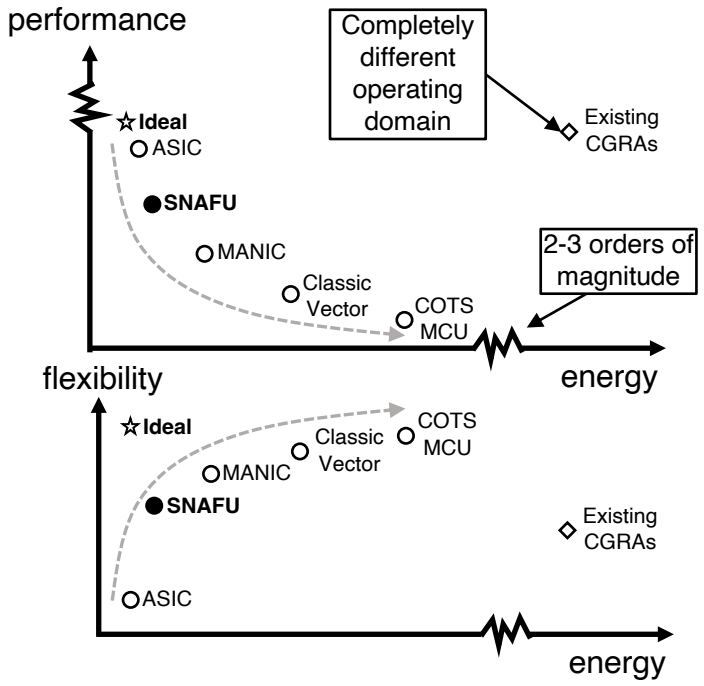
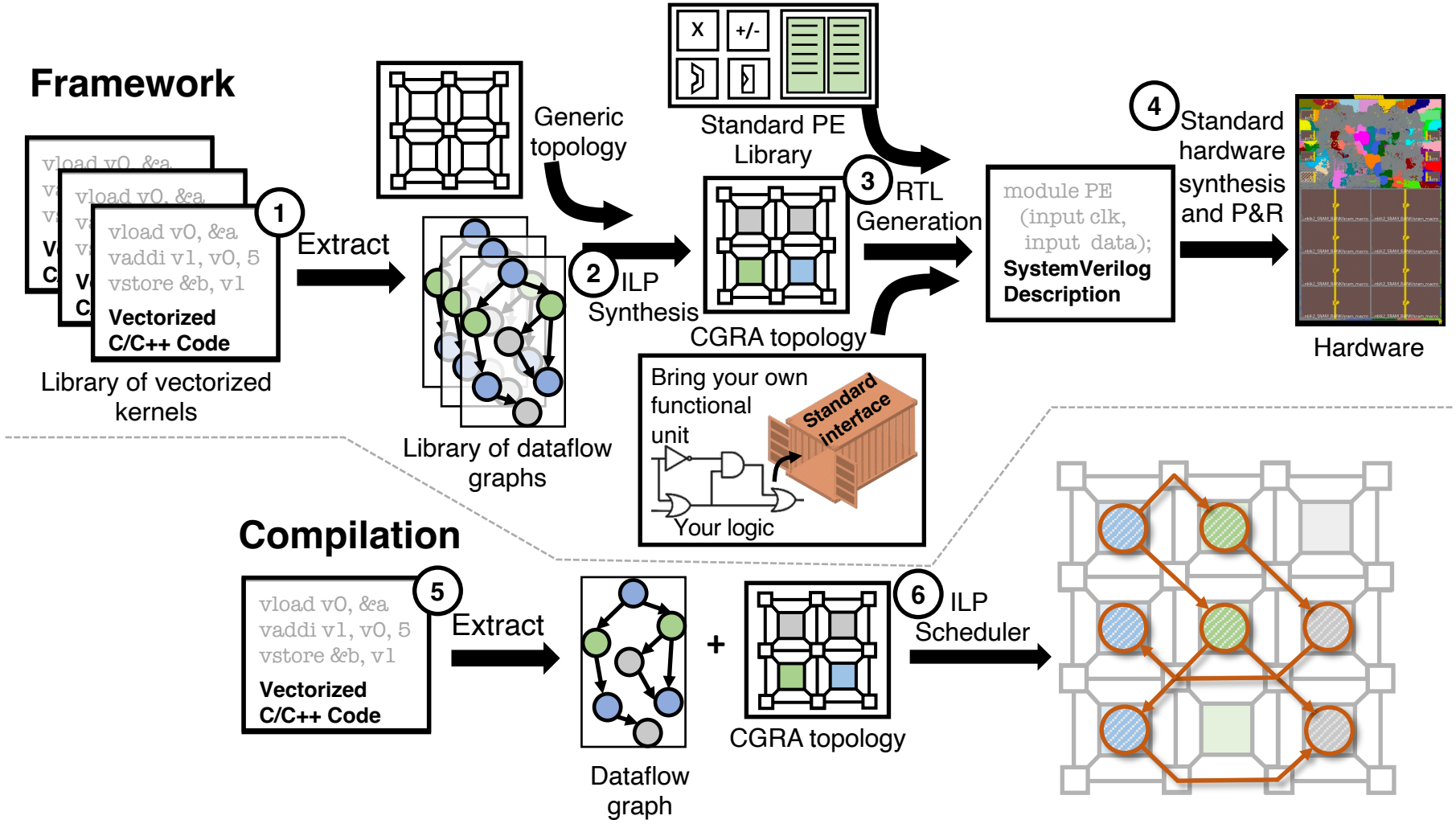
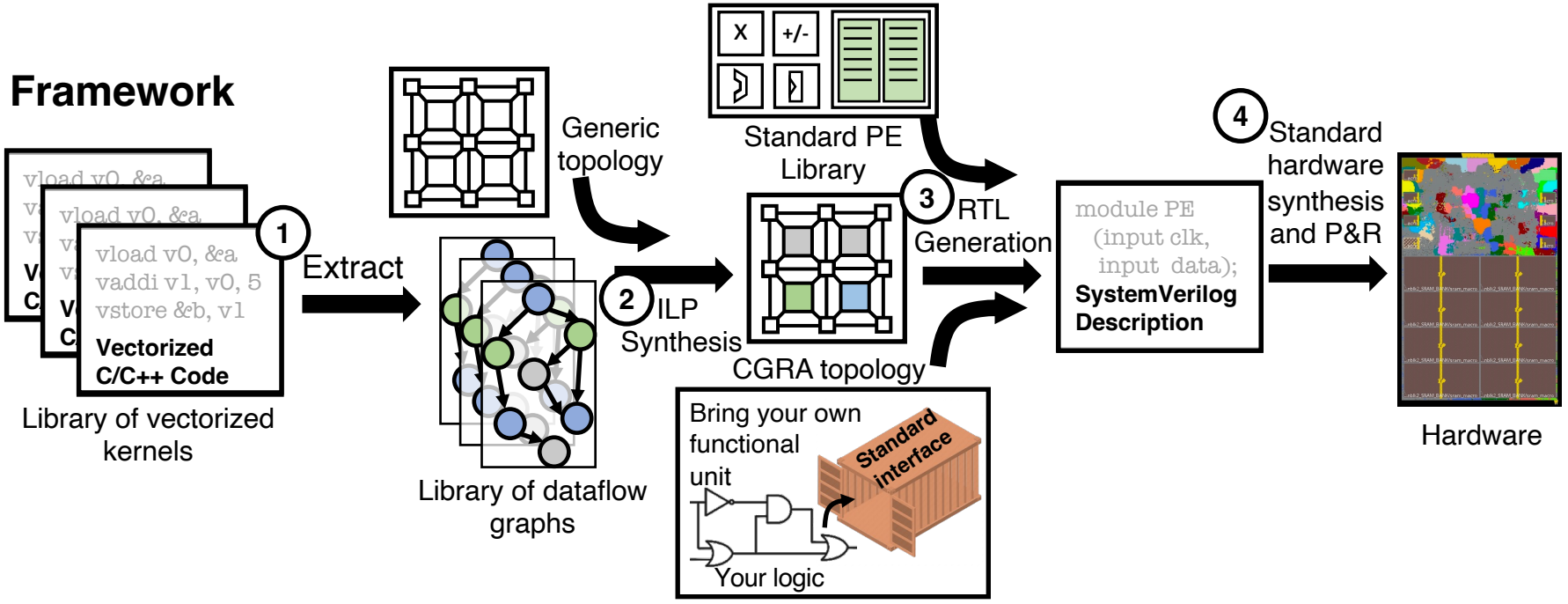
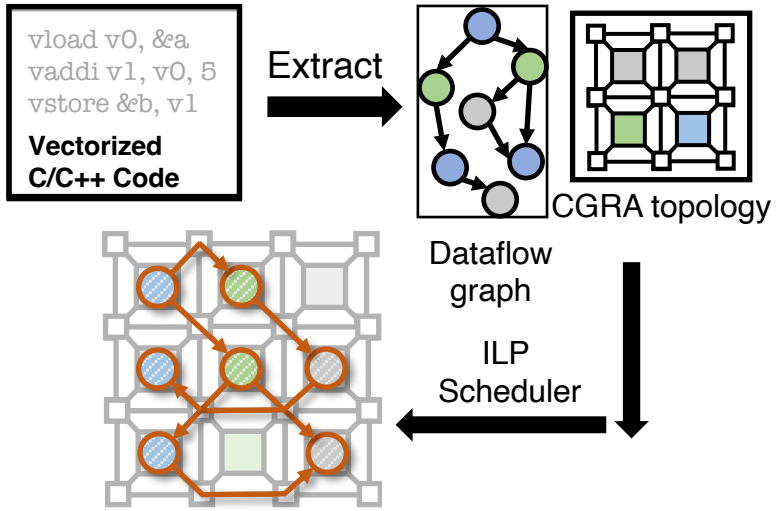
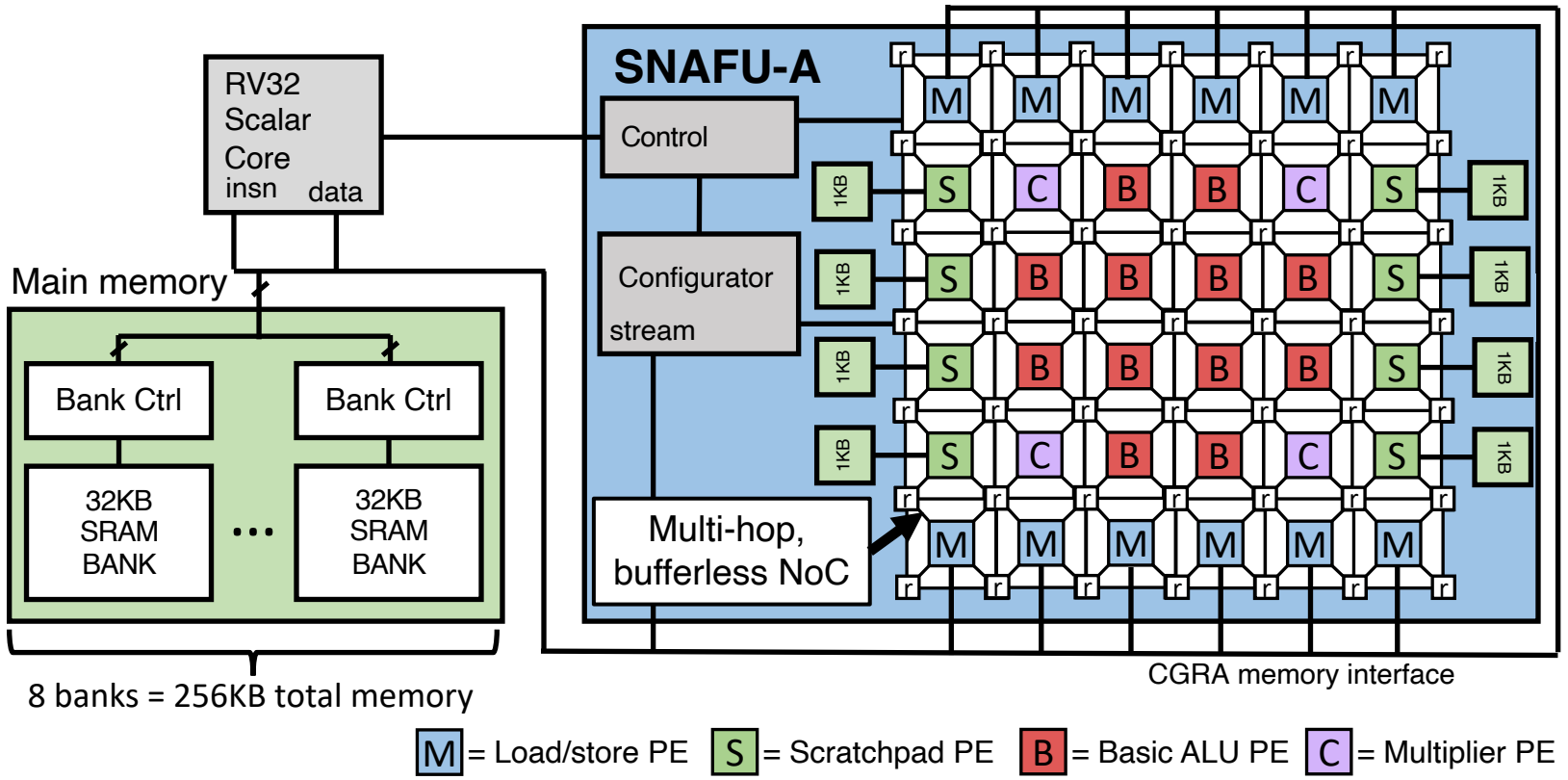
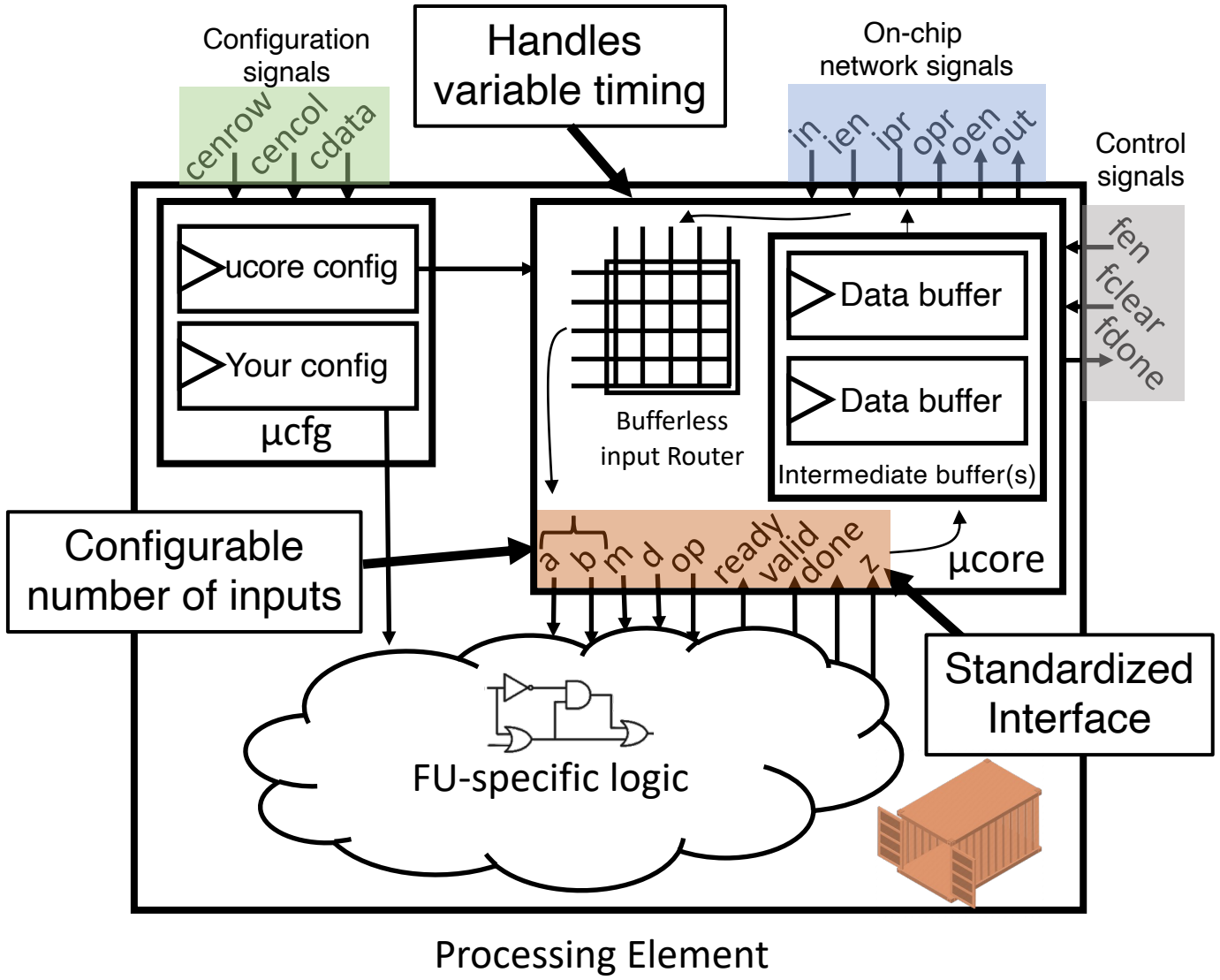

performance
Completely different operating domain
Existing CGRAs
Ideal
ASIC
SNAFU
MANIC
Classic Vector
COTS MCU
2-3 orders of magnitude
flexibility
energy
Ideal
COTS MCU
Classic Vector
MANIC
SNAFU
Existing CGRAs
ASIC
energy

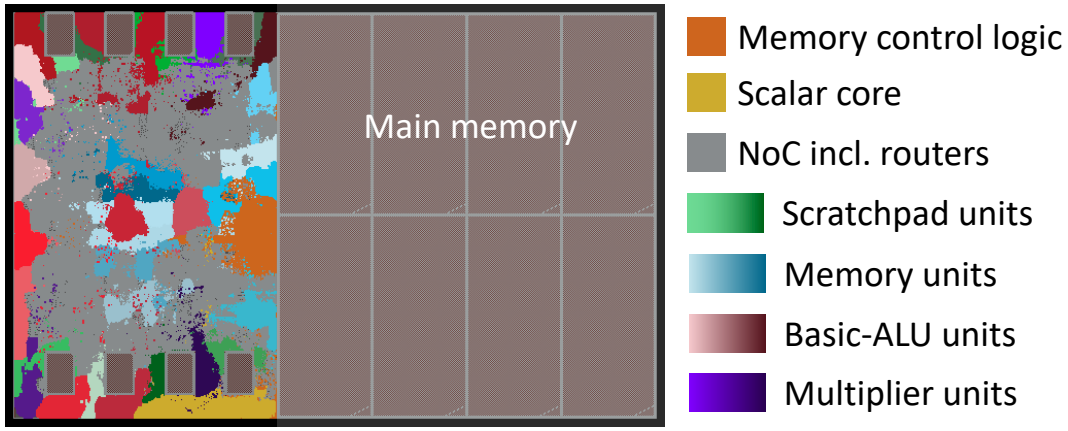

Framework
vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
Library of vectorized kernels
1
Extract
Library of dataflow graphs
Generic topology
2
ILP Synthesis
X
+/-
Standard PE Library
3
RTL Generation
CGRA topology
Bring your own functional unit
Standard interface
Your logic
module PE
(input clk,
input data);
SystemVerilog Description
4
Standard hardware synthesis and P&R
Hardware
Compilation
vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
5
Extract
Dataflow graph
+
CGRA topology
6
ILP Scheduler

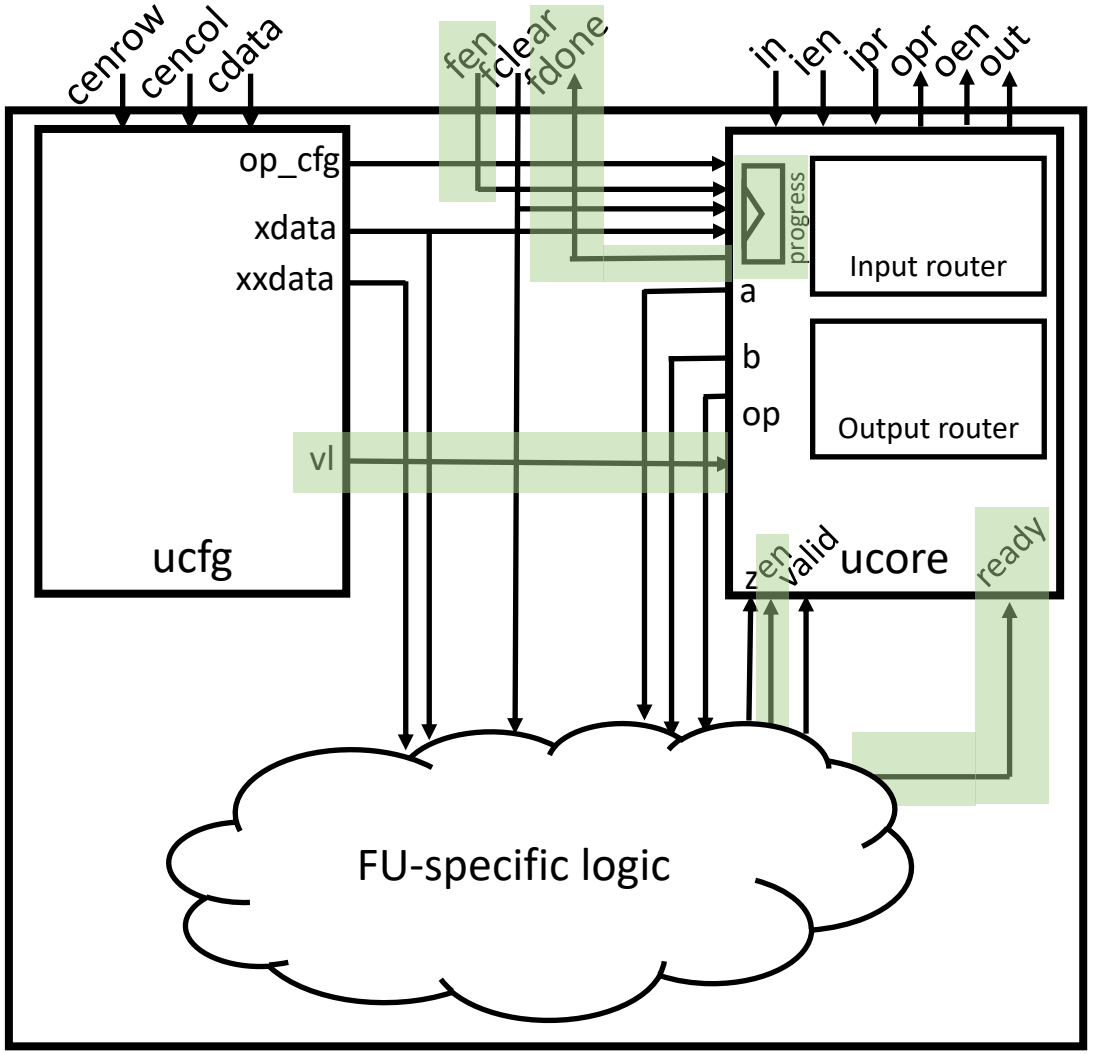

Framework
vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
Library of vectorized kernels
1
Extract
Library of dataflow graphs
Generic topology
2
ILP Synthesis
X
+/-
Standard PE Library
3
RTL Generation
CGRA topology
Bring your own functional unit
Standard interface
Your logic
module PE
(input clk,
input data);
SystemVerilog Description
4
Standard hardware synthesis and P&R
Hardware


vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
Extract
CGRA topology
Dataflow graph
ILP Scheduler

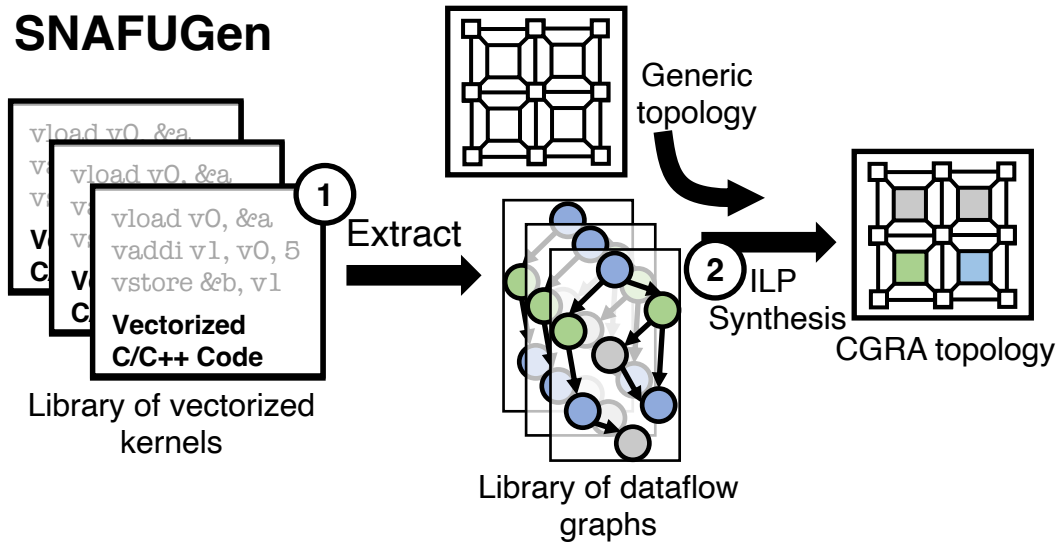

SNAFU-A
RV32
Scalar
Core
insn
data
Control
Configurator
stream
Multi-hop,
bufferless NoC
Main memory
Bank Ctrl
Bank Ctrl
32KB
SRAM
BANK
32KB
SRAM
BANK
8 banks = 256KB total memory
1KB
CGRA memory interface
M = Load/store PE
S = Scratchpad PE
B = Basic ALU PE
C = Multiplier PE

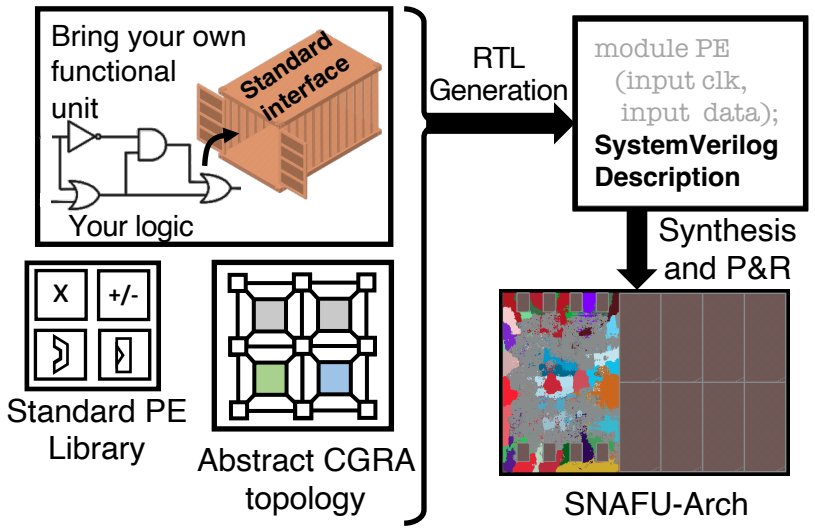

Configuration signals
cenrow
cencol
cdata
Handles variable timing
On-chip network signals
in
ien
ipr
opr
oen
out
Control signals
fen
fclear
fdone
ucore config
Your config
µcfg
Bufferless input Router
Data buffer
Data buffer
Intermediate buffer(s)
Configurable number of inputs
a
b
m
d
op
ready
valid
done
z
µcore
Standardized Interface
FU-specific logic
Processing Element

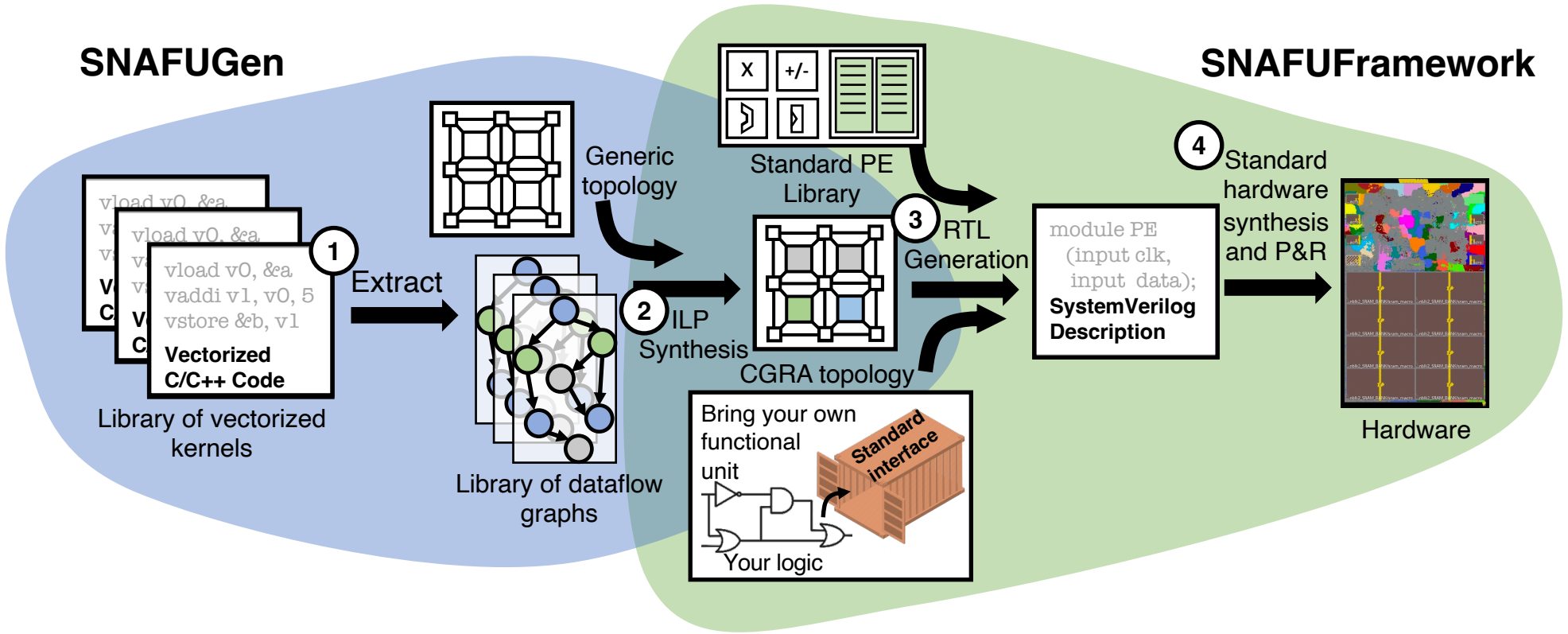

Main memory
Memory control logic
Scalar core
NoC incl. routers
Scratchpad units
Memory units
Basic-ALU units
Multiplier units

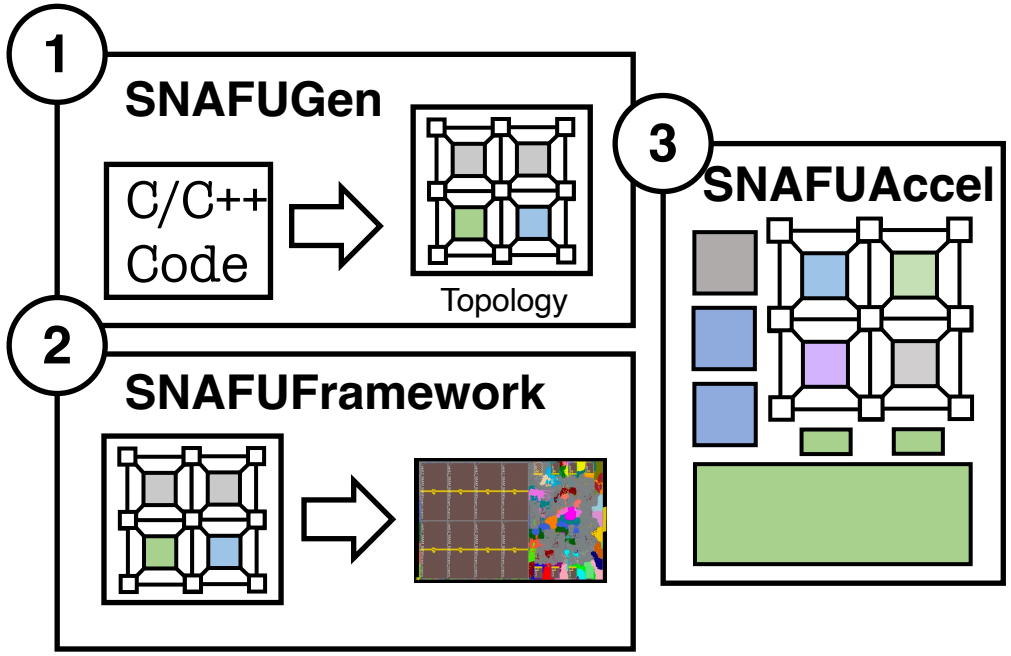


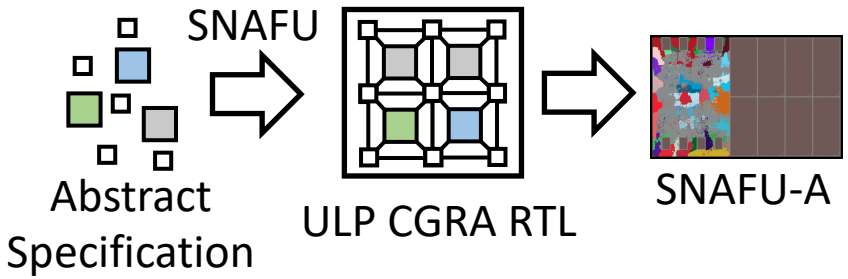
Functional unit

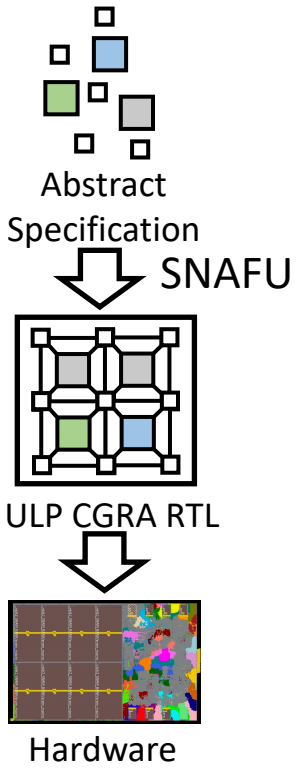

SNAFUGen
vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
Library of vectorized kernels
1
Extract
Library of dataflow graphs
Generic topology
2
ILP Synthesis
CGRA topology

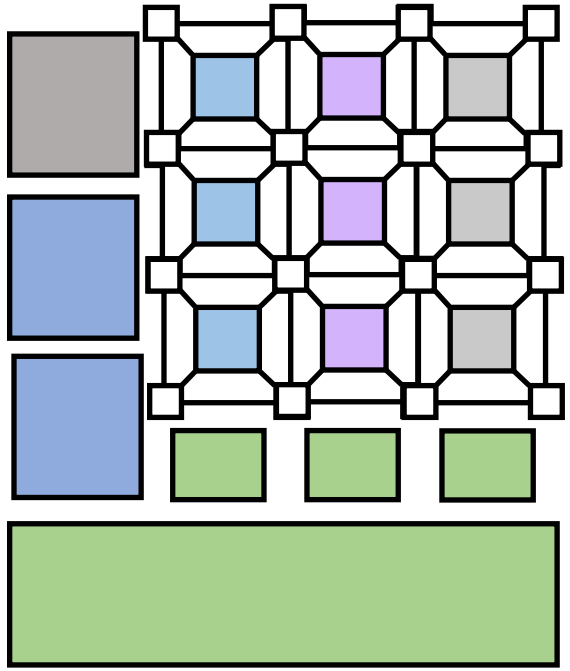

Bring your own functional unit
Standard interface
Your logic
X
+/-
Standard PE Library
Abstract CGRA topology
RTL Generation
module PE
(input clk,
input data);
SystemVerilog Description
Synthesis and P&R
SNAFU-Arch

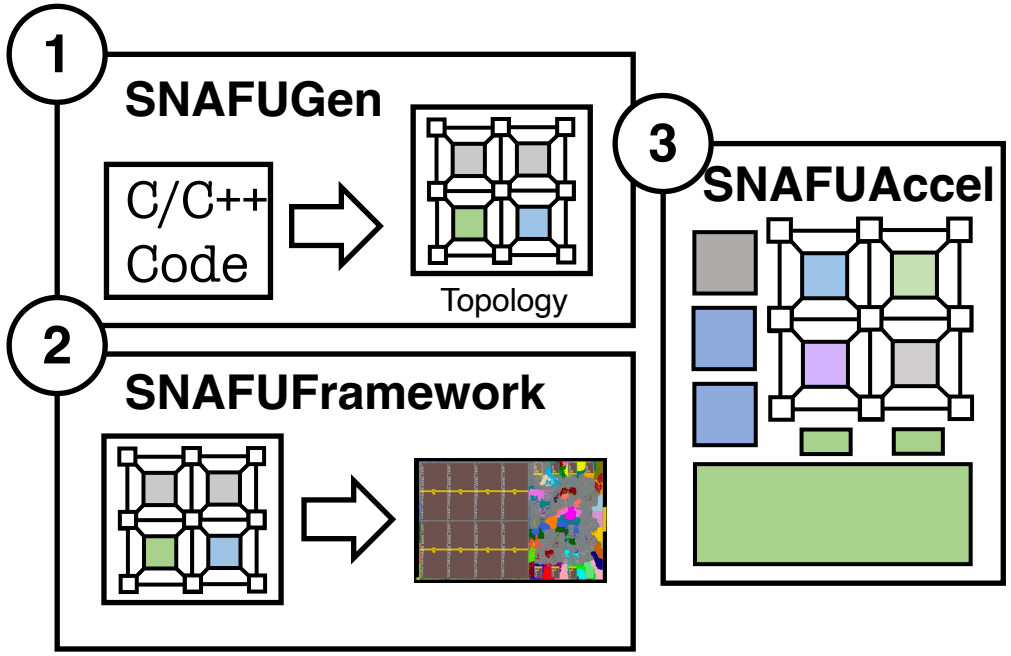

SNAFUGen
SNAFUFramework
vload v0, &a
vaddi v1, v0, 5
vstore &b, v1
Vectorized C/C++ Code
Library of vectorized kernels
1
Extract
Library of dataflow graphs
Generic topology
2
ILP Synthesis
X
+/-
Standard PE Library
3
RTL Generation
CGRA topology
Bring your own functional unit
Standard interface
Your logic
module PE
(input clk,
input data);
SystemVerilog Description
4
Standard hardware synthesis and P&R
Hardware

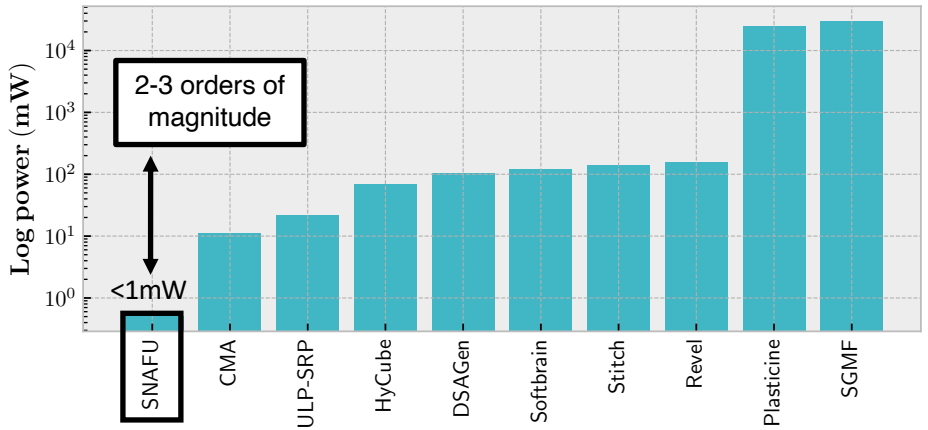

1
SNAFUGen
C/C++
Code
Topology
2
SNAFUFramework
3
SNAFUAccel

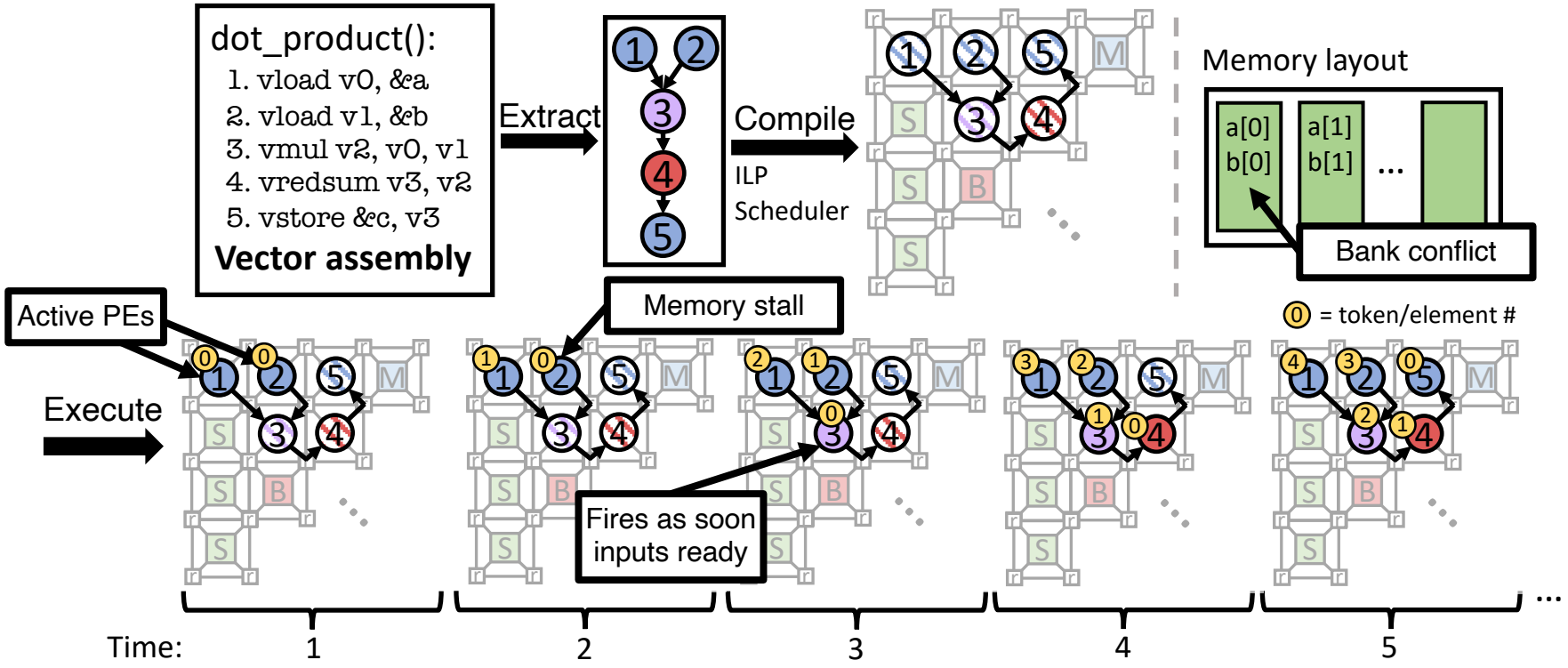

SNAFU
Abstract Specification
ULP CGRA RTL
SNAFU-A

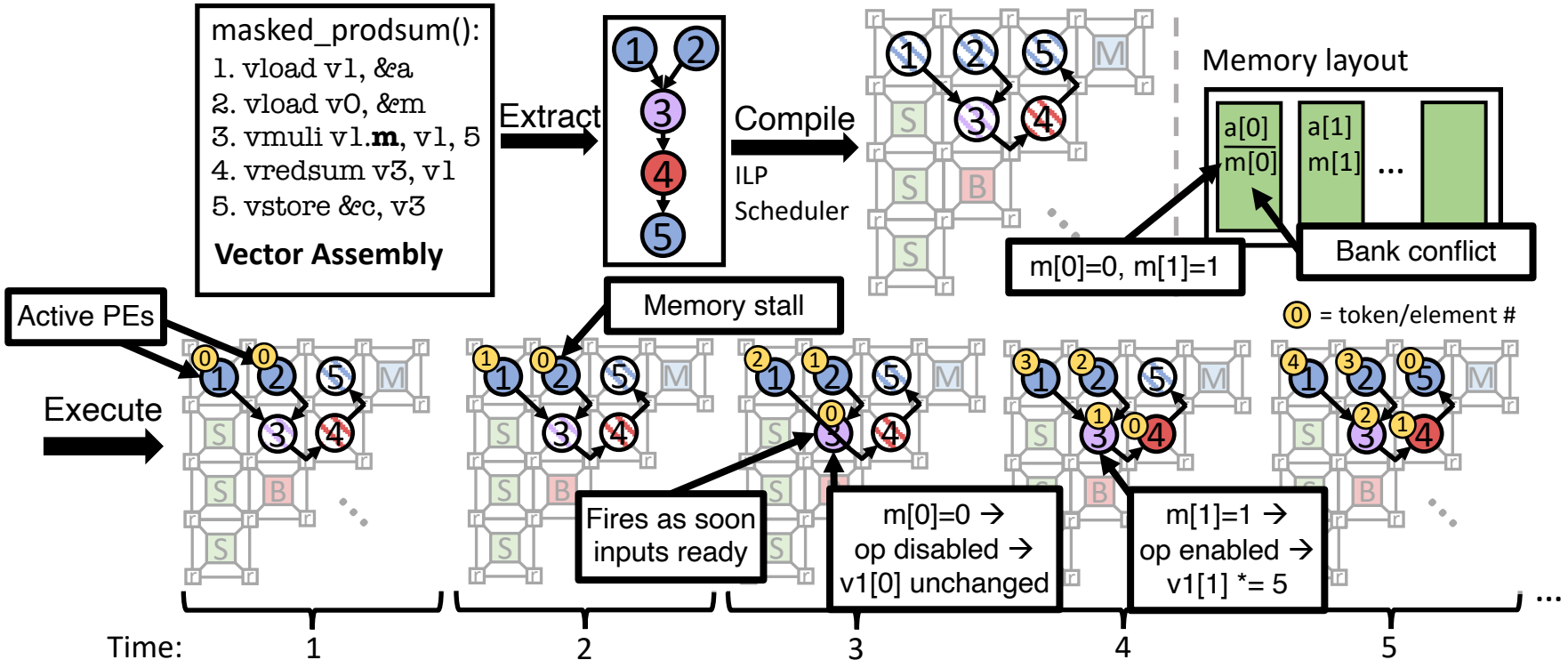

Abstract
Specification
SNAFU
ULP CGRA RTL
Hardware

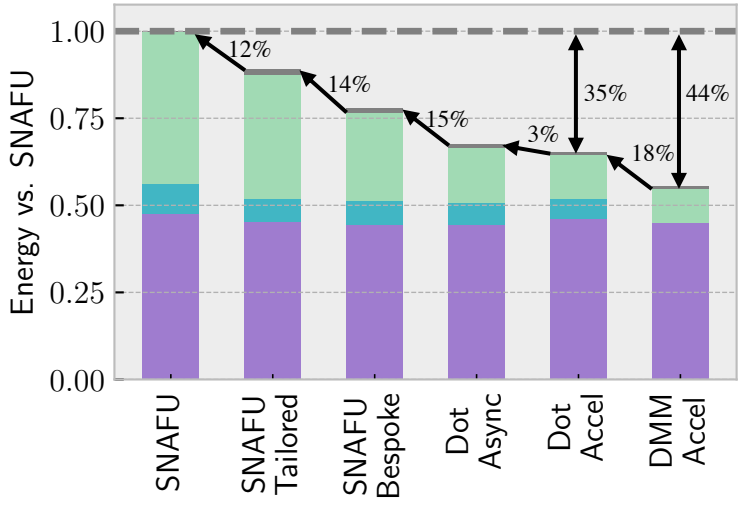


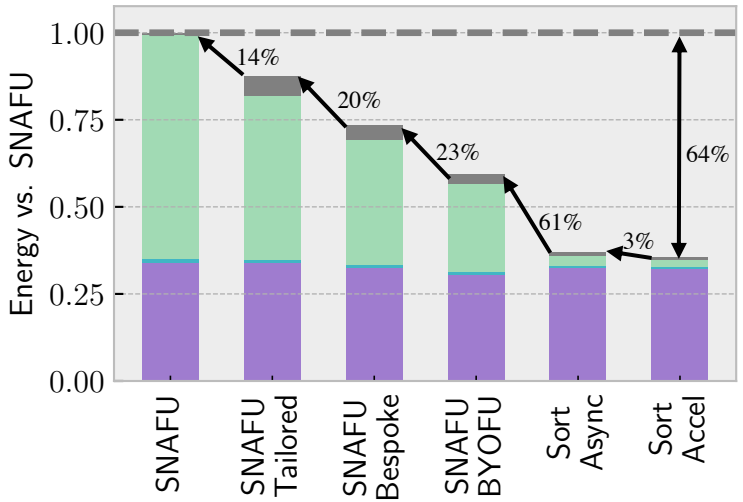

1
SNAFUGen
C/C++
Code
Topology
2
SNAFUFramework
3
SNAFUAccel

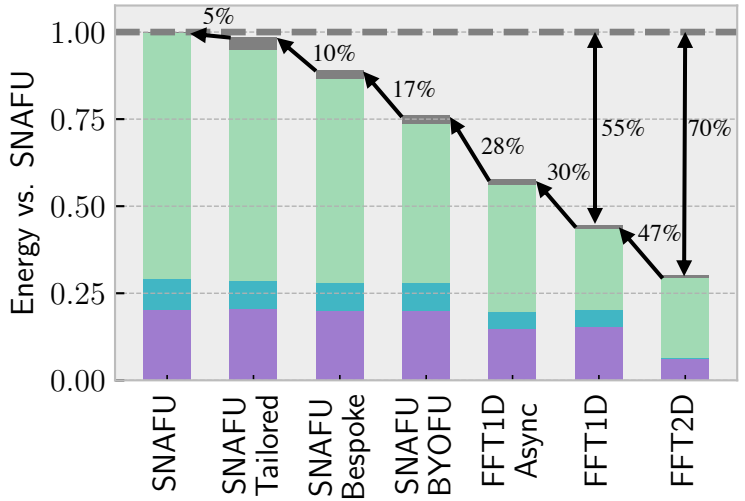

Log power (mW)
10^4
10^3
10^2
10^1
10^0
2-3 orders of magnitude
<1mW
SNAFU
CMA
ULP-SRP
HyCube
DSAGen
Softbrain
Stitch
Revel
Plasticine
SGMF


dot_product():
1. vload v0, &a
2. vload v1, &b
3. vmul v2, v0, v1
4. vredsum v3, v2
5. vstore &c, v3
Vector assembly
Extract
Compile
ILP
Scheduler
Memory layout
a[0]
b[0]
a[1]
b[1]
...
Bank conflict
Active PEs
Execute
Memory stall
Fires as soon
inputs ready
0 = token/element #
Time:
1
2
3
4
5
...


masked_prodsum():
1. vload v1, &a
2. vload v0, &m
3. vmuli v1.m, v1, 5
4. vredsum v3, v1
5. vstore &c, v3
Vector Assembly
Extract
Compile
ILP
Scheduler
Memory layout
a[0]
m[0]
a[1]
m[1]
...
Bank conflict
m[0]=0, m[1]=1
0 = token/element #
Active PEs
Execute
Memory stall
Fires as soon
inputs ready
m[0]=0 →
op disabled →
v1[0] unchanged
m[1]=1 →
op enabled →
v1[1] *= 5
Time:
1
2
3
4
5
...

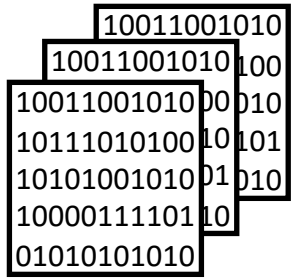

Energy vs. SNAFU
1.00
0.75
0.50
0.25
0.00
12%
14%
15%
3%
18%
35%
44%
SNAFU
SNAFU Tailored
SNAFU Bespoke
Dot Async
Dot Accel
DMM Accel


Energy vs. SNAFU
1.00
0.75
0.50
0.25
0.00
14%
20%
23%
61%
3%
64%
SNAFU
SNAFU Tailored
SNAFU Bespoke
SNAFU BYOFU
Sort Async
Sort Accel

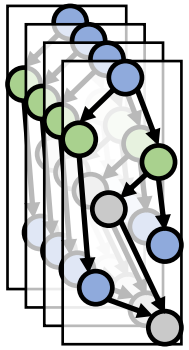

Energy vs. SNAFU
1.00
0.75
0.50
0.25
0.00
5%
10%
17%
28%
30%
47%
55%
70%
SNAFU
SNAFU Tailored
SNAFU Bespoke
SNAFU BYOFU
FFT1D Async
FFT1D
FFT2D

Compilation Pipeline

1. Extract DFGs



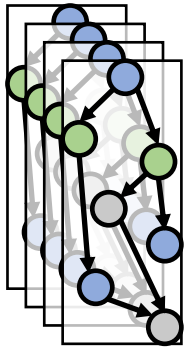
Extract



Library of DAGs

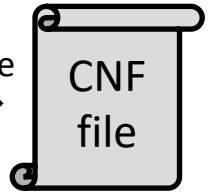
Status:
Can extract from C-like code

2. Synthesize fabric



Library of DAGs

Synthesize



SAT

~A, B, C, ~D

Variable assignments

Convert

links

0	0 0 0 1 0 0 1
1	0 0 1 0 0 0 1
2	0 1 0 0 0 0 0
3	0 0 0 1 0 0 1

Topology adjacency matrix

Status:
Redid SAT transform, bugs remain in formulation

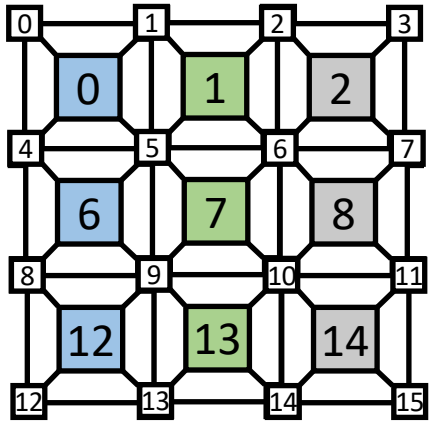
3. Generate RTL

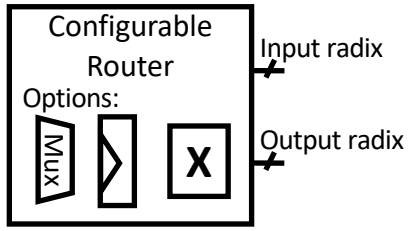
links

0	0 0 0 1 0 0 1
1	0 0 1 0 0 0 1
2	0 1 0 0 0 0 0
3	0 0 0 1 0 0 1

Topology adjacency matrix

Genus



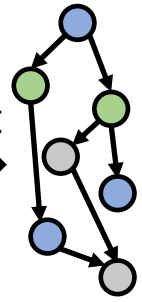


Status:
Genus accepts and synthesizes basic CGRA

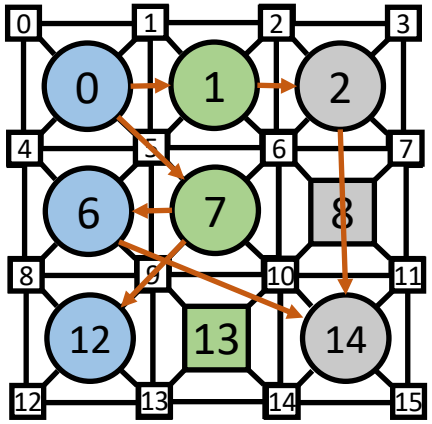
4. Schedule

10011001010
10111010100
10101001010
10000111101
01010101010

Extract



ILP

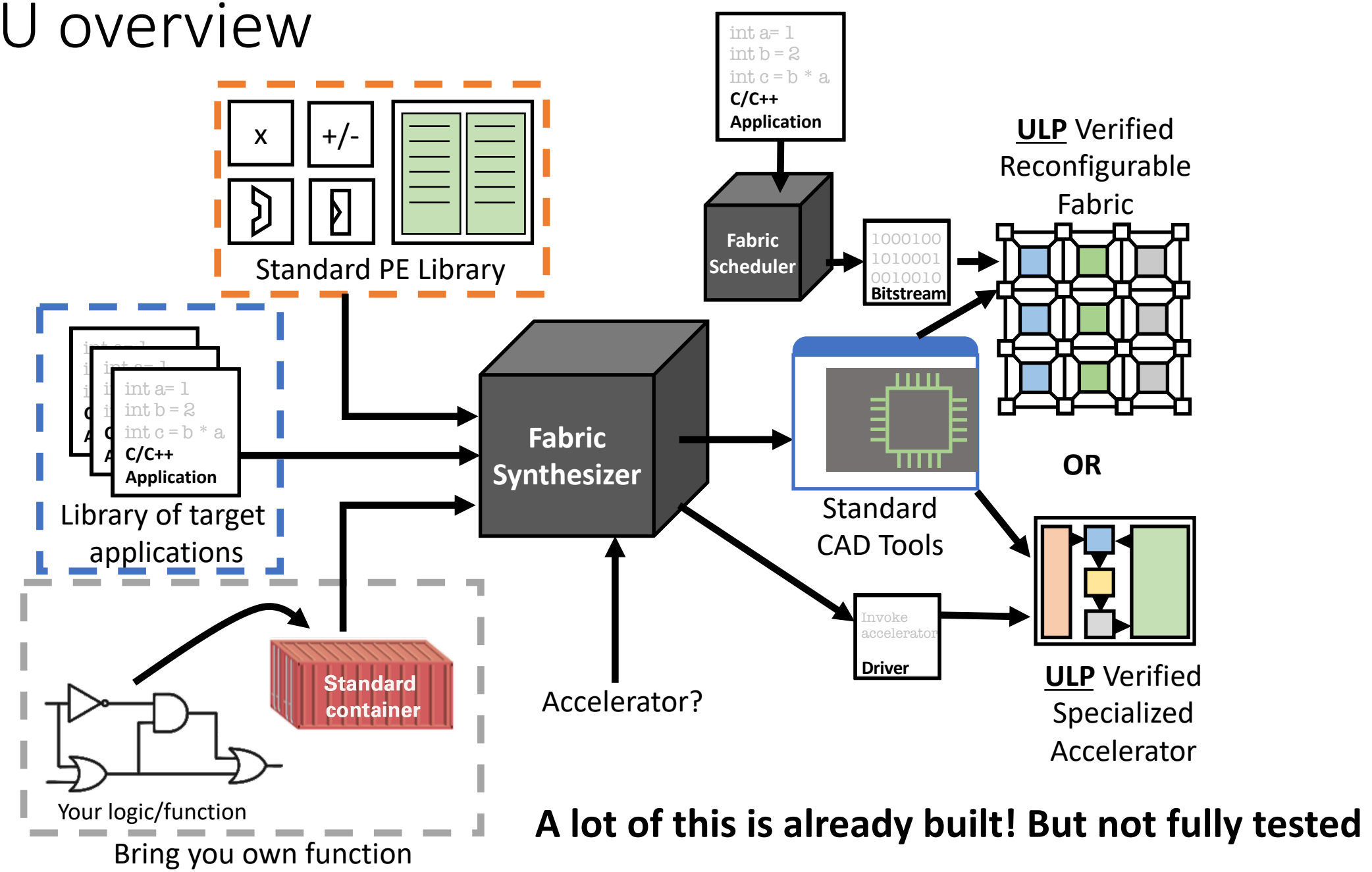


Status:
Work in progress

0000100010001000111001100001101010 10

Bitstream

SNAFU overview



A lot of this is already built! But not fully tested