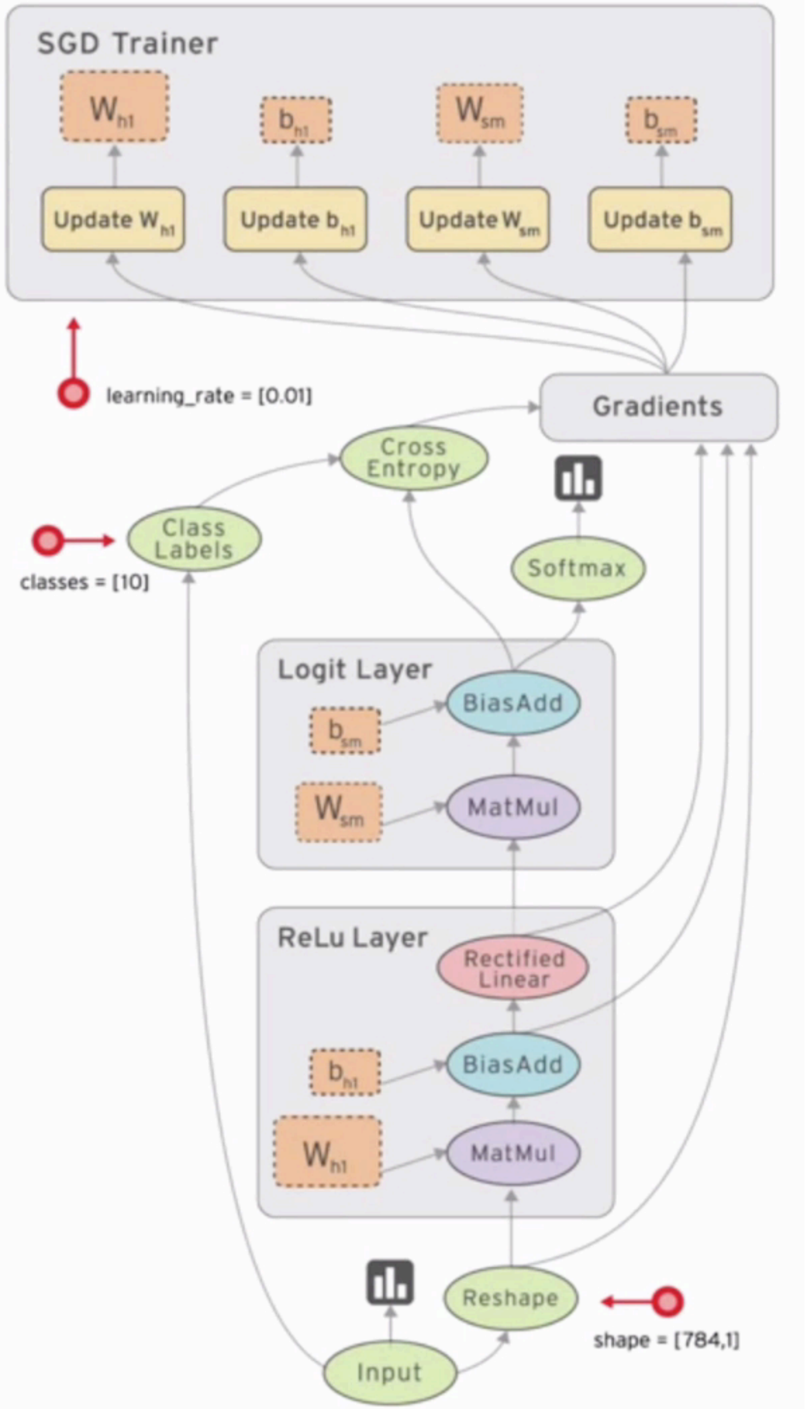
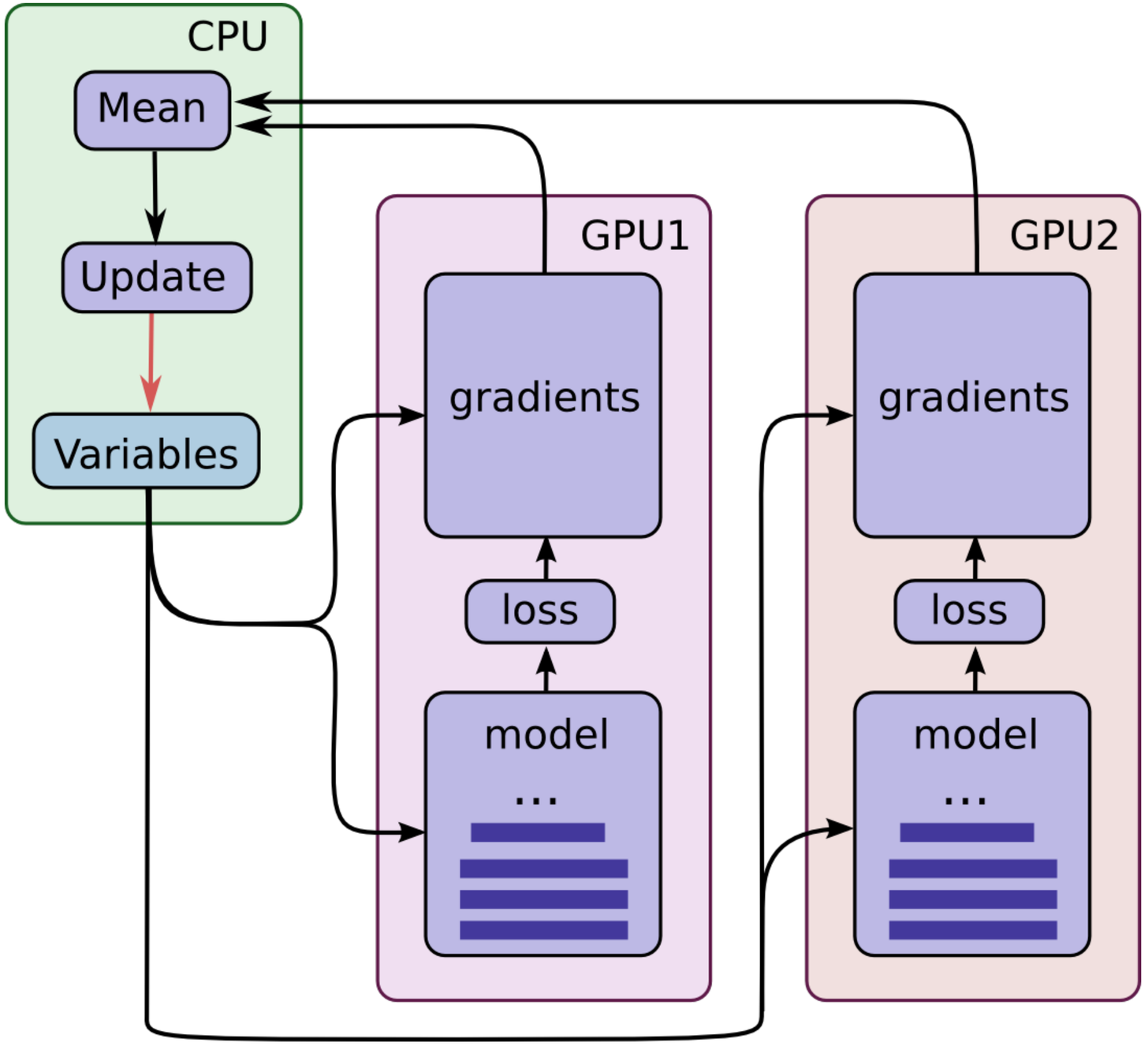
JOHN MCALLISTER
QUEEN'S UNIVERSITY BELFAST

HETEROGENEOUS DATAFLOW FOR HETEROGENEOUS MPSOC FPGA


SGD Trainer
W_{h1}
b_{h1}
W_{sm}
b_{sm}
Update W_{h1}
Update b_{h1}
Update W_{sm}
Update b_{sm}
learning_rate = [0.01]
Gradients
Cross Entropy
Class Labels
classes = [10]
Softmax
Logit Layer
BiasAdd
b_{sm}
W_{sm}
MatMul
ReLu Layer
Rectified Linear
BiasAdd
b_{h1}
W_{h1}
MatMul
Reshape
shape = [784,1]
Input


CPU
Mean
Update
Variables
GPU1
gradients
loss
model
...
GPU2
gradients
loss
model
...

A MAJOR CHALLENGE IN UTILIZING HETEROGENEOUS RESOURCES IS THE DIVERSITY OF DEVICES. . . .

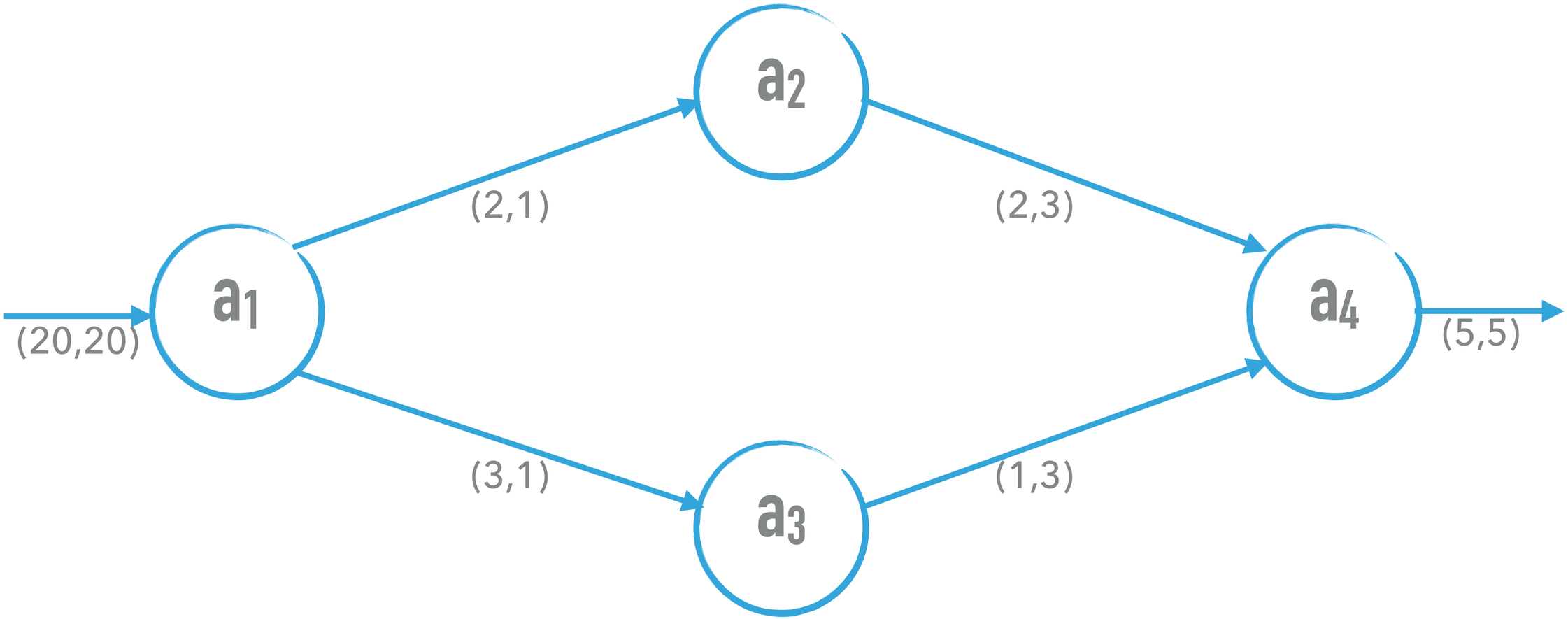
- A PROGRAM OPTIMIZED FOR GPU IS OFTEN VERY DIFFERENT FROM ONE OPTIMIZED FOR CPU
- ON ONE MACHINE, A SPECIFIC COMPUTATION MAY RUN BEST ON CPU, ON ANOTHER MACHINE IT MAY RUN BEST ON GPU.
- EVEN IF A SPECIFIC KERNEL MAY PERFORM BETTER ON THE GPU, IF THE GPU IS OVERLOADED AND THE CPU IS IDLE, IT MAY BE BEST TO BALANCE THE WORKLOAD BETWEEN THE TWO, OR IT MAY BE BEST TO PLACE COMPUTATION SOMEWHERE IT RUNS MORE SLOWLY BUT NEARER TO WHERE ITS OUTPUT WILL BE USED.

Portable Performance on Heterogeneous Architectures

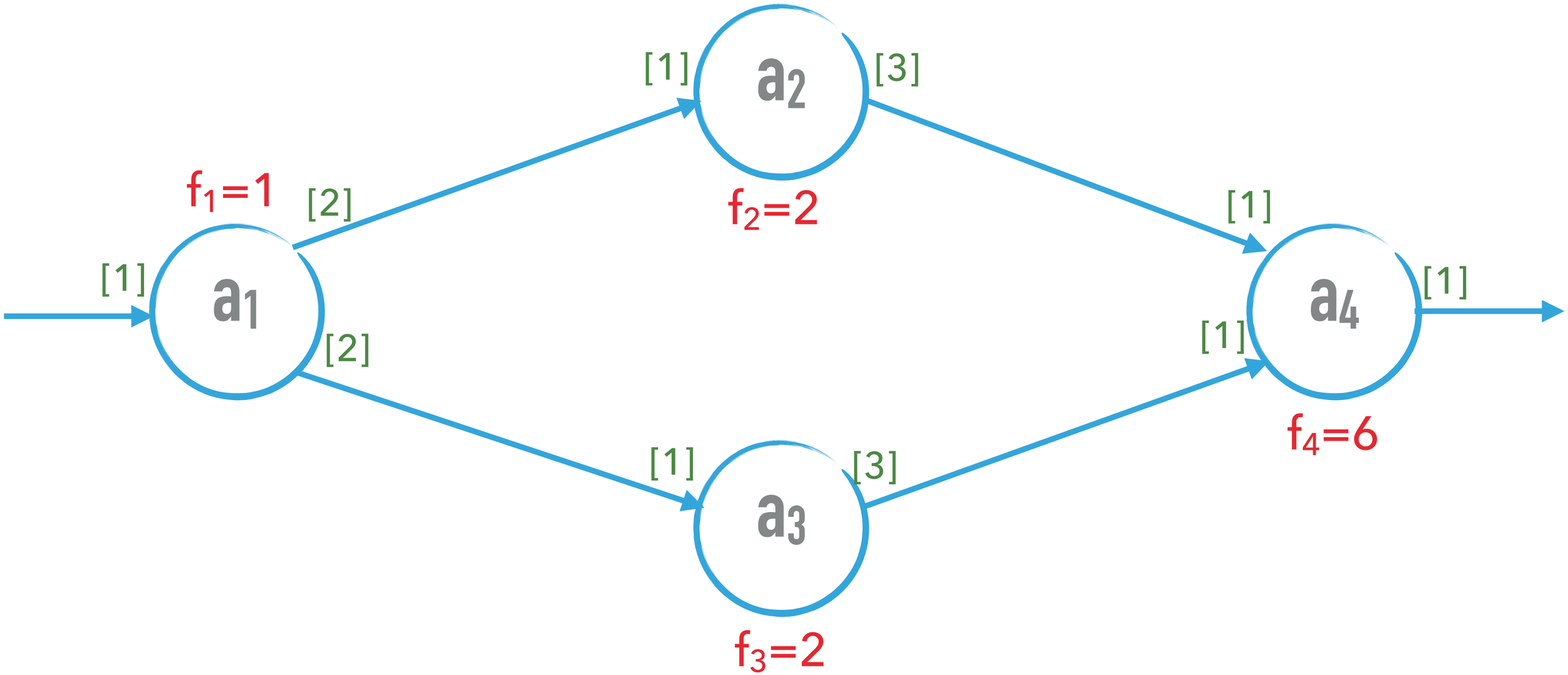
Phothilimthana, Ansel , Ragan-Kelley, Amarasinghe

ASPLOS 2013.

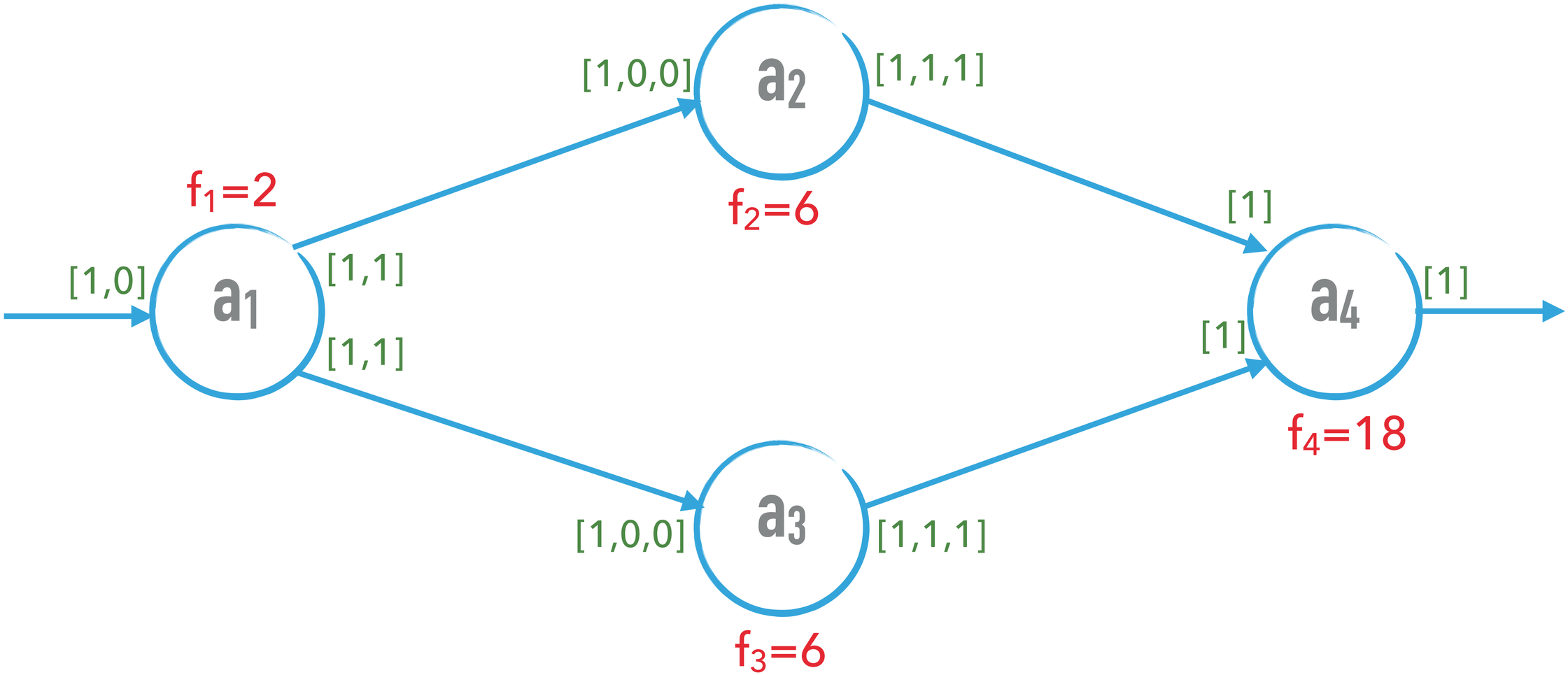
DATAFLOW



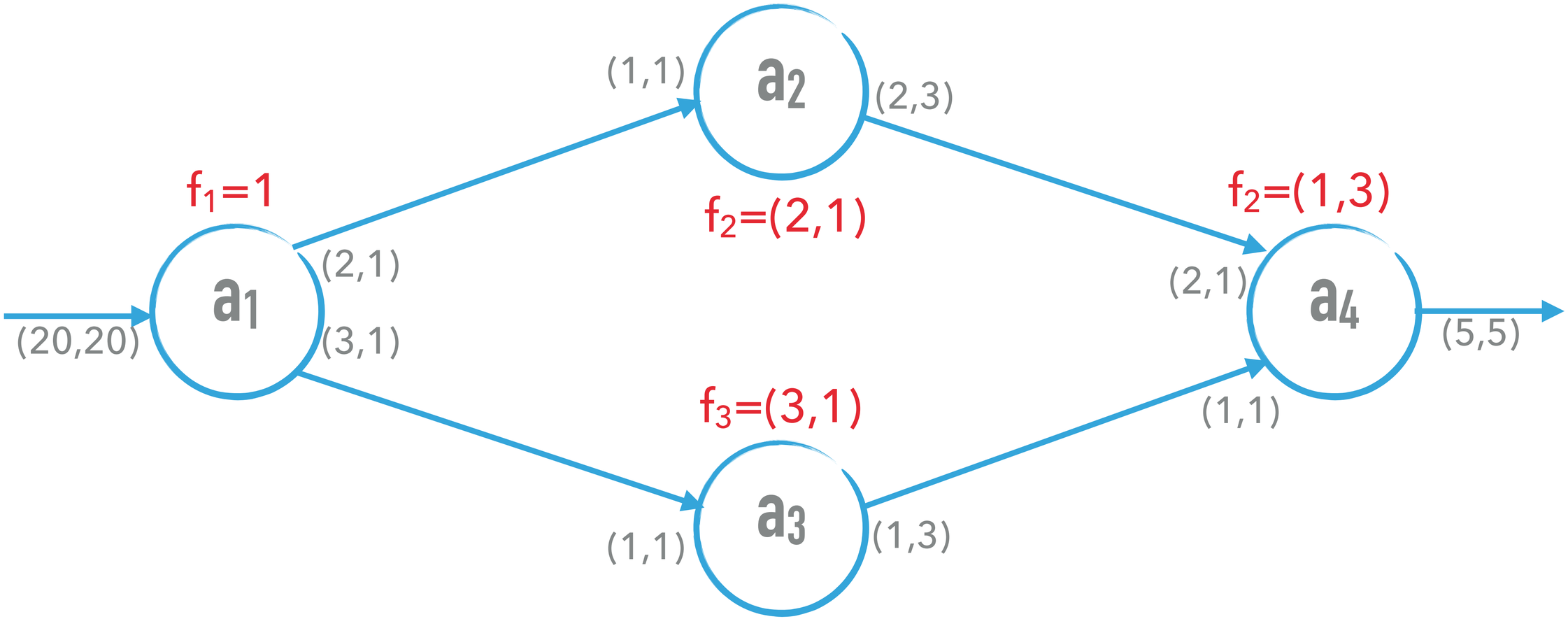
DATAFLOW - SYNCHRONOUS (SDF)



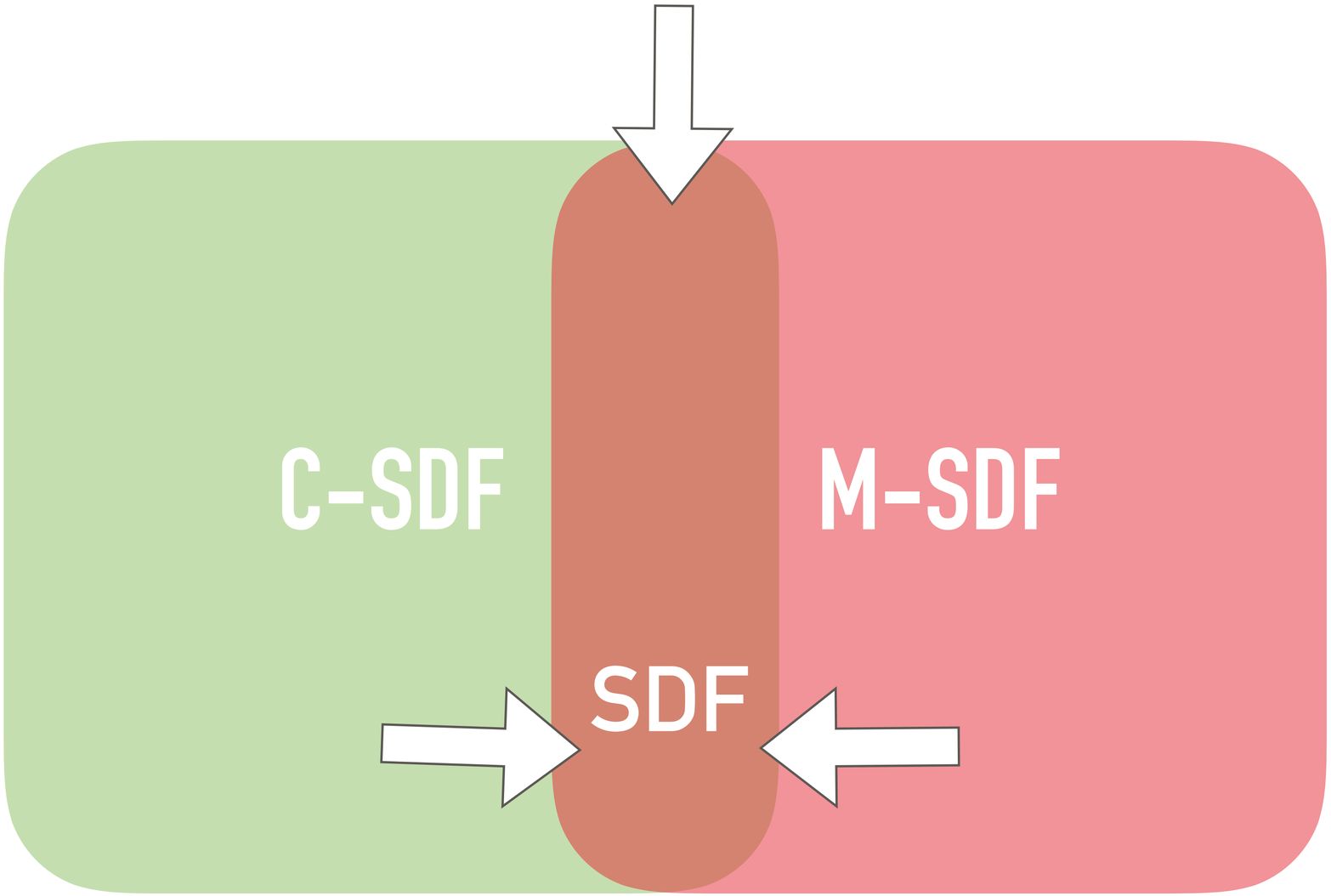
DATAFLOW – CYCLO-STATIC (C-SDF)

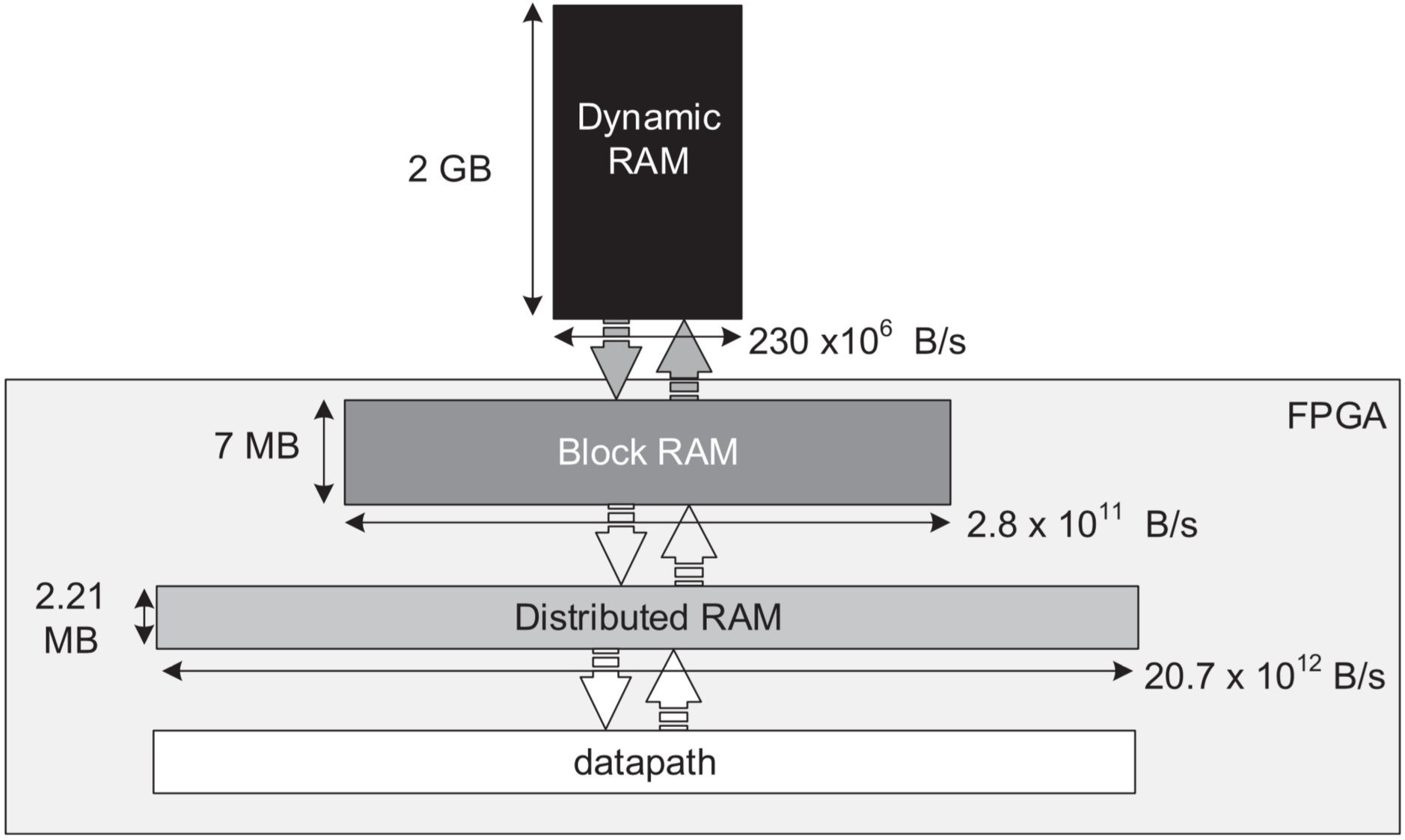


DATAFLOW – MULTIDIMENSIONAL (M-SDF)

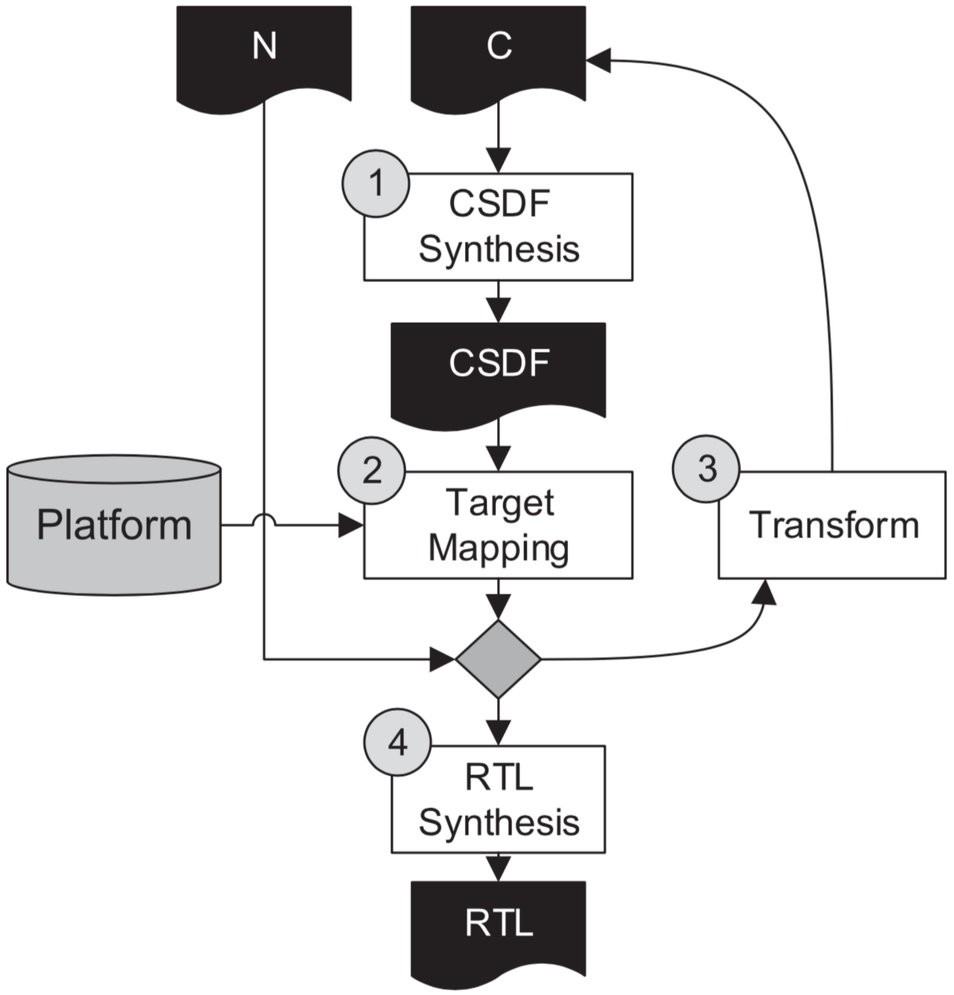


DYNAMIC

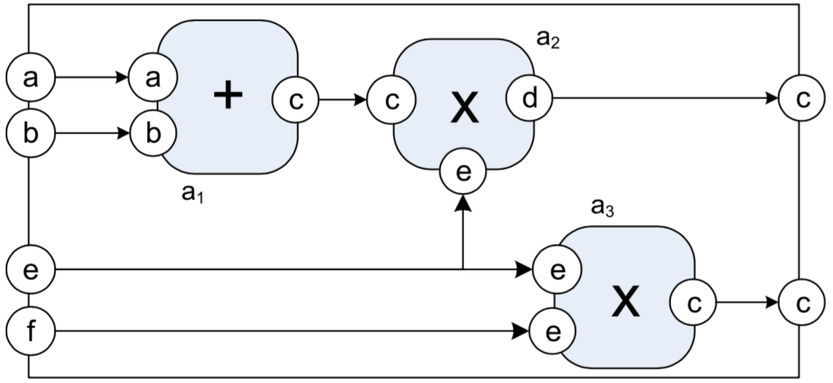



Dynamic RAM
2 GB
230 x10^6 B/s
FPGA
7 MB
Block RAM
2.8 x 10^11 B/s
2.21 MB
Distributed RAM
20.7 x 10^12 B/s
datapath

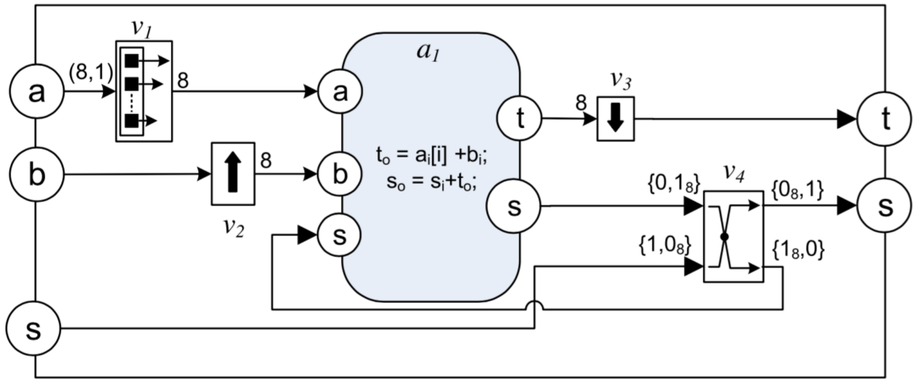
```
1:  procedure FSME (R, C)
2:      for i ← 0 : 17 do
3:          for j ← 0 : 21 do
4:              for k ← 0 : 31 do
5:                  for l ← 0 : 31 do
6:                      s ← 0
7:                      for m ← 0 : 15 do
8:                          for n ← 0 : 15 do
9:                              s ← s + R[16i + k + m][16j + l + n]
                                    −C[i + m][j + n]
10: return s
```



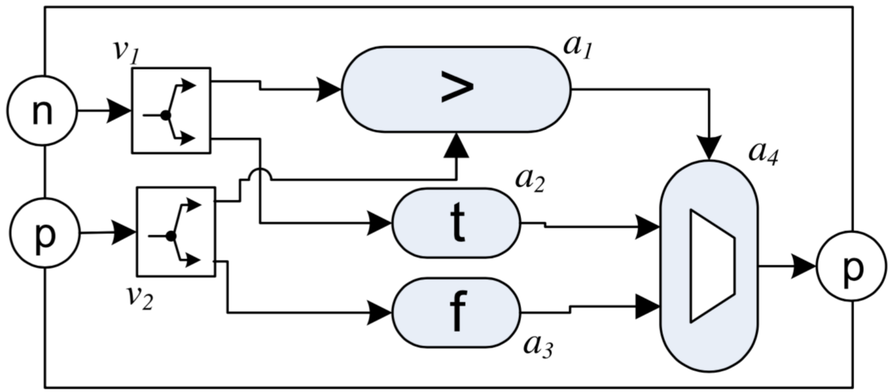
```
1: procedure BLOCKSTATEMENT (a, b, e, f)
2:     c ← a + b;
3:     d ← c ∗ e;
4:     c ← e ∗ f;
5:     return c, d
```

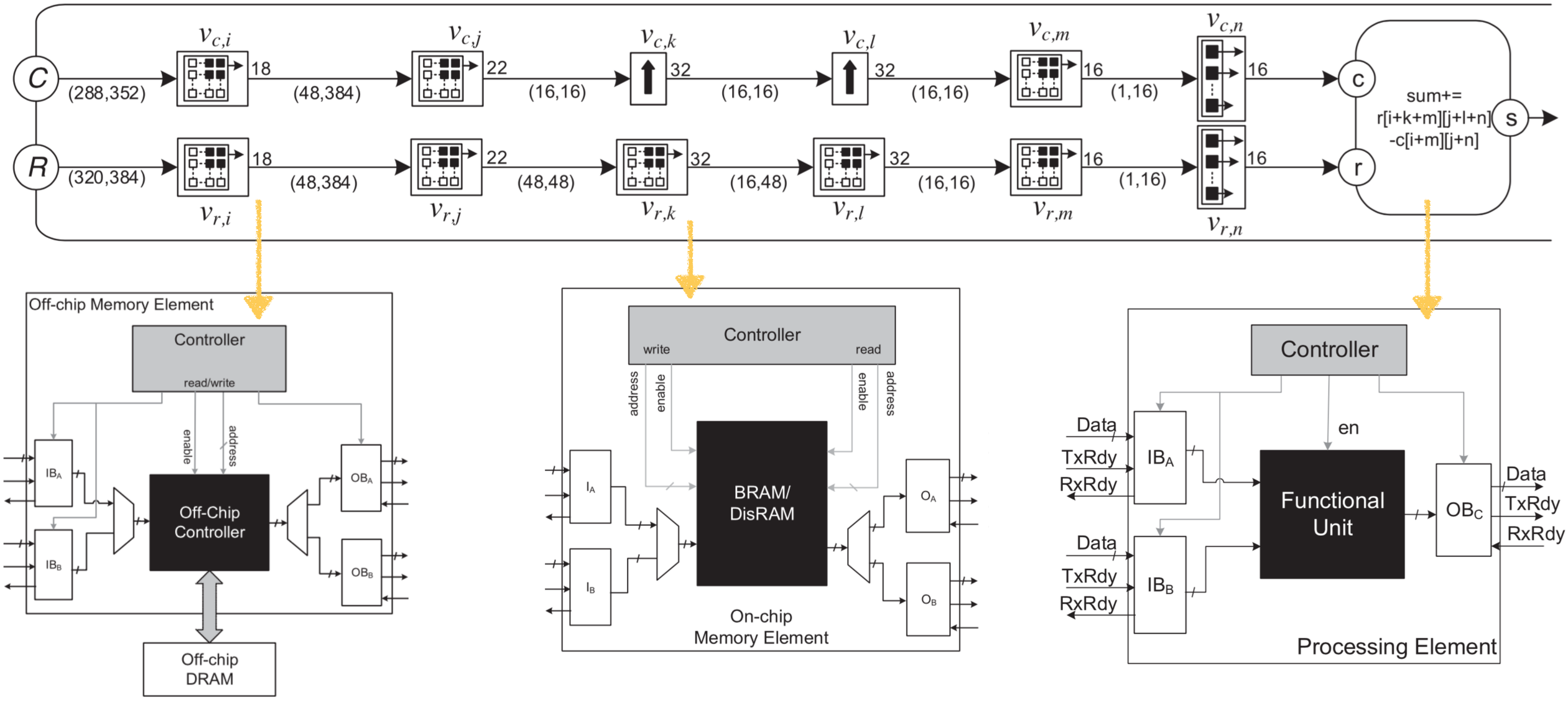


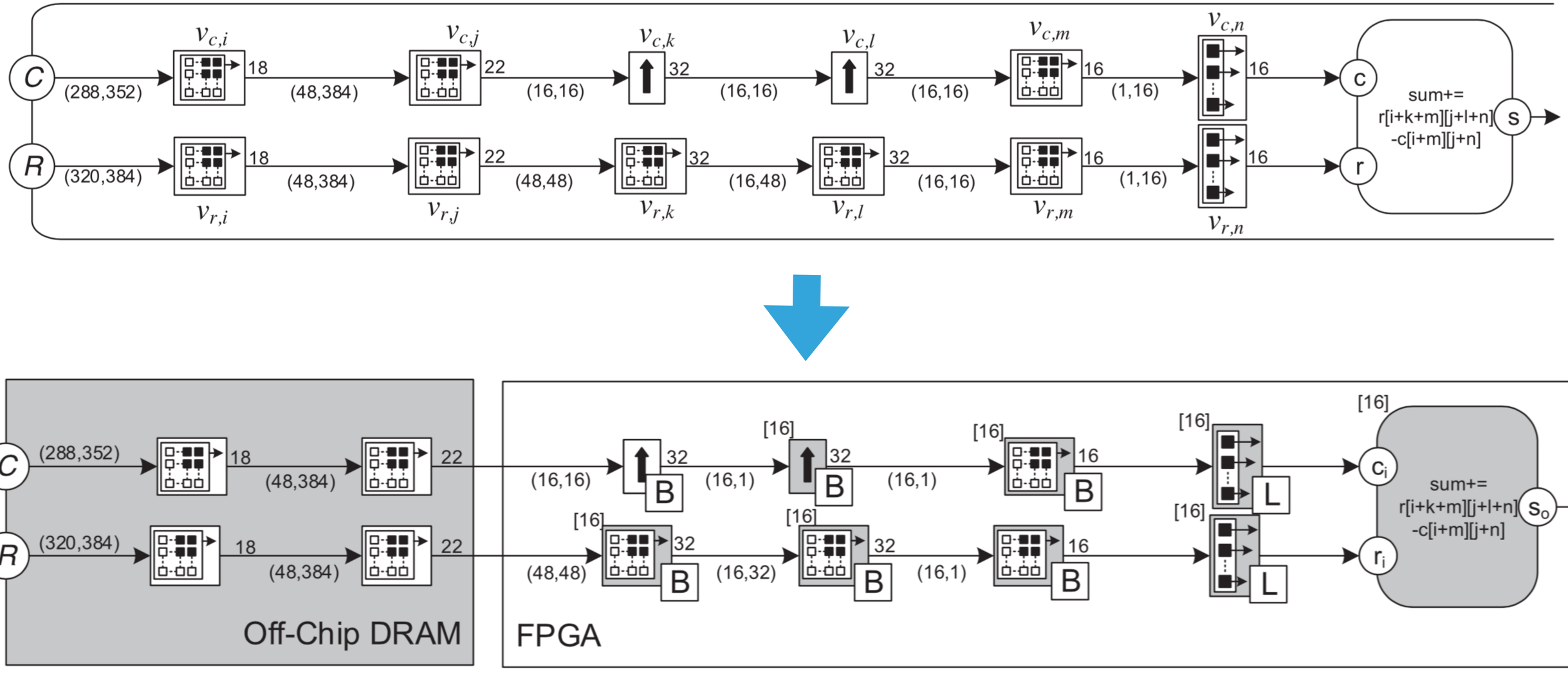

```
1: procedure LOOPSTATEMENT (m [8], b, s, t)
2:     s ← 0
3:     for i ← 1 : 8 do
4:         t ← m [i] + b
5:         s ← s + t;
6:     return s
```




```
1: procedure CONDITIONALSTATEMENT (n, p)
2:     if n > p then
3:         p = n
4:     else
5:         p = p
6:     return p
```




C
R
$v_{c,i}$
$v_{c,j}$
$v_{c,k}$
$v_{c,l}$
$v_{c,m}$
$v_{c,n}$
$v_{r,i}$
$v_{r,j}$
$v_{r,k}$
$v_{r,l}$
$v_{r,m}$
$v_{r,n}$
(288,352)
(320,384)
18
22
32
16
(48,384)
(16,16)
(48,48)
(16,48)
(1,16)
c
r
s
sum+=
r[i+k+m][j+l+n]
-c[i+m][j+n]
Off-chip Memory Element
Controller
read/write
enable
address
IB_A
IB_B
OB_A
OB_B
Off-Chip
Controller
Off-chip
DRAM
write
read
I_A
I_B
O_A
O_B
BRAM/
DisRAM
On-chip
Memory Element
Data
TxRdy
RxRdy
en
Functional
Unit
OB_C
Processing Element


C
R
(288,352)
(320,384)
$v_{c,i}$
$v_{c,j}$
$v_{c,k}$
$v_{c,l}$
$v_{c,m}$
$v_{c,n}$
$v_{r,i}$
$v_{r,j}$
$v_{r,k}$
$v_{r,l}$
$v_{r,m}$
$v_{r,n}$
18
22
32
16
(48,384)
(16,16)
(48,48)
(16,48)
(1,16)
c
r
s
sum+=
r[i+k+m][j+l+n]
-c[i+m][j+n]
Off-Chip DRAM
FPGA
B
L
[16]
(16,1)
(16,32)
c_i
r_i
s_o

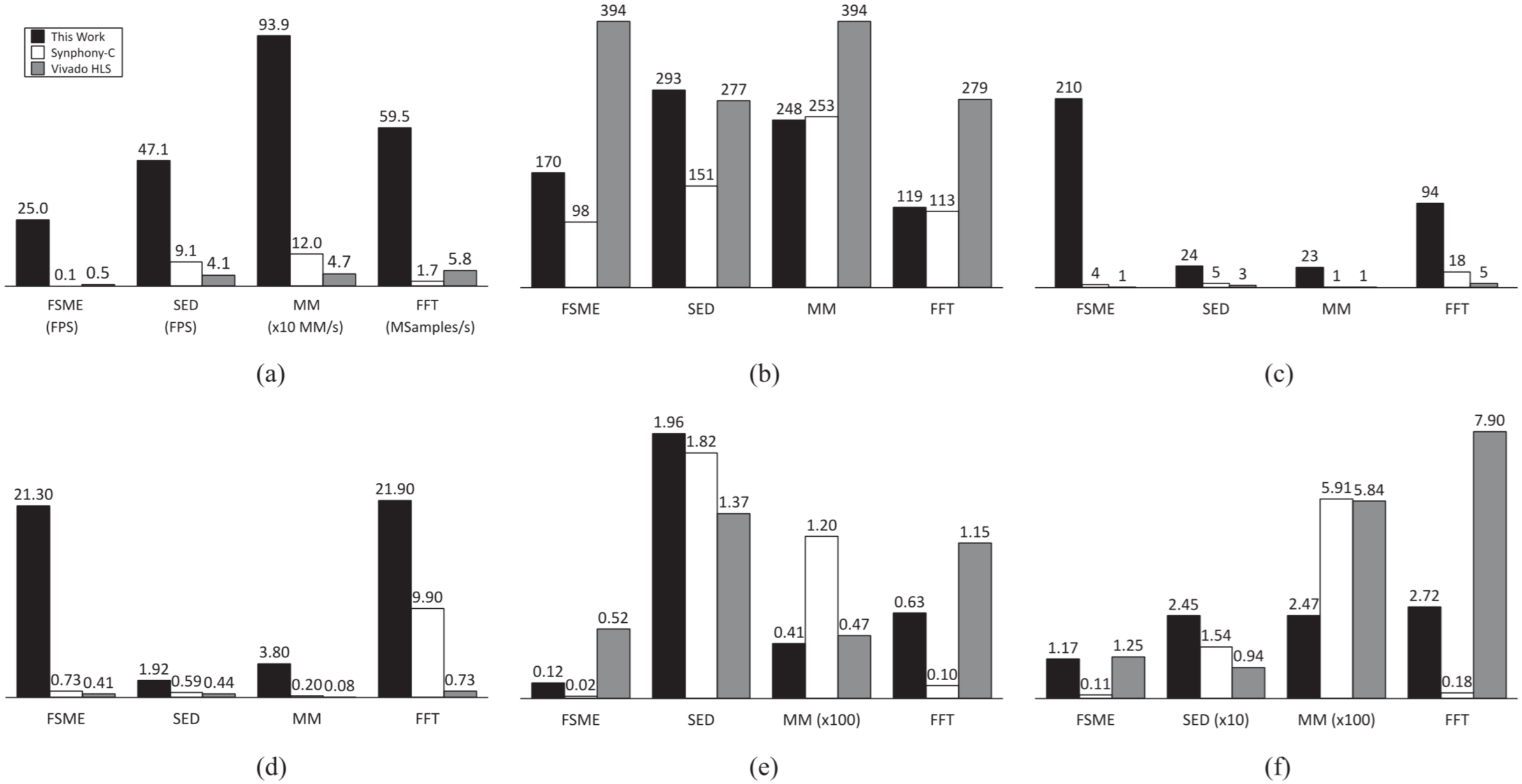
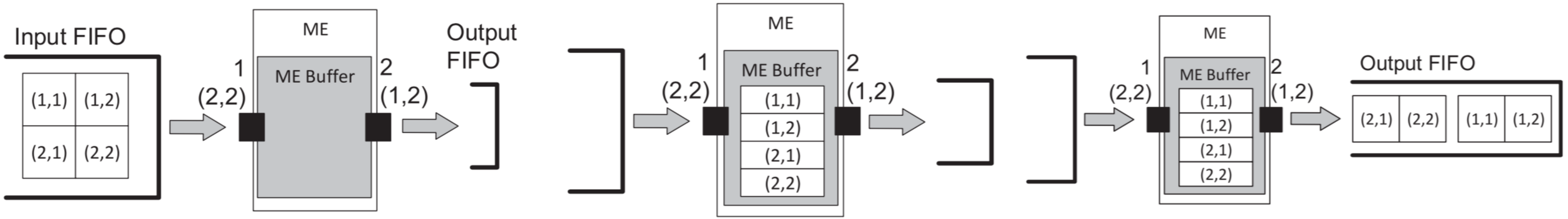
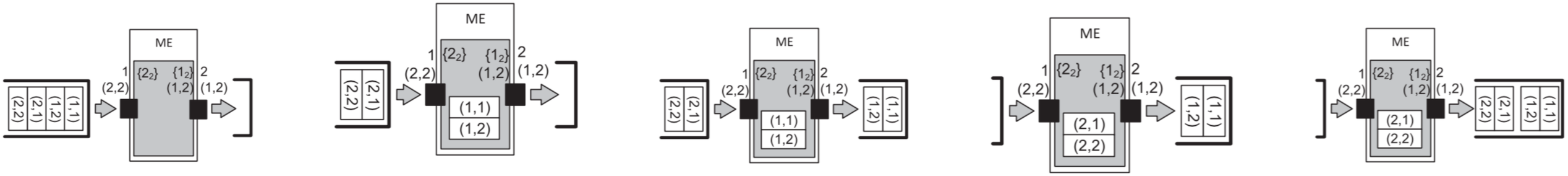


Fig. 10. Synthesis results and comparison. (a) T; (b) Clk (MHz); (c) DSP48E; (c) LUTs $\left(\times 10^3\right)$; (e) T/DSP; and (f) T/LUT.

Matrix-Vector Conversion: SDF



Matrix-Vector Conversion: C-SDF



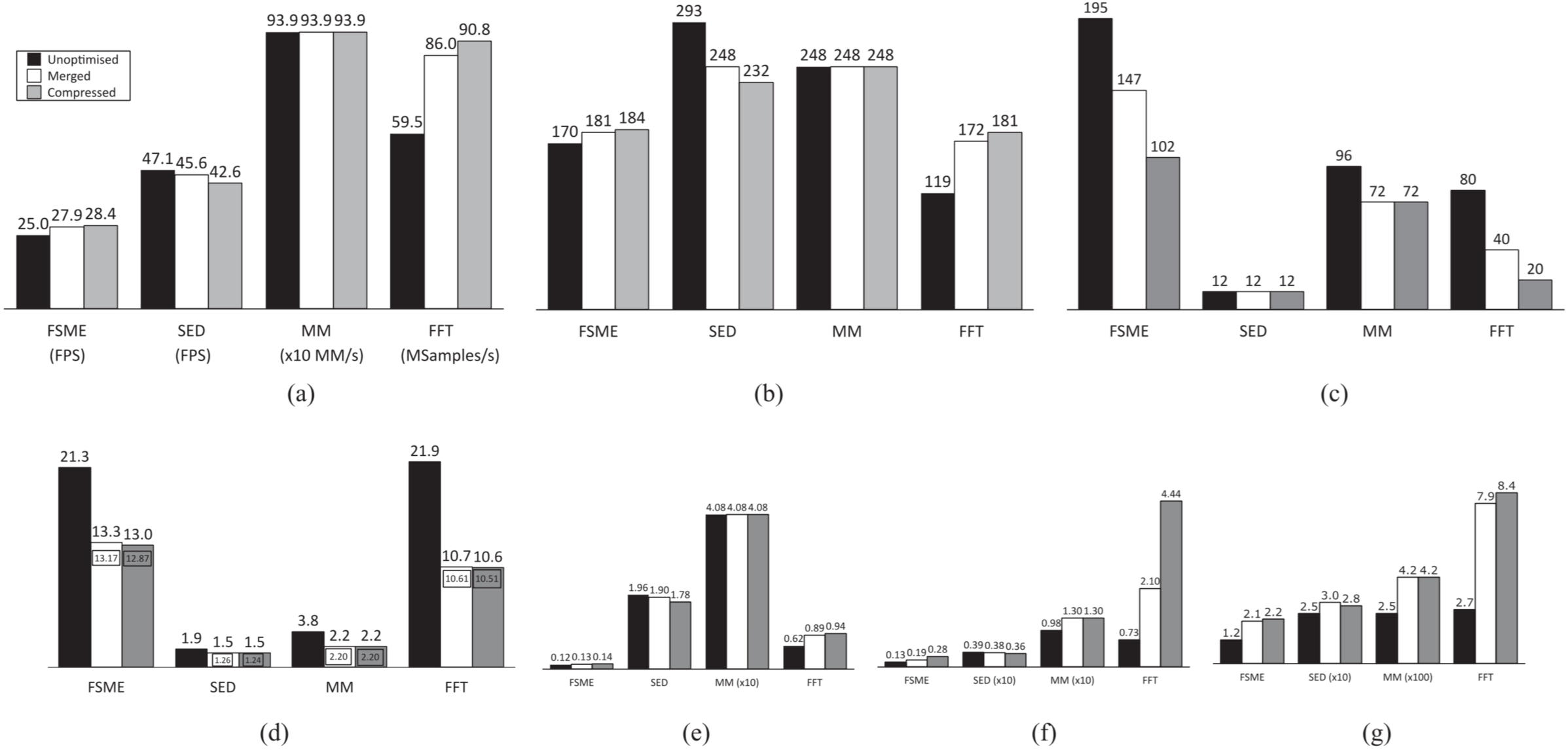


Fig. 14. Optimized synthesis—cost, performance, efficiency: (a) T; (b) clk (MHz); (c) BRAM; (d) LUT; (e) T/DSP; (f) T/BRAM; and (g) T/LUT.

DYNAMIC