NHR FAU

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

OSACA – Open-Source Architecture Code Analyzer

Introduction and tutorial

OSACA modeling

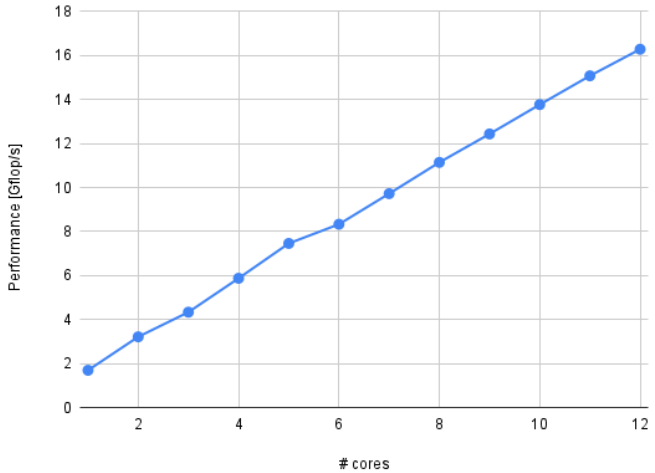
Remember the SpMV model?

$$P = \min\left(P_{max}, \frac{b_S}{B_C}\right)$$

OSACA modeling

- Remember the SpMV model?

$$P = \min\left(P_{max}, \frac{b_S}{B_C}\right)$$

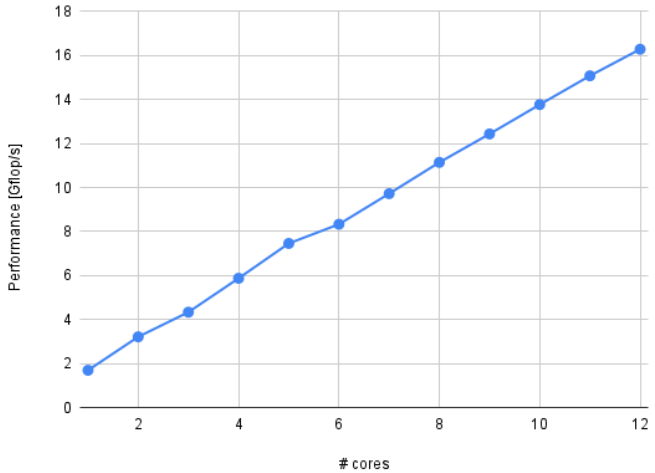


- Obviously, the code is not memory bound

OSACA modeling

- Remember the SpMV model?

$$P = \min\left(P_{max}, \frac{b_S}{B_C}\right)$$



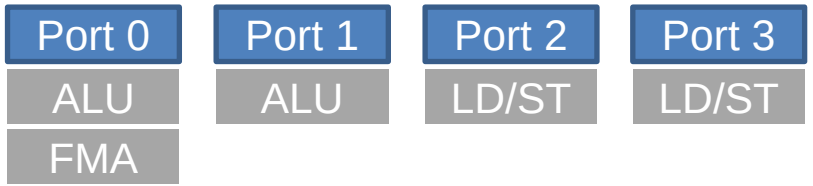
- Obviously, the code is not memory bound
- The assumption that $P_{\max}$ is insignificant may be wrong
  → $P_{\max}$, i.e., in-core modeling required!
  → OSACA!

Assumptions for in-core modeling

- Steady-state execution, *i.e.,* ***no*** *warm-up or cool-down phase*
- All data comes from L1 cache
  - Other tools (→ Kerncraft) can be used for modeling beyond L1
- Architecture-specific hardware model (“port model”)

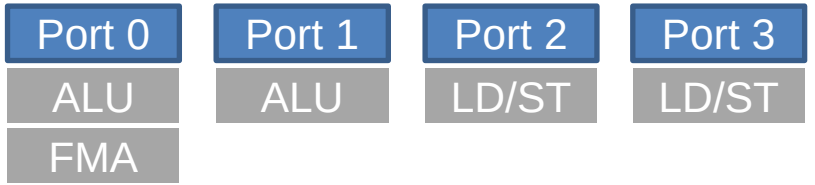
Assumptions for in-core modeling

- Steady-state execution, *i.e., **no** warm-up or cool-down phase*
- All data comes from L1 cache
  - Other tools (→ Kerncraft) can be used for modeling beyond L1
- Architecture-specific hardware model (“port model”)



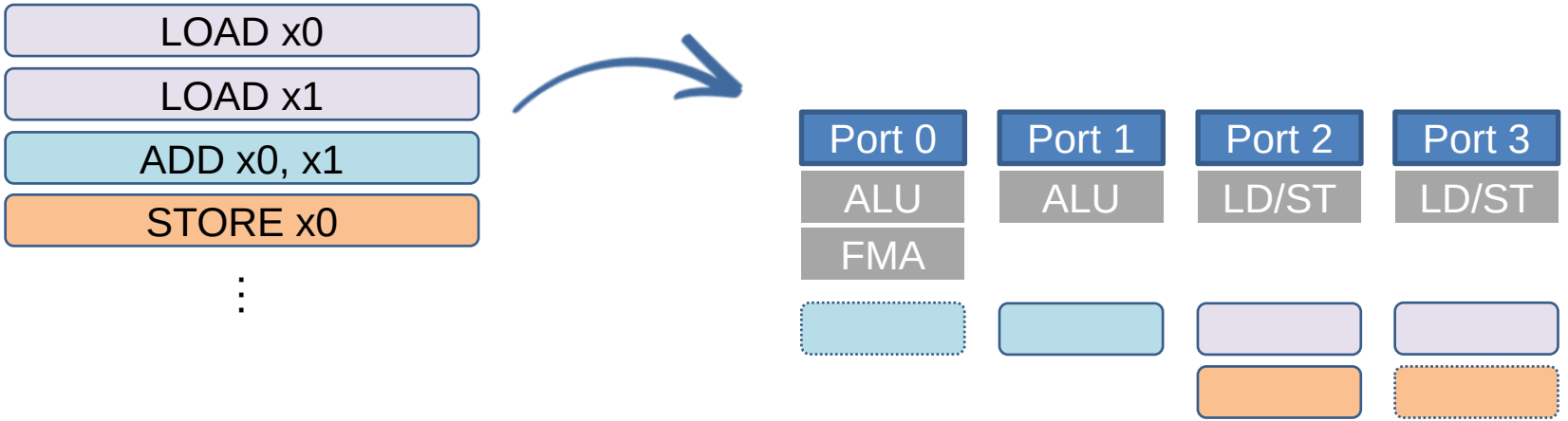
Assumptions for in-core modeling

- Steady-state execution, *i.e., **no** warm-up or cool-down phase*
- All data comes from L1 cache
  - Other tools (→ Kerncraft) can be used for modeling beyond L1
- Architecture-specific hardware model (“port model”)



Assumptions for in-core modeling

- Steady-state execution, *i.e., **no** warm-up or cool-down phase*
- All data comes from L1 cache
  - Other tools (→ Kerncraft) can be used for modeling beyond L1
- Architecture-specific hardware model (“port model”)



In-core metrics

Three metrics to estimate the in-core runtime:

- **Throughput** (TP)
- **Latency** (LT)
- **Loop-carried dependencies** (LCD)

Simplified runtime estimation: $t_c = \max(TP, LCD)$

In-core metrics

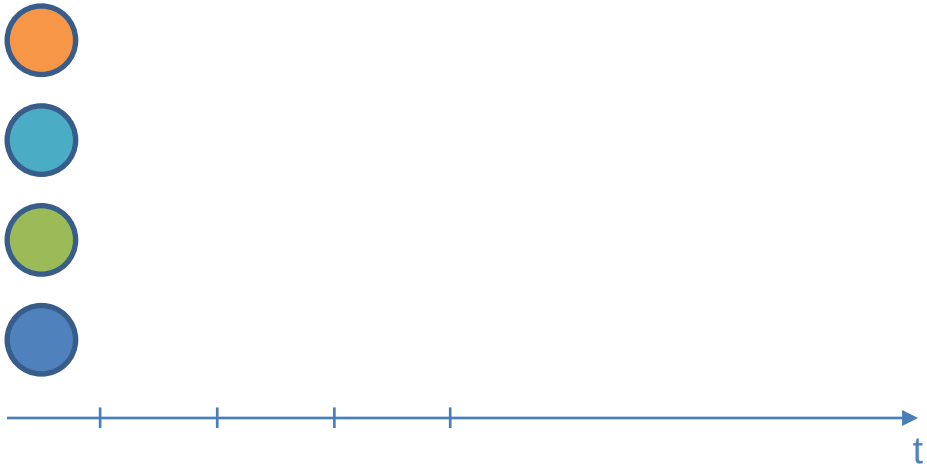
- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -

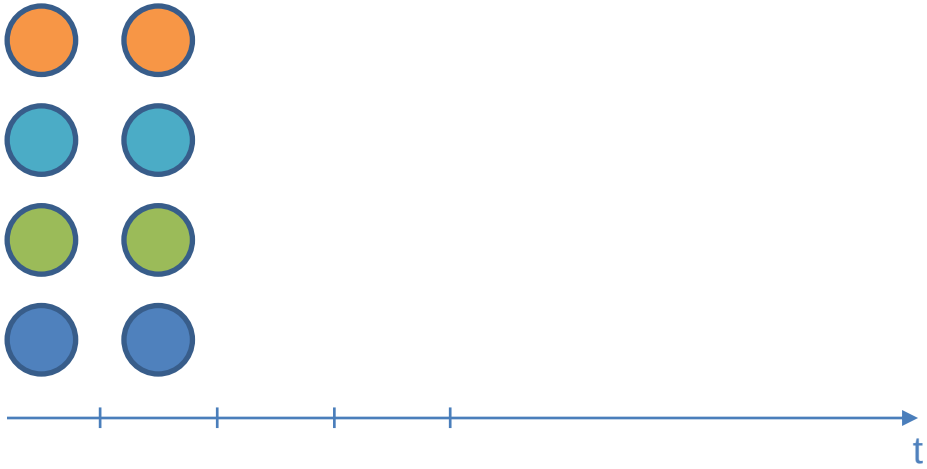
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -



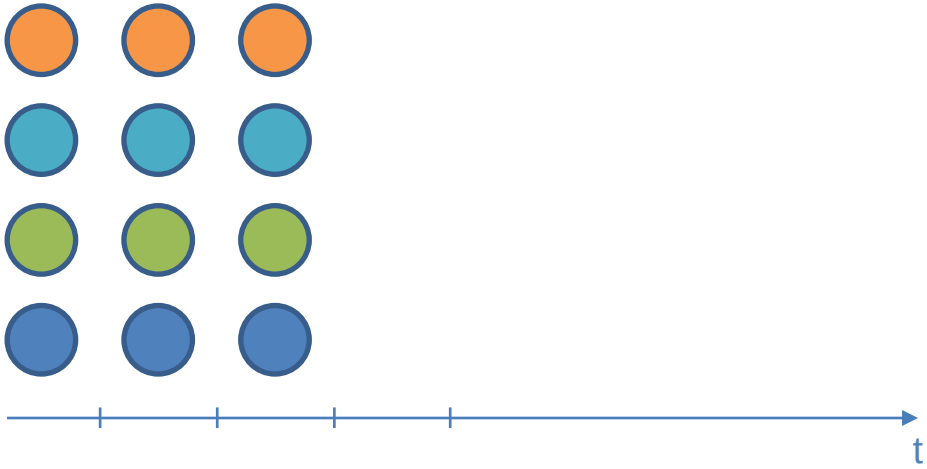
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -



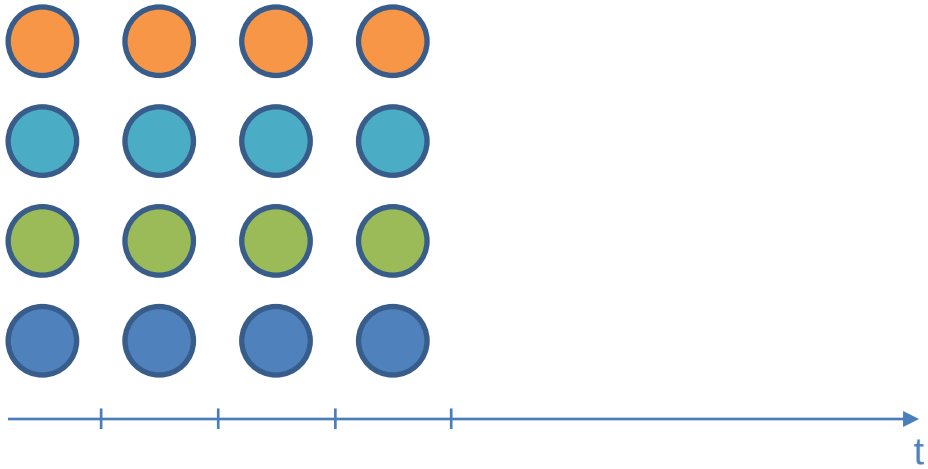
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -



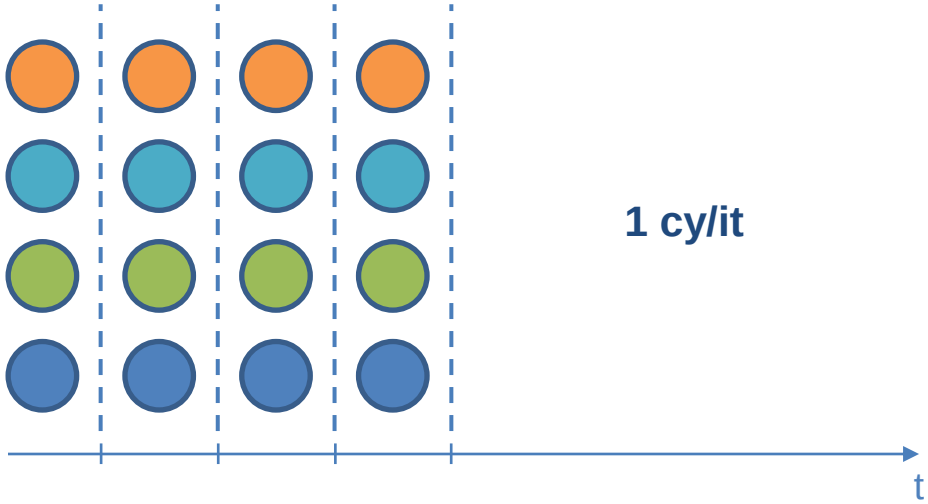
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -



In-core metrics

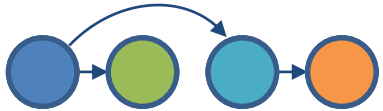
- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:
  - No dependencies within loop
  - No intra-loop dependencies
  - TP prediction: 1cy
  - CP prediction: 1 cy
  - LCD prediction: -



In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 

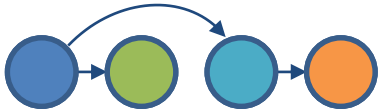
In-core metrics

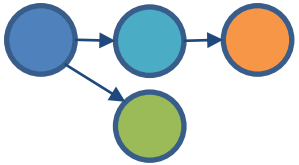
- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - Dependencies within loop
  - No intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



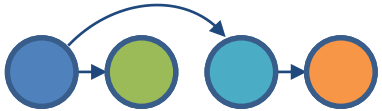


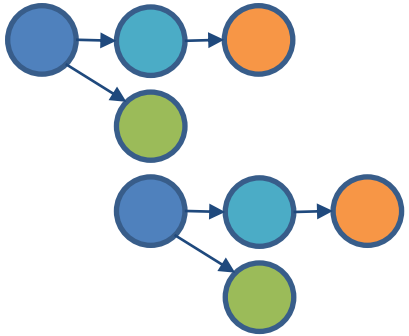
  - Dependencies within loop
  - No intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



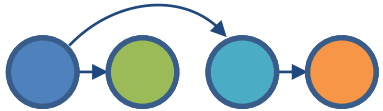


  - Dependencies within loop
  - No intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -

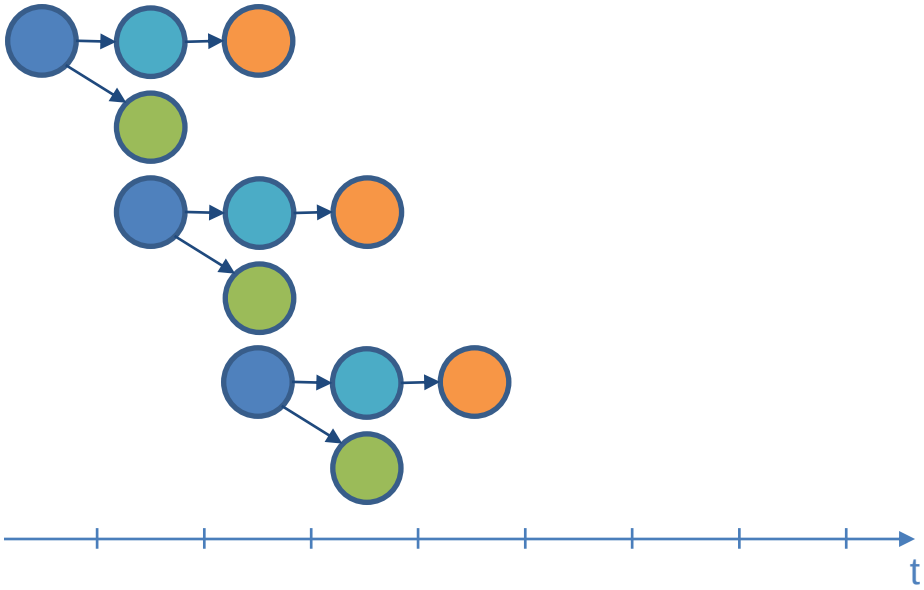
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



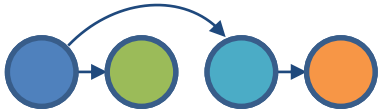
  - Dependencies within loop
  - No intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -



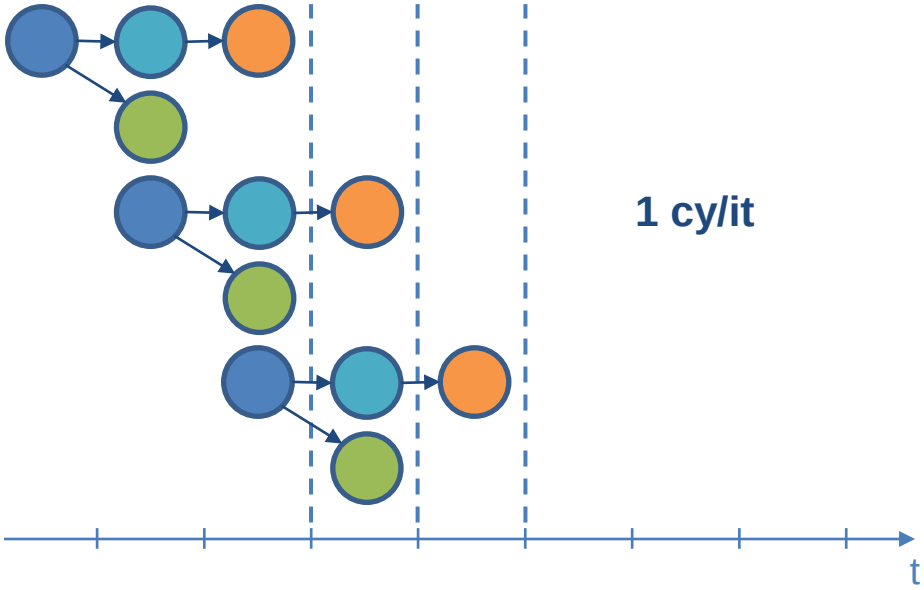
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



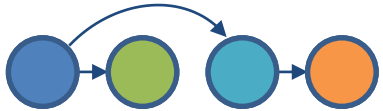
  - Dependencies within loop
  - No intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -



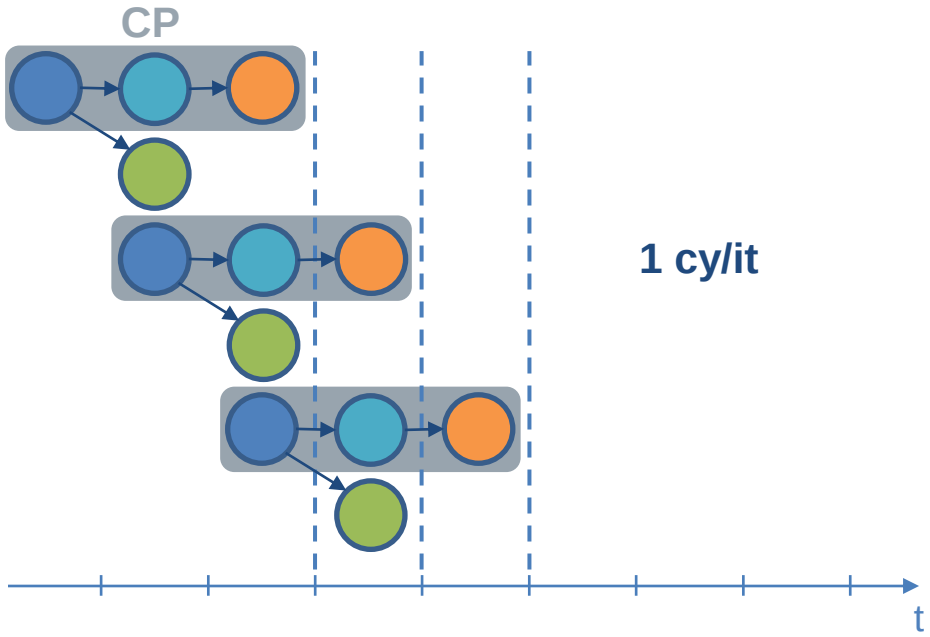
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:

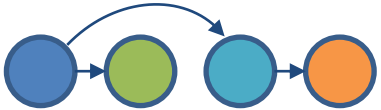


  - Dependencies within loop
  - No intra-loop dependencies

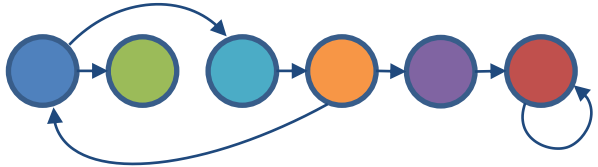
  - TP prediction: 1cy
  - CP prediction: 3 cy
  - LCD prediction: -



In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 

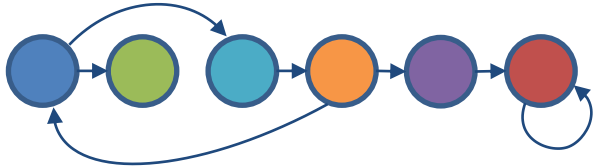
In-core metrics

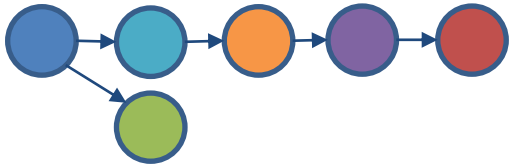
- Exemplary port model:
  - Six types of functional units (i.e., types of instructions): 
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code: 
  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



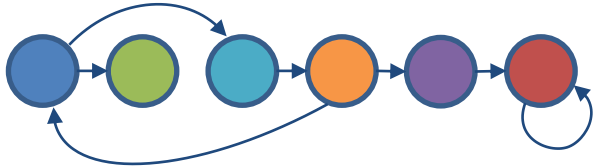


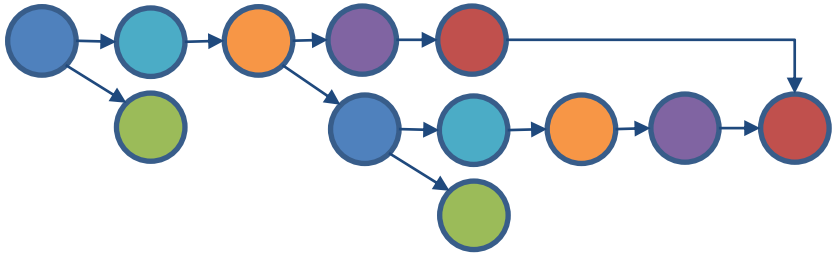
  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



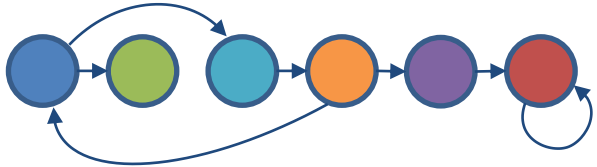


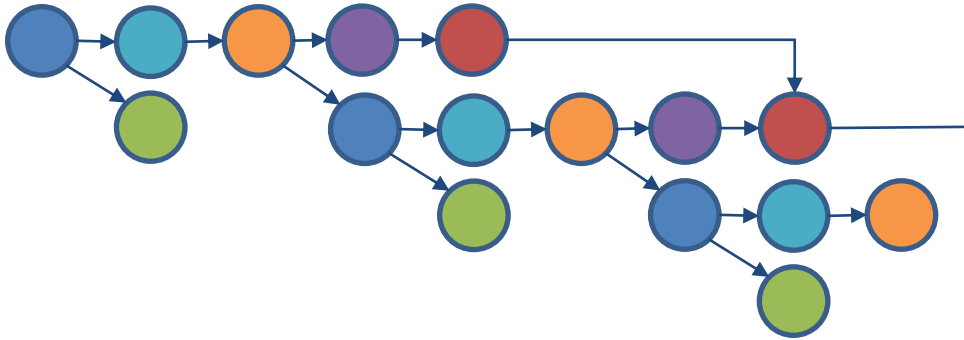
  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



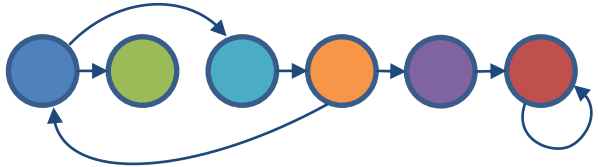


  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

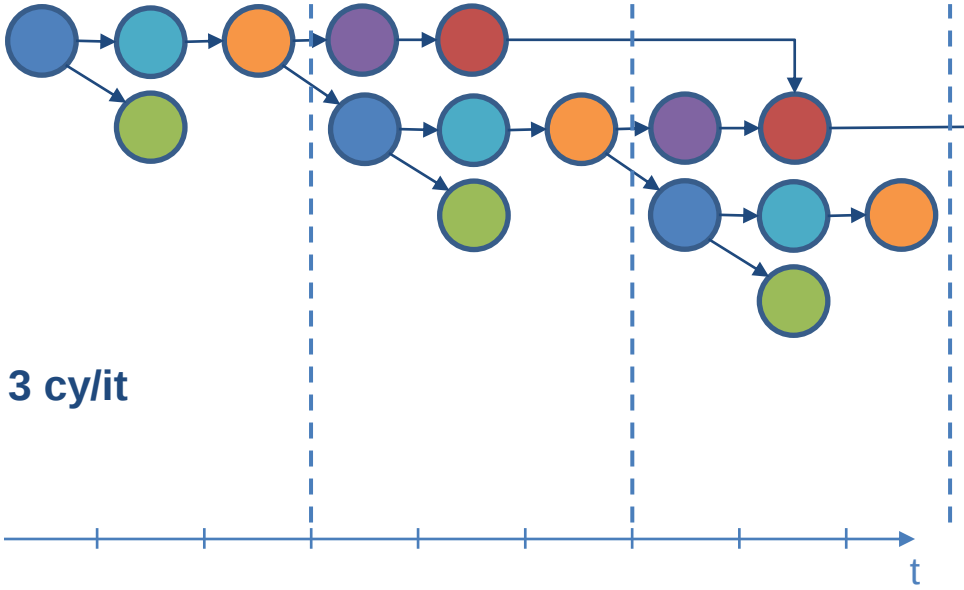
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



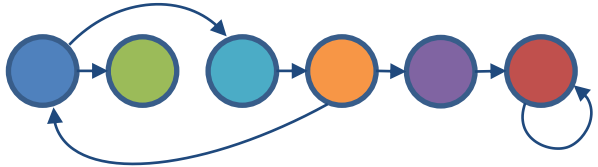
  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy



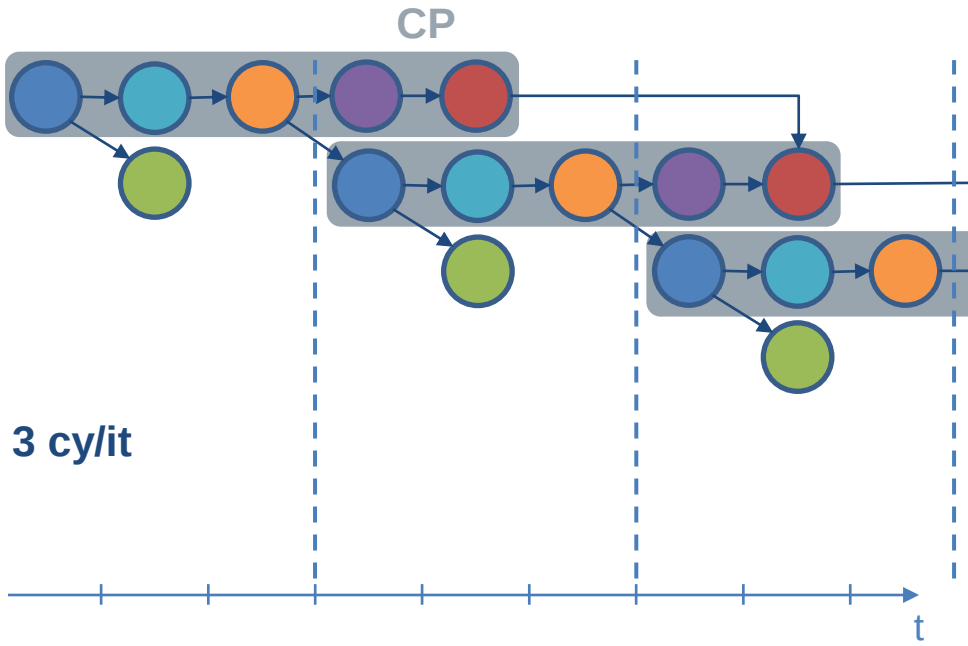
In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:



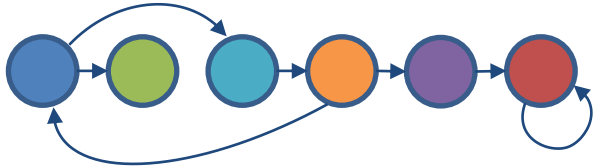
  - Dependencies within loop
  - Intra-loop dependencies

  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

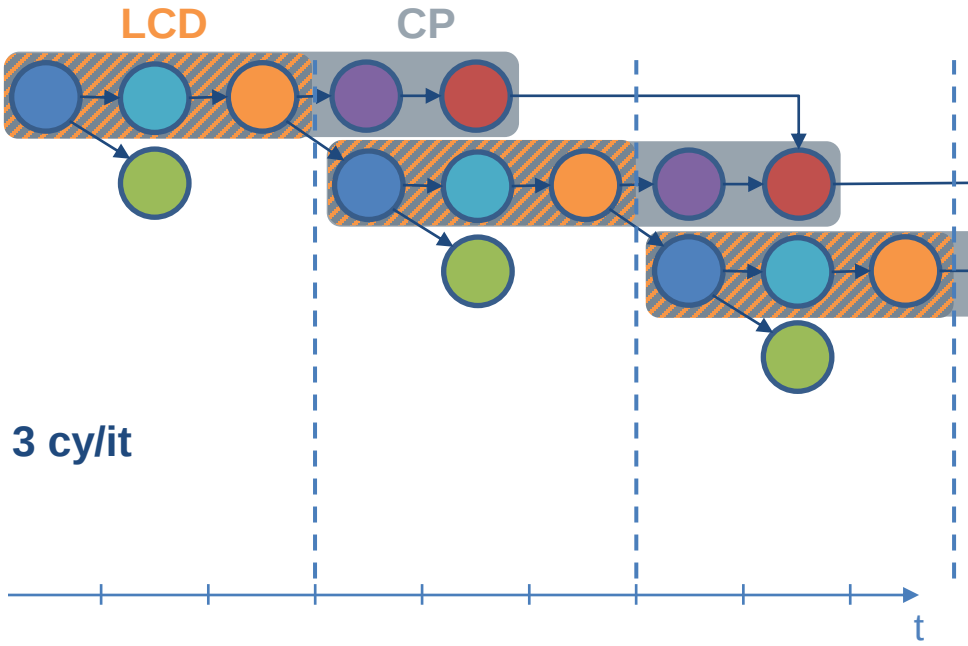


In-core metrics

- Exemplary port model:
  - Six types of functional units (i.e., types of instructions):
  - Reciprocal throughput for each instruction: 1cy
  - Latency for each instruction: 1cy
- Code:

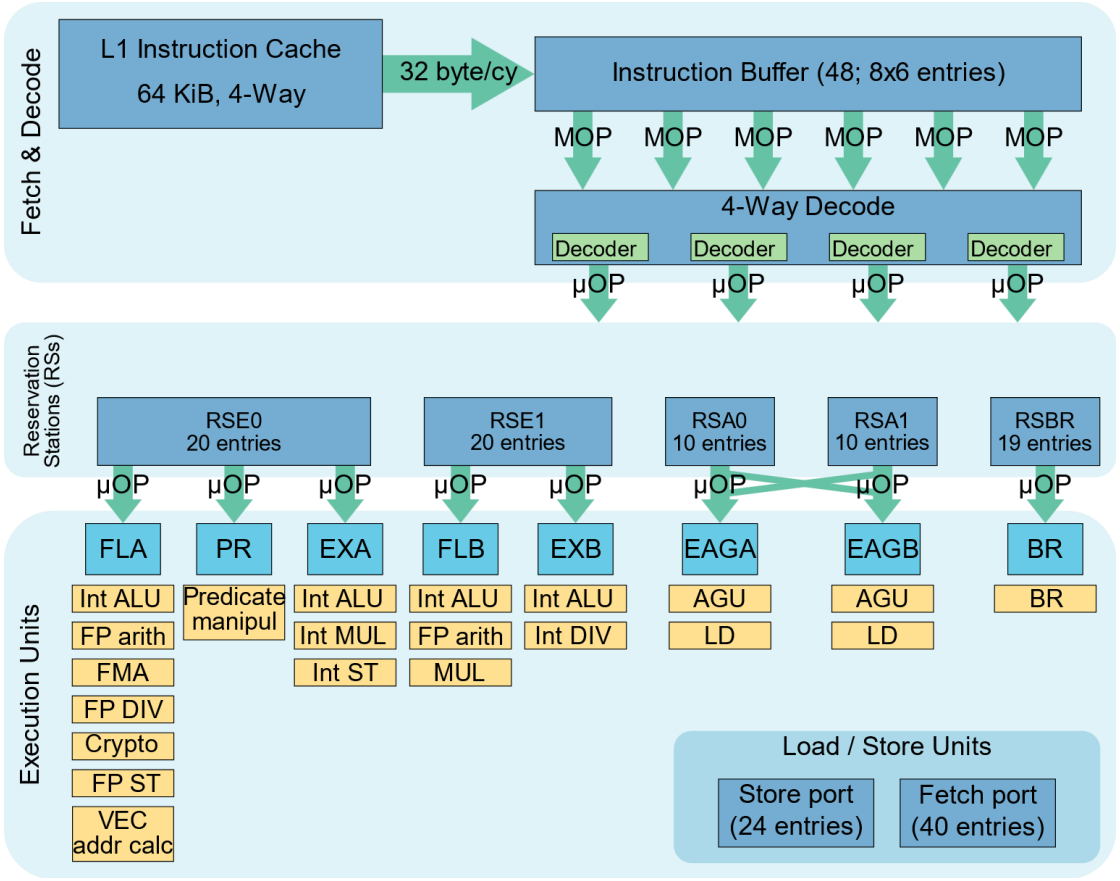


  - Dependencies within loop
  - Intra-loop dependencies



  - TP prediction: 1cy
  - CP prediction: 5 cy
  - LCD prediction: 3 cy

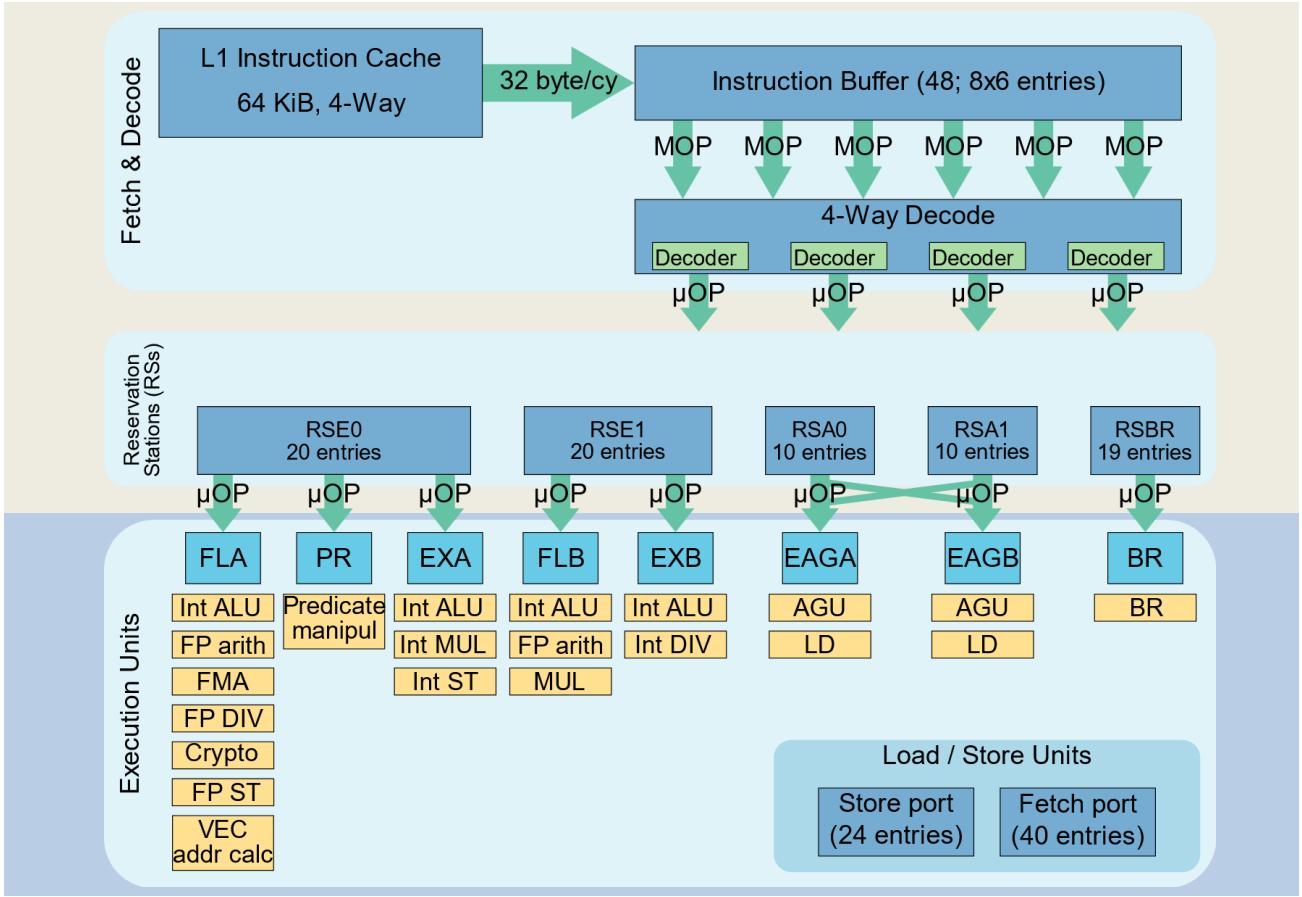
Port model for the A64FX



Port model for the A64FX

Frontend

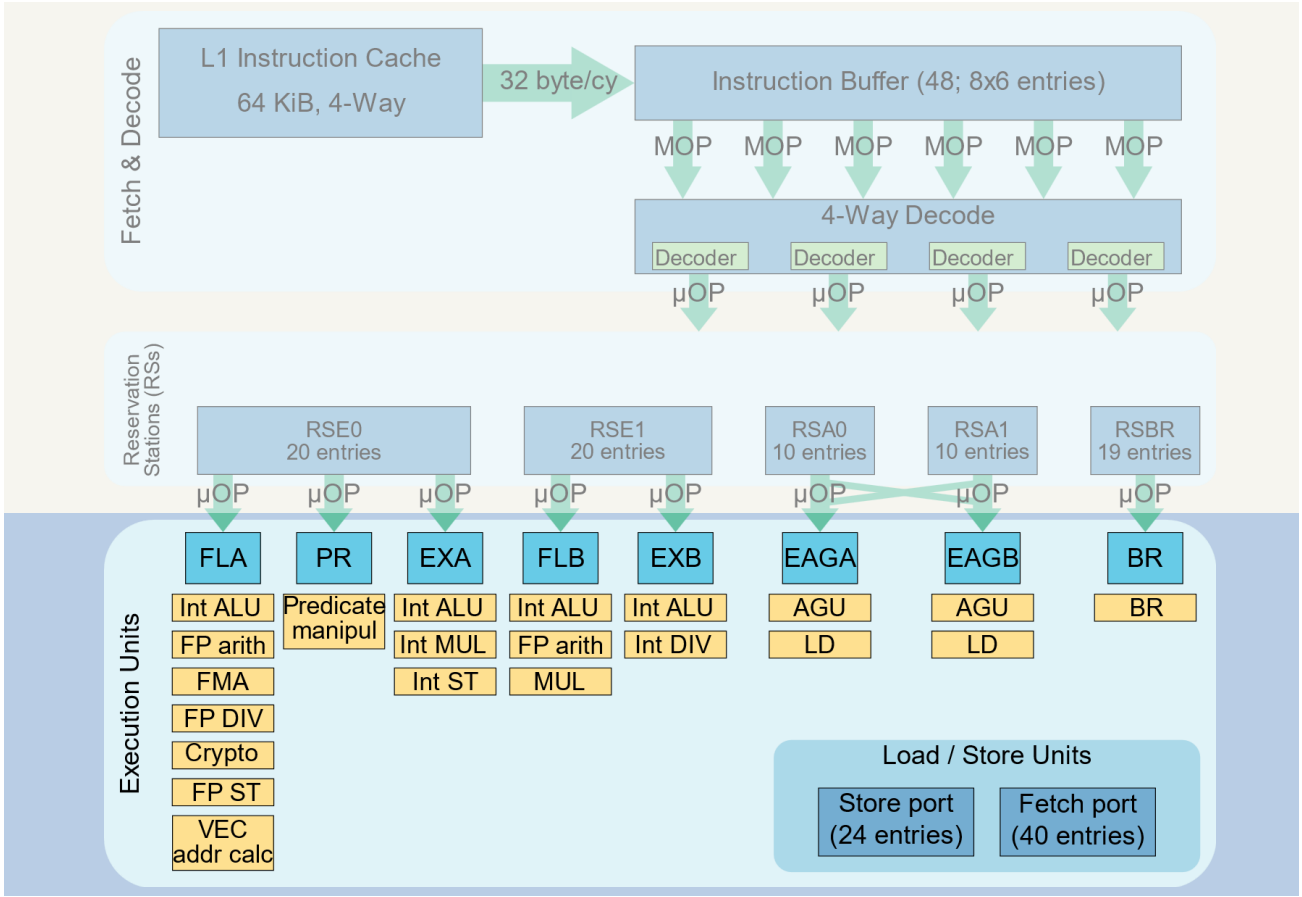
Backend



Port model for the A64FX

Frontend

Backend

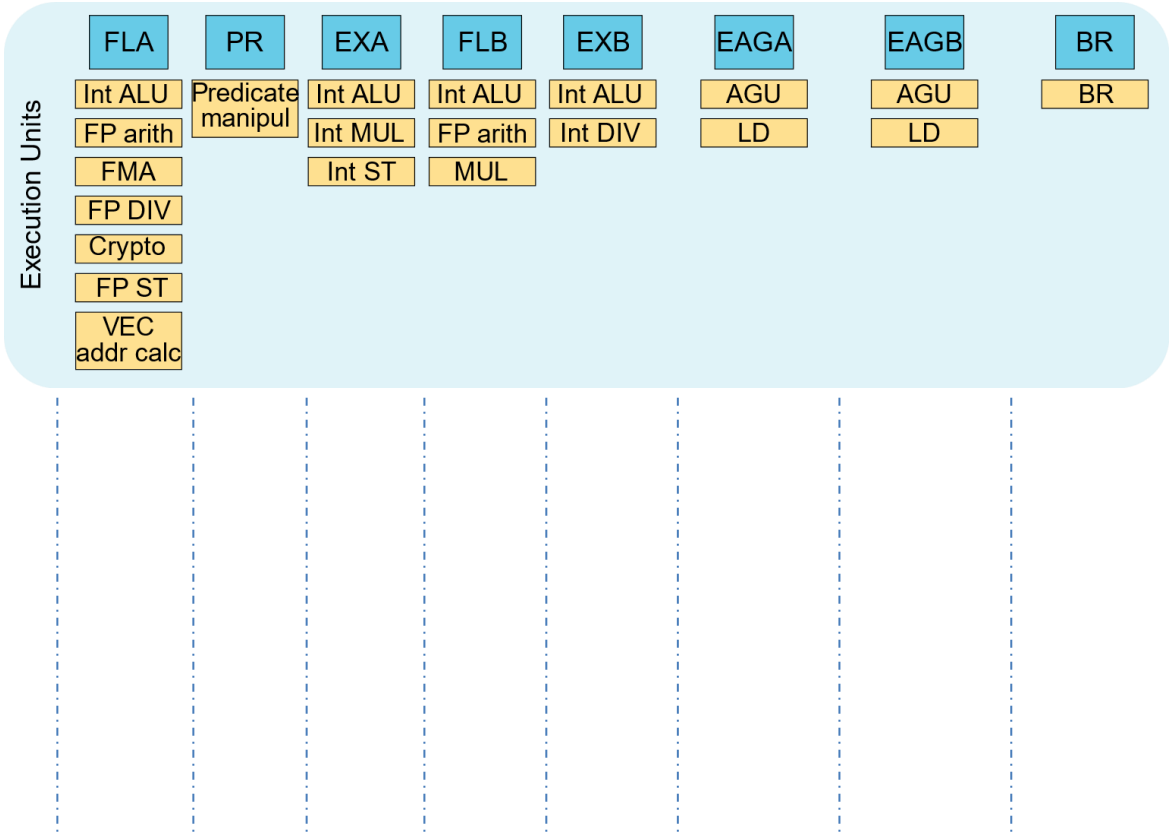


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

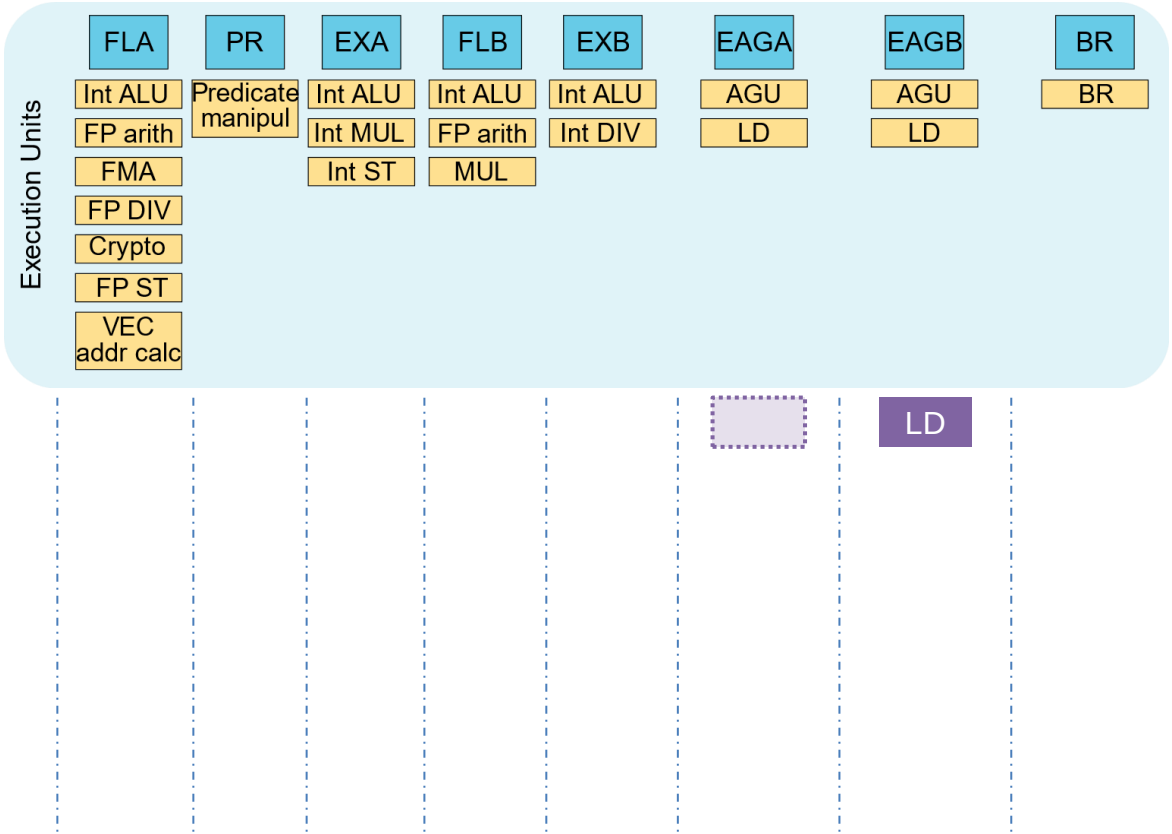


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

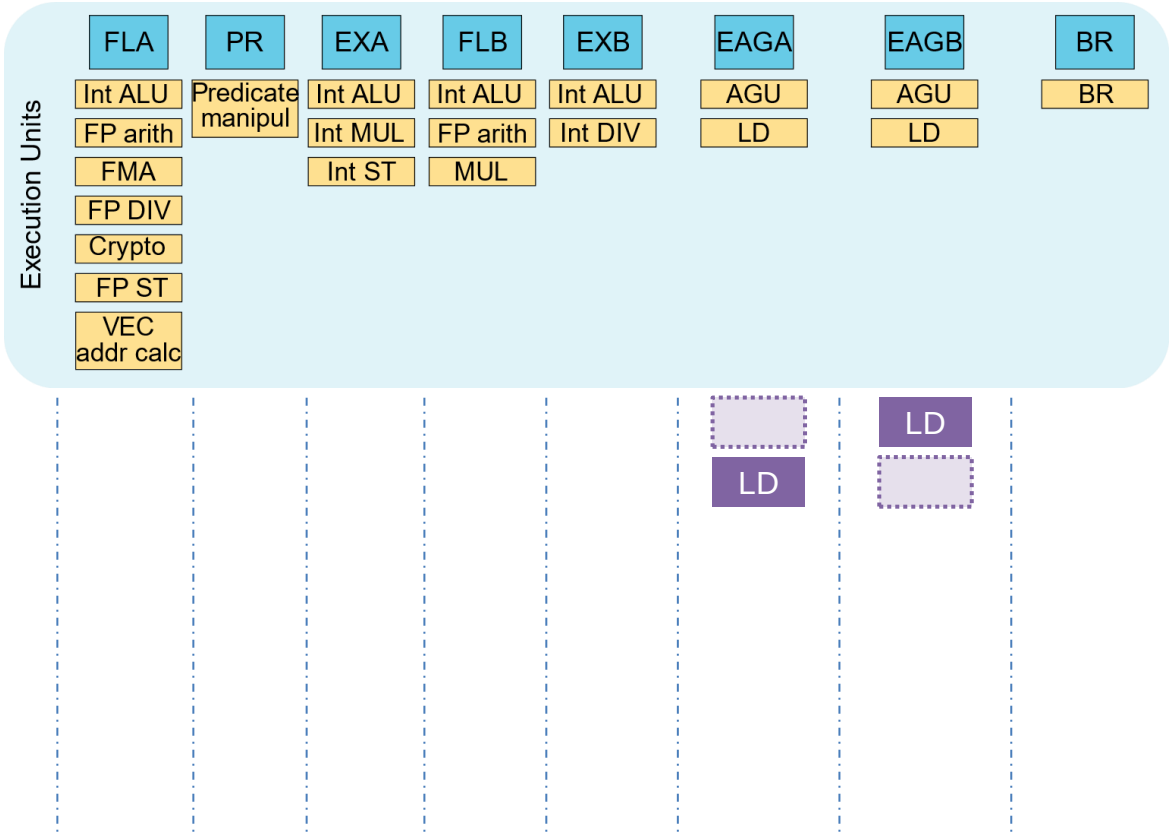


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

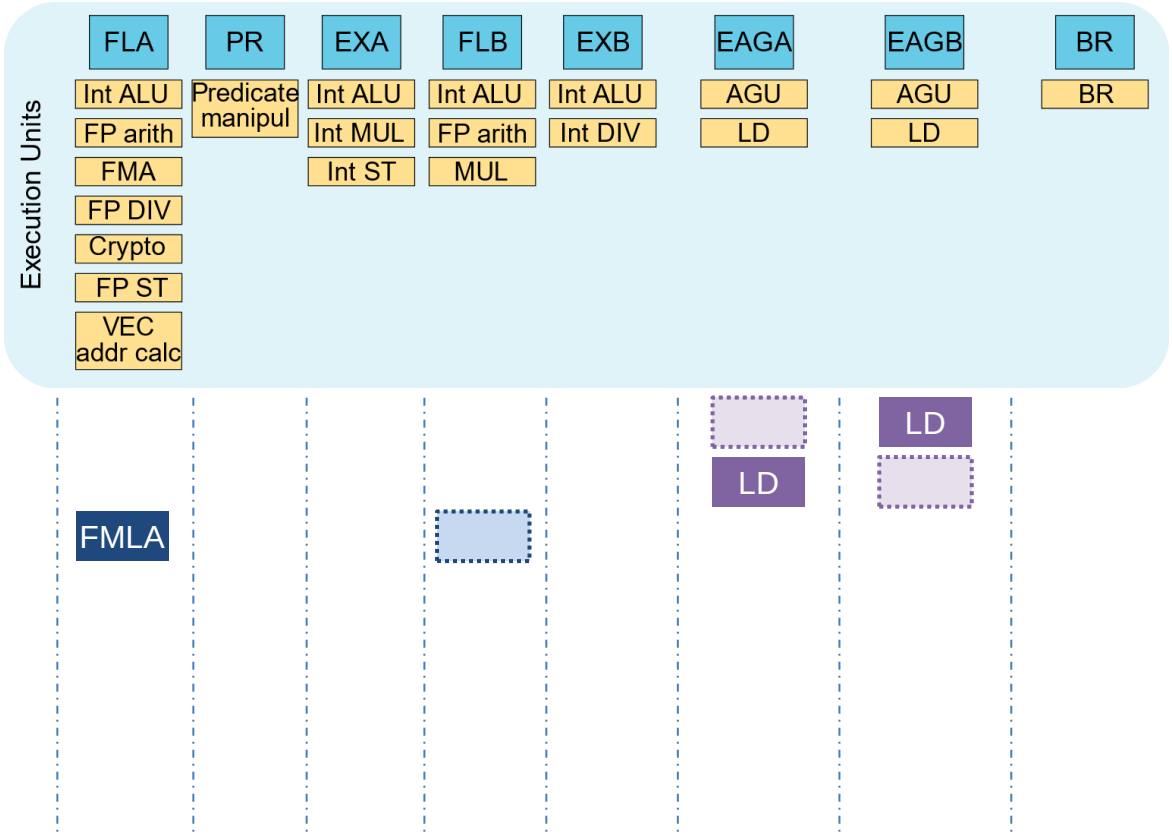


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

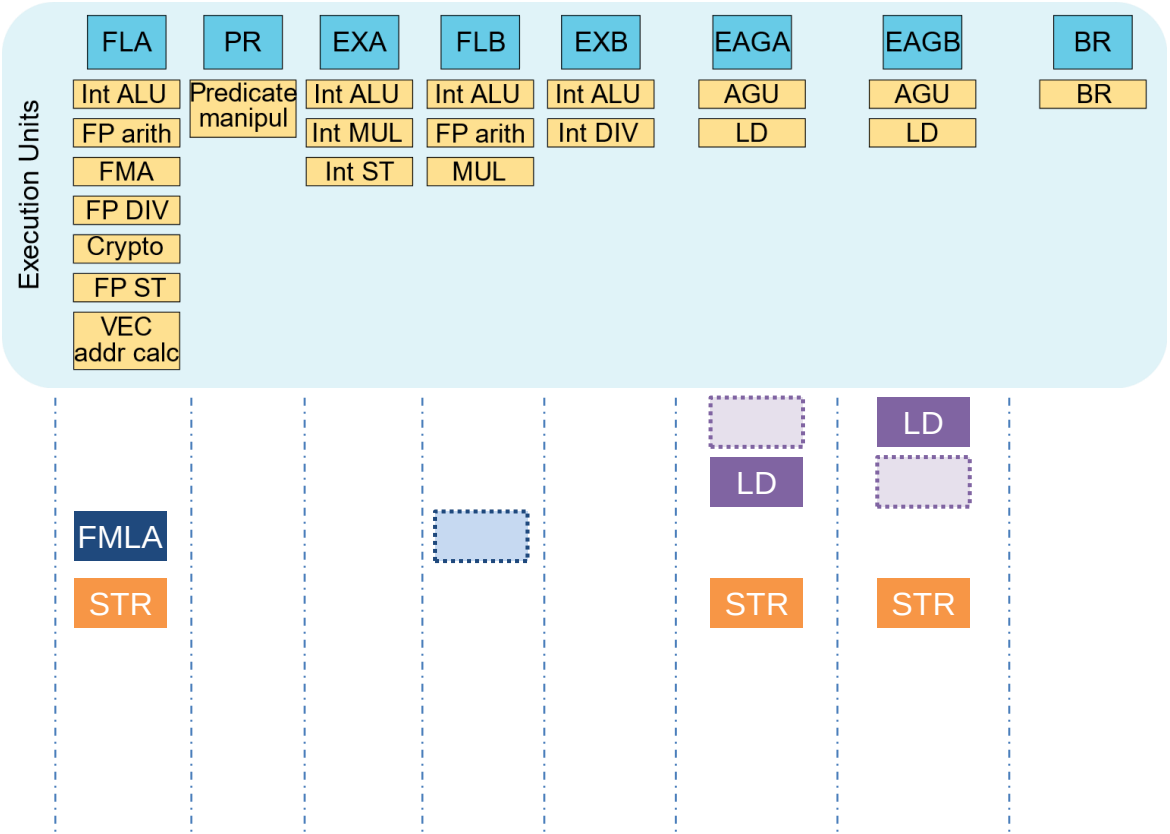


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

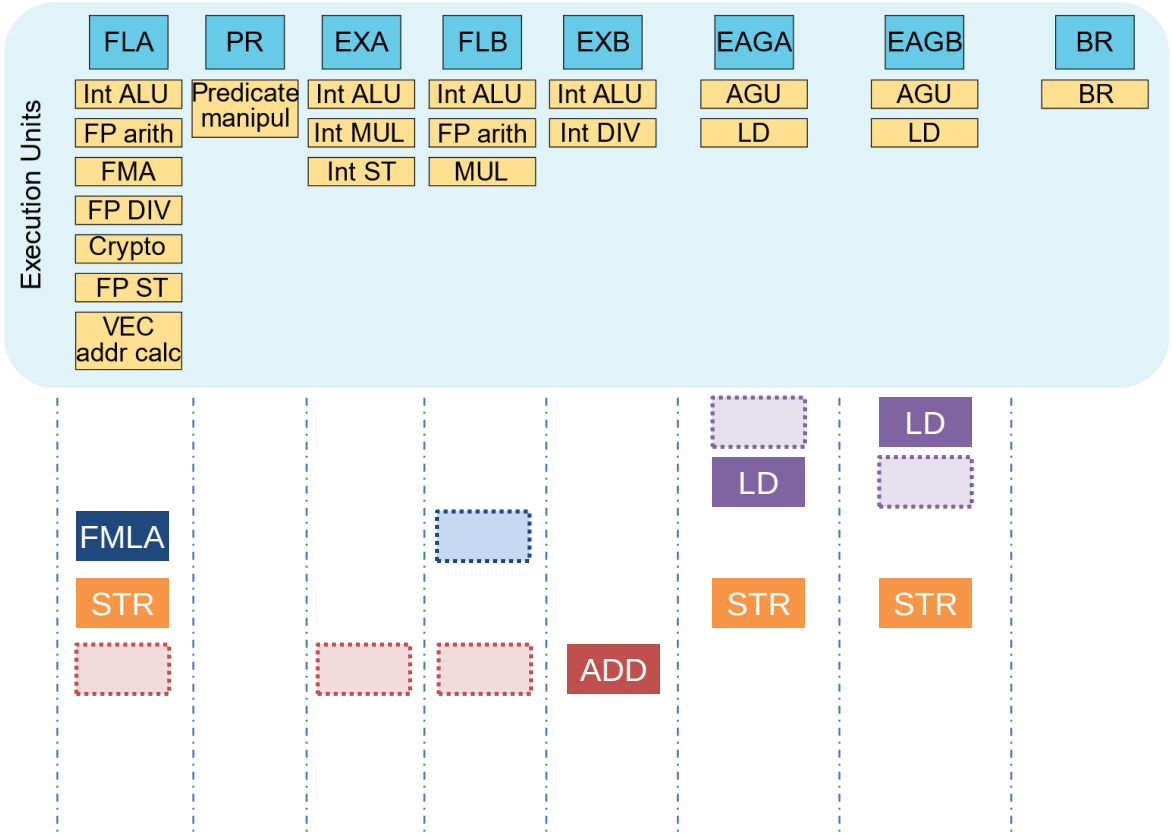


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

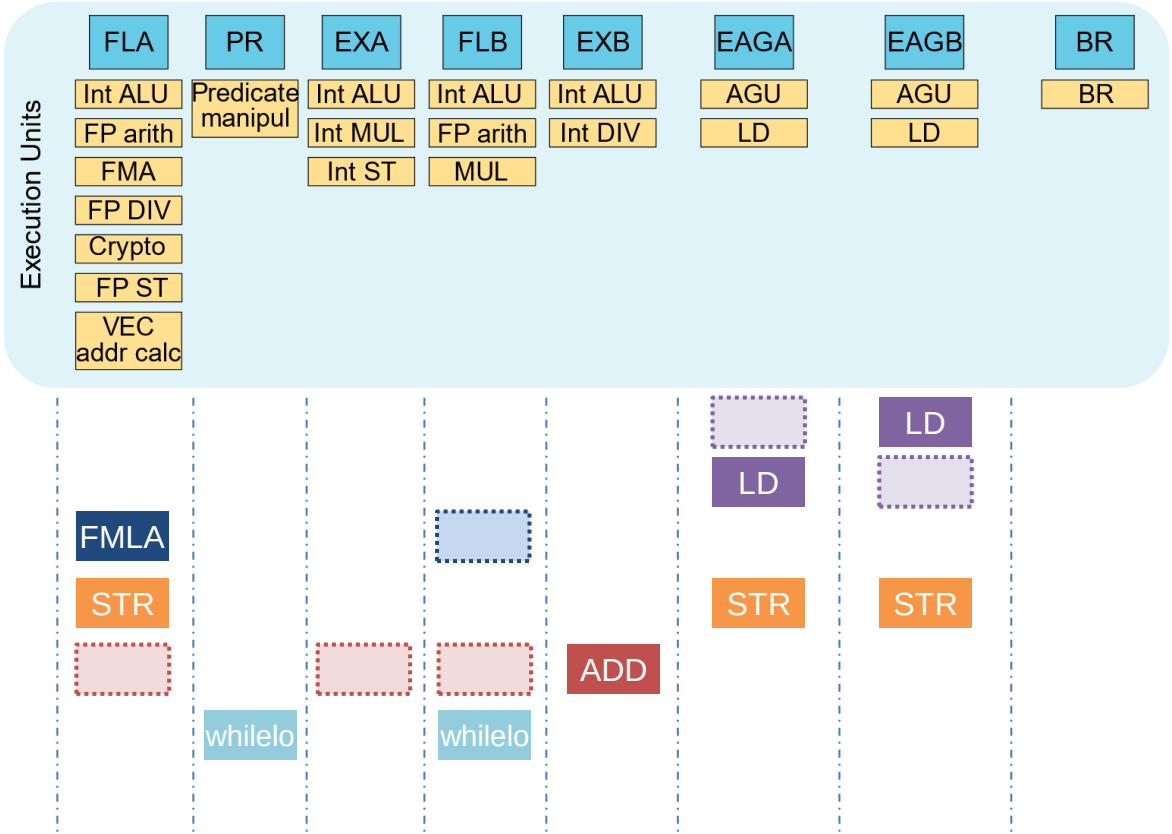


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

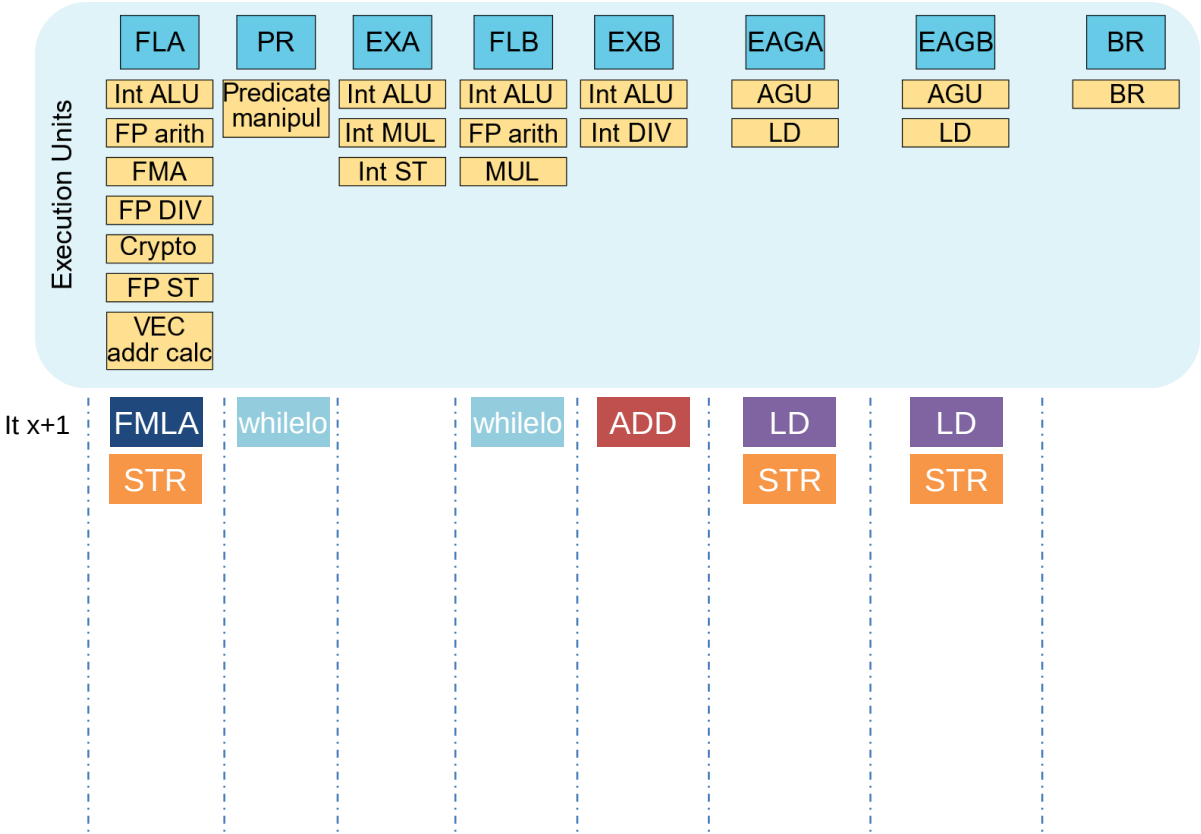


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

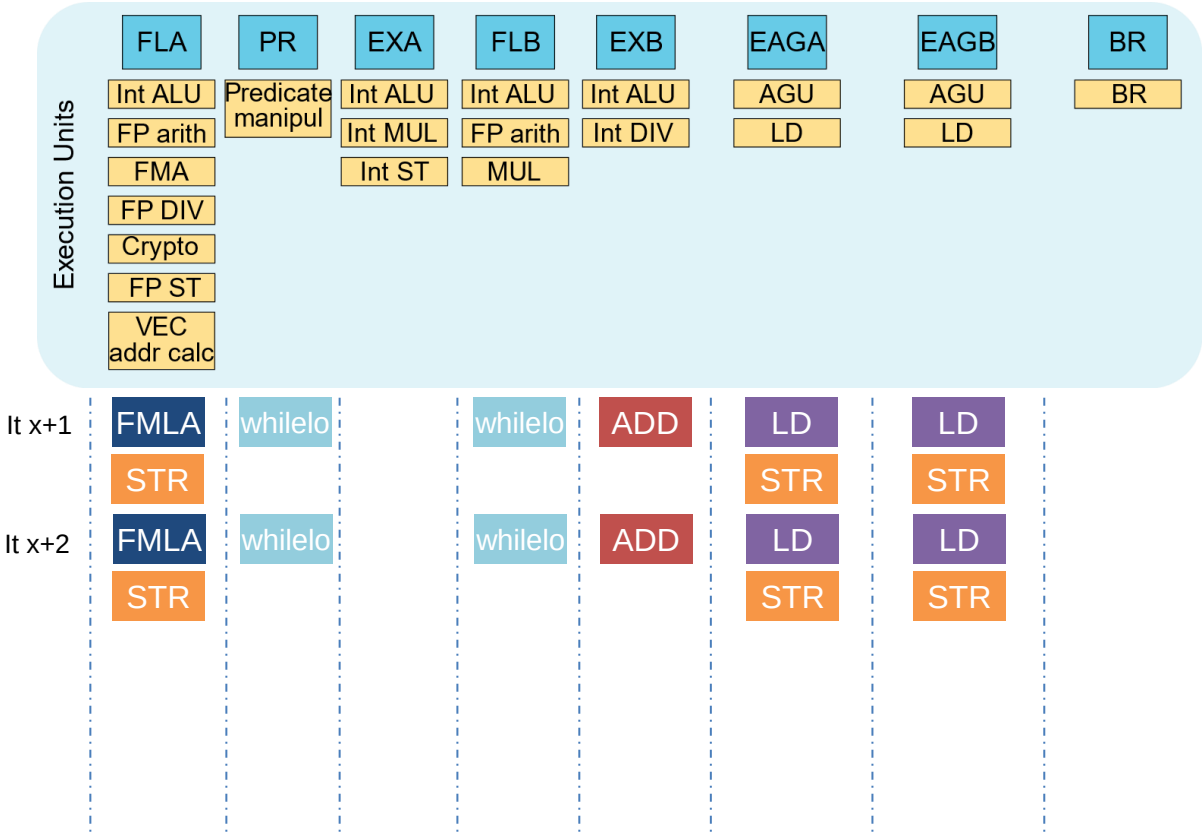


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

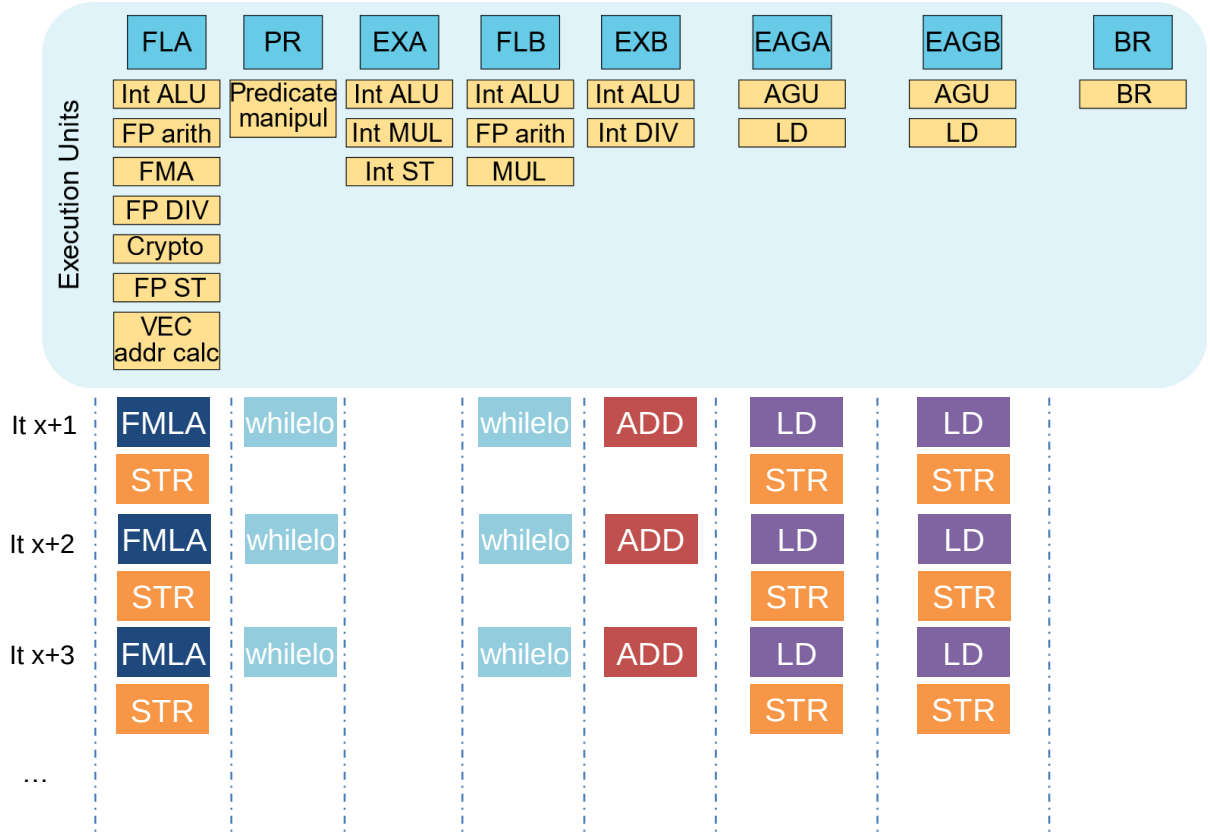


Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```



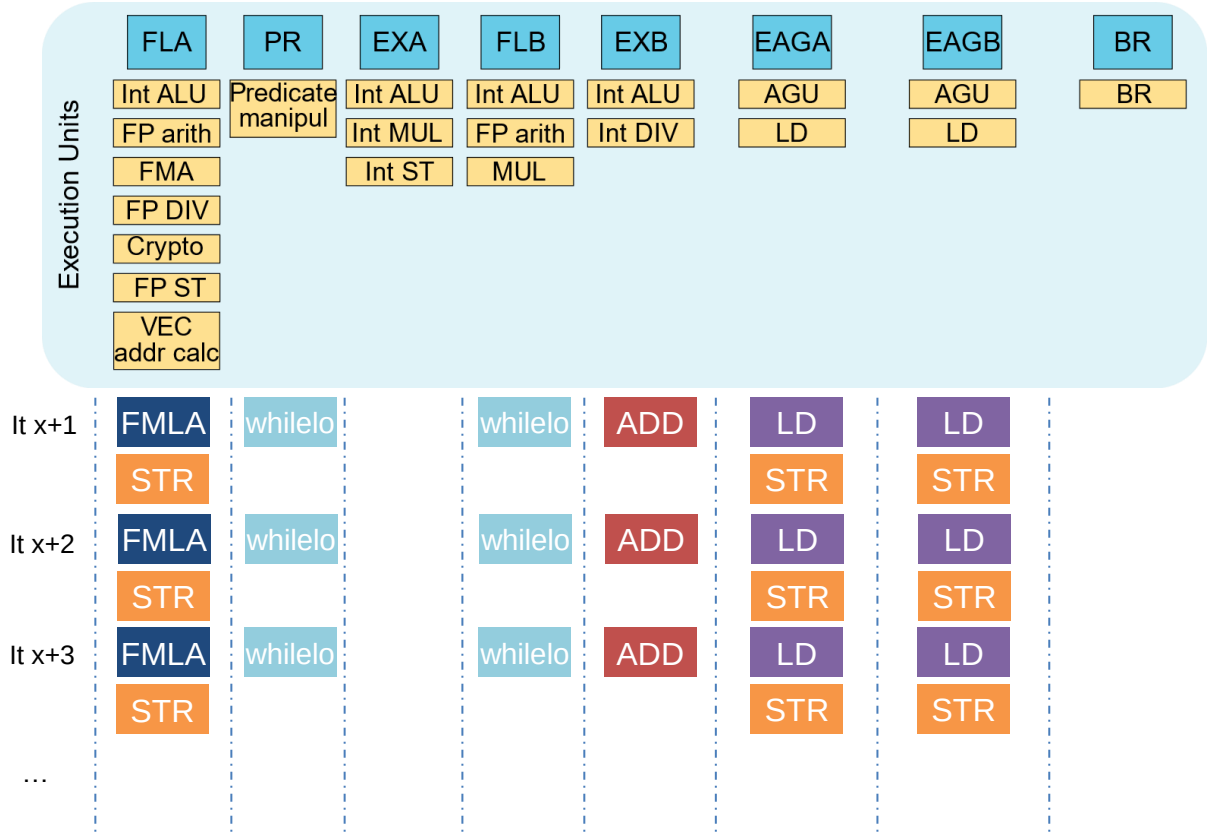
Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
.L18:
  ld1d z4.d, p5/z, [x21, x9, lsl 3]
  ld1d z5.d, p5/z, [x20, x9, lsl 3]
  fmad z5.d, p5/m, z2.d, z4.d
  st1d z5.d, p5, [x19, x9, lsl 3]
  add x8, x9, 8
  whilelo p5.d, w8, w7
  b.any .L18
```

2 cy / VL



Port model for the A64FX

STREAM TRIAD

```
a[i] = b[i] + s * c[i]
```

```
$ osaca --arch a64fx triad.s
```

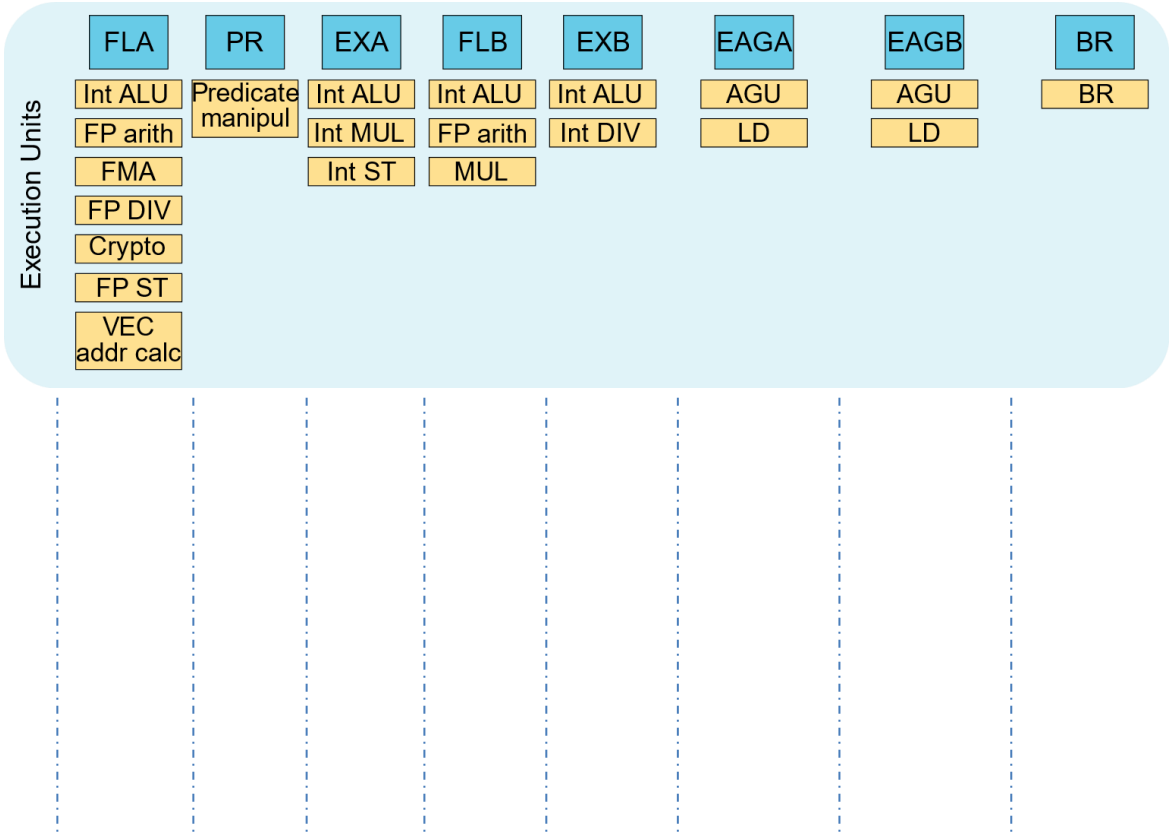
```
Combined Analysis Report
------------------------
                                 Port pressure in cycles
   |  0  - 0DV |  1   |  2   |  3   |  4   |  5  - 5D  |  6  - 6D  |  7   ||  CP | LCD |
-------------------------------------------------------------------------------------------
2  |           |      |      |      |      |           |           |      ||     |     |   .L18:
3  |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      ||     |     |   ld1d    z4.d, p5/z, [x21, x9, lsl 3]
4  |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      ||     |     |   ld1d    z5.d, p5/z, [x20, x9, lsl 3]
6  | 0.12      |      | 0.88 |      |      |           |           |      || 9.0 |     |   fmla    z5.d, p5/m, z1.d, z2.d
5  | 1.00      |      |      |      |      | 1.00      | 1.00      |      ||     |     |   st1d    z5.d, p5, [x19, x9, lsl 3]
5  | 0.00      |      | 0.25 | 0.00 | 0.75 |           |           |      ||     |     |   add     x8, x9, 8
7  |           | 1.00 |      | 1.00 |      |           |           |      ||     |     |   whilelo p5.d, w8, w7
8  |           |      |      |      |      |           |           | 1.00 ||     |     |   b.any   .L18

    1.12        1.0    1.13   1.0    0.75   2.00 1.00   2.00 1.00   1.00    9.0   0.0
```

Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
  ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
  ld1d    z2.d, p0/z, [x20, x0, lsl 3]
  ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
  add     x0, x0, 8
  fmla    z1.d, p0/m, z3.d, z2.d
  whilelo p0.d, x0, x17
  b.any   .L6
  faddv   d4, p1, z1.d
```

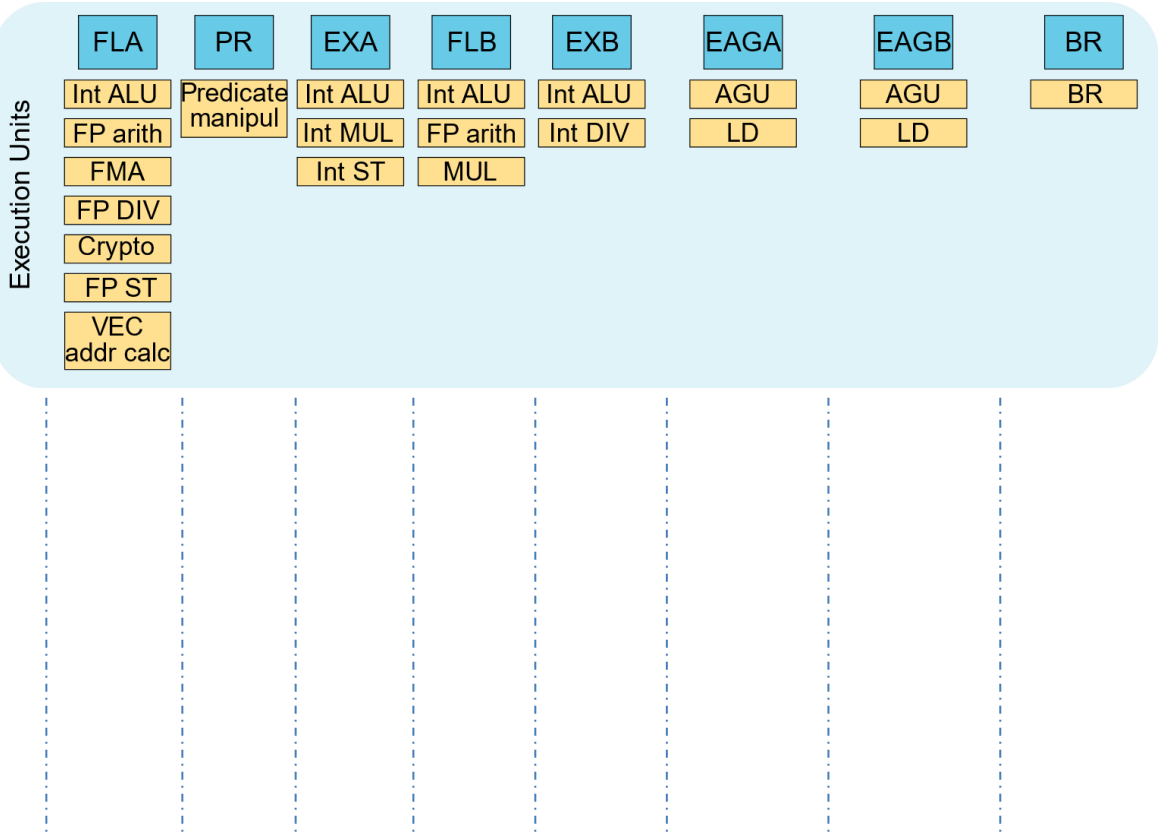


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

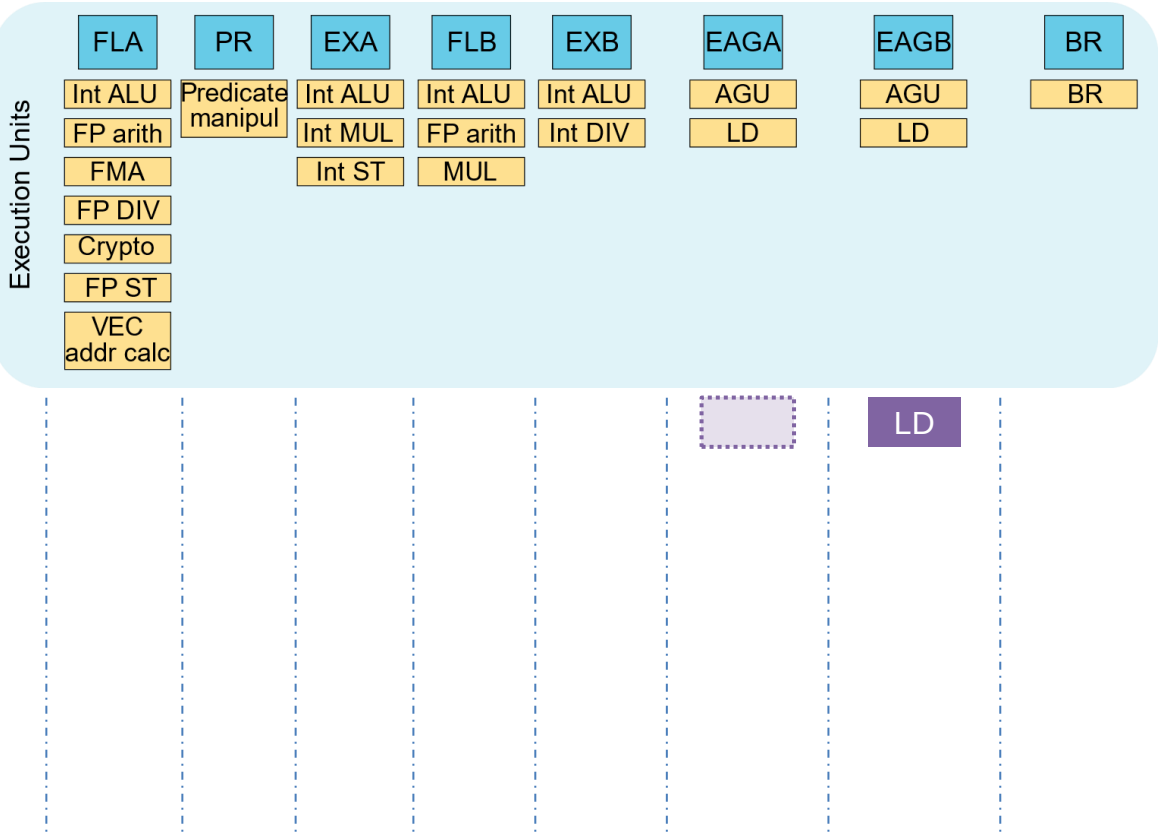


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

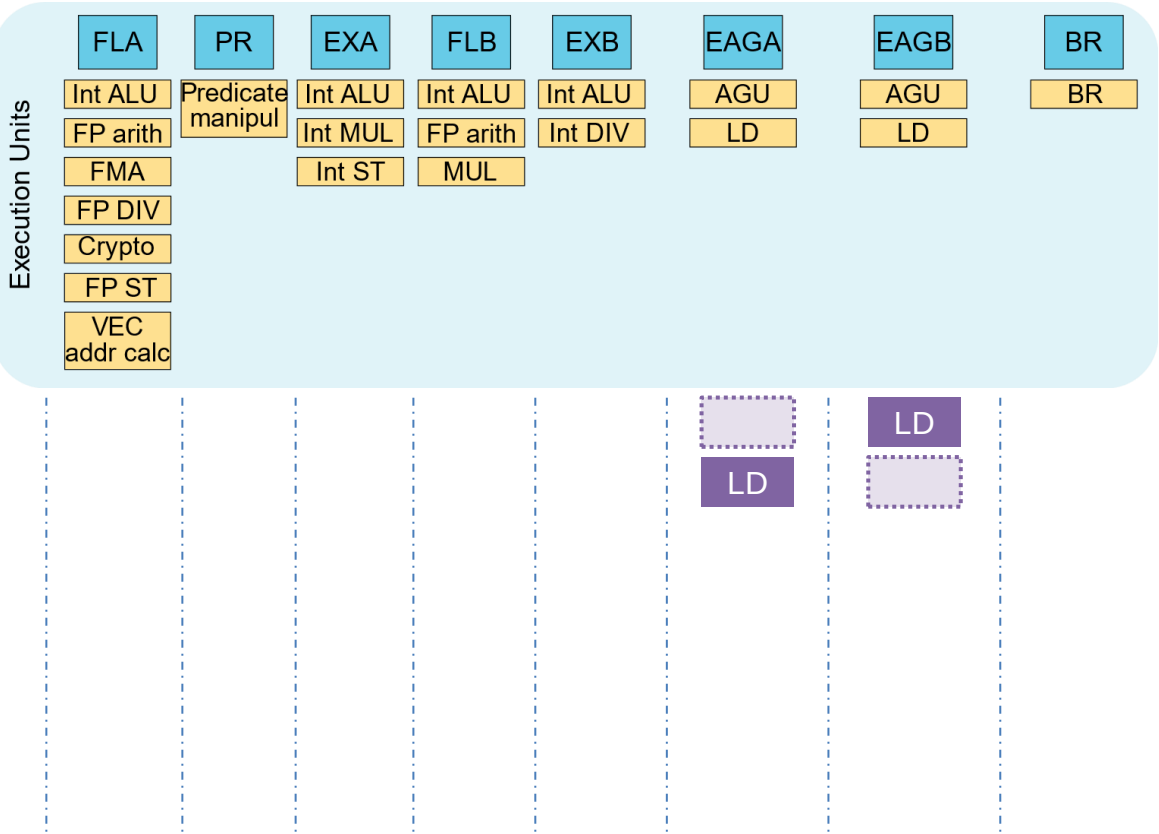


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

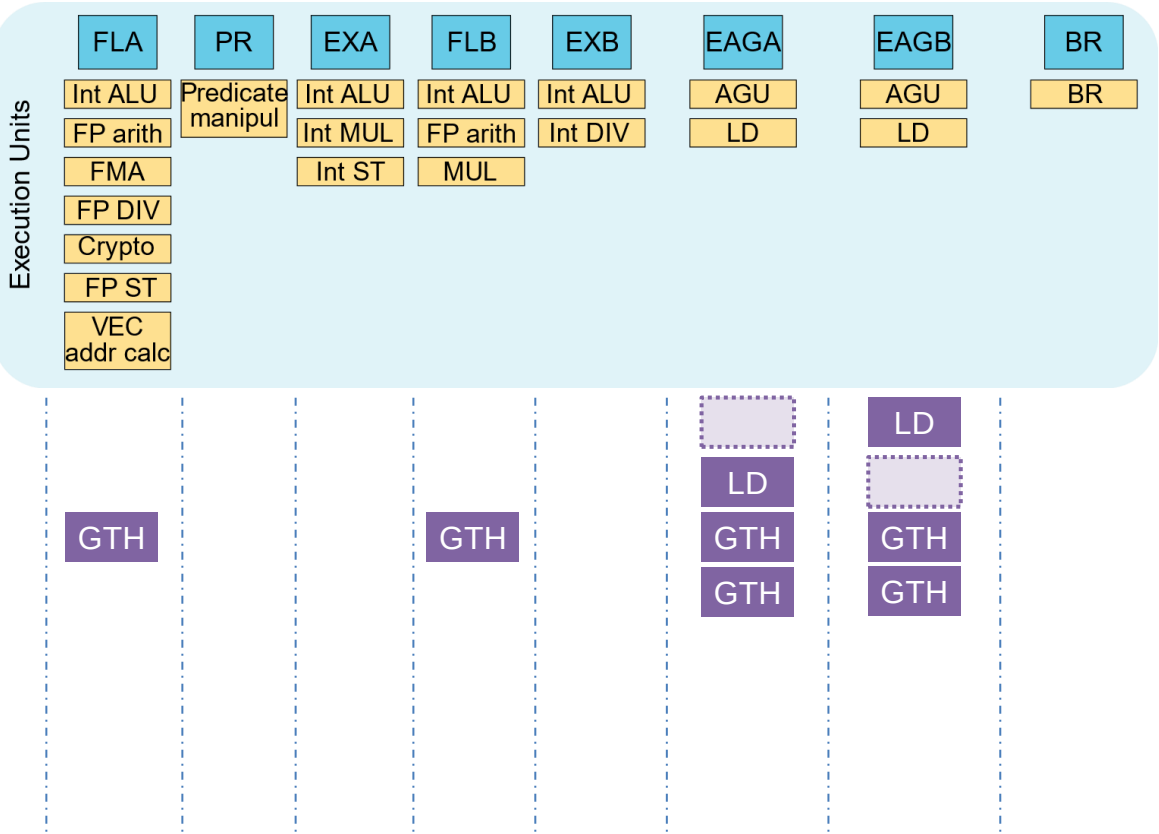


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

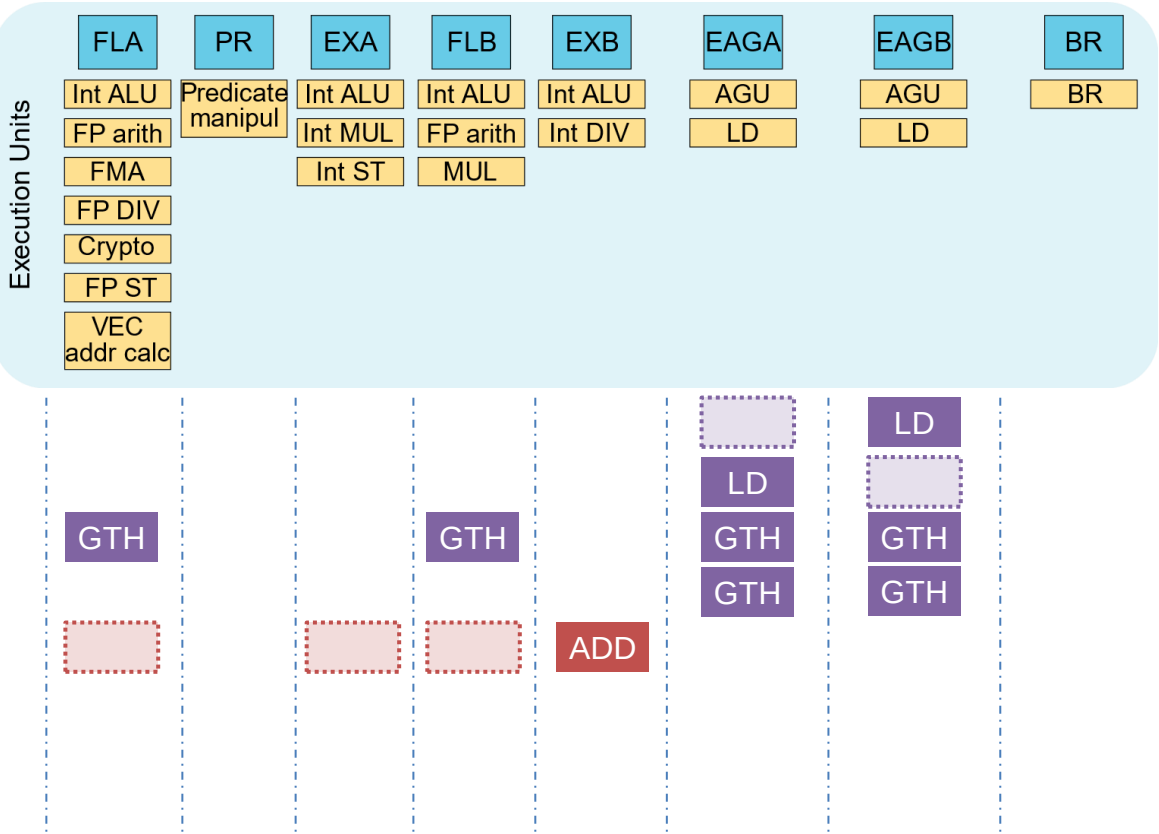


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

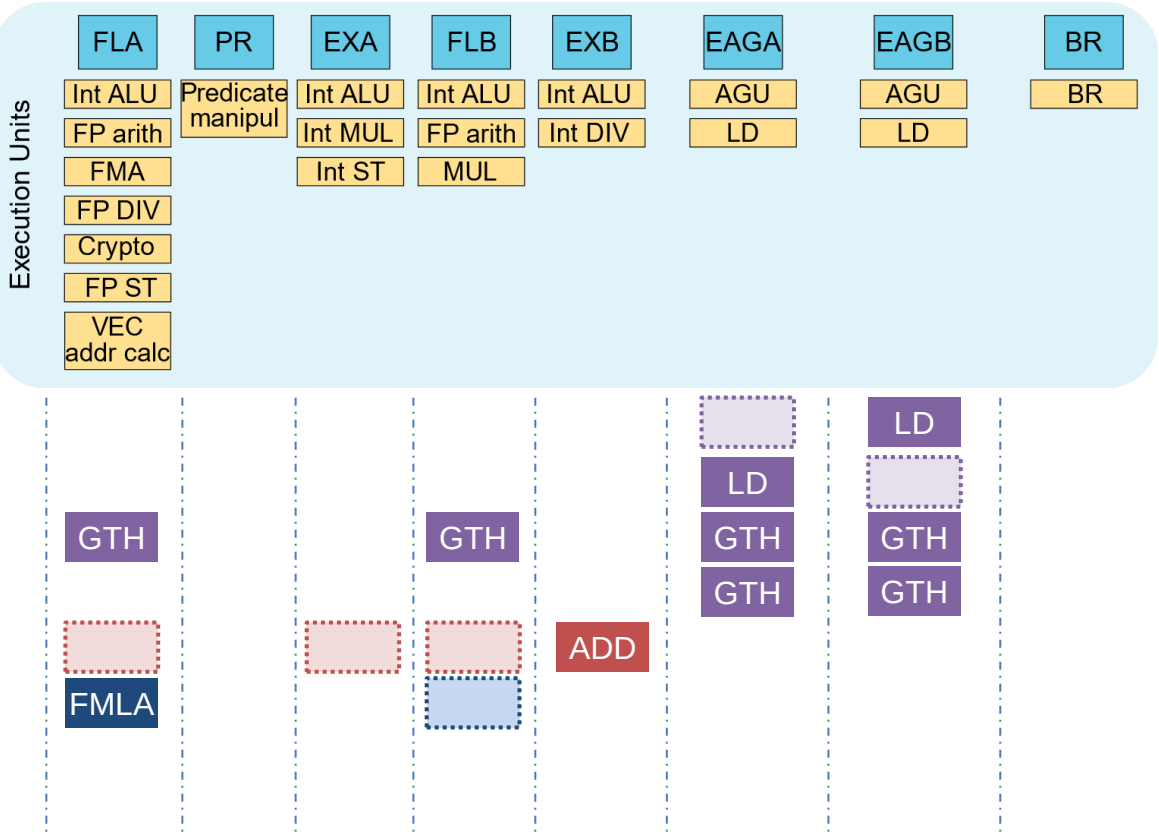


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

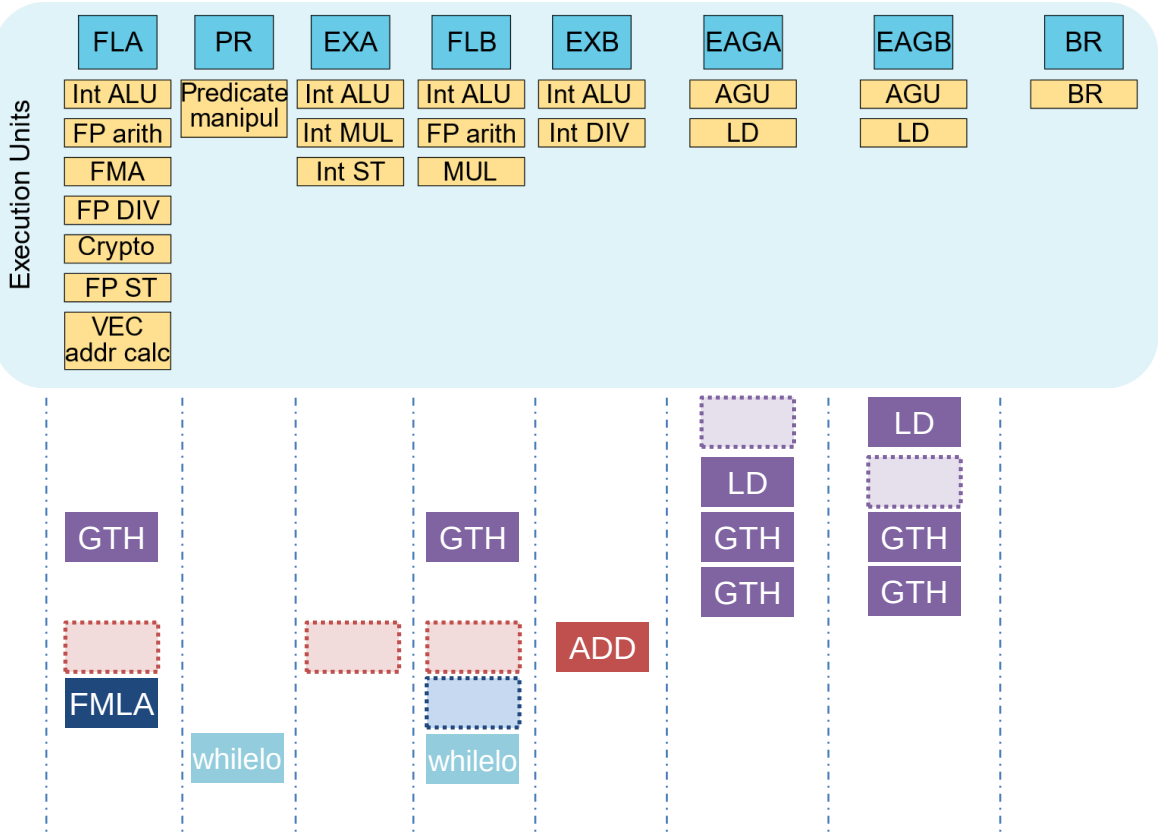


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

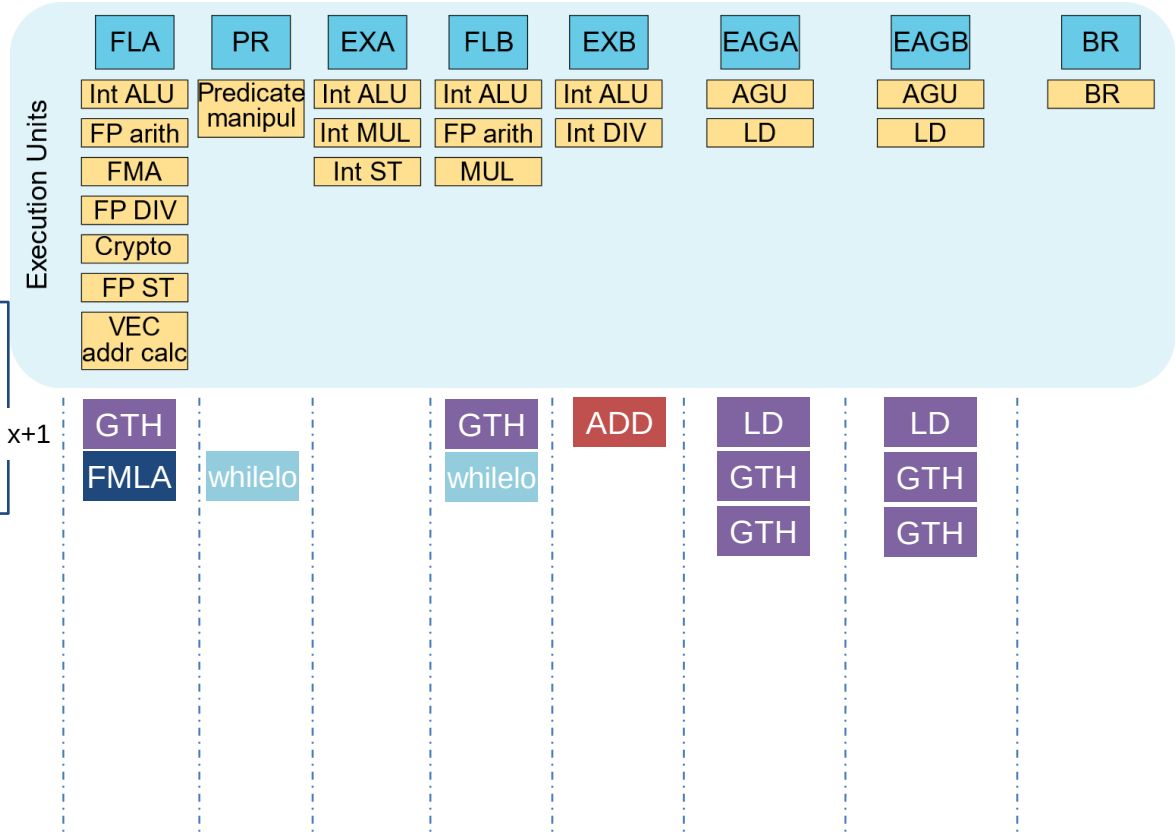


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

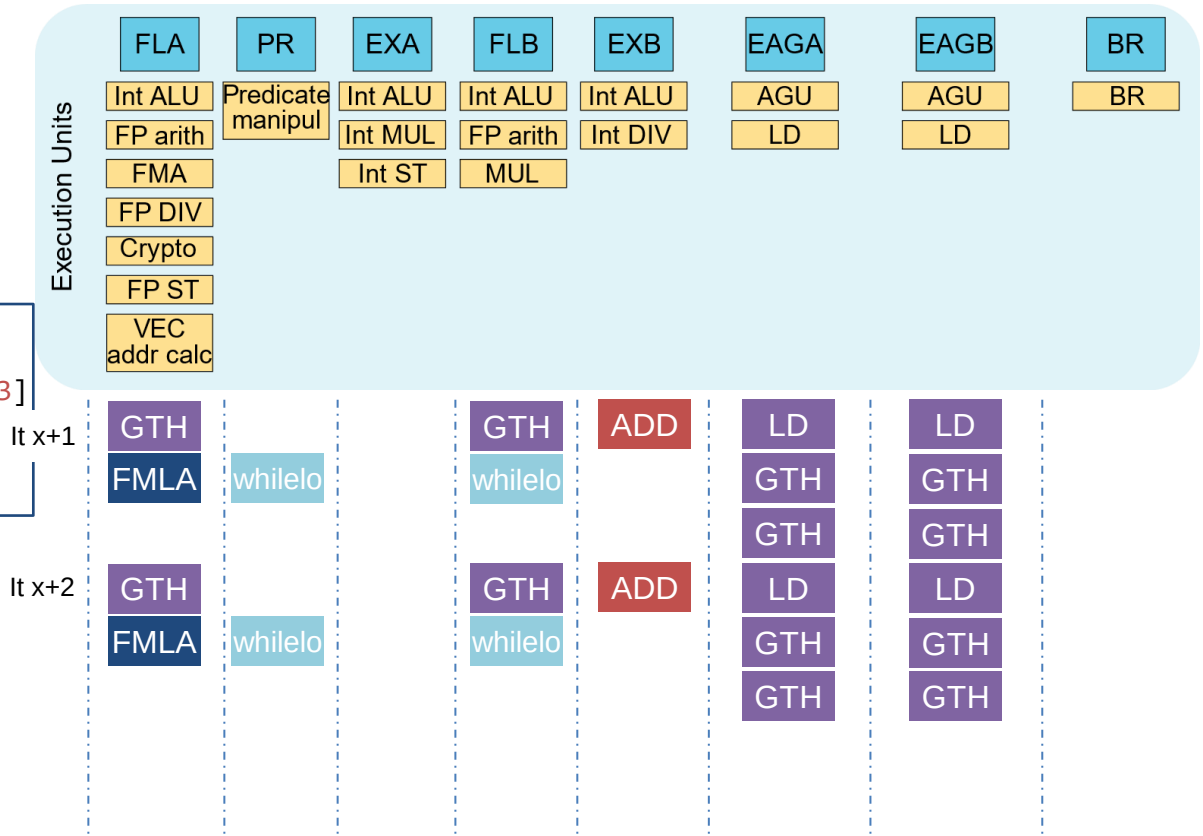


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

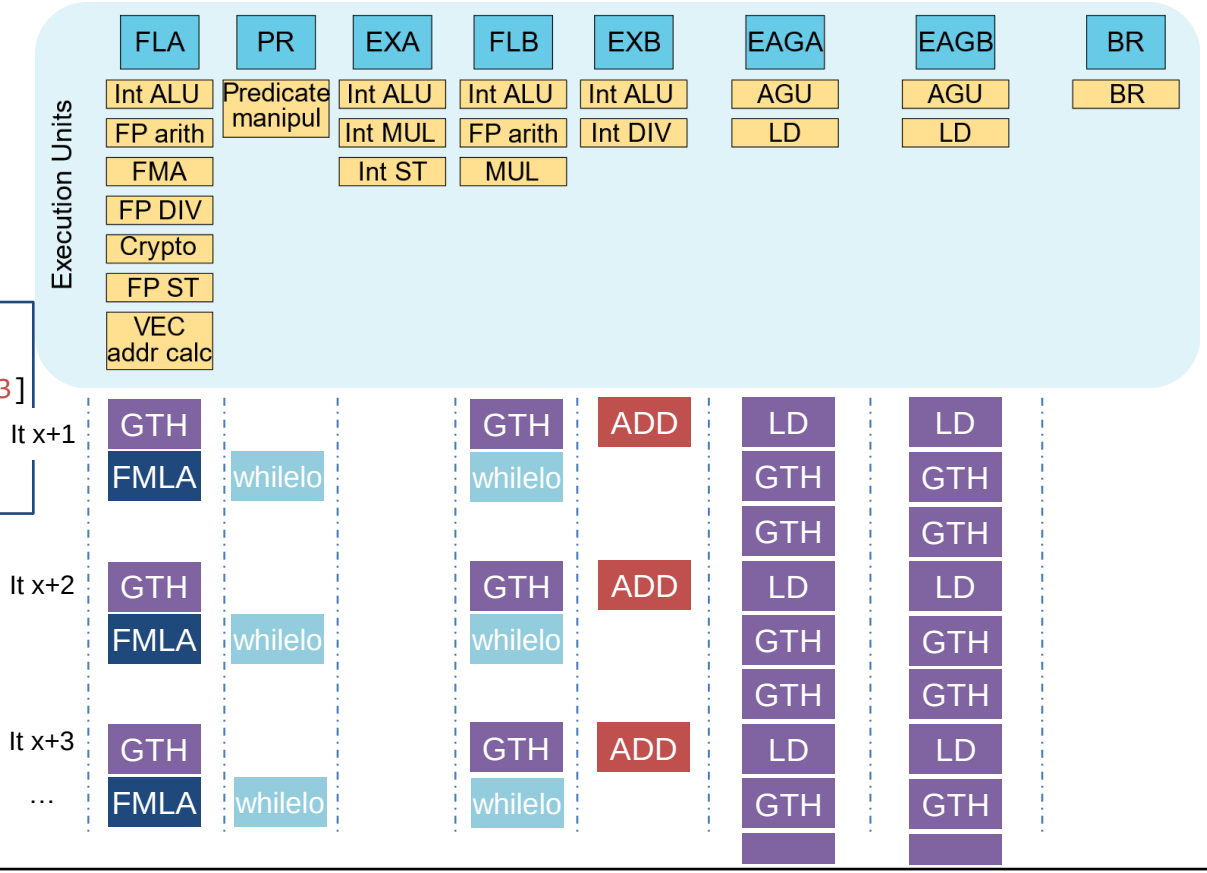


Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```



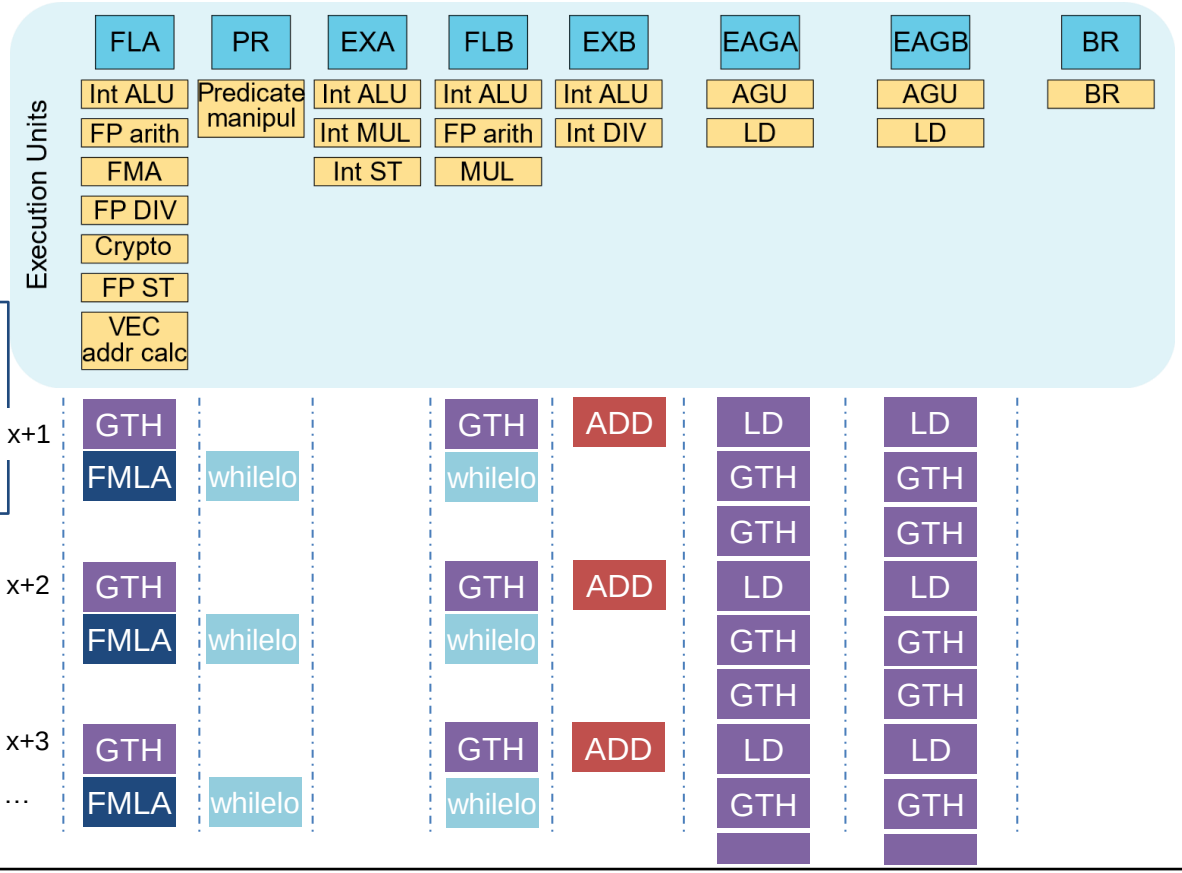
Port model for the A64FX

SPMV

$$y[i] = y[i] + A[i][j] * x[j]$$

```
.L6:
 ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
 ld1d    z2.d, p0/z, [x20, x0, lsl 3]
 ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
 add     x0, x0, 8
 fmla    z1.d, p0/m, z3.d, z2.d
 whilelo p0.d, x0, x17
 b.any   .L6
 faddv   d4, p1, z1.d
```

3 cy / VL



Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

```
$ osaca --arch a64fx spmv.s
```

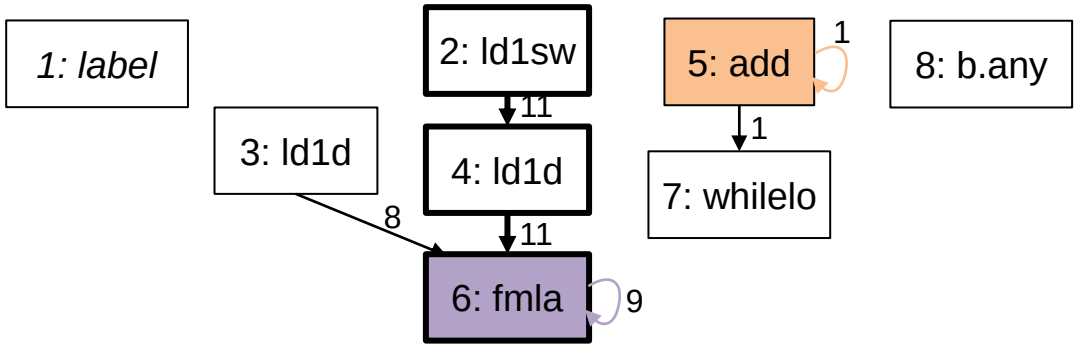
```
Combined Analysis Report
------------------------
                                Port pressure in cycles
  |  0  - 0DV |  1   |  2   |  3   |  4   |  5  - 5D  |  6  - 6D  |  7   ||  CP | LCD |
--------------------------------------------------------------------------------------
1 |           |      |      |      |      |           |           |      ||     |     |  .L6:
2 |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      || 8.0 |     |  ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
3 |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      ||     |     |  ld1d    z2.d, p0/z, [x20, z0, lsl 3]
4 | 1.00      |      |      | 1.00 |      | 2.00 2.00 | 2.00 2.00 |      ||11.0 |     |  ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
5 | 0.00      |      | 0.00 | 0.00 | 1.00 |           |           |      ||     |     |  add     x0, x0, 8
6 | 0.00      |      | 1.00 |      |      |           |           |      || 0.0 | 9.0 |  fmla    z1.d, p0/m, z3.d, z2.d
7 |           | 1.00 |      | 1.00 |      |           |           |      ||     |     |  whilelo p0.d, x0, x17
8 |           |      |      |      |      |           |           | 1.00 ||     |     |  b.any   .L6

   1.00       1.00   1.00   2.00   1.00   3.00 3.00   3.00 3.00   1.00   19.0   9.0
```

Port model for the A64FX

SPMV `y[i] = y[i] + A[i][j] * x[j]`

```
$ osaca --arch a64fx spmv.s --export-graph ./spmv-graph.dot
```

```
Combined Analysis Report
------------------------
                                Port pressure in cycles
   |  0  - 0DV |  1   |  2   |  3   |  4   |  5  - 5D  |  6  - 6D  |  7   ||  CP  | LCD |
-----------------------------------------------------------------------------------------
1  |           |      |      |      |      |           |           |      ||      |     |  .L6:
2  |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      ||  8.0 |     |  ld1sw   z0.d, p0/z, [x6, x0, lsl 2]
3  |           |      |      |      |      | 0.50 0.50 | 0.50 0.50 |      ||      |     |  ld1d    z2.d, p0/z, [x20, z0, lsl 3]
4  | 1.00      |      |      | 1.00 |      | 2.00 2.00 | 2.00 2.00 |      ||11.0  |     |  ld1d    z3.d, p0/z, [x18, z0.d, lsl 3]
5  | 0.00      |      | 0.00 | 0.00 | 1.00 |           |           |      ||      |     |  add     x0, x0, 8
6  | 0.00      |      | 1.00 |      |      |           |           |      ||  0.0 | 9.0 |  fmla    z1.d, p0/m, z3.d, z2.d
7  |           | 1.00 |      | 1.00 |      |           |           |      ||      |     |  whilelo p0.d, x0, x17
8  |           |      |      |      |      |           |           | 1.00 ||      |     |  b.any   .L6

   1.00        1.00   1.00   2.00   1.00   3.00 3.00   3.00 3.00   1.00   19.0   9.0
```



Port model for the A64FX

SPMV

```
y[i] = y[i] + A[i][j] * x[j]
```

- Example from C code: https://godbolt.org/z/8cb8aWaWr
- Example from ASM snippet: https://godbolt.org/z/Kar3o9jjo

Marker insertion

Byte marker

x86

```
movl    $111,%ebx    #START MARKER
.byte   100,103,144  #START MARKER
.L22:
  vmovapd 0(%r13,%rax),%ymm0
  vfmadd213pd (%r14,%rax),%ymm1,%ymm0
  vmovapd %ymm0,(%r12,%rax)
  addq $32,%rax
  cmpq %rax,%r15
  jne .L22
movl    $222,%ebx    #END MARKER
.byte   100,103,144  #END MARKER
```

arm

```
 mov    x1,#111     //START
 .byte  213,3,32,31 //START
 .L18:
   ldr q2, [x20, x0]
   ldr q1, [x21, x0]
   fmla v1.2d, v2.2d, v0.2d
   str q1, [x19, x0]
   add x0, x0, #16
   cmp x22, x0
   bne .L18
 mov    x1,#222     //END
 .byte  213,3,32,31 //END
```

Marker insertion

Byte marker

x86

```
movl    $111,%ebx    #START MARKER
.byte   100,103,144  #START MARKER
.L22:
  vmovapd 0(%r13,%rax),%ymm0
  vfmadd213pd (%r14,%rax),%ymm1,%ymm0
  vmovapd %ymm0,(%r12,%rax)
  addq $32,%rax
  cmpq %rax,%r15
  jne .L22
movl    $222,%ebx    #END MARKER
.byte   100,103,144  #END MARKER
```

arm

```
 mov    x1,#111      //START
 .byte  213,3,32,31 //START
 .L18:
   ldr q2, [x20, x0]
   ldr q1, [x21, x0]
   fmla v1.2d, v2.2d, v0.2d
   str q1, [x19, x0]
   add x0, x0, #16
   cmp x22, x0
   bne .L18
 mov    x1,#222      //END
 .byte  213,3,32,31  //END
```

Comment marker

x86

```
# OSACA-BEGIN
.L22:
  vmovapd 0(%r13,%rax),%ymm0
  vfmadd213pd (%r14,%rax),%ymm1,%ymm0
  vmovapd %ymm0,(%r12,%rax)
  addq $32,%rax
  cmpq %rax,%r15
  jne .L22
# OSACA-END
```

arm

```
 // OSACA-BEGIN
 .L18:
   ldr q2, [x20, x0]
   ldr q1, [x21, x0]
   fmla v1.2d, v2.2d, v0.2d
   str q1, [x19, x0]
   add x0, x0, #16
   cmp x22, x0
   bne .L18
 // OSACA-END
```

Marker insertion

Byte marker

x86

```
movl    $111,%ebx    #START MARKER
.byte   100,103,144  #START MARKER
.L22:
  vmovapd 0(%r13,%rax),%ymm0
  vfmadd213pd (%r14,%rax),%ymm1,%ymm0
  vmovapd %ymm0,(%r12,%rax)
  addq $32,%rax
  cmpq %rax,%r15
  jne .L22
movl    $222,%ebx    #END MARKER
.byte   100,103,144  #END MARKER
```

arm

```
 mov    x1,#111     //START
 .byte  213,3,32,31 //START
 .L18:
   ldr q2, [x20, x0]
   ldr q1, [x21, x0]
   fmla v1.2d, v2.2d, v0.2d
   str q1, [x19, x0]
   add x0, x0, #16
   cmp x22, x0
   bne .L18
 mov    x1,#222     //END
 .byte  213,3,32,31 //END
```

Comment marker

x86

```
# OSACA-BEGIN
.L22:
  vmovapd 0(%r13,%rax),%ymm0
  vfmadd213pd (%r14,%rax),%ymm1,%ymm0
  vmovapd %ymm0,(%r12,%rax)
  addq $32,%rax
  cmpq %rax,%r15
  jne .L22
# OSACA-END
```

arm

```
 // OSACA-BEGIN
 .L18:
   ldr q2, [x20, x0]
   ldr q1, [x21, x0]
   fmla v1.2d, v2.2d, v0.2d
   str q1, [x19, x0]
   add x0, x0, #16
   cmp x22, x0
   bne .L18
 // OSACA-END
```

Insertion tool

```
$ osaca --arch ARCH --insert-marker

Blocks found in assembly file:
.L_A
                ...
.L_B
                ...
----------------------------
Possible blocks to be marked:
.L_A
.L_B
Choose block to be marked [.L_B]: _
```

Accuracy and related tools

	OSACA	LLVM-MCA	IACA	Ithemal
Throughput	✓	✓	✓	✓
Critical Path	✓	✓	✗	✗
Loop-Carried Dependencies	✓	✓	✓	✗
Supported ISAs	x86, arm64	Many	x86	x86
Supported µArchs	**Intel**: SNB, IVB, HSW, BDW, SKX, CSX, ICX; **AMD**: Zen1, Zen2; **Arm**: Neoverse N1, ThunderX2, A64FX	Many	**Intel**: SNB, IVB, HSW, BDW, SKX	**Intel**: IVB, HSW, SKL
Speed	 - 			
Open Source	✓	✓	✗	✓
Future Development	✓	✓	✗	

Getting started

- Via PyPI:

```
$ pip install osaca
```

- From source:

```
$ git clone https://github.com/RRZE-HPC/OSACA.git
```

- Webbased with Compiler Explorer: https://godbolt.org/
  a) Choose “Analysis” as language
  b) Add OSACA as tool to your compiler

- OOKAMI module:

```
$ module load archiconda/3
$ source activate OSACA
```

Getting started

- Via PyPI:

```
$ pip install osaca
```

- From source:

```
$ git clone https://github.com/RRZE-HPC/OSACA.git
```

For questions, issues, and feature requests

- Webbased with Compiler Explorer: https://godbolt.org/
  a) Choose “Analysis” as language
  b) Add OSACA as tool to your compiler

- OOKAMI module:

```
$ module load archiconda/3
$ source activate OSACA
```

NHR FAU

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Thank you

Questions