pp4fpga Sparse Matrix Vector Multiplication report

電子所碩一 R09943019 李唐

➢ System introduction

Sparse matrix vector (SpMV) multiplication performs the operation

$$\mathrm{y} = \mathrm{M} * \mathrm{x}$$

where x and y are vectors, and M is a sparse matrix stored in the compressed row storage (CRS) form.

CRS form is commonly used when the matrix contains only a small number of nonzero elements (typically 10 percent or less), enabling a matrix to be stored +++with less memory and manipulated with fewer operations.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & 0 & 0 \\ 0 & m_{22} & m_{23} & 0 \\ m_{31} & 0 & m_{33} & m_{34} \\ 0 & m_{42} & 0 & m_{44} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Given a sparse matrix M in 2-dim representation as shown above, the corresponding CRS representation consists of three arrays:

✧ **values**

The values array holds the value of each nonzero element in the matrix.

$$[m_{11} \quad m_{12} \quad m_{22} \quad m_{23} \quad m_{31} \quad m_{33} \quad m_{34} \quad m_{42} \quad m_{44}]$$

✧ **columnIndex**

The columnIndex array stores the column index of each nonzero element.

$$[0 \quad 1 \quad 1 \quad 2 \quad 0 \quad 2 \quad 3 \quad 1 \quad 3]$$

✧ **rowPtr**

The rowPtr array contains the index in values array of the first element in each row. In this example, the index of m_{11}, m_{22}, m_{31}, m_{42}, and the total number of nonzero elements in values array are stored in rowPtr.

$$[0 \quad 2 \quad 4 \quad 7 \quad 9]$$

※ If there are rows in the matrix without a nonzero element, then values in the rowPtr array will repeat.

Given the CRS form representation, we can conclude that

- if $\mathrm{values}[\mathrm{k}] = m_{ij}$, then $\mathrm{colIndex}[\mathrm{k}] = \mathrm{j}$
- if $\mathrm{values}[\mathrm{k}] = m_{ij}$, then $\mathrm{rowPtr}[\mathrm{i}] \leq \mathrm{k} < \mathrm{rowPtr}[\mathrm{i}+1]$
- the number of nonzero elements in row i is $\mathrm{rowPtr}[\mathrm{i}+1] - \mathrm{rowPtr}[\mathrm{i}]$

Sparse matrix vector multiplication can be more efficiently computed by representing the matrix in CRS form, which can significantly reduce the operations required especially when the matrix is highly sparse.

- Screen dump and Observations
  - **Solution 1: Baseline implementation**

```
L1: for (int i = 0; i < NUM_ROWS; i++) {
        y0 = 0;
        L2: for (int k = rowPtr[i]; k < rowPtr[i+1]; k++) {
                y0 += values[k] * x[columnIndex[k]];
        }
        y[i] = y0;
}
```

L1 iterates through each row of the matrix, and L2 iterates across the column of the matrix to accumulate the partial sum.

  - Synthesis report/RTL Co-simulation

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.024 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
13	189	0.130 us	1.890 us	13	189	none

Detail

Instance

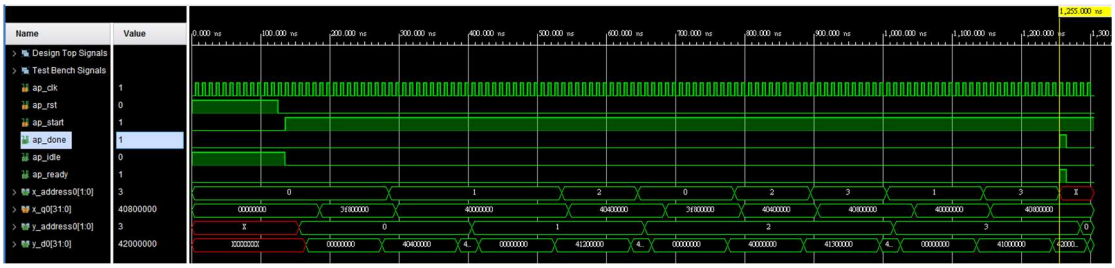
Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- L1	12	188	3 ~ 47	-	-	4	no
+ L2	0	44	11	-	-	0 ~ 4	no

Utilization Estimates

Summary

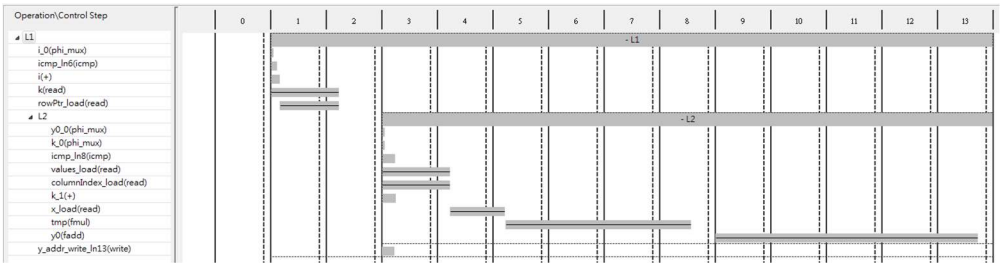
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	78	-
FIFO	-	-	-	-	-
Instance	-	5	348	711	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	89	-
Register	-	-	215	-	-
Total	0	5	563	878	0
Available	280	220	106400	53200	0
Utilization (%)	0	2	~0	1	0



Total latency (cycles)	L2 max iteration latency (cycles)
112	36

The sparse matrix-vector multiplication requires various number of iterations through the partial sum accumulation. Thus, the latency estimated in synthesis report is not accurate. Instead, we measure the latency from the cosim waveform (L2 max iteration latency means the maximum iteration latency appeared in L2).

✧ Analysis perspective



	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
spmv	-	13~189	-	14 ~ 190	-
L1	no	12 ~ 188	3 ~ 47	-	4
	no	0 ~ 44	11	-	0~4

✧ **Solution 2: Pipeline L1**

```
L1: for (int i = 0; i < NUM_ROWS; i++) {
# pragma HLS PIPELINE
      y0 = 0;
      L2: for (int k = rowPtr[i]; k < rowPtr[i+1]; k++) {
            y0 += values[k] * x[columnIndex[k]];
      }
      y[i] = y0;
}
```

✧ Synthesis report/RTL Co-simulation

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.024 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
13	189	0.130 us	1.890 us	13	189	none

Detail

Instance

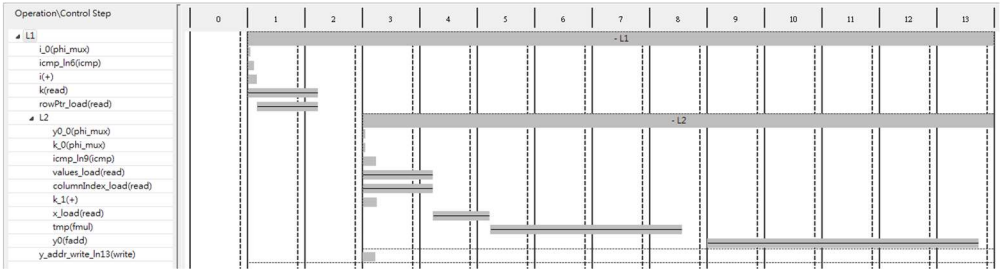
Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- L1	12	188	3 ~ 47	-	-	4	no
+ L2	0	44	11	-	-	0 ~ 4	no

Total latency (cycles)	L2 max iteration latency (cycles)
112	36

There is no change to the overall performance since the L2 loop iteration is variable bounded. Thus, the L2 loop cannot be flatten.

✧ Analysis perspective



	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
spmv	-	13~189	-	14 ~ 190	-
L1	no	12 ~ 188	3 ~ 47	-	4
	no	0 ~ 44	11	-	0~4

✧ **Solution 3: Pipeline L2**

```
L1: for (int i = 0; i < NUM_ROWS; i++) {
        y0 = 0;
        L2: for (int k = rowPtr[i]; k < rowPtr[i+1]; k++) {
# pragma HLS PIPELINE
                y0 += values[k] * x[columnIndex[k]];
        }
        y[i] = y0;
}
```

✧ Synthesis report/RTL Co-simulation

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.024 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
17	117	0.170 us	1.170 us	17	117	none

Detail

Instance

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- L1	16	116	4 ~ 29	-	-	4	no
+ L2	0	25	11	5	1	0 ~ 4	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	80	-
FIFO	-	-	-	-	-
Instance	-	5	348	711	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	92	-
Register	-	-	215	-	-
Total	0	5	563	883	0
Available	280	220	106400	53200	0
Utilization (%)	0	2	~0	1	0

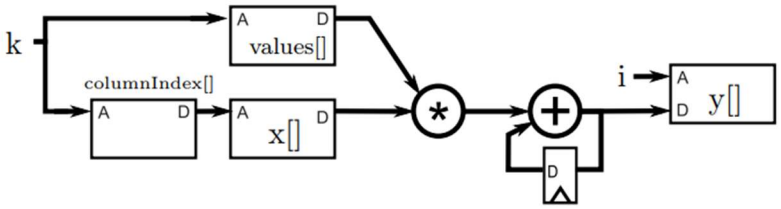
Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
spmv_fadd_32ns_32bkb_U1	spmv_fadd_32ns_32bkb	0	2	205	390	0
spmv_fmul_32ns_32cud_U2	spmv_fmul_32ns_32cud	0	3	143	321	0
Total		2	0	5	348	711	0

Interface

Summary

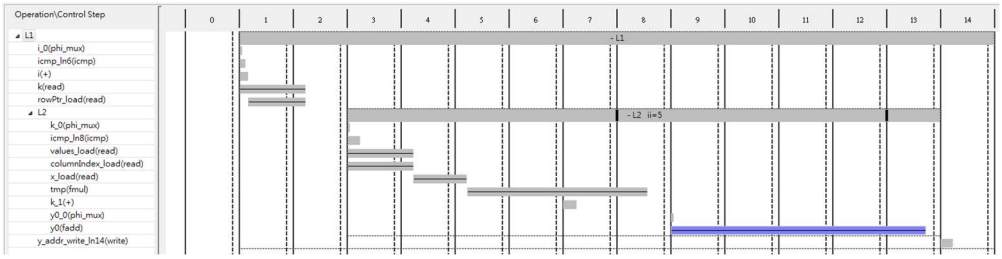
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	spmv	return value
ap_rst	in	1	ap_ctrl_hs	spmv	return value
ap_start	in	1	ap_ctrl_hs	spmv	return value
ap_done	out	1	ap_ctrl_hs	spmv	return value
ap_idle	out	1	ap_ctrl_hs	spmv	return value
ap_ready	out	1	ap_ctrl_hs	spmv	return value
rowPtr_address0	out	3	ap_memory	rowPtr	array
rowPtr_ce0	out	1	ap_memory	rowPtr	array
rowPtr_q0	in	32	ap_memory	rowPtr	array
rowPtr_address1	out	3	ap_memory	rowPtr	array
rowPtr_ce1	out	1	ap_memory	rowPtr	array
rowPtr_q1	in	32	ap_memory	rowPtr	array
columnIndex_address0	out	4	ap_memory	columnIndex	array
columnIndex_ce0	out	1	ap_memory	columnIndex	array
columnIndex_q0	in	32	ap_memory	columnIndex	array
values_address0	out	4	ap_memory	values	array
values_ce0	out	1	ap_memory	values	array
values_q0	in	32	ap_memory	values	array
y_address0	out	2	ap_memory	y	array
y_ce0	out	1	ap_memory	y	array
y_we0	out	1	ap_memory	y	array
y_d0	out	32	ap_memory	y	array
x_address0	out	2	ap_memory	x	array
x_ce0	out	1	ap_memory	x	array
x_q0	in	32	ap_memory	x	array



Total latency (cycles)	L2 max iteration latency (cycles)
82	24

Pipelining to the inner L2 loop can reduce the latency required. However, the II is limited by the floating-point addition in each iteration.

✧ Analysis perspective



	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
spmv	-	17~117	-	18 ~ 118	-
L1	no	16 ~ 116	4 ~ 29	-	4
	yes	0 ~ 25	11	5	0~4

The blue mark shows the limiting operation.

✧ **Solution 4: fixed-point data type instead of floating-point data type**

Use arbitrary precision data type ap_fixed with 16 bits in representation.

✧ Synthesis report/RTL Co-simulation

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.702 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
17	37	0.170 us	0.370 us	17	37	none

Detail

Instance

N/A

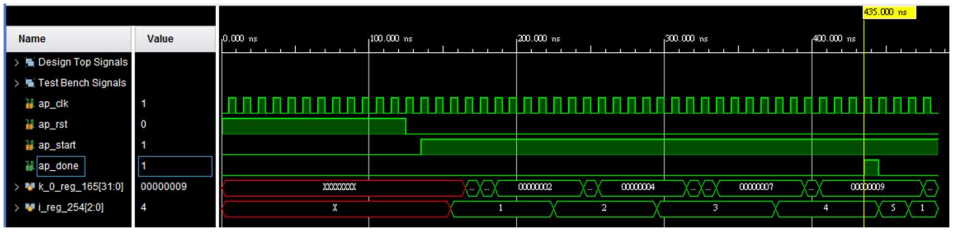
Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- L1	16	36	4 ~ 9	-	-	4	no
+ L2	0	5	3	1	1	0 ~ 4	yes

Utilization Estimates

Summary

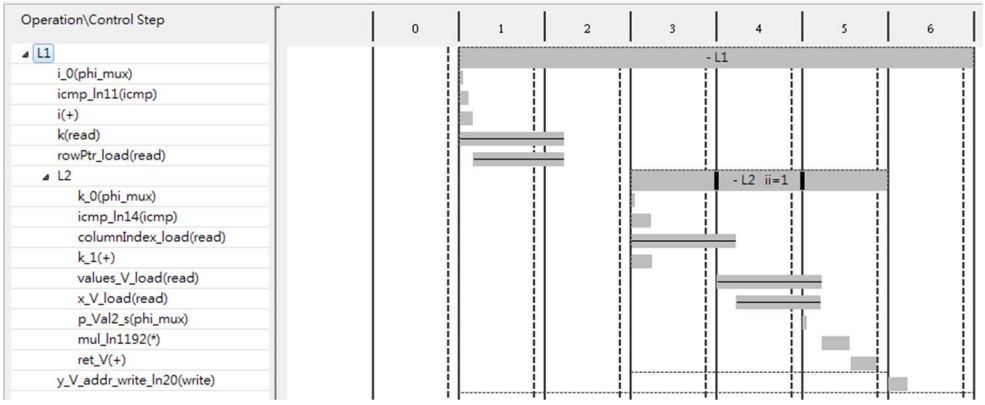
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	-	0	82	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	78	-
Register	-	-	163	-	-
Total	0	1	163	160	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0



Total latency (cycles)	L2 max iteration latency (cycles)
30	8

With 16-bit fixed-point representation, we can realize II=1 without floating-point addition bottleneck. Thus, the cosim latency is significantly reduced to only 30 cycles in total. Moreover, the hardware resource is also greatly reduced (no floating-point instance).

✧ Analysis perspective



	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
spmv	-	17~37	-	18 ~ 38	-
L1	no	16 ~ 36	4 ~ 9	-	4

➢ GitHub submission

https://github.com/nodetibylno/MSoC_self_paced_learning

➢ Reference

1. R. Kastner, J. Matai, and S. Neuendorffer , Parallel Programming for FPGAs, arXiv , 2018.