

# Умножение матриц

---

Соболь Валентин  
3540901/81502

26 декабря 2019 г.

Санкт-Петербургский политехнический университет Петра Великого

# Произведение матриц

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{bmatrix}$$

$$\mathbf{AB} = \begin{bmatrix} (\mathbf{AB})_{11} & (\mathbf{AB})_{12} & \cdots & (\mathbf{AB})_{1p} \\ (\mathbf{AB})_{21} & (\mathbf{AB})_{22} & \cdots & (\mathbf{AB})_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{AB})_{n1} & (\mathbf{AB})_{n2} & \cdots & (\mathbf{AB})_{np} \end{bmatrix}$$

$$(AB)_{ij} = \sum_{k=1}^m A_{ik} B_{kj}$$

## «Простой» алгоритм

```
void matmul(int A[N][M], int B[M][P], int AB[N][P]) {  
    /* for each row and column of AB */  
    row: for(int i = 0; i < N; ++i) {  
        col: for(int j = 0; j < P; ++j) {  
            /* compute (AB)i,j */  
            int ABij = 0;  
            product: for(int k = 0; k < M; ++k) {  
                ABij += A[i][k] * B[k][j];  
            }  
            AB[i][j] = ABij;  
        }  
    }  
}
```

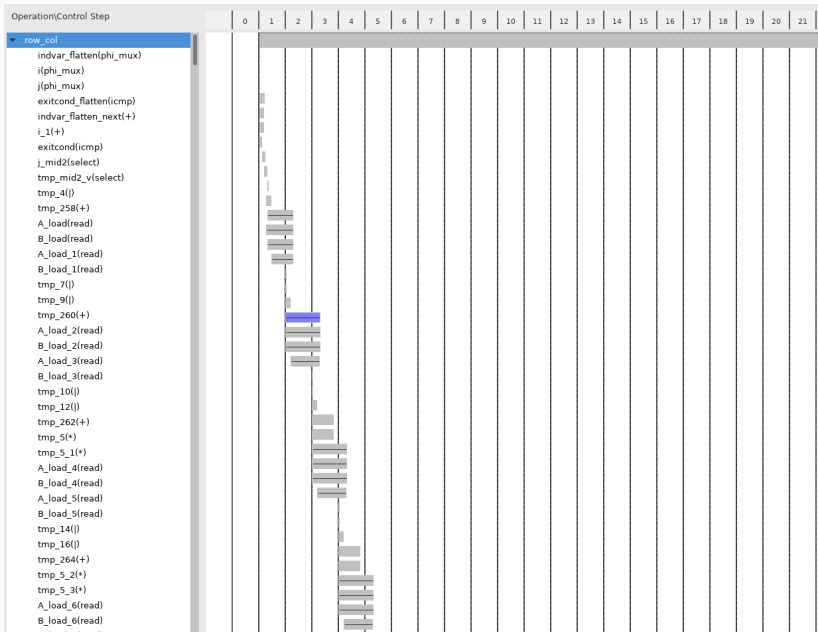
# «Простой» алгоритм

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
▼ ● matmul	-	8421633	-	8421634	-
▼ ● row	no	8421632	65794	-	128
▼ ● col	no	65792	514	-	128
● product	no	512	4	-	128

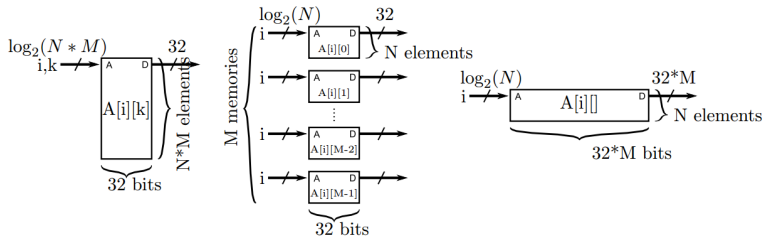
# «Простой» алгоритм: Конвейеризация

Performance Profile ⓘ		Resource Profile					
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count		
▼ ● matmul	-	1048582	-	1048583	-		
● row_col	yes	1048580	69	64	16384		

## «Простой» алгоритм: Конвейеризация



# Разбиение данных



Исходный массив | array\_partition | array\_reshape

# «Простой» алгоритм: Конвейеризация + разбиение

Performance Profile		Resource Profile			
	Pipelined	Latency	Iteration Latency	Initiation Interval	Trip count
▼ ● matmul	-	16391	-	16392	-
● row_col	yes	16389	7	1	16384



# «Простой» алгоритм: Конвейеризация всего устройства

```
WARNING: [ANALYSIS 214-1] Tool encounters 16384 load/store instructions to analyze which may result in long runtime.
ERROR: [XFORM 203-1403] Unsupported enormous number of load/store instructions: 'matmul' .
ERROR: [HLS 200-70] Failed building synthesis data model.
command 'ap_source' returned error code
    while executing
    "source [lindex $::argv 1] "
      ("uplevel" body line 1)
        invoked from within
    "uplevel \#0 { source [lindex $::argv 1] } "
INFO: [HLS 200-112] Total elapsed time: 900.983 seconds; peak allocated memory: 75.784 MB.
INFO: [Common 17-206] Exiting vivado_hls at Mon Dec 23 16:27:01 2019...
```

## «Простой» алгоритм: Сравнение

### Performance Estimates

#### ▣ Timing (ns)

Clock		no_flags	pipeline_inner	reshape_part_inner
ap_clk	Target	10.00	10.00	10.00
	Estimated	8.470	9.010	9.010

#### ▣ Latency (clock cycles)

		no_flags	pipeline_inner	reshape_part_inner
Latency	min	8421633	1048582	16391
	max	8421633	1048582	16391
Interval	min	8421633	1048582	16391
	max	8421633	1048582	16391

### Utilization Estimates

	no_flags	pipeline_inner	reshape_part_inner
BRAM_18K	0	0	0
DSP48E	3	6	384
FF	213	1604	14362
LUT	290	6364	7357

## Блочный алгоритм

a)

$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,4}$
$A_{4,1}$	$A_{4,2}$	$A_{4,3}$	$A_{4,4}$

$\times$

$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$
$B_{2,1}$	$B_{2,2}$	$B_{2,3}$	$B_{2,4}$
$B_{3,1}$	$B_{3,2}$	$B_{3,3}$	$B_{3,4}$
$B_{4,1}$	$B_{4,2}$	$B_{4,3}$	$B_{4,4}$

$=$

$AB_{1,1}$	$AB_{1,2}$	$AB_{1,3}$	$AB_{1,4}$
$AB_{2,1}$	$AB_{2,2}$	$AB_{2,3}$	$AB_{2,4}$
$AB_{3,1}$	$AB_{3,2}$	$AB_{3,3}$	$AB_{3,4}$
$AB_{4,1}$	$AB_{4,2}$	$AB_{4,3}$	$AB_{4,4}$

b)

$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,4}$
$A_{4,1}$	$A_{4,2}$	$A_{4,3}$	$A_{4,4}$

 $\times$ 

$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$
$B_{2,1}$	$B_{2,2}$	$B_{2,3}$	$B_{2,4}$
$B_{3,1}$	$B_{3,2}$	$B_{3,3}$	$B_{3,4}$
$B_{4,1}$	$B_{4,2}$	$B_{4,3}$	$B_{4,4}$

 $=$ 

$AB_{1,1}$	$AB_{1,2}$	$AB_{1,3}$	$AB_{1,4}$
$AB_{2,1}$	$AB_{2,2}$	$AB_{2,3}$	$AB_{2,4}$
$AB_{3,1}$	$AB_{3,2}$	$AB_{3,3}$	$AB_{3,4}$
$AB_{4,1}$	$AB_{4,2}$	$AB_{4,3}$	$AB_{4,4}$

c)

$A_{11}$	$A_{12}$	$A_{13}$	$A_{14}$
$A_{21}$	$A_{22}$	$A_{23}$	$A_{24}$
$A_{31}$	$A_{32}$	$A_{33}$	$A_{34}$
$A_{41}$	$A_{42}$	$A_{43}$	$A_{44}$

$\times$

$B_{11}$	$B_{12}$	$B_{13}$	$B_{14}$
$B_{21}$	$B_{22}$	$B_{23}$	$B_{24}$
$B_{31}$	$B_{32}$	$B_{33}$	$B_{34}$
$B_{41}$	$B_{42}$	$B_{43}$	$B_{44}$

$=$

$AB_{11}$	$AB_{12}$	$AB_{13}$	$AB_{14}$
$AB_{21}$	$AB_{22}$	$AB_{23}$	$AB_{24}$
$AB_{31}$	$AB_{32}$	$AB_{33}$	$AB_{34}$
$AB_{41}$	$AB_{42}$	$AB_{43}$	$AB_{44}$

d)

$A_{11}$	$A_{12}$	$A_{13}$	$A_{14}$
$A_{21}$	$A_{22}$	$A_{23}$	$A_{24}$
$A_{31}$	$A_{32}$	$A_{33}$	$A_{34}$
$A_{41}$	$A_{42}$	$A_{43}$	$A_{44}$

 $\times$ 

$B_{11}$	$B_{12}$	$B_{13}$	$B_{14}$
$B_{21}$	$B_{22}$	$B_{23}$	$B_{24}$
$B_{31}$	$B_{32}$	$B_{33}$	$B_{34}$
$B_{41}$	$B_{42}$	$B_{43}$	$B_{44}$

 $=$ 

$AB_{11}$	$AB_{12}$	$AB_{13}$	$AB_{14}$
$AB_{21}$	$AB_{22}$	$AB_{23}$	$AB_{24}$
$AB_{31}$	$AB_{32}$	$AB_{33}$	$AB_{34}$
$AB_{41}$	$AB_{42}$	$AB_{43}$	$AB_{44}$

# Блочный алгоритм


```
void matmul(hls::stream<blockvec> &Arows, hls::stream<blockvec> &Bcols, blockmat & ABpartial,
#pragma HLS DATAFLOW

static DTYPE A[BLOCK_SIZE][SIZE];
if(iteration % (SIZE/BLOCK_SIZE) == 0){ //only load the A rows when necessary
    loadA: for(int i = 0; i < SIZE; i++) {
        blockvec tempA = Arows.read();
        for(int j = 0; j < BLOCK_SIZE; j++) {
            A[j][i] = tempA.block[j];
        }
    }
}

DTYPE AB[BLOCK_SIZE][BLOCK_SIZE] = { 0 };
partialsum: for(int k=0; k < SIZE; k++) {
    blockvec tempB = Bcols.read();
    innerB: for(int i = 0; i < BLOCK_SIZE; i++) {
        for(int j = 0; j < BLOCK_SIZE; j++) {
            AB[i][j] = AB[i][j] + A[i][k] * tempB.block[j];
        }
    }
}

writeoutput: for(int i = 0; i < BLOCK_SIZE; i++) {
    for(int j = 0; j < BLOCK_SIZE; j++) {
        ABpartial.matrix[i][j] = AB[i][j];
    }
}
}
```


# Блочный алгоритм

	Negative Slack	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
▼  matmul	-	2	3	594	1055	7487	7446	dataflow
● Loop_memset_AB_proc9	-	0	3	274	407	7445	7445	none
● Block_proc8	-	0	0	152	223	1~769	1 ~ 769	none
● Loop_writeoutput_pro	-	0	0	26	119	41	41	none
● matmul_entry5	-	0	0	2	26	0	0	none

# Блочный алгоритм: Конвейеризация innerB

	Negative Slack	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
▼ matmul	-	4	12	589	1371	2851~3108	2838	dataflow
❗ Loop_memset_AB_proc9	-	0	12	448	667	2837	2837	none
❗ Loop_writeoutput_pro	-	0	0	29	210	10	10	none
● Block_proc8	-	0	0	96	277	1~258	1 ~ 258	none
● matmul_entry6	-	0	0	2	26	0	0	none

# Блочный алгоритм: Конвейеризация partialsum

	Negative Slack	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
▼  matmul	-	10	48	1735	2364	2081	2071	dataflow
<i>ii</i> Loop_memset_AB_proc9	-	0	48	1451	1463	2070	2070	none
<i>ii</i> Loop_writeoutput_pro	-	0	0	29	210	10	10	none
● Block_proc8	-	0	0	15	222	1~130	1 ~ 130	none
● matmul_entry5	-	0	0	2	26	0	0	none



## Performance Estimates

### ▣ Timing (ns)

Clock		no_flags	pipeline_all	pipeline_inner
ap_clk	Target	10.00	10.00	10.00
	Estimated	8.470	8.470	8.470

### ▣ Latency (clock cycles)

		no_flags	pipeline_all	pipeline_inner
Latency	min	7487	2081	2851
	max	7487	2081	3108
Interval	min	7446	2071	2838
	max	7446	2071	2838

### Utilization Estimates

	no_flags	pipeline_all	pipeline_inner
BRAM_18K2	2	10	4
DSP48E	3	48	12
FF	594	1735	589
LUT	1055	2364	1371

## Преимущества блочного алгоритма

- Большие матрицы
- Поточковые данные
- Параллелизация вычислений блоков