한양대학교 2021-2 SoC 설계방법론 (최정욱 교수님)

Final project

HLS를 이용한 Sparse Convolution용 CSR 기반 Rule generator 설계

2021.12.22.

Team 1

2020252244 이장환

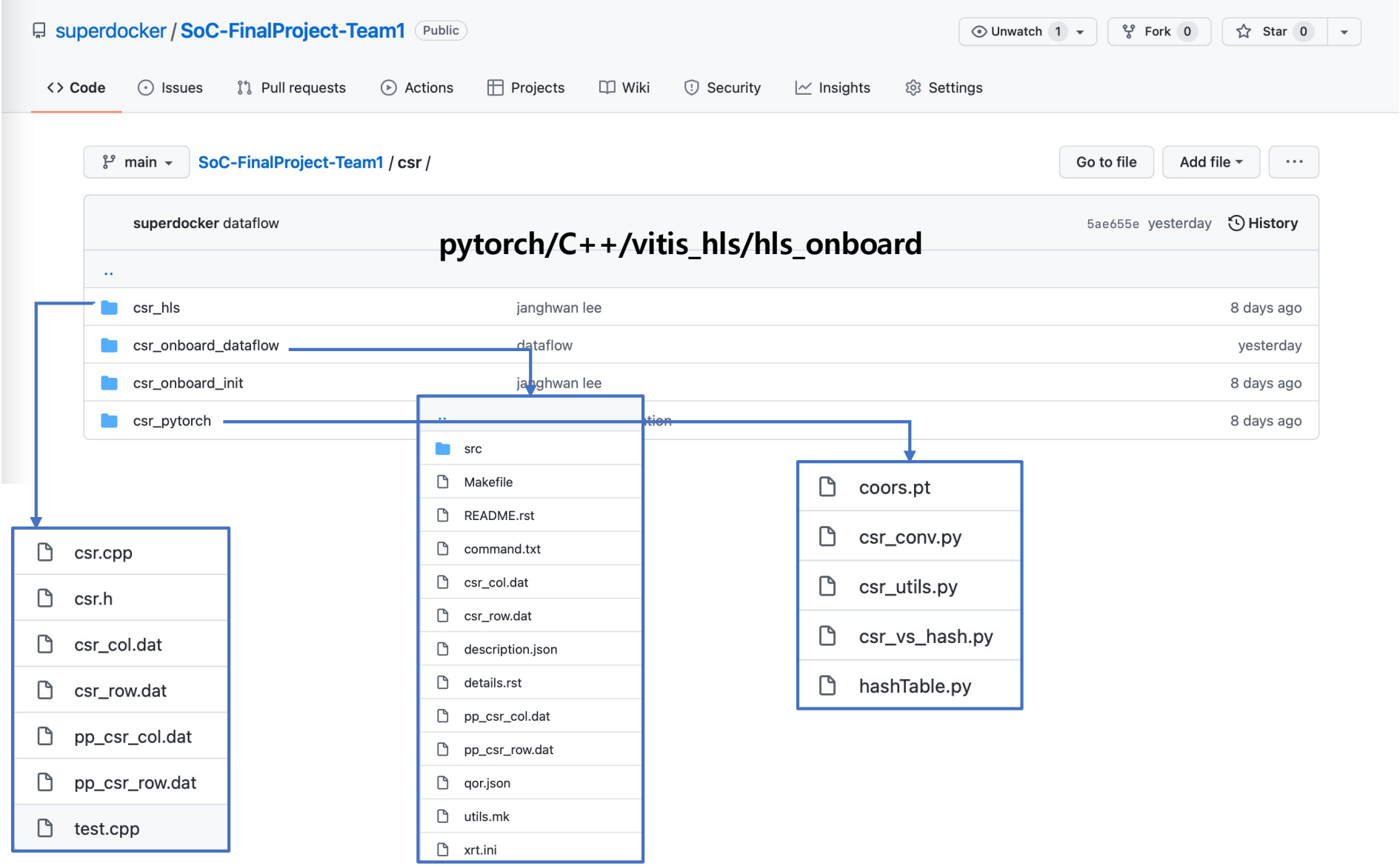
2020298097 박성민

2019200648 이민재

목차

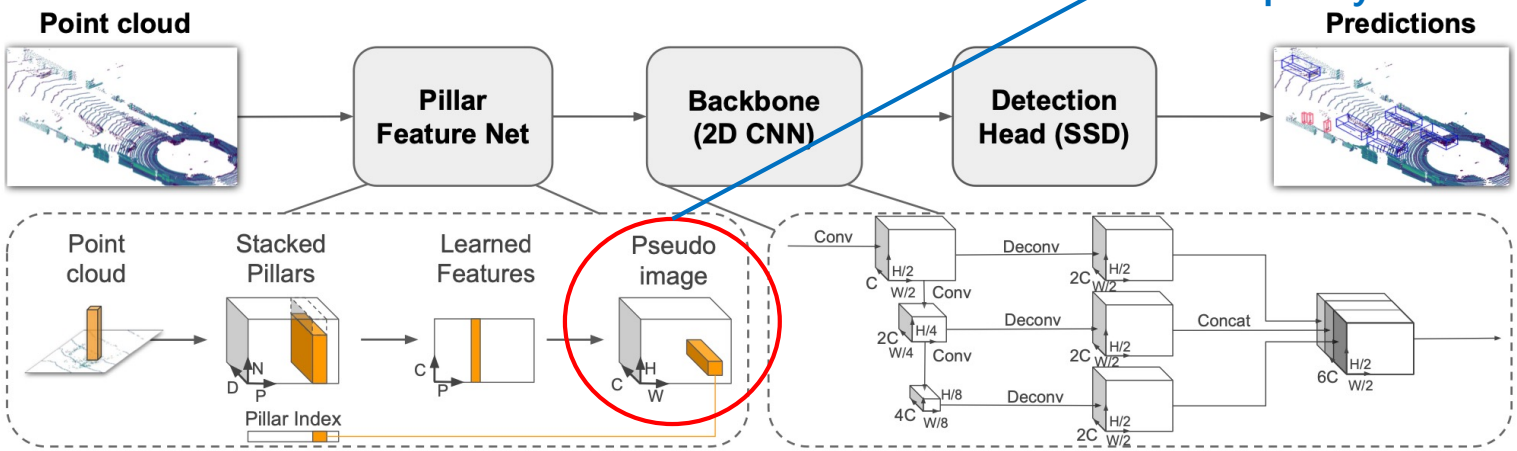
Source codes

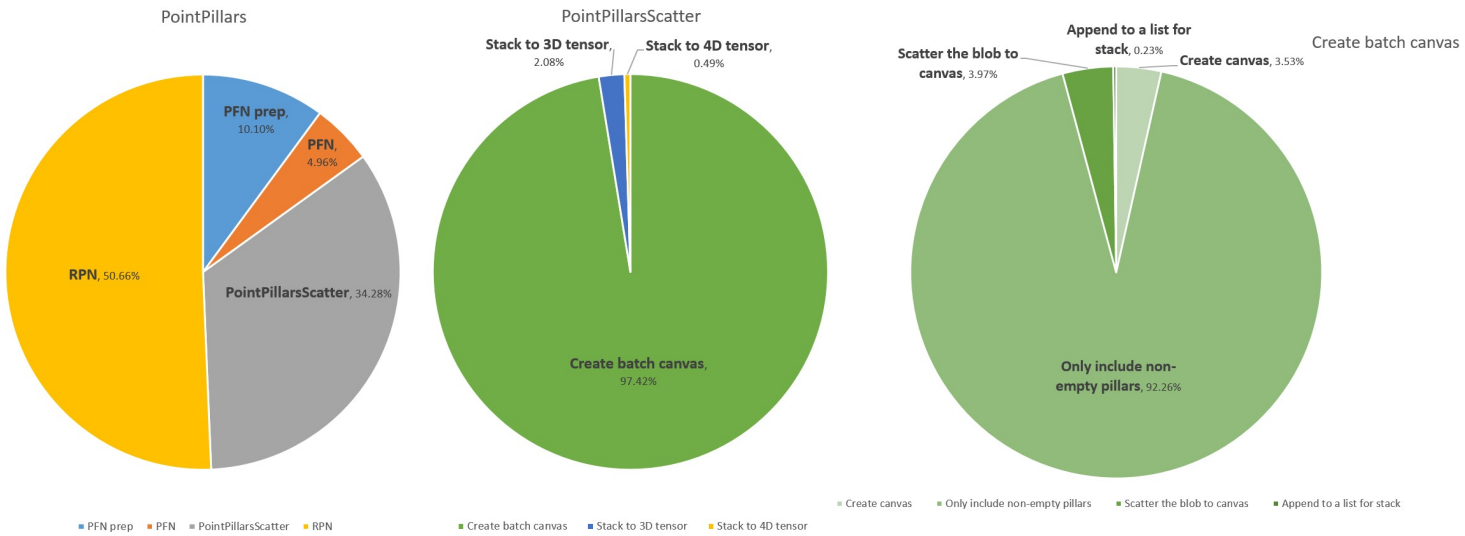
- **Github: https://github.com/superdocker/SoC-FinalProject-Team1/tree/main/csr**



프로젝트 배경

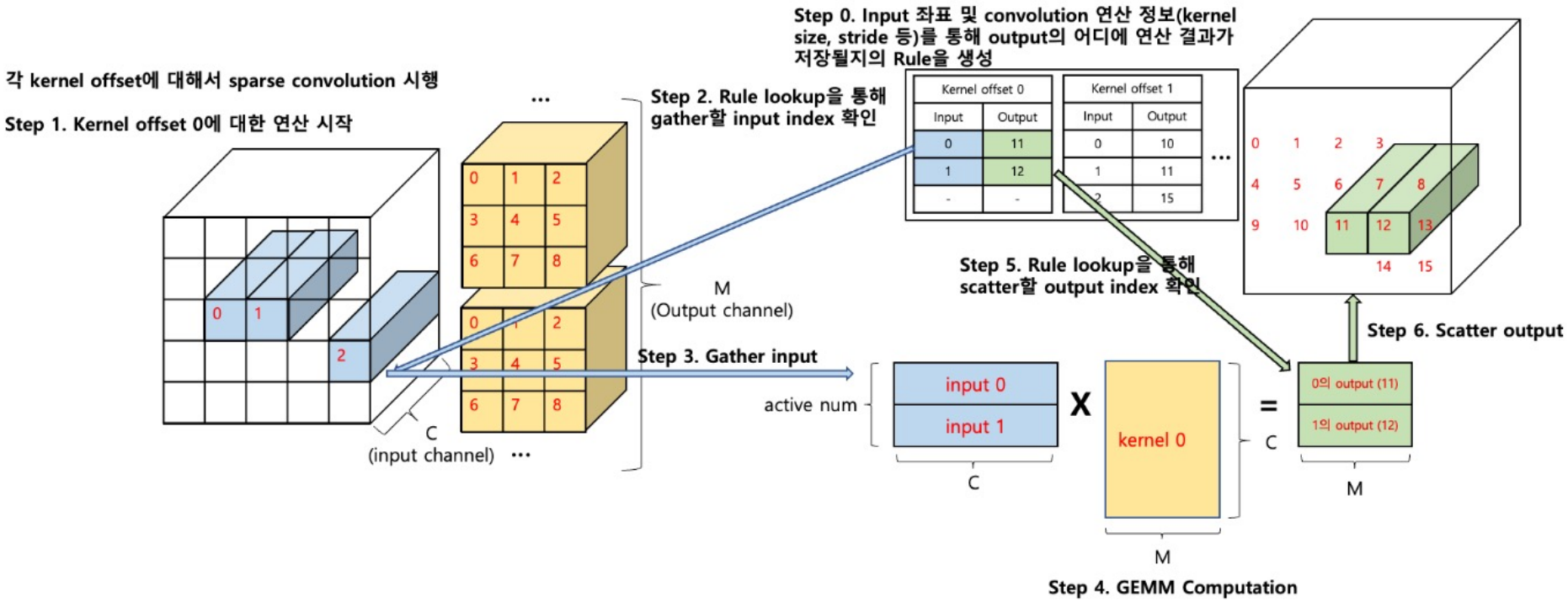
Batch Size = 2
None-zero Pillars = 6452 ([6452, 64])
Pseudo-image = [2, 64, 320, 280]
Sparsity = 1 - 6452/(2*320*280) = 96.40%





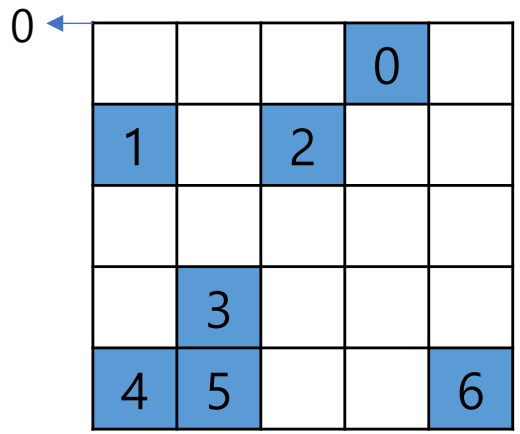
- 3D object detection 모델에 sparse convolution이 많이 사용됨
- PointPillars[1]의 실제 데이터를 pytorch에서 분석해본 결과, 실제 유효한 값은 공간의 4% 정도만 차지
- 그러나 이러한 sparse한 데이터의 위치 정보를 처리하는 데 전체 연산 시간의 34% 정도를 소요할 정도로 비효율적임

Rule을 사용한 Sparse convolution



- 3D object detection의 유명한 모델 중 하나인 SECOND[2]에서는 이러한 sparse convolution 연산을 하기 위해 Rule이라는 look-up table을 사용함
- Rule에는 각 kernel offset에 따라 input index가 output index 어디에 mapping되는지의 정보를 담고 있음

CSR format의 개념



csr_row: [0, 1, 3, 3, 4, 7]
csr_col: [3, 0, 2, 1, 0, 1, 4]

- **CSR(Compressed Sparse Row)는 Sparse encoding의 한 형태로, 각 coordinate들을 [x, y]로 저장하지 않고 CSR row와 CSR col 두 개의 정보로 저장하고 있음**
- **CSR row의 경우 각 row에 데이터가 존재할 경우 첫 번째로 매겨지는 index를 저장하고 있음**
  - **따라서 총 M개의 row가 있다고 할 때 M+1 개의 CSR row element를 가짐**
- **CSR col의 경우 nnz개의 non-zero element가 존재할 경우 각 element의 column index를 저장하고 있음 (위의 예시의 경우, 7개)**
- **M row x N col 의 2D 공간 내에 nnz 개의 non-zero element가 존재한다고 할 때, 기존에는 M*N 개의 위치 data를 저장하고 있다면, CSR format은 M+1+nnz개의 위치 데이터를 저장하고 있음**

프로젝트 목표

			0	
1		2		
	3			
4	5			6

	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

0	1	2	3	4
5	6	7	8	9
10	11	12	13	
14	15	16	17	18
19	20	21	22	23

csr_row: [0, 1, 3, 3, 4, 7]
csr_col: [3, 0, 2, 1, 0, 1, 4]

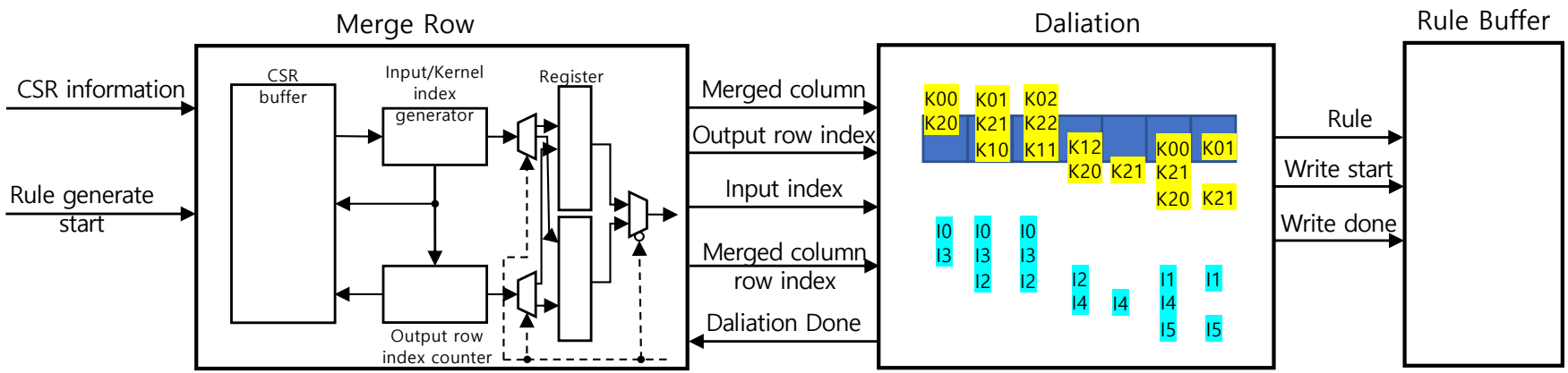
Rule

Kernel 0	Kernel 1	Kernel 2	Kernel 3	Kernel 4	Kernel 5	Kernel 6	Kernel 7	Kernel 8
0 → 9	0 → 8	0 → 7	0 → 4	0 → 3	0 →2	1 →1	1 →0	2 →1
1 → 11	1 → 10	2 → 11	1 → 6	1→5	2 →6	2 →3	2 →2	3 →10
2 → 13	2 → 12	3 → 19	2 → 8	2→7	3 →14	3 →12	3 →11	5 →14
3 → 21	3 → 20		3 → 16	3→15	5 →19	4 →15	4 →14	6 →17
			4 → 20	4→19	6 →22	5 →16	5 →15	
			5 → 21	5→20			6 →18	
				6→23				

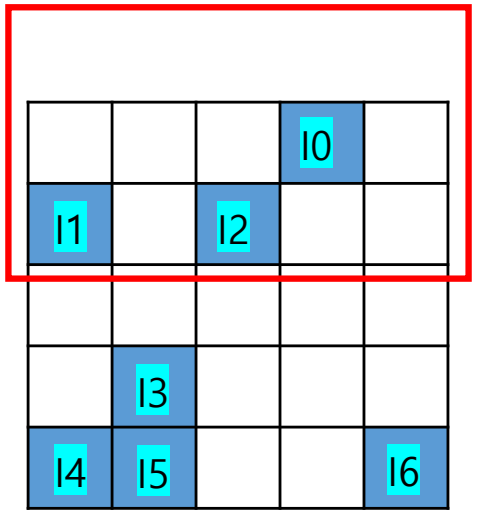
- **CSR format의 특성**
  - **CSR row의 각 요소는 각 row의 시작 input index를 저장하고 있음**
  - **CSR row의 r+1 번째 요소에서 r번째 요소를 빼면 해당 row의 non zero element 수를 알 수 있음**
  - **CSR row의 값을 통해 특정 row에 존재하는 column index들을 알 수 있음**

→ 프로젝트의 목표: CSR format의 특성을 활용하여 Rule을 generation하는 하드웨어 설계

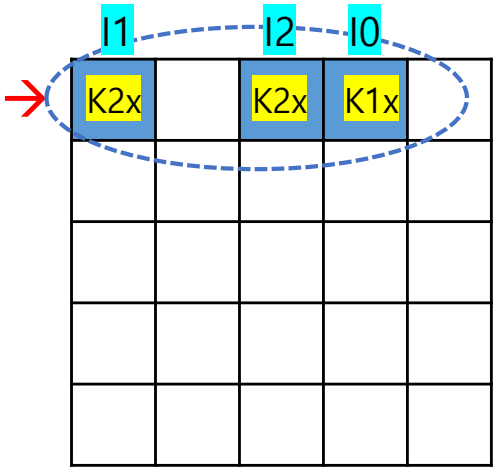
CSR format을 활용한 Rule generator



Step 1. Merge rows



csr_row: [0, 1, 3, 3, 4, 7]
csr_col: [3, 0, 2, 1, 0, 1, 4]



csr_row: [0, 1, 3, 3, 4, 7]
csr_col: [3, 0, 2, 1, 0, 1, 4]

Kernel(vertical)-Input indice pairs

-	-	-	-	-
-	-	0	-	-
1	2	-	-	-

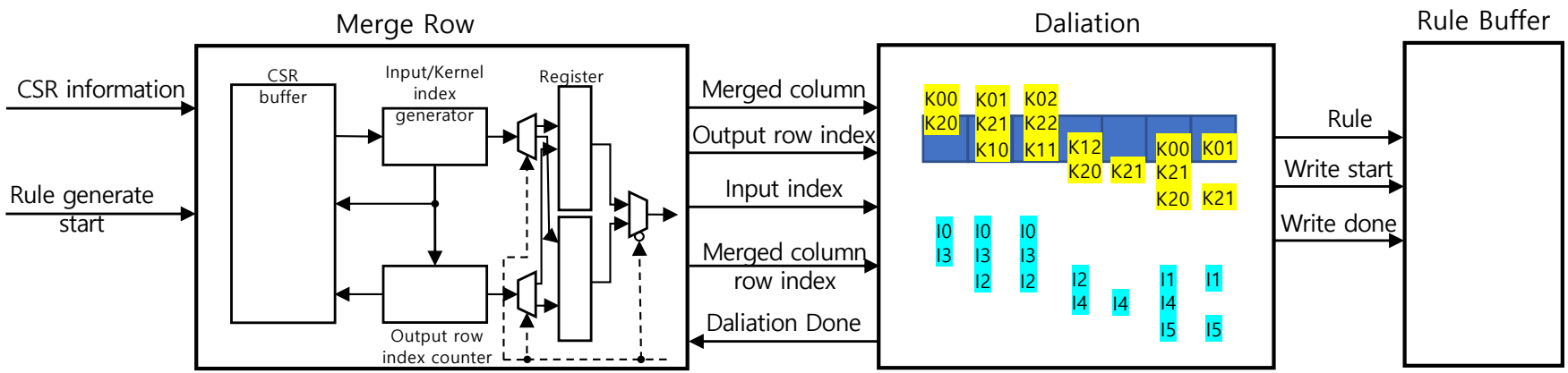
Merged column

Kernel

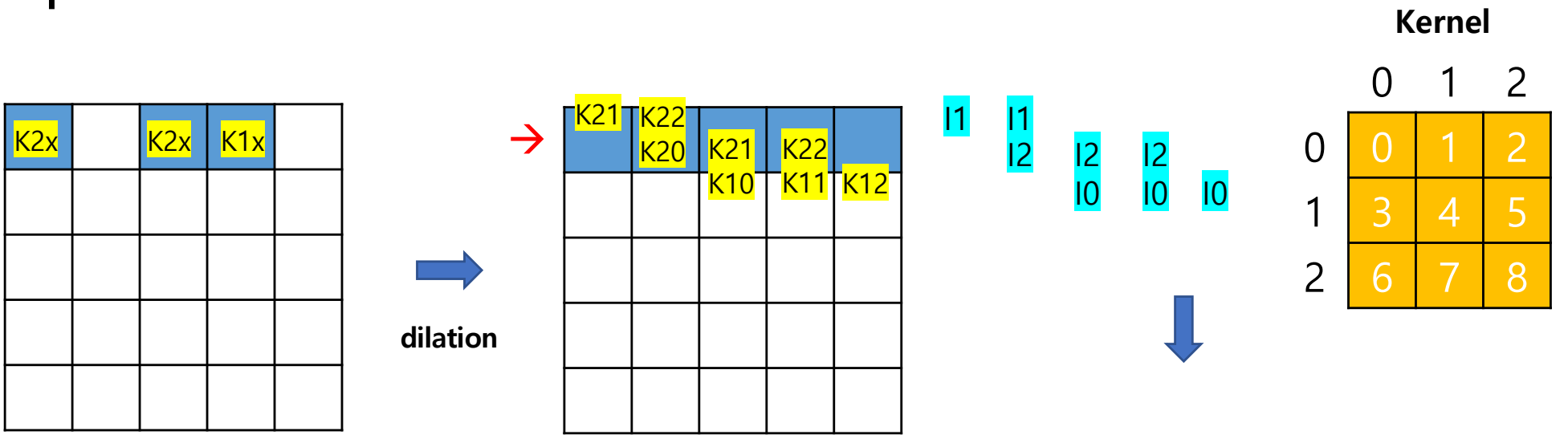
	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

csr_row를 보고 0번째(0)부터 2번째 row의 시작 column idx(3) 전까지 merge (3,0,2)

CSR format을 활용한 Rule generator



Step 2. Dilate column & Generate Rule



Rule

Kernel 3	Kernel 4	Kernel 5	Kernel 6	Kernel 7	Kernel 8
0 → 4	0 → 3	0 →2	1 →1 2 →3	1 →0 2 →2	2 →1

실험 결과

- **Rule generator 구조**

  for (int i=0; i<output row; i++)
  Merge → Dilate → Write Rule

- **Dataset**
  - **Simple example: 32x32 의 2D space에 random generation 한 40개의 데이터 사용 (Sparsity: 96.1%)**
  - **Point Pillars dataset: 320x280의 2D space에 4551개의 pillar 존재 (sparsity 95%)**

Dataset	Simple example	PointPillars data	
SW emulation	7.225 ms	830.828 ms	**x115**
HW emulation	0.096 ms	6.466 ms	**x67**
Run on F1	0.207 ms	8.600 ms	**x42**

Simple example + No dataflow optimization

row 32 num data 40

- **col 32: 0.090 ms**
- **col 64: 0.113 ms**
- **col 128: 0.155 ms**

```
[JHLEE] FPGA kernel exec time is 11.001679 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 22 seconds, Emulation time: 0.0900063 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 15.001670 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 26 seconds, Emulation time: 0.112961 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 24.002198 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 35 seconds, Emulation time: 0.155026 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

---

col 32 num data 40

- **row 32: 0.090 ms**
- **row 64: 0.111 ms**
- **row 128: 0.105 ms**

```
[JHLEE] FPGA kernel exec time is 11.001679 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 22 seconds, Emulation time: 0.0900063 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 15.001504 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 26 seconds, Emulation time: 0.111174 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 13.001595 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 24 seconds, Emulation time: 0.104796 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

---

col 32 row 32

- **num 40: 0.090 ms**
- **num80: 0.118 ms**
- **num160: 0.153 ms**
- **num320: 0.230ms**
- **num640: 0.244 ms**

```
[JHLEE] FPGA kernel exec time is 11.001679 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 22 seconds, Emulation time: 0.0900063 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 15.001600 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 26 seconds, Emulation time: 0.117735 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 25.002464 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 36 seconds, Emulation time: 0.152505 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 40.003962 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 52 seconds, Emulation time: 0.229934 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

```
[JHLEE] FPGA kernel exec time is 43.004209 s
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 54 seconds, Emulation time: 0.243754 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.129 KB              WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.156 KB              WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB              WR = 2.812 KB
```

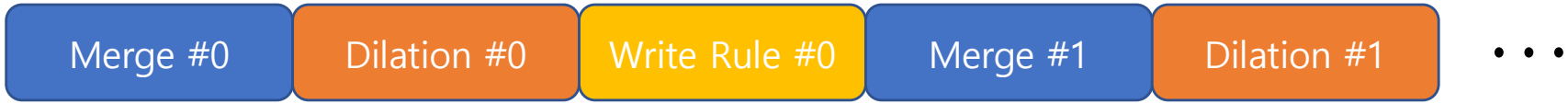
Column size 와 # of data에 dependent함

↓

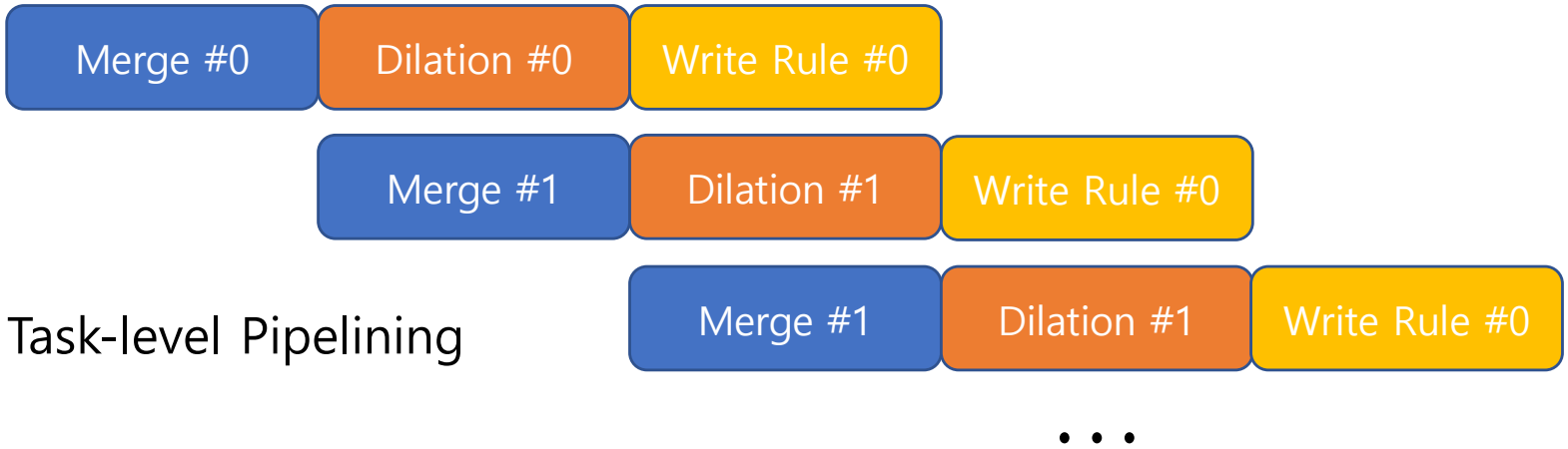
Dataflow optimization필요

Dataflow 개선 방안

- **Dataflow 적용 안함**

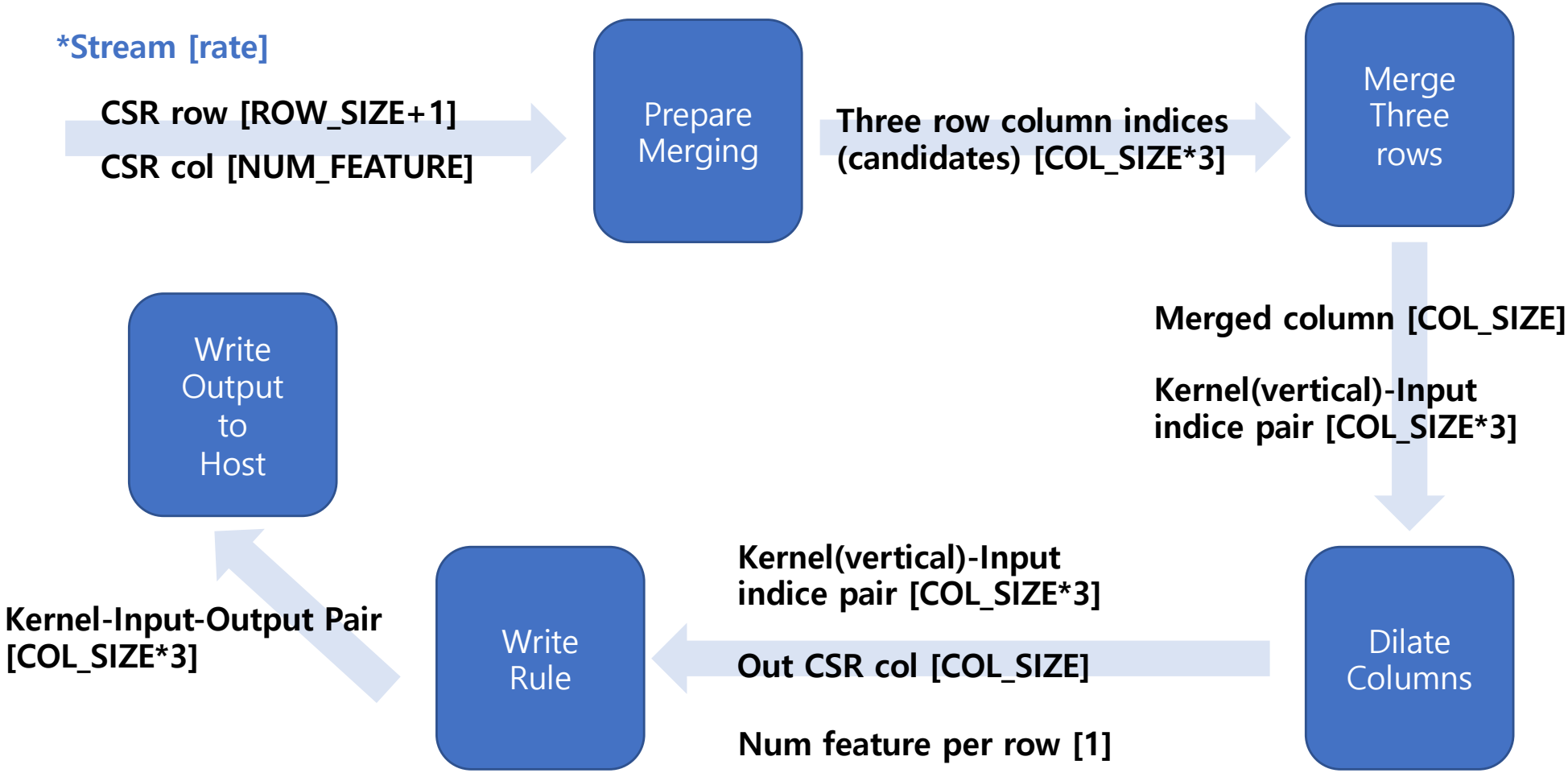


- **Dataflow 적용**



HLS code overview

Task level pipelining을 위한 dataflow optimization 적용



HLS code overview

Task lev

*Stream

CSR r

CSR c

Kernel-Inpu
[COL_SIZE*3

```
extern "C" {
void vadd(hls::vector<DTYPE,(ROW_SIZE+1)> *csr_row,
          hls::vector<DTYPE,(NUM_FEATURE)> *csr_col,
          hls::vector<DTYPE,3> *out_Rule){
#pragma HLS INTERFACE mode=m_axi bundle=gmem0 port=csr_row depth=600
#pragma HLS INTERFACE mode=m_axi bundle=gmem1 port=csr_col depth=600
#pragma HLS INTERFACE mode=m_axi bundle=gmem2 port=out_Rule depth=600

#pragma HLS DATAFLOW
    hls::stream<hls::vector<DTYPE,(ROW_SIZE+1)>> CsrRow("CsrRowStream");
    hls::stream<hls::vector<DTYPE,(ROW_SIZE+1)>> CsrRow_("CsrRow2Stream");
    hls::stream<hls::vector<DTYPE,NUM_FEATURE>> CsrCol("CsrColStream");
    hls::stream<hls::vector<DTYPE,COL_SIZE*3>> Candidates("CandidatesStream");
    hls::stream<hls::vector<DTYPE,COL_SIZE>> MergedCol("MergedColStream");
    hls::stream<hls::vector<DTYPE,COL_SIZE>> MergedCol_("MergedCol2Stream");
    hls::stream<hls::vector<DTYPE,COL_SIZE*3>> KIPairs("KIPiarsStream");
    hls::stream<hls::vector<DTYPE,COL_SIZE*3>> KIPairs_("KIPiars2Stream");
    hls::stream<hls::vector<DTYPE,3*KERNEL_SIZE*KERNEL_SIZE>> RuleStream("RuleStream"); // kernel, input, output
    hls::stream<DTYPE> RuleLength("RuleLength");
    hls::stream<DTYPE> OutNumFeaturePerRow("OutNumFeaturePerRowStream");
    hls::stream<hls::vector<DTYPE,COL_SIZE>> OutCsrCol("OutCsrColStream");
    readCsrData(CsrRow, CsrCol, csr_row, csr_col);

    int rule_size;

    /* Step 0. Prepare merger
    Input: csr_row, csr_col
    Output: Three candidatess
    */
    prepare_merge(Candidates, CsrRow_, CsrRow, CsrCol);

    /* Step 1. Three column merger
    Input: csr_row, csr_col
    Output: Merged cols, Input-Kernel(vertical) pairs
    */
    merge(MergedCol, Candidates, KIPairs, CsrRow_);

    /* Step 2. Row dilator
    Input: Merged cols
    Output: out_csr_row, out_csr_col
    */
    dilate(MergedCol_, KIPairs_, MergedCol, KIPairs, OutNumFeaturePerRow, OutCsrCol);

    /* Step 3. Rule generator
    Input: Input-Kernel(vertical) pairs, out_csr_row, out_csr_col
    Output: Rule
    */
    rule_size = write_rule(RuleStream, RuleLength, MergedCol_, KIPairs_, OutNumFeaturePerRow, OutCsrCol);

    gen_rule(RuleStream, RuleLength, out_Rule);

}

}
```

ge
ee
/s

COL_SIZE]

put
SIZE*3]

te
mns

Dataflow Optimization 실험 결과

- **Optimization**
  - **Dataflow 적용 안함**
    - **하나의 Output row에 대해서 Merge Rows → Dilate Columns → Write Rule 순차적으로 진행**
  - **Dataflow (R):**
    - **Dataflow를 Row loop 단위로 적용**

    → Column에 대해서는 요소를 나누지 못해서 scalable 하지 않음
    (Data set의 크기가 커지면 Stream 사이즈가 너무 커져서 사용할 수 없음)
  - **Dataflow (RC):**
    - **Dataflow Row loop 보다 하위의 Column loop 단위로 적용**
- **Dataset**
  - **Simple example: 32x32 의 2D space에 random generation 한 40개의 데이터 사용**

Optimization	Dataflow X	Dataflow (R)	Dataflow (RC)
SW emulation	7.225 ms	11.364 ms	2.202 ms
HW emulation	0.096 ms	0.078 ms	0.174 ms
Run on F1	0.207 ms	-	-

Dataflow Optimization 실험 결과

- **HW Emulation**

```
[Result] Rule generation done
   Kernel  0      Kernel  1      Kernel  2      Kernel  3      Kernel  4      Kernel  5      Kernel  6      Kernel  7      Kernel  8
 Input Output   Input Output   Input Output   Input Output   Input Output   Input Output   Input Output   Input Output   Input Output
|      0    0 ||      0    1 ||      0    2 ||      0    6 ||      0    7 ||      0    8 ||      0   12 ||      0   13 ||      0   14 |
|      1    3 ||      1    4 ||      1    5 ||      1    9 ||      1   10 ||      1   11 ||      1   15 ||      1   16 ||      1   17 |
|      2   18 ||      2   19 ||      2   20 ||      2   29 ||      2   30 ||      2   31 ||      2   50 ||      2   51 ||      2   52 |
|      3   21 ||      3   22 ||      3   23 ||      3   33 ||      3   34 ||      3   35 ||      3   55 ||      3   56 ||      3   57 |
|      4   24 ||      4   25 ||      4   26 ||      4   42 ||      4   43 ||      4   44 ||      4   64 ||      4   65 ||      4   66 |
|      5   26 ||      5   27 ||      5   28 ||      5   44 ||      5   45 ||      5   46 ||      5   66 ||      5   67 ||      5   68 |
|      6   30 ||      6   31 ||      6   32 ||      6   51 ||      6   52 ||      6   53 ||      6   73 ||      6   74 ||      6   75 |
|      7   36 ||      7   37 ||      7   38 ||      7   58 ||      7   59 ||      7   60 ||      7   80 ||      7   81 ||      7   82 |
|      8   39 ||      8   40 ||      8   41 ||      8   61 ||      8   62 ||      8   63 ||      8   83 ||      8   84 ||      8   85 |
|      9   47 ||      9   48 ||     10   51 ||      9   69 ||      9   70 ||     10   73 ||      9   86 ||      9   87 ||     10   90 |
|     10   49 ||     10   50 ||     11   56 ||     10   71 ||     10   72 ||     11   78 ||     10   88 ||     10   89 ||     11   95 |
|     11   54 ||     11   55 ||     13   81 ||     11   76 ||     11   77 ||     13   98 ||     11   93 ||     11   94 ||     13  109 |
|     12   69 ||     12   70 ||     14   92 ||     12   86 ||     12   87 ||     14  106 ||     12  102 ||     12  103 ||     14  115 |
|     13   79 ||     13   80 ||     15  101 ||     13   96 ||     13   97 ||     15  112 ||     13  107 ||     13  108 ||     15  118 |
|     14   90 ||     14   91 ||     16  121 ||     14  104 ||     14  105 ||     16  127 ||     14  113 ||     14  114 ||     16  136 |
|     15   99 ||     15  100 ||     17  124 ||     15  110 ||     15  111 ||     17  130 ||     15  116 ||     15  117 ||     17  140 |
|     16  119 ||     16  120 ||     18  133 ||     16  125 ||     16  126 ||     18  143 ||     16  134 ||     16  135 ||     18  155 |
|     17  122 ||     17  123 ||     19  139 ||     17  128 ||     17  129 ||     19  149 ||     17  138 ||     17  139 ||     19  162 |
|     18  131 ||     18  132 ||     20  146 ||     18  141 ||     18  142 ||     20  158 ||     18  153 ||     18  154 ||     20  169 |
|     19  137 ||     19  138 ||     21  152 ||     19  147 ||     19  148 ||     21  166 ||     19  160 ||     19  161 ||     21  178 |
|     20  144 ||     20  145 ||     22  161 ||     20  156 ||     20  157 ||     22  173 ||     20  167 ||     20  168 ||     22  185 |
|     21  150 ||     21  151 ||     23  164 ||     21  164 ||     21  165 ||     23  176 ||     21  176 ||     21  177 ||     23  188 |
|     22  159 ||     22  160 ||     24  172 ||     22  171 ||     22  172 ||     24  184 ||     22  183 ||     22  184 ||     24  197 |
|     23  162 ||     23  163 ||     25  181 ||     23  174 ||     23  175 ||     25  191 ||     23  186 ||     23  187 ||     25  200 |
|     24  170 ||     24  171 ||     26  194 ||     24  182 ||     24  183 ||     26  203 ||     24  195 ||     24  196 ||     26  209 |
|     25  179 ||     25  180 ||     27  206 ||     25  189 ||     25  190 ||     27  212 ||     25  198 ||     25  199 ||     27  218 |
|     26  192 ||     26  193 ||     28  215 ||     26  201 ||     26  202 ||     28  221 ||     26  207 ||     26  208 ||     28  229 |
|     27  204 ||     27  205 ||     29  224 ||     27  210 ||     27  211 ||     29  232 ||     27  216 ||     27  217 ||     29  239 |
|     28  213 ||     28  214 ||     30  225 ||     28  219 ||     28  220 ||     30  233 ||     28  227 ||     28  228 ||     30  240 |
|     29  222 ||     29  223 ||     31  228 ||     29  230 ||     29  231 ||     31  236 ||     29  237 ||     29  238 ||     31  243 |
|     30  223 ||     30  224 ||     32  246 ||     30  231 ||     30  232 ||     32  253 ||     30  238 ||     30  239 ||     32  260 |
|     31  226 ||     31  227 ||     33  247 ||     31  234 ||     31  235 ||     33  254 ||     31  241 ||     31  242 ||     33  261 |
|     32  244 ||     32  245 ||     34  250 ||     32  251 ||     32  252 ||     34  257 ||     32  258 ||     32  259 ||     34  267 |
|     33  245 ||     33  246 ||     35  260 ||     33  252 ||     33  253 ||     35  270 ||     33  259 ||     33  260 ||     35  279 |
|     34  248 ||     34  249 ||     36  264 ||     34  255 ||     34  256 ||     36  273 ||     34  265 ||     34  266 ||     36  285 |
|     35  258 ||     35  259 ||     37  276 ||     35  268 ||     35  269 ||     37  288 ||     35  277 ||     35  278 ||     -1   -1 |
|     36  262 ||     36  263 ||     38  280 ||     36  271 ||     36  272 ||     38  291 ||     36  283 ||     36  284 ||     -1   -1 |
|     37  274 ||     37  275 ||     39  282 ||     37  286 ||     37  287 ||     39  293 ||     -1   -1 ||     -1   -1 ||     -1   -1 |
|     38  278 ||     38  279 ||     -1   -1 ||     38  289 ||     38  290 ||     -1   -1 ||     -1   -1 ||     -1   -1 ||     -1   -1 |
|     39  280 ||     39  281 ||     -1   -1 ||     39  291 ||     39  292 ||     -1   -1 ||     -1   -1 ||     -1   -1 ||     -1   -1 |
INFO::[ Vitis-EM 22 ] [Time elapsed: 0 minute(s) 56 seconds, Emulation time: 0.17417 ms]
Data transfer between kernel(s) and global memory(s)
vadd_1:m_axi_gmem0-DDR[1]          RD = 0.250 KB               WR = 0.000 KB
vadd_1:m_axi_gmem1-DDR[1]          RD = 0.250 KB               WR = 0.000 KB
vadd_1:m_axi_gmem2-DDR[1]          RD = 0.000 KB               WR = 25.312 KB

INFO: [HW-EMU 06-0] Waiting for the simulator process to exit
INFO: [HW-EMU 06-1] All the simulator processes exited successfully
[centos@ip-172-31-21-26 csr_v2_mj]$ make run TARGET=hw DEVICE=$AWS_PLATFORM
```

Dataflow Optimization 실험 결과

- **F1 instance - HW**

```
[16:34:36] Phase 3.3.1 Small Shape Clustering
[16:36:08] Phase 3.3.2 Flow Legalize Slice Clusters
[16:36:08] Phase 3.3.3 Slice Area Swap
[16:39:11] Phase 3.4 Place Remaining
[16:39:41] Phase 3.5 Re-assign LUT pins
[16:41:14] Phase 3.6 Pipeline Register Optimization
[16:41:14] Phase 3.7 Fast Optimization
[16:44:49] Phase 4 Post Placement Optimization and Clean-Up
[16:44:49] Phase 4.1 Post Commit Optimization
[16:49:56] Phase 4.1.1 Post Placement Optimization
[16:50:27] Phase 4.1.1.1 BUFG Insertion
[16:50:27] Phase 1 Physical Synthesis Initialization
[16:53:30] Phase 4.1.1.2 BUFG Replication
[16:53:30] Phase 4.1.1.3 Post Placement Timing Optimization
[16:58:34] Phase 4.1.1.4 Replication
[17:03:40] Phase 4.2 Post Placement Cleanup
[17:03:40] Phase 4.3 Placer Reporting
[17:03:40] Phase 4.3.1 Print Estimated Congestion
[17:04:10] Phase 4.4 Final Placement Cleanup
[17:26:35] Finished 4th of 6 tasks (FPGA logic placement). Ela

[17:26:35] Starting logic routing..
[17:29:08] Phase 1 Build RT Design
[17:35:14] Phase 2 Router Initialization
[17:35:14] Phase 2.1 Fix Topology Constraints
[17:35:45] Phase 2.2 Pre Route Cleanup
[17:36:15] Phase 2.3 Global Clock Net Routing
[17:37:47] Phase 2.4 Update Timing
[17:43:54] Phase 2.5 Update Timing for Bus Skew
[17:43:54] Phase 2.5.1 Update Timing
[17:47:28] Phase 3 Initial Routing
[17:47:28] Phase 3.1 Global Routing
[17:56:08] Phase 4 Rip-up And Reroute
[17:56:08] Phase 4.1 Global Iteration 0
[18:19:03] Phase 4.2 Global Iteration 1
[18:25:09] Phase 4.3 Global Iteration 2
[18:30:13] Phase 4.4 Global Iteration 3
[18:35:48] Phase 4.5 Global Iteration 4
[18:40:22] Phase 4.6 Global Iteration 5
[18:43:54] Phase 4.7 Global Iteration 6
[18:46:26] Phase 5 Delay and Skew Optimization
[18:46:26] Phase 5.1 Delay CleanUp
[18:46:57] Phase 5.1.1 Update Timing
[18:50:00] Phase 5.2 Clock Skew Optimization
[18:51:01] Phase 6 Post Hold Fix
[18:51:01] Phase 6.1 Hold Fix Iter
[18:51:01] Phase 6.1.1 Update Timing
[18:54:35] Phase 7 Leaf Clock Prog Delay Opt
[18:57:09] Phase 8 Route finalize
[18:57:40] Phase 9 Verifying routed nets
[18:58:10] Phase 10 Depositing Routes
[18:59:41] Phase 11 Post Router Timing
[18:59:41] Phase 11.1 Update Timing
[19:06:51] Phase 12 Physical Synthesis in Router
[19:06:51] Phase 12.1 Physical Synthesis Initialization
[19:11:27] Phase 12.2 Critical Path Optimization
[19:13:29] Finished 5th of 6 tasks (FPGA routing). Elapsed tim
```

```
****** v++ v2021.1 (64-bit)
  **** SW Build 3246112 on 2021-06-09-14:19:56
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

INFO: [v++ 60-1306] Additional information associated with this v++ package can be found at:
        Reports: /home/centos/src/project_data/aws-fpga/Vitis/examples/xilinx_2021.1/csr_v2_mj/_x/reports/package
        Log files: /home/centos/src/project_data/aws-fpga/Vitis/examples/xilinx_2021.1/csr_v2_mj/_x/logs/package
Running Dispatch Server on port: 40883
INFO: [v++ 60-1548] Creating build summary session with primary output /home/centos/src/project_data/aws-fpga/Vitis/example
01920_2/vadd.xclbin.package_summary, at Tue Dec 21 19:35:04 2021
INFO: [v++ 60-1316] Initiating connection to rulecheck server, at Tue Dec 21 19:35:04 2021
Running Rule Check Server on port:40675
INFO: [v++ 60-1315] Creating rulecheck session with output '/home/centos/src/project_data/aws-fpga/Vitis/examples/xilinx_20
ec 21 19:35:06 2021
INFO: [v++ 60-895]   Target platform: /home/centos/src/project_data/aws-fpga/Vitis/aws_platform/xilinx_aws-vu9p-f1_shell-v0
INFO: [v++ 60-1578]   This platform contains Xilinx Shell Archive '/home/centos/src/project_data/aws-fpga/Vitis/aws_platfor
04261818_201920_2.xsa'
INFO: [v++ 74-78] Compiler Version string: 2021.1
INFO: [v++ 60-1302] Platform 'xilinx_aws-vu9p-f1_shell-v04261818_201920_2.xpfm' has been explicitly enabled for this releas
INFO: [v++ 60-2256] Packaging for hardware
INFO: [v++ 60-2460] Successfully copied a temporary xclbin to the output xclbin: /home/centos/src/project_data/aws-fpga/Vit
-v04261818_201920_2/vadd.xclbin
INFO: [v++ 60-2343] Use the vitis_analyzer tool to visualize and navigate the relevant reports. Run the following command.
    vitis_analyzer /home/centos/src/project_data/aws-fpga/Vitis/examples/xilinx_2021.1/csr_v2_mj/build_dir.hw.xilinx_aws-vu
INFO: [v++ 60-791] Total elapsed time: 0h 0m 20s
INFO: [v++ 60-1653] Closing dispatch client.
emconfigutil --platform /home/centos/src/project_data/aws-fpga/Vitis/aws_platform/xilinx_aws-vu9p-f1_shell-v04261818_201920
-vu9p-f1_shell-v04261818_201920_2

****** configutil v2021.1 (64-bit)
  **** SW Build 3246112 on 2021-06-09-14:19:56
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

INFO: [ConfigUtil 60-895]   Target platform: /home/centos/src/project_data/aws-fpga/Vitis/aws_platform/xilinx_aws-vu9p-f1_s
INFO: [ConfigUtil 60-1578]   This platform contains Xilinx Shell Archive '/home/centos/src/project_data/aws-fpga/Vitis/aws_
INFO: [ConfigUtil 60-1032]
emulation configuration file `emconfig.json` is created in ./_x.hw.xilinx_aws-vu9p-f1_shell-v04261818_201920_2 directory
./hello_world ./build_dir.hw.xilinx_aws-vu9p-f1_shell-v04261818_201920_2/vadd.xclbin
terminate called after throwing an instance of 'xrt_xocl::error'
  what():  No devices found
make: *** [run] Aborted (core dumped)
[centos@ip-172-31-21-26 csr_v2_mj]$ df -h
```

결론 및 개선점

- Rule Generator의 경우 각 Output row별로 Merge & Deliation하는데 걸리는 Clock Cycle이 다르지만, Dataflow Optimization도 잘 적용됨
  (예를 들어,
  ```
  for ( ){
      if ( ){
          break;
  } }
  ```
  → Dataflow 적용 가능
- Dataflow Optimization 할 때, 적용되는 Task 간의 주고 받는 Data-stream의 크기가 2048-bit를 넘게 되면 합성되지 않는 문제 존재

- 개선여지
  - Dataflow Optimization에서 Data-stream을 저 작게 쪼개면 Scalable 하게 적용가능

Reference

- **PointPillars: Fast Encoders for Object Detection from Point Clouds, Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, Oscar Beijbom, CVPR2019 (arXiv:1812.05784)**
- **Yan Y, Mao Y, Li B. SECOND: Sparsely Embedded Convolutional Detection. Sensors. 2018; 18(10):3337. https://doi.org/10.3390/s18103337**