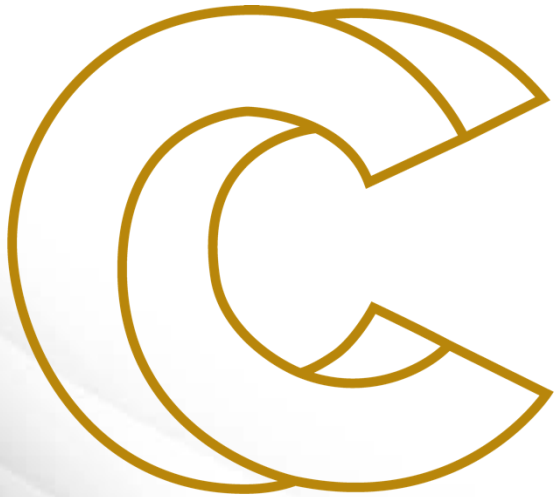


EURO^2

GPU Optimization of Advanced Encryption Standard

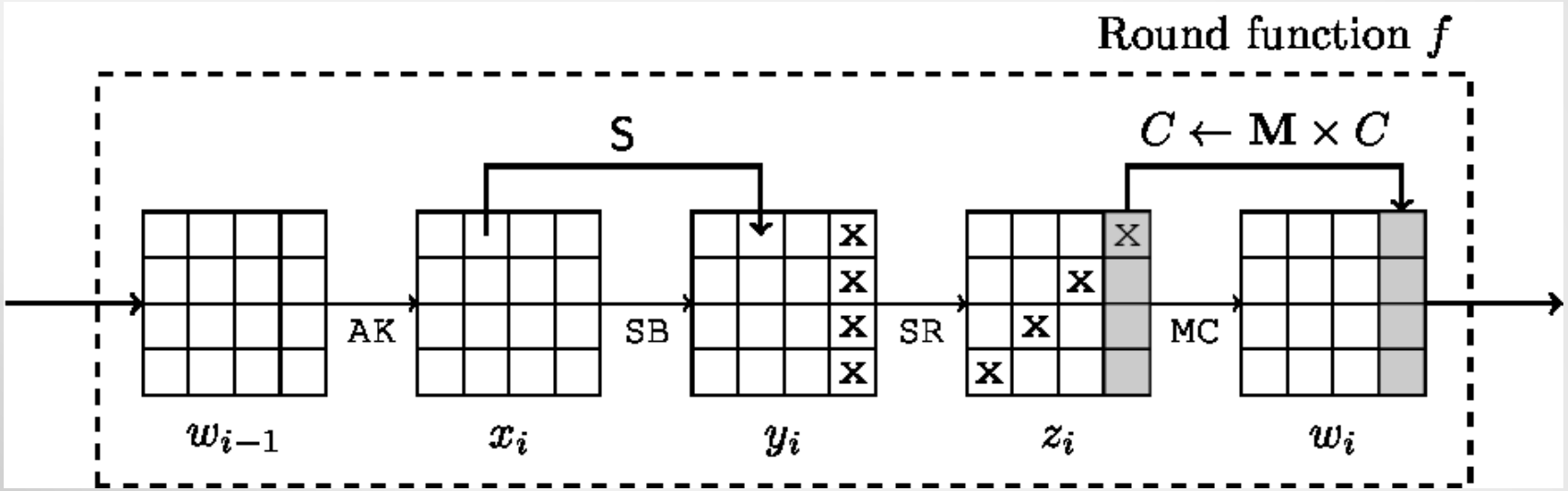
Cihangir Tezcan, PhD

Graduate School of Informatics, METU, Ankara

ncc@ulakbim.gov.tr

Lesson 5: CUDA Optimization of AES

Advanced Encryption Standard (AES)

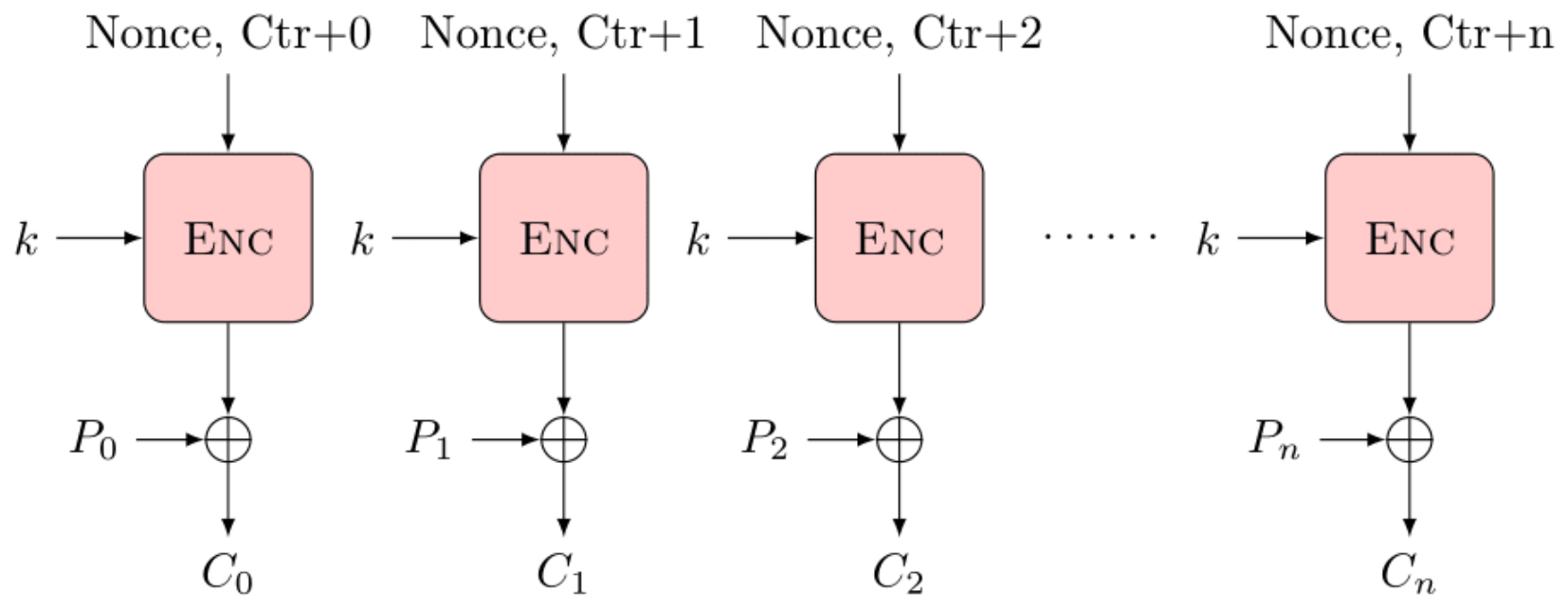


Three implementation techniques are common for GPUs

1. **Naive**
2. **Table based** *(bottleneck: bank conflicts)*
3. **Bitsliced** *(bottleneck: available register count)*

Lesson 5: CUDA Optimization of AES

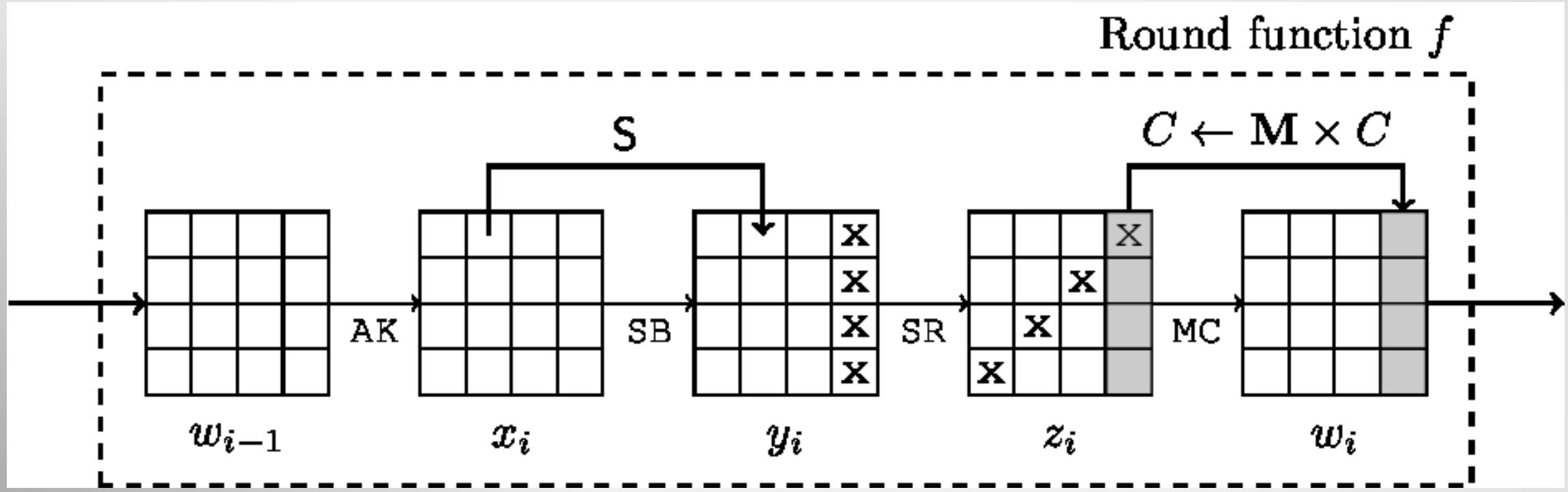
Advanced Encryption Standard (AES)



Let each GPU thread encrypt different counter and send the result to RAM to avoid plaintext transfer from RAM to GPU.

Lesson 5: CUDA Optimization of AES

Advanced Encryption Standard (AES)



Lesson 5: CUDA Optimization of AES

EURO²

Table-based Implementation

$$T_0[x] = \begin{bmatrix} 2 \cdot S[x] \\ S[x] \\ S[x] \\ 3 \cdot S[x] \end{bmatrix} \quad T_1[x] = \begin{bmatrix} 3 \cdot S[x] \\ 2 \cdot S[x] \\ S[x] \\ S[x] \end{bmatrix}$$

$$T_2[x] = \begin{bmatrix} S[x] \\ 3 \cdot S[x] \\ 2 \cdot S[x] \\ S[x] \end{bmatrix} \quad T_3[x] = \begin{bmatrix} S[x] \\ S[x] \\ 3 \cdot S[x] \\ 2 \cdot S[x] \end{bmatrix}$$

```
t0 = T0[c0>>24] ^ T1[(c1>>16)&0xFF] ^ T2[(c2>>8)&0xFF] ^ T3[c3&0xFF] ^ rk0;
t1 = T0[c1>>24] ^ T1[(c2>>16)&0xFF] ^ T2[(c3>>8)&0xFF] ^ T3[c0&0xFF] ^ rk1;
t2 = T0[c2>>24] ^ T1[(c3>>16)&0xFF] ^ T2[(c0>>8)&0xFF] ^ T3[c1&0xFF] ^ rk2;
t3 = T0[c3>>24] ^ T1[(c0>>16)&0xFF] ^ T2[(c1>>8)&0xFF] ^ T3[c2&0xFF] ^ rk3;
```

Lesson 5: CUDA Optimization of AES

Table0: 32-bit output for byte input from the 0th row:

```
u32 table0[256]={0xc66363a5, 0xf87c7c84, 0xee777799, 0xf67b7b8d, 0xfff2f20d, 0xd66b6bbd, 0xde6f6fb1, 0x91c5c554, 0x60303050, 0x20 10103,
        0xce6767a9, 0x562b2b7d, 0xe7fefe19, 0xb5d7d762, 0x4dababe6, 0xec76769a, 0x8fcaca45, 0x1f82829d, 0x89c9c940, 0xfa7d7d87, 0xeffafa15,
        0xb25959eb, 0x8e4747c9, 0xfbf0f00b, 0x41adadec, 0xb3d4d467, 0x5fa2a2fd, 0x45afafea, 0x239c9cbf, 0x53a4a4f7, 0xe4727296, 0x9bc0c05b,
        0x75b7b7c2, 0xe1fdfd1c, 0x3d9393ae, 0x4c26266a, 0x6c36365a, 0x7e3f3f41, 0xf5f7f702, 0x83cccc4f, 0x6834345c, 0x51a5a5f4, 0xd1e5e534,
        0xf9f1f108, 0xe2717193, 0xabd8d873, 0x62313153, 0x2a15153f, 0x804040c, 0x95c7c752, 0x46232365, 0x9dc3c35e, 0x30181828, 0x379696a1,
        0xa05050f, 0x2f9a9ab5, 0xe070709, 0x24121236, 0x1b80809b, 0xdfe2e23d, 0xcdebeb26, 0x4e272769, 0x7fb2b2cd, 0xea75759f, 0x1209091b,
        0x1d83839e, 0x582c2c74, 0x341a1a2e, 0x361b1b2d, 0xdc6e6eb2, 0xb45a5aee, 0x5ba0a0fb, 0xa45252f6, 0x763b3b4d, 0xb7d6d661, 0x7db3b3ce,
        0x5229297b, 0xdde3e33e, 0x5e2f2f71, 0x13848497, 0xa65353f5, 0xb9d1d168, 0x0, 0xc1eded2c, 0x40202060, 0xe3fcfc1f, 0x79b1b1c8,
        0xb65b5bed, 0xd46a6abe, 0x8dcbcb46, 0x67bebed9, 0x7239394b, 0x944a4ade, 0x984c4cd4, 0xb05858e8, 0x85cfcf4a, 0xbbd0d06b, 0xc5efef2a,
        0x4faaaae5, 0xedfbfb16, 0x864343c5, 0x9a4d4dd7, 0x66333355, 0x11858594, 0x8a4545cf, 0xe9f9f910, 0x4020206, 0xfe7f7f81, 0xa05050f0,
        0x783c3c44, 0x259f9fba, 0x4ba8a8e3, 0xa25151f3, 0x5da3a3fe, 0x804040c0, 0x58f8f8a, 0x3f9292ad, 0x219d9dbc, 0x70383848, 0xf1f5f504,
        0x63bcbcdf, 0x77b6b6c1, 0xafdada75, 0x42212163, 0x20101030, 0xe5ffff1a, 0xfdf3f30e, 0xbfd2d26d, 0x81cdcd4c, 0x180c0c14, 0x26131335,
        0xc3ecec2f, 0xbe5f5fe1, 0x359797a2, 0x884444cc, 0x2e171739, 0x93c4c457, 0x55a7a7f2, 0xfc7e7e82, 0x7a3d3d47, 0xc86464ac, 0xba5d5de7,
        0x3219192b, 0xe6737395, 0xc06060a0, 0x19818198, 0x9e4f4fd1, 0xa3dcdc7f, 0x44222266, 0x542a2a7e, 0x3b9090ab, 0xb888883, 0x8c4646ca,
        0xc7eeee29, 0x6bb8b8d3, 0x2814143c, 0xa7dede79, 0xbc5e5ee2, 0x160b0b1d, 0xaddbdb76, 0xdbe0e03b, 0x64323256, 0x743a3a4e, 0x140a0a1e,
        0x924949db, 0xc06060a, 0x4824246c, 0xb85c5ce4, 0x9fc2c25d, 0xbdd3d36e, 0x43acacef, 0xc46262a6, 0x399191a8, 0x319595a4, 0xd3e4e437,
        0xf279798b, 0xd5e7e732, 0x8bc8c843, 0x6e373759, 0xda6d6db7, 0x18d8d8c, 0xb1d5d564, 0x9c4e4ed2, 0x49a9a9e0, 0xd86c6cb4, 0xac5656fa,
        0xf3f4f407, 0xcfeaea25, 0xca6565af, 0xf47a7a8e, 0x47aeaee9, 0x10080818, 0x6fbabad5, 0xf0787888, 0x4a25256f, 0x5c2e2e72, 0x381c1c24,
        0x57a6a6f1, 0x73b4b4c7, 0x97c6c651, 0xcbe8e823, 0xa1dddd7c, 0xe874749c, 0x3e1f1f21, 0x964b4bdd, 0x61bdbddc, 0xd8b8b86, 0xf8a8a85,
        0xe0707090, 0x7c3e3e42, 0x71b5b5c4, 0xcc6666aa, 0x904848d8, 0x6030305, 0xf7f6f601, 0x1c0e0e12, 0xc26161a3, 0x6a35355f, 0xae5757f9,
        0x69b9b9d0, 0x17868691, 0x99c1c158, 0x3a1d1d27, 0x279e9eb9, 0xd9e1e138, 0xebf8f813, 0x2b9898b3, 0x22111133, 0xd26969bb, 0xa9d9d970,
        0x78e8e89, 0x339494a7, 0x2d9b9bb6, 0x3c1e1e22, 0x15878792, 0xc9e9e920, 0x87cece49, 0xaa5555ff, 0x50282878, 0xa5dfdf7a, 0x38c8c8f,
        0x59a1a1f8, 0x9898980, 0x1a0d0d17, 0x65bfbfda, 0xd7e6e631, 0x844242c6, 0xd06868b8, 0x824141c3, 0x299999b0, 0x5a2d2d77, 0x1e0f0f11,
        0x7bb0b0cb, 0xa85454fc, 0x6dbbbbd6, 0x2c16163a};
```

Lesson 5: CUDA Optimization of AES

EURO²

Table1: 32-bit output for byte input from the 1st row:

```
u32 table1[256]={0xa5c66363, 0x84f87c7c, 0x99ee7777, 0x8df67b7b, 0xdfff2f2, 0xbdd66b6b, 0xb1de6f6f, 0x5491c5c5, 0x50603030, 0x302 0101,
        0xa9ce6767, 0x7d562b2b, 0x19e7fefe, 0x62b5d7d7, 0xe64dabab, 0x9aec7676, 0x458fcaca, 0x9d1f8282, 0x4089c9c9, 0x87fa7d7d, 0x15effafa,
        0xebb25959, 0xc98e4747, 0xbfbf0f0, 0xec41adad, 0x67b3d4d4, 0xfd5fa2a2, 0xea45afaf, 0xbf239c9c, 0xf753a4a4, 0x96e47272, 0x5b9bc0c0,
        0xc275b7b7, 0x1ce1fdfd, 0xae3d9393, 0x6a4c2626, 0x5a6c3636, 0x417e3f3f, 0x2f5f7f7, 0x4f83cccc, 0x5c683434, 0xf451a5a5, 0x34d1e5e5,
        0x8f9f1f1, 0x93e27171, 0x73abd8d8, 0x53623131, 0x3f2a1515, 0xc080404, 0x5295c7c7, 0x65462323, 0x5e9dc3c3, 0x28301818, 0xa1379696,
        0xf0a0505, 0xb52f9a9a, 0x90e0707, 0x36241212, 0x9b1b8080, 0x3ddfe2e2, 0x26cdebeb, 0x694e2727, 0xcd7fb2b2, 0x9fea7575, 0x1b120909,
        0x9e1d8383, 0x74582c2c, 0x2e341a1a, 0x2d361b1b, 0xb2dc6e6e, 0xeeb45a5a, 0xfb5ba0a0, 0xf6a45252, 0x4d763b3b, 0x61b7d6d6, 0xce7db3b3,
        0x7b522929, 0x3edde3e3, 0x715e2f2f, 0x97138484, 0xf5a65353, 0x68b9d1d1, 0x0, 0x2cc1eded, 0x60402020, 0x1fe3fcfc, 0xc879b1b1,
        0xedb65b5b, 0xbed46a6a, 0x468dcbcb, 0xd967bebe, 0x4b723939, 0xde944a4a, 0xd4984c4c, 0xe8b05858, 0x4a85cfcf, 0x6bbbd0d0, 0x2ac5efef,
        0xe54faaaa, 0x16edfbfb, 0xc5864343, 0xd79a4d4d, 0x55663333, 0x94118585, 0xcf8a4545, 0x10e9f9f9, 0x6040202, 0x81fe7f7f, 0xf0a05050,
        0x44783c3c, 0xba259f9f, 0xe34ba8a8, 0xf3a25151, 0xfe5da3a3, 0xc0804040, 0x8a058f8f, 0xad3f9292, 0xbc219d9d, 0x48703838, 0x4f1f5f5,
        0xdf63bcbc, 0xc177b6b6, 0x75afdada, 0x63422121, 0x30201010, 0x1ae5ffff, 0xefdf3f3, 0x6dbfd2d2, 0x4c81cdcd, 0x14180c0c, 0x35261313,
        0x2fc3ecec, 0xe1be5f5f, 0xa2359797, 0xcc884444, 0x392e1717, 0x5793c4c4, 0xf255a7a7, 0x82fc7e7e, 0x477a3d3d, 0xacc86464, 0xe7ba5d5d,
        0x2b321919, 0x95e67373, 0xa0c06060, 0x98198181, 0xd19e4f4f, 0x7fa3dcdc, 0x66442222, 0x7e542a2a, 0xab3b9090, 0x830b8888, 0xca8c4646,
        0x29c7eeee, 0xd36bb8b8, 0x3c281414, 0x79a7dede, 0xe2bc5e5e, 0x1d160b0b, 0x76addbdb, 0x3bdbe0e0, 0x56643232, 0x4e743a3a, 0x1e140a0a,
        0xdb924949, 0xa0c0606, 0x6c482424, 0xe4b85c5c, 0x5d9fc2c2, 0x6ebdd3d3, 0xef43acac, 0xa6c46262, 0xa8399191, 0xa4319595, 0x37d3e4e4,
        0x8bf27979, 0x32d5e7e7, 0x438bc8c8, 0x596e3737, 0xb7da6d6d, 0x8c018d8d, 0x64b1d5d5, 0xd29c4e4e, 0xe049a9a9, 0xb4d86c6c, 0xfaac5656,
        0x7f3f4f4, 0x25cfeaea, 0xafca6565, 0x8ef47a7a, 0xe947aeae, 0x18100808, 0xd56fbaba, 0x88f07878, 0x6f4a2525, 0x725c2e2e, 0x24381c1c,
        0xf157a6a6, 0xc773b4b4, 0x5197c6c6, 0x23cbe8e8, 0x7ca1dddd, 0x9ce87474, 0x213e1f1f, 0xdd964b4b, 0xdc61bdbd, 0x860d8b8b, 0x850f8a8a,
        0x90e07070, 0x427c3e3e, 0xc471b5b5, 0xaacc6666, 0xd8904848, 0x5060303, 0x1f7f6f6, 0x121c0e0e, 0xa3c26161, 0x5f6a3535, 0xf9ae5757,
        0xd069b9b9, 0x91178686, 0x5899c1c1, 0x273a1d1d, 0xb9279e9e, 0x38d9e1e1, 0x13ebf8f8, 0xb32b9898, 0x33221111, 0xbbd26969, 0x70a9d9d9,
        0x89078e8e, 0xa7339494, 0xb62d9b9b, 0x223c1e1e, 0x92158787, 0x20c9e9e9, 0x4987cece, 0xffaa5555, 0x78502828, 0x7aa5dfdf, 0x8f038c8c,
        0xf859a1a1, 0x80098989, 0x171a0d0d, 0xda65bfbf, 0x31d7e6e6, 0xc6844242, 0xb8d06868, 0xc3824141, 0xb0299999, 0x775a2d2d, 0x111e0f0f,
        0xcb7bb0b0, 0xfca85454, 0xd66dbbbb, 0x3a2c1616};
```

Lesson 5: CUDA Optimization of AES

Table2: 32-bit output for byte input from the 2nd row:

```
u32 table2[256]={0x63a5c663, 0x7c84f87c, 0x7799ee77, 0x7b8df67b, 0xf20dfff2, 0x6bbdd66b, 0x6fb1de6f, 0xc55491c5, 0x30506030, 0x10 30201,
        0x67a9ce67, 0x2b7d562b, 0xfe19e7fe, 0xd762b5d7, 0xabe64dab, 0x769aec76, 0xca458fca, 0x829d1f82, 0xc94089c9, 0x7d87fa7d, 0xfa15effa,
        0x59ebb259, 0x47c98e47, 0xf00bfbf0, 0xadec41ad, 0xd467b3d4, 0xa2fd5fa2, 0xafea45af, 0x9cbf239c, 0xa4f753a4, 0x7296e472, 0xc05b9bc0,
        0xb7c275b7, 0xfd1ce1fd, 0x93ae3d93, 0x266a4c26, 0x365a6c36, 0x3f417e3f, 0xf702f5f7, 0xcc4f83cc, 0x345c6834, 0xa5f451a5, 0xe534d1e5,
        0xf108f9f1, 0x7193e271, 0xd873abd8, 0x31536231, 0x153f2a15, 0x40c0804, 0xc75295c7, 0x23654623, 0xc35e9dc3, 0x18283018, 0x96a13796,
        0x50f0a05, 0x9ab52f9a, 0x7090e07, 0x12362412, 0x809b1b80, 0xe23ddfe2, 0xeb26cdeb, 0x27694e27, 0xb2cd7fb2, 0x759fea75, 0x91b1209,
        0x839e1d83, 0x2c74582c, 0x1a2e341a, 0x1b2d361b, 0x6eb2dc6e, 0x5aeeb45a, 0xa0fb5ba0, 0x52f6a452, 0x3b4d763b, 0xd661b7d6, 0xb3ce7db3,
        0x297b5229, 0xe33edde3, 0x2f715e2f, 0x84971384, 0x53f5a653, 0xd168b9d1, 0x0, 0xed2cc1ed, 0x20604020, 0xfc1fe3fc, 0xb1c879b1,
        0x5bedb65b, 0x6abed46a, 0xcb468dcb, 0xbed967be, 0x394b7239, 0x4ade944a, 0x4cd4984c, 0x58e8b058, 0xcf4a85cf, 0xd06bbbd0, 0xef2ac5ef,
        0xaae54faa, 0xfb16edfb, 0x43c58643, 0x4dd79a4d, 0x33556633, 0x85941185, 0x45cf8a45, 0xf910e9f9, 0x2060402, 0x7f81fe7f, 0x50f0a050,
        0x3c44783c, 0x9fba259f, 0xa8e34ba8, 0x51f3a251, 0xa3fe5da3, 0x40c08040, 0x8f8a058f, 0x92ad3f92, 0x9dbc219d, 0x38487038, 0xf504f1f5,
        0xbcdf63bc, 0xb6c177b6, 0xda75afda, 0x21634221, 0x10302010, 0xff1ae5ff, 0xf30efdf3, 0xd26dbfd2, 0xcd4c81cd, 0xc14180c, 0x13352613,
        0xec2fc3ec, 0x5fe1be5f, 0x97a23597, 0x44cc8844, 0x17392e17, 0xc45793c4, 0xa7f255a7, 0x7e82fc7e, 0x3d477a3d, 0x64acc864, 0x5de7ba5d,
        0x192b3219, 0x7395e673, 0x60a0c060, 0x81981981, 0x4fd19e4f, 0xdc7fa3dc, 0x22664422, 0x2a7e542a, 0x90ab3b90, 0x88830b88, 0x46ca8c46,
        0xee29c7ee, 0xb8d36bb8, 0x143c2814, 0xde79a7de, 0x5ee2bc5e, 0xb1d160b, 0xdb76addb, 0xe03bdbe0, 0x32566432, 0x3a4e743a, 0xa1e140a,
        0x49db9249, 0x60a0c06, 0x246c4824, 0x5ce4b85c, 0xc25d9fc2, 0xd36ebdd3, 0xacef43ac, 0x62a6c462, 0x91a83991, 0x95a43195, 0xe437d3e4,
        0x798bf279, 0xe732d5e7, 0xc8438bc8, 0x37596e37, 0x6db7da6d, 0x8d8c018d, 0xd564b1d5, 0x4ed29c4e, 0xa9e049a9, 0x6cb4d86c, 0x56faac56,
        0xf407f3f4, 0xea25cfea, 0x65afca65, 0x7a8ef47a, 0xaee947ae, 0x8181008, 0xbad56fba, 0x7888f078, 0x256f4a25, 0x2e725c2e, 0x1c24381c,
        0xa6f157a6, 0xb4c773b4, 0xc65197c6, 0xe823cbe8, 0xdd7ca1dd, 0x749ce874, 0x1f213e1f, 0x4bdd964b, 0xbddc61bd, 0x8b860d8b, 0x8a850f8a,
        0x7090e070, 0x3e427c3e, 0xb5c471b5, 0x66aacc66, 0x48d89048, 0x3050603, 0xf601f7f6, 0xe121c0e, 0x61a3c261, 0x355f6a35, 0x57f9ae57,
        0xb9d069b9, 0x86911786, 0xc15899c1, 0x1d273a1d, 0x9eb9279e, 0xe138d9e1, 0xf813ebf8, 0x98b32b98, 0x11332211, 0x69bbd269, 0xd970a9d9,
        0x8e89078e, 0x94a73394, 0x9bb62d9b, 0x1e223c1e, 0x87921587, 0xe920c9e9, 0xce4987ce, 0x55ffaa55, 0x28785028, 0xdf7aa5df, 0x8c8f038c,
        0xa1f859a1, 0x89800989, 0xd171a0d, 0xbfda65bf, 0xe631d7e6, 0x42c68442, 0x68b8d068, 0x41c38241, 0x99b02999, 0x2d775a2d, 0xf111e0f,
        0xb0cb7bb0, 0x54fca854, 0xbbd66dbb, 0x163a2c16};
```

Lesson 5: CUDA Optimization of AES

Table3: 32-bit output for byte input from the 3rd row:

```
u32 table3[256]={0x6363a5c6, 0x7c7c84f8, 0x777799ee, 0x7b7b8df6, 0xf2f20dff, 0x6b6bbdd6, 0x6f6fb1de, 0xc5c55491, 0x30305060, 0x10 10302,
        0x6767a9ce, 0x2b2b7d56, 0xfefe19e7, 0xd7d762b5, 0xababe64d, 0x76769aec, 0xcaca458f, 0x82829d1f, 0xc9c94089, 0x7d7d87fa, 0xfafa15ef,
        0x5959ebb2, 0x4747c98e, 0xf0f00bfb, 0xadadec41, 0xd4d467b3, 0xa2a2fd5f, 0xafafea45, 0x9c9cbf23, 0xa4a4f753, 0x727296e4, 0xc0c05b9b,
        0xb7b7c275, 0xfdfd1ce1, 0x9393ae3d, 0x26266a4c, 0x36365a6c, 0x3f3f417e, 0xf7f702f5, 0xcccc4f83, 0x34345c68, 0xa5a5f451, 0xe5e534d1,
        0xf1f108f9, 0x717193e2, 0xd8d873ab, 0x31315362, 0x15153f2a, 0x4040c08, 0xc7c75295, 0x23236546, 0xc3c35e9d, 0x18182830, 0x9696a137,
        0x5050f0a, 0x9a9ab52f, 0x707090e, 0x12123624, 0x80809b1b, 0xe2e23ddf, 0xebeb26cd, 0x2727694e, 0xb2b2cd7f, 0x75759fea, 0x9091b12,
        0x83839e1d, 0x2c2c7458, 0x1a1a2e34, 0x1b1b2d36, 0x6e6eb2dc, 0x5a5aeeb4, 0xa0a0fb5b, 0x5252f6a4, 0x3b3b4d76, 0xd6d661b7, 0xb3b3ce7d,
        0x29297b52, 0xe3e33edd, 0x2f2f715e, 0x84849713, 0x5353f5a6, 0xd1d168b9, 0x0, 0xeded2cc1, 0x20206040, 0xfcfc1fe3, 0xb1b1c879,
        0x5b5bedb6, 0x6a6abed4, 0xcbcb468d, 0xbebed967, 0x39394b72, 0x4a4ade94, 0x4c4cd498, 0x5858e8b0, 0xcfcf4a85, 0xd0d06bbb, 0xefef2ac5,
        0xaaaae54f, 0xfbfb16ed, 0x4343c586, 0x4d4dd79a, 0x33335566, 0x85859411, 0x4545cf8a, 0xf9f910e9, 0x2020604, 0x7f7f81fe, 0x5050f0a0,
        0x3c3c4478, 0x9f9fba25, 0xa8a8e34b, 0x5151f3a2, 0xa3a3fe5d, 0x4040c080, 0x8f8f8a05, 0x9292ad3f, 0x9d9dbc21, 0x38384870, 0xf5f504f1,
        0xbcbcdf63, 0xb6b6c177, 0xdada75af, 0x21216342, 0x10103020, 0xffff1ae5, 0xf3f30efd, 0xd2d26dbf, 0xcdcd4c81, 0xc0c1418, 0x13133526,
        0xecec2fc3, 0x5f5fe1be, 0x9797a235, 0x4444cc88, 0x1717392e, 0xc4c45793, 0xa7a7f255, 0x7e7e82fc, 0x3d3d477a, 0x6464acc8, 0x5d5de7ba,
        0x19192b32, 0x737395e6, 0x6060a0c0, 0x81819819, 0x4f4fd19e, 0xdcdc7fa3, 0x22226644, 0x2a2a7e54, 0x9090ab3b, 0x8888830b, 0x4646ca8c,
        0xeeee29c7, 0xb8b8d36b, 0x14143c28, 0xdede79a7, 0x5e5ee2bc, 0xb0b1d16, 0xdbdb76ad, 0xe0e03bdb, 0x32325664, 0x3a3a4e74, 0xa0a1e14,
        0x4949db92, 0x6060a0c, 0x24246c48, 0x5c5ce4b8, 0xc2c25d9f, 0xd3d36ebd, 0xacacef43, 0x6262a6c4, 0x9191a839, 0x9595a431, 0xe4e437d3,
        0x79798bf2, 0xe7e732d5, 0xc8c8438b, 0x3737596e, 0x6d6db7da, 0x8d8d8c01, 0xd5d564b1, 0x4e4ed29c, 0xa9a9e049, 0x6c6cb4d8, 0x5656faac,
        0xf4f407f3, 0xeaea25cf, 0x6565afca, 0x7a7a8ef4, 0xaeaee947, 0x8081810, 0xbabad56f, 0x787888f0, 0x25256f4a, 0x2e2e725c, 0x1c1c2438,
        0xa6a6f157, 0xb4b4c773, 0xc6c65197, 0xe8e823cb, 0xdddd7ca1, 0x74749ce8, 0x1f1f213e, 0x4b4bdd96, 0xbdbddc61, 0x8b8b860d, 0x8a8a850f,
        0x707090e0, 0x3e3e427c, 0xb5b5c471, 0x6666aacc, 0x4848d890, 0x3030506, 0xf6f601f7, 0xe0e121c, 0x6161a3c2, 0x35355f6a, 0x5757f9ae,
        0xb9b9d069, 0x86869117, 0xc1c15899, 0x1d1d273a, 0x9e9eb927, 0xe1e138d9, 0xf8f813eb, 0x9898b32b, 0x11113322, 0x6969bbd2, 0xd9d970a9,
        0x8e8e8907, 0x9494a733, 0x9b9bb62d, 0x1e1e223c, 0x87879215, 0xe9e920c9, 0xcece4987, 0x5555ffaa, 0x28287850, 0xdfdf7aa5, 0x8c8c8f03,
        0xa1a1f859, 0x89898009, 0xd0d171a, 0xbfbfda65, 0xe6e631d7, 0x4242c684, 0x6868b8d0, 0x4141c382, 0x9999b029, 0x2d2d775a, 0xf0f111e,
        0xb0b0cb7b, 0x5454fca8, 0xbbbbd66d, 0x16163a2c};
```

Lesson 5: CUDA Optimization of AES

One round of AES turns into the following:

```
b[0]= table0[a[0]>>24] ^ table1[(a[1]>>16)&0xff] ^ table2[(a[2]>>8)&0xff] ^ table3[a[3]&0xff] ^ rk0;
b[1]= table0[a[1]>>24] ^ table1[(a[2]>>16)&0xff] ^ table2[(a[3]>>8)&0xff] ^ table3[a[0]&0xff] ^ rk1;
b[2]= table0[a[2]>>24] ^ table1[(a[3]>>16)&0xff] ^ table2[(a[0]>>8)&0xff] ^ table3[a[1]&0xff] ^ rk2;
b[3]= table0[a[3]>>24] ^ table1[(a[0]>>16)&0xff] ^ table2[(a[1]>>8)&0xff] ^ table3[a[2]&0xff] ^ rk3;
```

- Instead of storing the tables in the global memory, we keep them at the shared memory.
- Need 1 KB per table.
- Almost every implementation in the literature uses this approach.
- Different threads in a warp accessing the same data lane causes shared memory bank conflict.

Lesson 5: CUDA Optimization of AES

EURO²

How to Avoid Shared Memory Bank Conflicts

- GPU threads are group into warps of 32 threads and there are 32 data lines for the shared memory
- e.g. T0[0], T0[32], T0[64], T0[96], T0[128], T0[160], T0[192], T0[224] can be accessed from the same data line
- 2 threads using the same data lines turns into serial requests (bank conflict)

Solution:

- Keep 32 copies of each table, each thread in a warp can use its own data lane
- Requires 32 KBs per table, not enough to store 4 tables (128 KBs) in shared memory
- Only keep 32 copies of Table0
- Other tables are byte rotations of Table0

Lesson 5: CUDA Optimization of AES

EURO²

How to Avoid Shared Memory Bank Conflicts

```
__shared__ u32 t0S[TABLE_SIZE][SHARED_MEM_BANK_SIZE];
__shared__ u32 t4S[TABLE_SIZE][S_BOX_BANK_SIZE];
__shared__ u32 rkS[AES_128_KEY_SIZE_INT];

if (threadIdx.x < TABLE_SIZE) {
    for (u8 bankIndex = 0; bankIndex < SHARED_MEM_BANK_SIZE; bankIndex++) {
        t0S[threadIdx.x][bankIndex] = t0G[threadIdx.x];
    }
    for (u8 bankIndex = 0; bankIndex < S_BOX_BANK_SIZE; bankIndex++) {
        t4S[threadIdx.x][bankIndex] = t4G[threadIdx.x];
    }
    if (threadIdx.x < AES_128_KEY_SIZE_INT) {
        rkS[threadIdx.x] = rk[threadIdx.x];
    }
}
```

Lesson 5: CUDA Optimization of AES

EURO²

One round of AES turns into the following:

```
__device__ u32 Shift(u32 x, u32 n) { return (x >> n) | (x << (-n & 31)); }

int warpThreadIndex = threadIdx.x & 31;

t0 = t0S[s0>>24][warpThreadIndex] ^ Shift(t0S[(s1>>16) & 0xFF][warpThreadIndex], 8) ^ Shift(t0S[(s2>>8) &
0xFF][warpThreadIndex], 16) ^ Shift(t0S[s3 & 0xFF][warpThreadIndex], 24) ^ rk0;

t1 = t0S[s1>>24][warpThreadIndex] ^ Shift(t0S[(s2>>16) & 0xFF][warpThreadIndex], 8) ^ Shift(t0S[(s3>>8) &
0xFF][warpThreadIndex], 16) ^ Shift(t0S[s0 & 0xFF][warpThreadIndex], 24) ^ rk1;

t2 = t0S[s2>>24][warpThreadIndex] ^ Shift(t0S[(s3>>16) & 0xFF][warpThreadIndex], 8) ^ Shift(t0S[(s0>>8) &
0xFF][warpThreadIndex], 16) ^ Shift(t0S[s1 & 0xFF][warpThreadIndex], 24) ^ rk2;

t3 = t0S[s3>>24][warpThreadIndex] ^ Shift(t0S[(s0>>16) & 0xFF][warpThreadIndex], 8) ^ Shift(t0S[(s1>>8) &
0xFF][warpThreadIndex], 16) ^ Shift(t0S[s2 & 0xFF][warpThreadIndex], 24) ^ rk3;
```

Next lecture

Encryption Performance of AES on GPUs

- Now that we optimized AES using CUDA, we are going to benchmark encryption performance on various GPUs

Thanks

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia