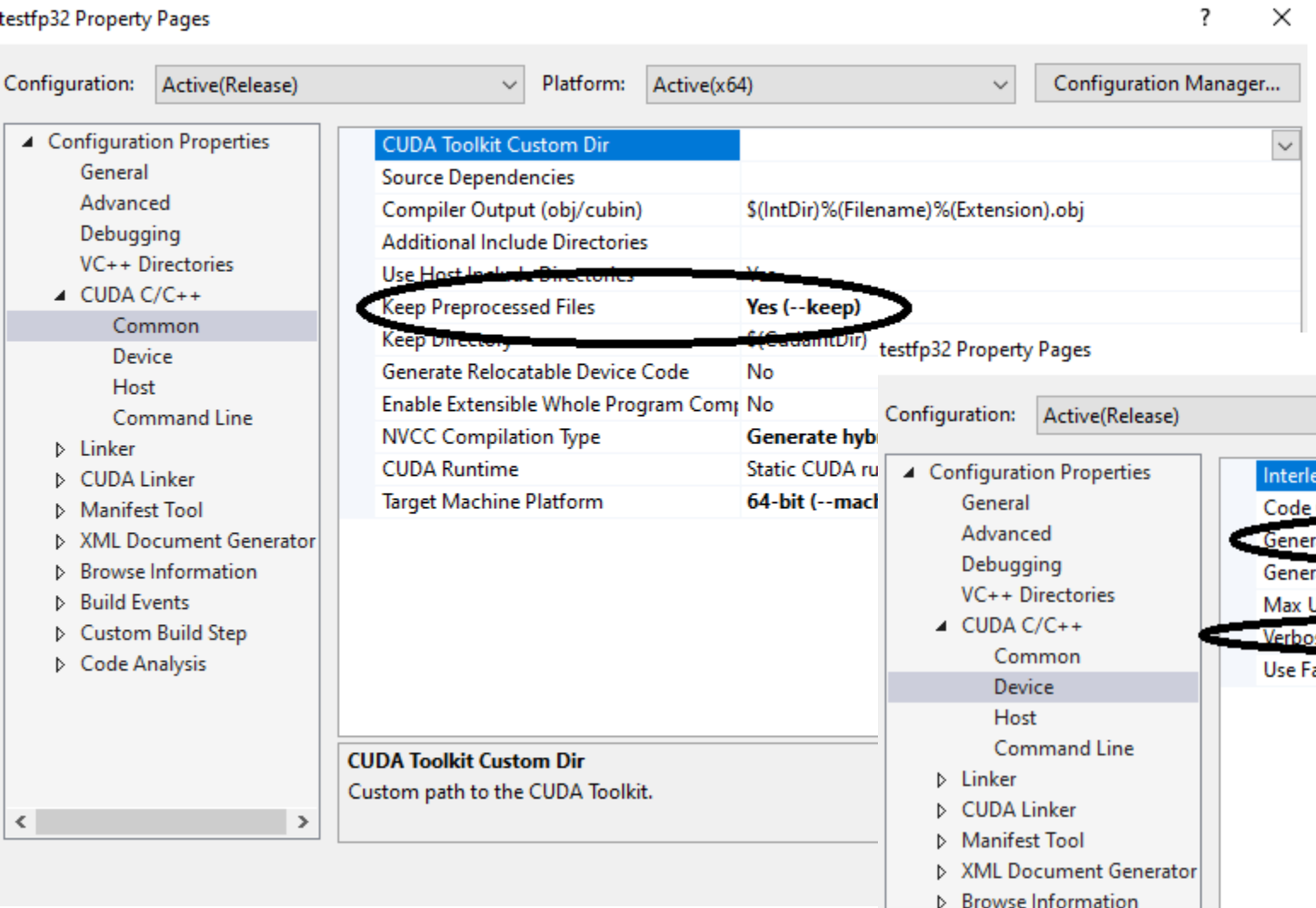
- 전역변수 인덱싱 구현
- 인덱스가 상수 값일 때는 단순한 구현이 가능

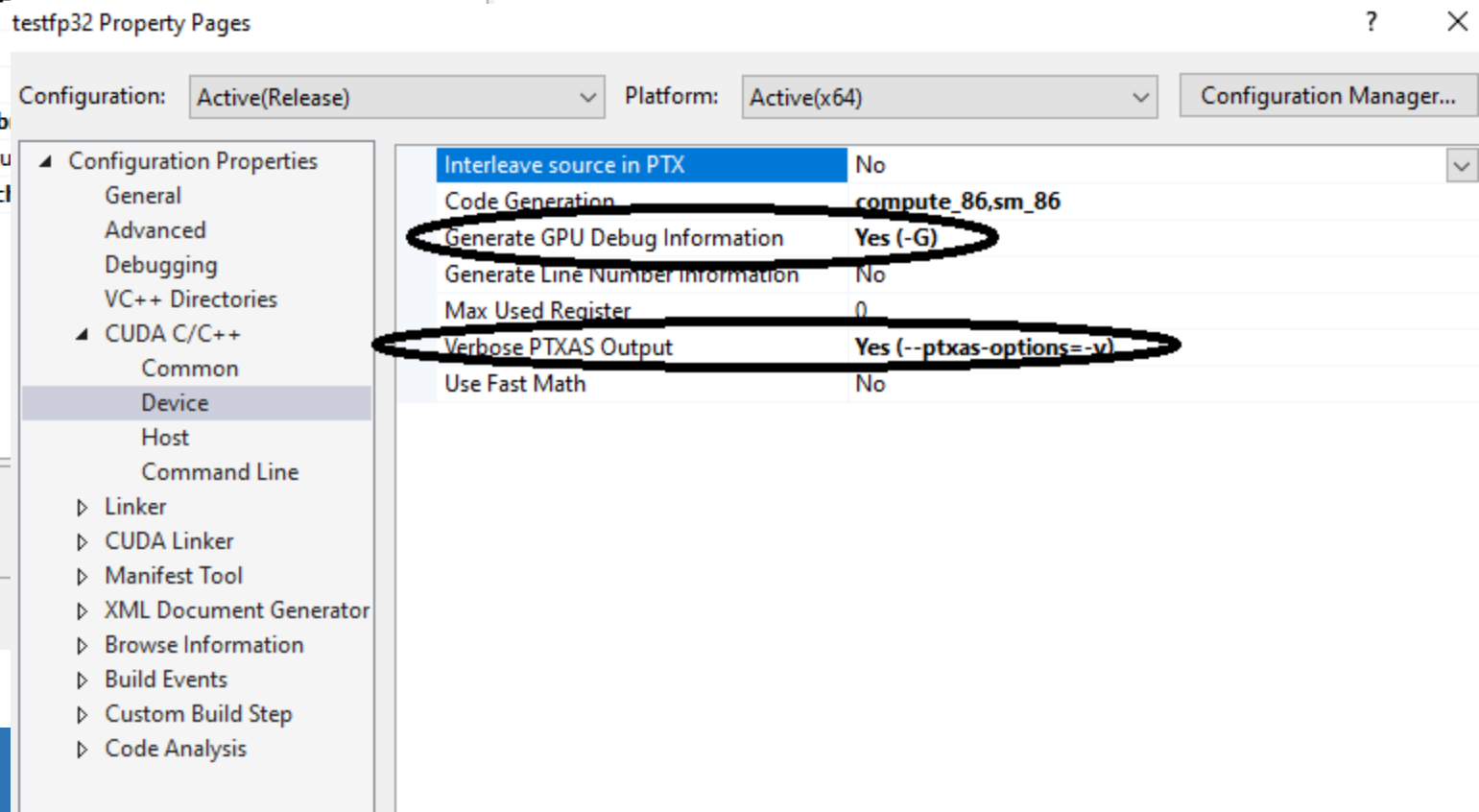
```
int a = b[0]
    ld.global.s32 a, [b+0];\n\t

uint64_t a = b[5]
    ld.global.u64 a, [b+5];\n\t
```

```
Aba ^= Da;
BCa = Aba;
Age ^= De;
BCe = ROL(Age, 44);
Aki ^= Di;
BCi = ROL(Aki, 43);
Amo ^= Do;
BCo = ROL(Amo, 21);
Asu ^= Du;
BCu = ROL(Asu, 14);
Eba = BCa ^ ((~BCe) & BCi);
Eba ^= (uint64_t)KeccakF_RoundConstants[round];
Ebe = BCe ^ ((~BCi) & BCo);
Ebi = BCi ^ ((~BCo) & BCu);
Ebo = BCo ^ ((~BCu) & BCa);
Ebu = BCu ^ ((~BCa) & BCe);
```

- 변수 인덱스일 경우 구현이 복잡해짐

문제점 해결


testfp32 Property Pages
Configuration: Active(Release)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
CUDA C/C++
Common
Device
Host
Command Line
Linker
CUDA Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
CUDA Toolkit Custom Dir
Source Dependencies
Compiler Output (obj/cubin)
$(IntDir)%(Filename)%(Extension).obj
Additional Include Directories
Keep Preprocessed Files
Yes (--keep)
Generate Relocatable Device Code
No
Enable Extensible Whole Program Comp
No
NVCC Compilation Type
CUDA Runtime
Static CUDA ru
Target Machine Platform
64-bit (--mach
CUDA Toolkit Custom Dir
Custom path to the CUDA Toolkit.


testfp32 Property Pages
Configuration: Active(Release)
Platform: Active(x64)
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
CUDA C/C++
Common
Device
Host
Command Line
Linker
CUDA Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Interleave source in PTX
No
Code Generation
compute_86,sm_86
Generate GPU Debug Information
Yes (-G)
Generate Line Number Information
No
Max Used Register
0
Verbose PTXAS Output
Yes (--ptxas-options=-v)
Use Fast Math
No

문제점 해결

- 단순한 프로젝트 생성
- 전역 변수 배열에서 상수 값을 가져오는 프로젝트

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

__device__ static const int TestArray[10] = { 10, 100, 110, 130, 150, 200, 500, 8500, 10000, 51545 };

__device__ static void get_round_constant(void)
{
    int round_counter, a;

    for (round_counter = 0; round_counter < 10; round_counter+=3)
    {
        a = TestArray[round_counter];
        printf("round %d: %d \n", round_counter, a);
    }
}

__global__ void call(void)
{
    get_round_constant();
}

int main()
{
    call <<< 1, 1 >>> ();

    return 0;
}
```

- PTX 코드 추출

```
tmp16:
    mov.u64     %rd1, TestArray;
    cvta.global.u64     %rd2, %rd1;
    cvt.s64.s32 %rd3, %r2;
    shl.b64     %rd4, %rd3, 2;
    add.s64     %rd5, %rd2, %rd4;
    ld.u32  %r5, [%rd5];
    mov.b32     %r6, %r5;
```

문제점 해결

- 여섯 줄로 구현이 가능

```
"mov.u64 tmp1, KeccakF_RoundConstants;₩n₩t"
"cvt.u64.s32 tmp2, %60;₩n₩t"
"mul.lo.u64 tmp2, tmp2, 8;₩n₩t"
"add.u64 tmp1, tmp1, tmp2;₩n₩t"
"ld.global.u64 tmp2, [tmp1];₩n₩t"
"xor.b64 %35, %35, tmp2;₩n₩t" // Eba ^= (uint64_t)KeccakF_RoundConstants[round];
```

결론 및 향후 과제

record_assembly.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

block:	1	thread:	32	time:	0.577133
block:	1	thread:	64	time:	0.576499
block:	1	thread:	128	time:	0.577939
block:	1	thread:	256	time:	0.629837
block:	1	thread:	512	time:	0.001517
block:	1	thread:	1024	time:	0.001904
block:	2	thread:	32	time:	0.576374
block:	2	thread:	64	time:	0.576704
block:	2	thread:	128	time:	0.619517
block:	2	thread:	256	time:	0.588954
block:	2	thread:	512	time:	0.001693
block:	2	thread:	1024	time:	0.001837
block:	4	thread:	32	time:	0.577040
block:	4	thread:	64	time:	0.618829
block:	4	thread:	128	time:	0.578493
block:	4	thread:	256	time:	0.675069
block:	4	thread:	512	time:	0.001869
block:	4	thread:	1024	time:	0.001923
block:	8	thread:	32	time:	0.648061
block:	8	thread:	64	time:	0.576982
block:	8	thread:	128	time:	0.632989
block:	8	thread:	256	time:	0.589354

record.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

block:	1	thread:	32	time:	0.615520
block:	1	thread:	64	time:	0.614752
block:	1	thread:	128	time:	0.618496
block:	1	thread:	256	time:	0.632672
block:	1	thread:	512	time:	0.002048
block:	1	thread:	1024	time:	0.001664
block:	2	thread:	32	time:	0.609984
block:	2	thread:	64	time:	0.608480
block:	2	thread:	128	time:	0.612576
block:	2	thread:	256	time:	0.628672
block:	2	thread:	512	time:	0.001728
block:	2	thread:	1024	time:	0.001760
block:	4	thread:	32	time:	0.609792
block:	4	thread:	64	time:	0.609664
block:	4	thread:	128	time:	0.802880
block:	4	thread:	256	time:	0.626976
block:	4	thread:	512	time:	0.001760
block:	4	thread:	1024	time:	0.002048
block:	8	thread:	32	time:	0.845376
block:	8	thread:	64	time:	0.610272
block:	8	thread:	128	time:	0.614400
block:	8	thread:	256	time:	0.627392

결론 및 향후 과제

- 동일한 Coarse grain 구현이지만, PTX 어셈블리의 결과물이 빠름
- SHAKE-128 Fine grain을 시도
  - 현재는 어느 정도 완성이 되었으나, 구현 적합성이 통과하지 못함
  - 이를 수정하기 위해 다양한 방법을 시도 중