

SYNOPSYS·新思

# Creating Custom RISC-V Processors Using ASIP Design Tools A Post-Quantum Cryptography Case Study

毛海雪, 新思科技解决方案事业部, 资深应用工程师  
August 22, 2024

# Problem Statement: RISC-V Extensibility

## ISA customization & extensibility drive RISC-V adoption



**RISC-V® Summit Europe 2023, 5-9 June 2023, Barcelona, Spain**

15:15 - Andrei Ivanov, ETH Zürich - RIVETS: An Efficient Training and Inference Library for RISC-V with **Snitch Extensions**

14:45 - Guy Lemieux, University Of British Columbia - From CCX to CIX: A Modest Proposal for **(Custom) Composable Instruction eXtensions**

16:45 - Marton Bognar, imec-DistriNet, KU Leuven - Proteus: **An Extensible RISC-V Core for Hardware Extensions**

12:00 - Stanislaw Kaushanski, MINRES GmbH - Automated Cross-level Verification Flow of a Highly Configurable RISC-V Core Family with **Custom Instructions**

9:00 - Philipp Tomsich, VRULL GmbH - SW-driven evolution of a uniquely **modular and extensible ISA**

15:00 - Tariq Kurd, Codasip - RISC-V code-size reduction with Zc extensions and **dictionary compression custom instruction**

Demo Theatre, 13.30 - Jon Taylor - Imperas Software Ltd. - RISC-V Models for Verification, Software Development and **Architectural Exploration**

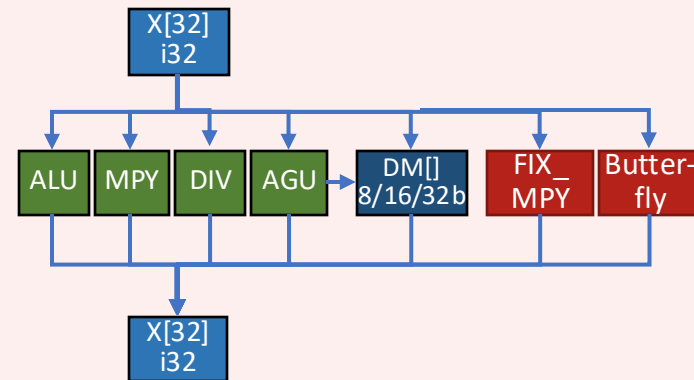
Pascal Gouédo - CV32E40P updates: **customizing an open-source RISC-V core** at industrial-grade; experiences and challenges

Daniel Vázquez - **Extending RISC-V Datapaths** with Coarse-Grained Reconfigurable Architectures

Gert Goossens - Taking the Risk out of **Optimizing Your Own RISC-V** Architecture Design

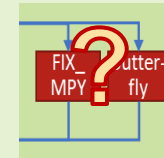
Jérôme Fereyre - VPSDK : a portability library for **extended arithmetic operations** targeting a RISC-V Variable eXtended Precision accelerator.

## This results in ASIPs with a RISC-V baseline ISA



- Better power, performance & area (PPA)
- Preserve RISC-V compatibility:
  - Execute SW code & libraries
  - Reuse HW peripherals that have been designed for general-purpose RISC-V

## Challenges...



**Which extensions** are best for the target application domain?



How to obtain a high-quality **SW Development Kit (SDK)**, including an optimizing compiler?



How to obtain a reliable **RTL implementation** with excellent PPA?



How to **verify** the design?

# Agenda

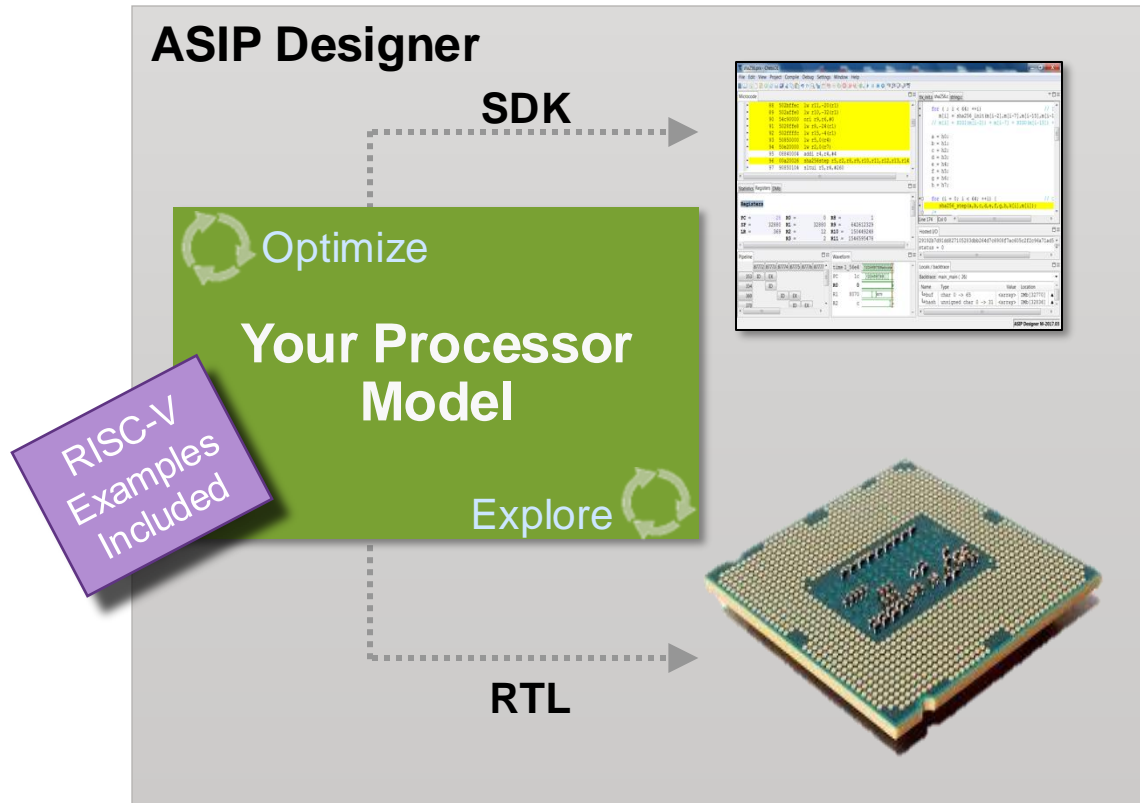
- Synopsys ASIP Designer introduction
  - a processor / DSP / accelerator development infrastructure
- Synopsys ASIP Designer RISC-V offering
  - RISC-V example processor models
- Synopsys RISC-V extension support
  - simple and large-scale extensions
- Case Study
  - An ASIP for Post-Quantum Cryptography

# Synopsys ASIP Designer introduction

## A Processor / DSP / Accelerator Development Infrastructure

# ASIP Designer™ Automates Design of Custom Processors/Accelerators

Architectural Exploration through RTL Generation Enables PPA-Optimized Implementation

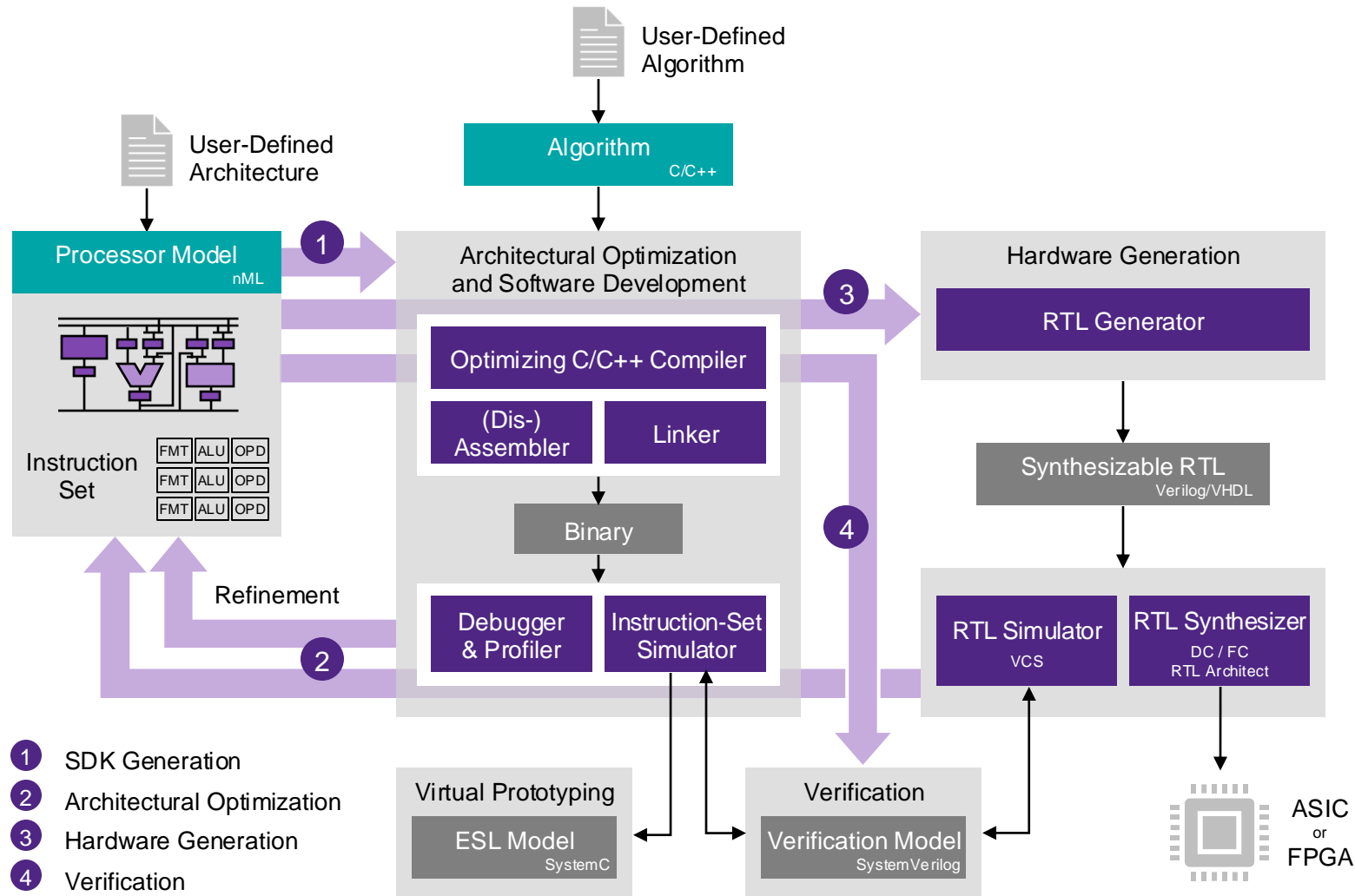


Used by 7 of the Top 10 Semiconductor Developers

- Industry's leading tool for creating Application-Specific Instruction-Set Processors (ASIPs)
  - Language-based description of ISA provides full architectural flexibility
  - Automatic generation of professional software development kit (SDK)
  - Automatic generation of synthesizable RTL and debug infrastructure
  - Accelerated verification and virtual prototyping
  - Integrated with Synopsys' Reference Design & Verification Flows
  - Increased engineering productivity
- More than 2 dozen example models included to accelerate engineering productivity
  - From microprocessors, DSPs, vector processors, to programmable datapaths, and RISC-V cores
  - Easily customizable
  - Provided in source code to jump-start your design

# ASIP Designer

## Tool Flow



## Supported design steps

- Modeling of instruction-set architectures: nML language
- Automatic generation of software development kit, including an efficient C/C++ compiler
- Algorithm-driven architectural exploration: **“Compiler-in-the-Loop”**
- Automatic generation of synthesizable RTL **“Synthesis-in-the-Loop”**
- Design verification

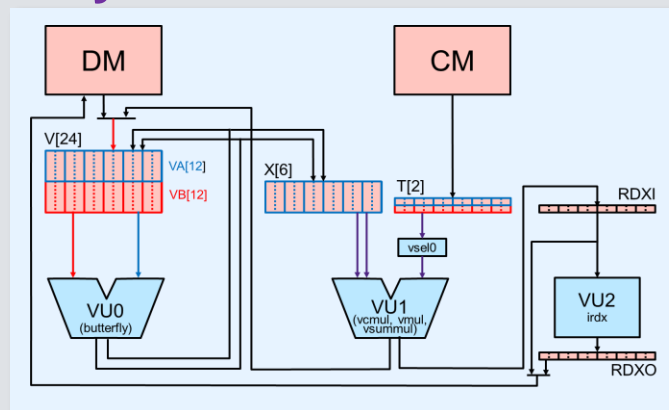
# ASIP Design – Some Cases Studies

## Full Architectural Freedom, Not Based on a Template

### Add Programmability to RTL Accelerator

#### FFT / DFT Accelerator

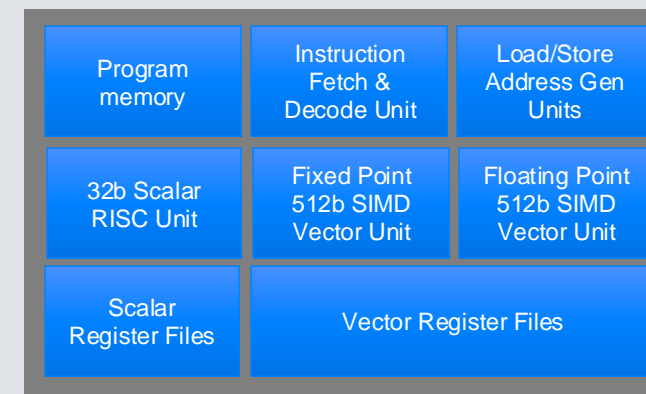
- FFT: all power-of-2 sizes from 8 to 2048, algorithm up to radix-8
- DFT: all prime-factorizable sizes from 6 to 1536
- 8-lane SIMD, 5-way VLIW



### Custom DSP Development

#### Vector-DSP

- Combined SIMD / VLIW design
- Broad data type support: Int 4/8/16/32 Float, Bfloat 16, fixed-point, complex
- Fully C-programmable

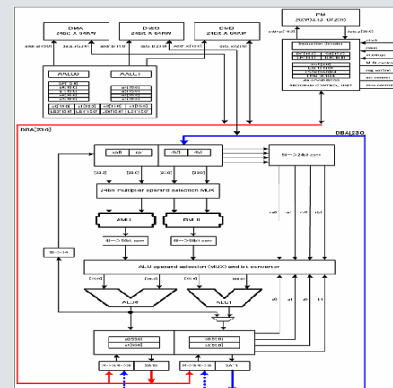
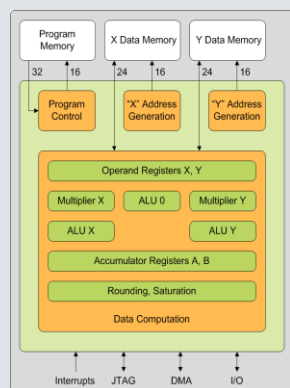


### (Extreme) Low-Power with Flexibility



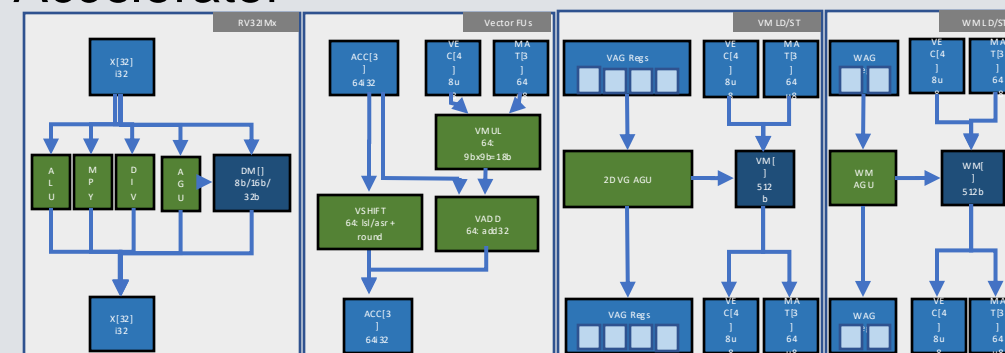
CoolFlux DSP onsemi LPDSP32

- Ultra-low power DSP
- Hearing instruments & audio wearables
- 24-bit precision
- Dual Harvard
- ILP: 8 parallel ops
- 43K gates
- 25μW/MHz @ 0.9 V (65nm CMOS)



### ASIP Designer RISC-V Design Solutions

#### AI Accelerator



**RISC-V Scalar ISA +Vector Ext. +AGUs & Load/Store**

# Synopsys ASIP Designer RISC-V Offering “Trv” Processor Model Family



# Synopsys Processor Portfolio

ARC Processor IP

Unrivaled PPA Efficiency for Modern Processing Workloads

NEW

ARC-V™

## RMX Family Ultra Low Power Embedded

- 32-bit embedded processor, DSP option
- High efficiency 3- and 5-stage pipeline configs

## RHX Family Real-Time Performance

- 32-bit real-time processor, 1-16 cores
- High-speed, dual-issue 10-stage pipeline

## RPX Family Host Processor

- 64-bit host processor, 1-16 cores
- SMP Linux, L2 cache support

Specialty

## VPX Family Vector DSP

- SIMD/VLW design for parallel processing
- Multiple vector FP units for high precision

## NPX Family NPU

- Scalable neural processor units (1K-96K MACs)
- Supports latest AI networks (e.g., transformers)

## EV Family Vision Processor

- Heterogeneous multicore for vision processing
- DNN (Deep Neural Network) Engine

Classic

## EM Family Embedded MPU

- 3-stage pipeline with high efficiency DSP
- Optimized for low power IoT

## SEM Family Security CPU

- Protection against HW, SW, side channel attacks
- SecureShield to create TEE's

## HS Family High Speed CPU

- High performance CPUs, CPU + DSP
- Single- and multi-core configs

## Functional Safety (FS) Processors

- Integrated hardware safety features for all processor families
- Accelerates ISO 26262 certification for safety-critical automotive SoCs

Design Tool

## ASIP Designer™ Processor / DSP / Accel. Design Tool

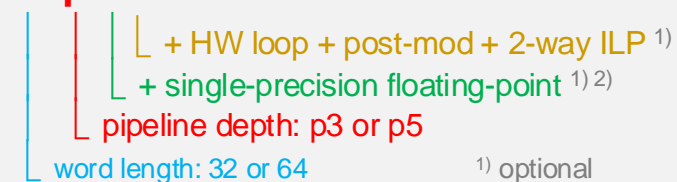
- Tool automating the creation of application-specific instruction-set processors (ASIPs)
- When processor IP cannot meet PPA requirements and fixed hardware is not flexible enough

## Functional Safety (FS) Tool Features

# Trv (RISC-V ISA) Models

Shipped with ASIP Designer

Trv**32**p**3**f**x**



## Integer models: Trv<mm>p<n>**[x]**

	32-bit datapath	64-bit datapath
3-stage pipeline	Trv32p3 Trv32p3x	Trv64p3 Trv64p3x
5-stage pipeline	Trv32p5 Trv32p5x	Trv64p5 Trv64p5x

- ISA: RV64IM, RV32IM
  - Integer and multiply instructions
- Micro architecture
  - Protected pipeline, 3 or 5 stages
  - Hardware multiplier
  - Iterative divider
- Optional extensions: Trv<mm>p<n>**x**
  - Two-way static ILP
  - Zero overhead hardware loops
  - Load/stores with post-modify addressing

## Floating-point models: Trv32p<n>**f[x]**

	32-bit datapath
3-stage pipeline	Trv32p3f Trv32p3fx
5-stage pipeline	Trv32p5f Trv32p5fx

- ISA: RV32IMZfinx
  - Integer and multiply instructions
  - Single precision float instructions
- Micro architecture
  - Protected pipeline, 3 or 5 stages
  - FPU models based on HardFloat [Hauser]
  - Iterative divider and square-root units
- Optional extensions: Trv32p<n>**fx**
  - Two-way static ILP
  - Zero overhead hardware loops
  - Load/stores with post-modify addressing

# Synopsys RISC-V Extension Support

## Simple extensions and large-scale extensions

# Trv (RISC-V ISA) Models

## SDX: Simple Datapath eXtensions

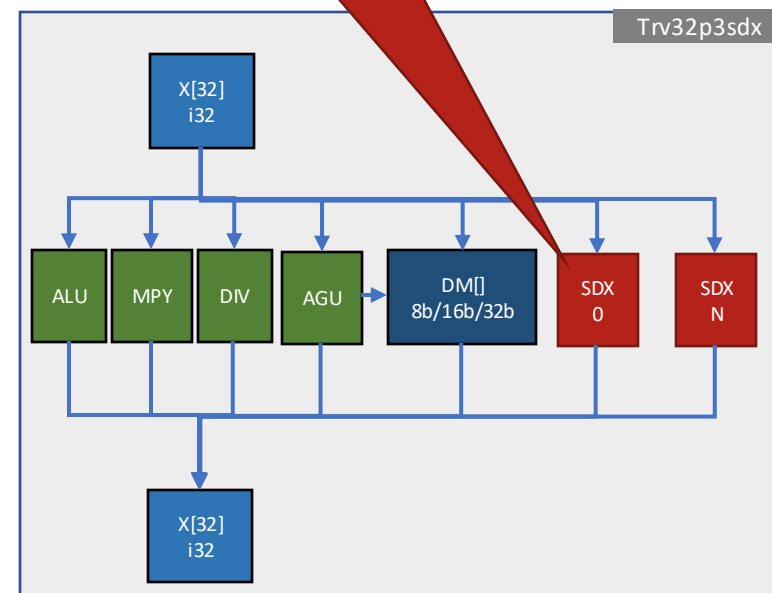
- SDX is a mechanism to add simple extension instructions to Trv (RISC-V)
  - nML model of Trv32p3 with predefined **stubs** for extension instructions
  - User codes the **behavior** of the stubs in PDG (bit-accurate C code)
  - Compiler intrinsics that target the extension instructions are provided (and can be modified)
- Benefits
  - No (deep) nML knowledge required: extensions can be created by SW engineers
  - Fast exploration of extensions with compiler-in-the-loop & synthesis-in-the-loop
- Extensions encoded in RISC-V custom-2 space
- Operand options
  - 3-register (32 and 64-bit)
  - Accumulate
  - Additional single register inputs and outputs

### Behavioral model (PDG)

```
w32 sdx0(w32 a, w32 b)
{
  w32 r;
  r[15:0] = a[15:0] + b[15:0];
  r[31:16] = a[31:16] + b[31:16];
  return r;
}
```

### Compiler intrinsics


```
int sdx0(int,int);
int add2(int,int);
```

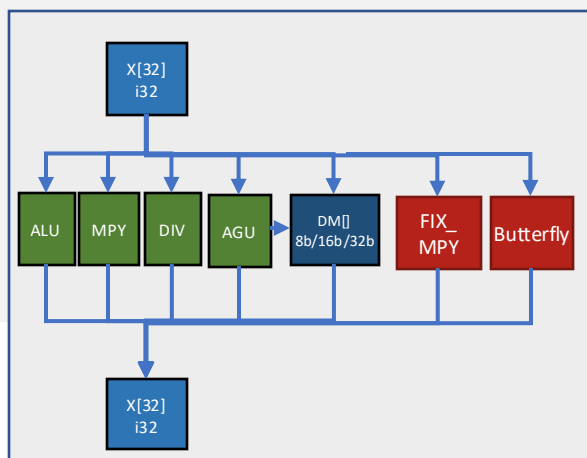


# Trv (RISC-V ISA) Models


## SDX Examples Provided With ASIP Designer

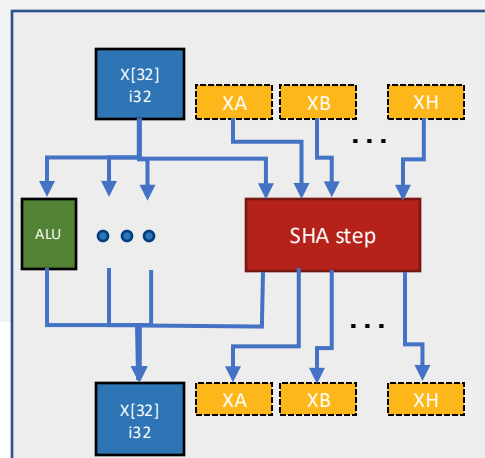
### FFT

- SDX instructions accelerating
  - Complex fixed-point multiplication & scaling  
`sdx1 rd,rs1,rs2`
  - ABS(x) function: `sdx2 rd, rs1, rs2(x0)`
  - FFT Butterfly: `sdx5 rd,rs1,rs1`
- Specialization:
  - Fractional data types
  - Complex numbers (16bit/16bit -> 32bit register)
- Speedup: 280%** Area increase: 31% 

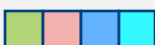



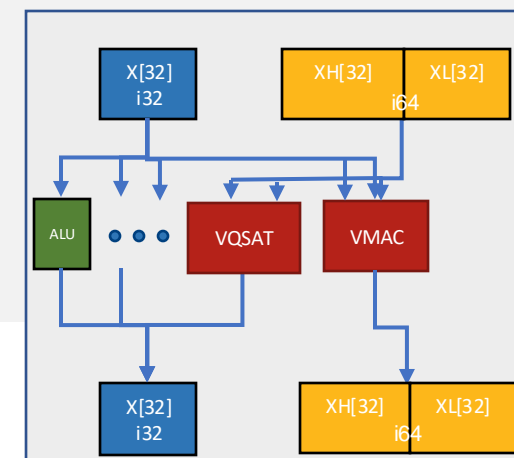
### SHA256

- Computes a hash of message W using bitwise AND, OR, XOR operations, shift operations and additions
- Custom data path is ideal to implement the complex hash function in one instruction
- Additional state of the hash functions (8 state variables) require an SDX variant that supports **8 additional register reads and writes**  
`sdx7 rd, rs1, rs2, x24,x25,..., x32`
- Speedup: 270%** Area increase: 16% 



### Keyword Spotting

- Based on small sized Neural Network (3.3M MACs)
- SDX architecture feature: **packed SIMD**
  - 32-bit register contains vector of 4x 8-bit values 
  - Use of register pairs, enabling 64-bit access  
`sdx4a_dr dd,rs1,rs2,mode // vmac`  
`sdx0_dd rd, ds1,ds2 // vqsat`
- Speedup: 1160%** Area increase: 16% 



# Large-Scale Example: “Tmoby”

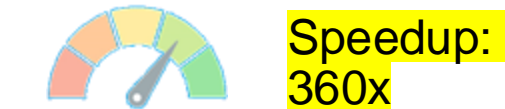
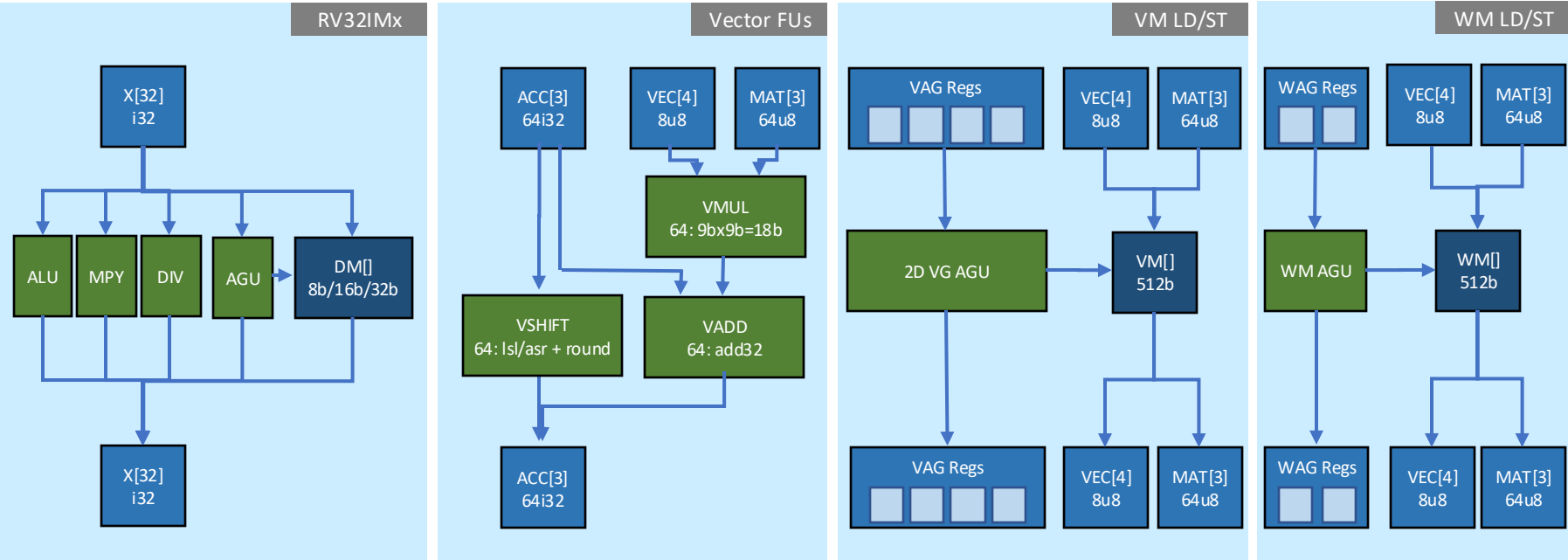
## RISC-V Based ASIP for MobileNet v3

### Application: MobileNet V3

- TensorFlow code converted to C code with Tmoby-specific vector intrinsics
- Main function
  - Implementation of neural-network graph
  - Memory copies
  - Kernels
- Kernels
  - Conv\_2D (pointwise)
  - Depthwise\_Conv\_2D
  - Add
  - Average\_Pool\_2D
  - Softmax

tmoby															
sc: bit32_ifmt				vc: slot_vc				vm: slot_vm				wm: slot_wm			
				pnpop_instr				pnpop_instr				pnpop_instr			
majOP	0110011														
majOP_IMM	0010011	x	x	xxx1xxx1x110xxx110	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
majLOAD	0000011			vec_move	01xxx110			vec_vm_ldst	0			vec_wm_ldst	0		
majSTORE	0100011			vec_ext_upd				agv_standalone	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	agw_standalone	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
majBRANCH	1100011			vec_bool	00xxx110			agv_moves	xxxx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	agw_moves	xxxx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
majJAL	1101111			vec_bc	xxx0x110xxx110										
majJALR	1100111			vec_mac											
majLUI	0110111			mux_trans_add_instr	xxx010										
majAUIPC	0010111			vec_sh	x010xxx110										
majCUSTOM0	0001011			vec_misc	xxxx0xxx1x110xxx110										
majCUSTOM1	0101011														
majCUSTOM3	1111011														

- VLIW extension of Trv32p3x
  - 90-bit instruction word
  - 4-way ILP
    - Scalar slot: Trv32p3
    - Vector arithmetic slot: SIMD64 MAC 8x8→32
    - 2 vector load/store slots: VM: features WM: weights Vector addressing

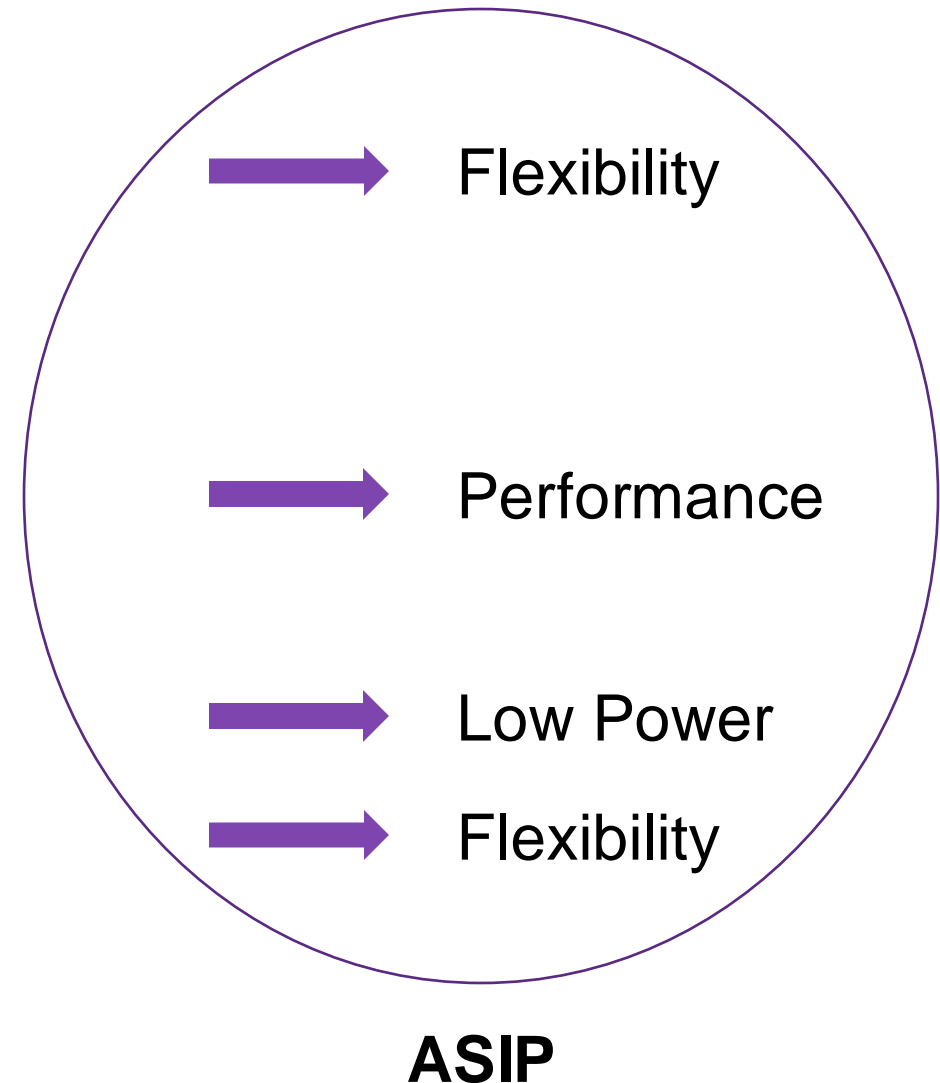


# Case Study

## An ASIP for Post-Quantum Cryptography

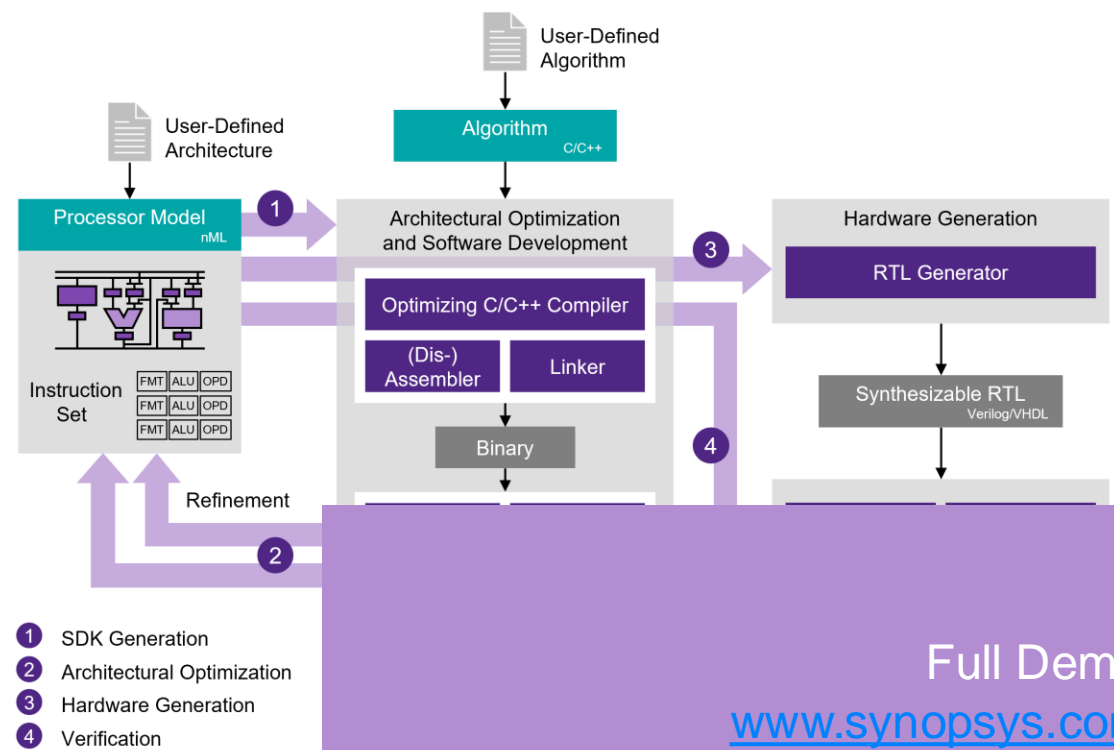
# Kyber: Summary of Requirements

- PQC Standard/algorithms still evolving
- Computational complexity:
  - Extensive Hashing
  - Modulo Arithmetic:
    - Modular Addition/Multiplication/Reduction
    - Polynomial Multiplication
  - Random Sampling (uniform, binomial)
- Mobile Devices
- Common features with other Cryptography algorithms





# Concept



- Starting point:

- Ready-to-go software (Trv32p5x)
- Compile on Trv32p5x
- Simulate & F

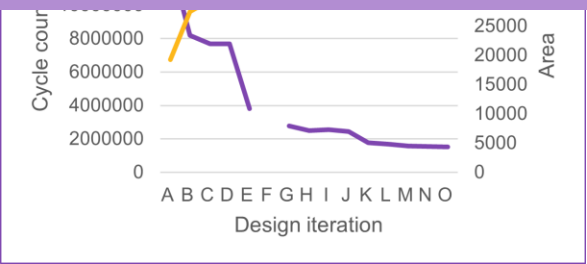
Demo

- Iteratively refine:

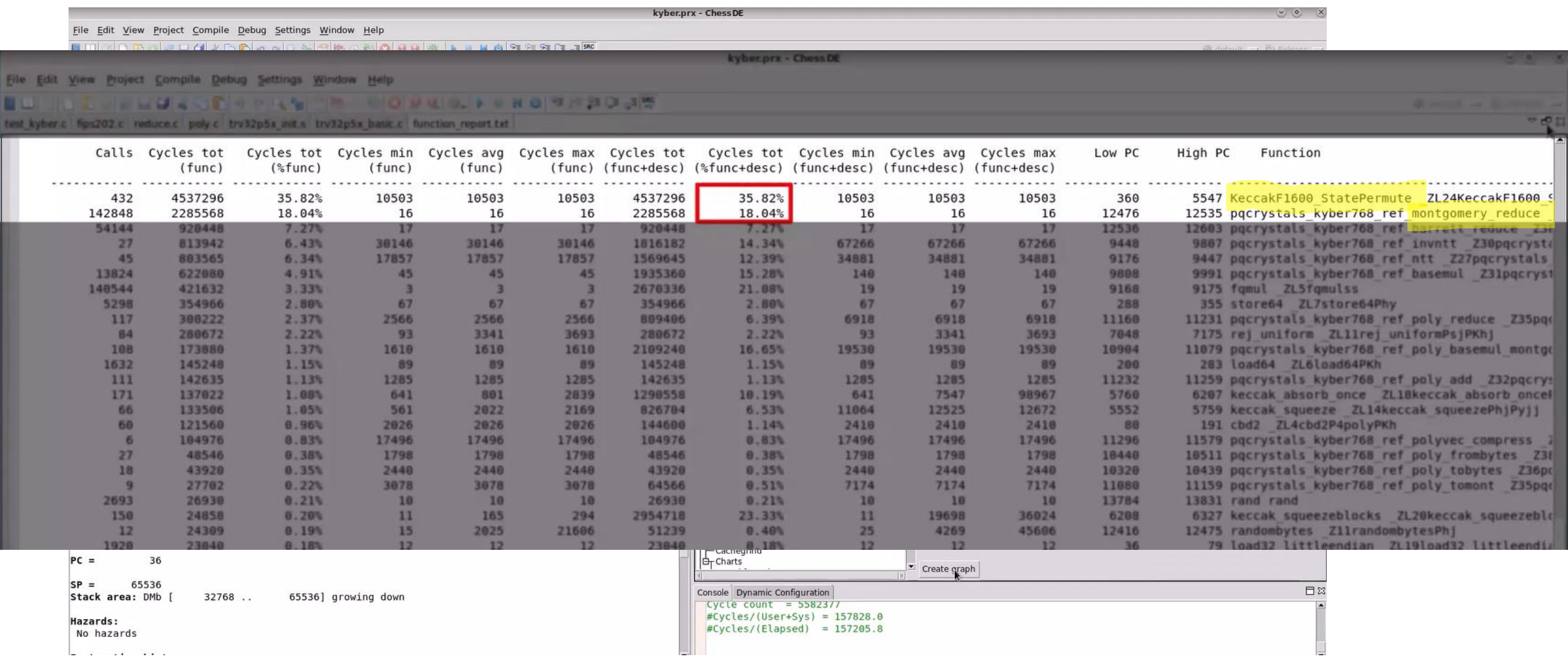
Full Demo available on:  
[www.synopsys.com/asip](http://www.synopsys.com/asip) -> ASIP Tutorials

Joint architect  
ASIP Designer:

- Compiler-in-the-loop™
- Synthesis-in-the-loop™



# Initial Profiling on Trv32p5x (Dual-issue)



# Accelerating Montgomery/Barrett Reduction

C code

```
int16_t montgomery_reduce(int32_t a)
{
    int16_t t;

    t = (int16_t)a*QINV;
    t = (a - (int32_t)t*KYBER_Q) >> 16;
    return t;
}
```



PDG code

```
#define KYBER_Q 3329
#define QINV -3327

w32 reduce_m(w32 a) {
    // a: int32_t
    int16_t t;
    t = (int16_t)a*QINV;
    t = (a - (int32_t)t*KYBER_Q) >> 16;
    return t;
}
```

reduce\_m /  
reduce\_b

C code

```
int16_t barrett_reduce(int16_t a) {
    int16_t t;
    const int16_t v = ((1<<26) + KYBER_Q/2)/KYBER_Q;

    t = ((int32_t)v*a + (1<<25)) >> 26;
    t *= KYBER_Q;
    return a - t;
}
```



PDG code

```
w32 reduce_b(w32 a) {
    int16_t t;
    int16_t v = ((1<<26) + KYBER_Q/2)/KYBER_Q;
    t = ((int32_t)v*a + (1<<25)) >> 26;
    t *= KYBER_Q;
    t = a - t;
    return t;
}
```

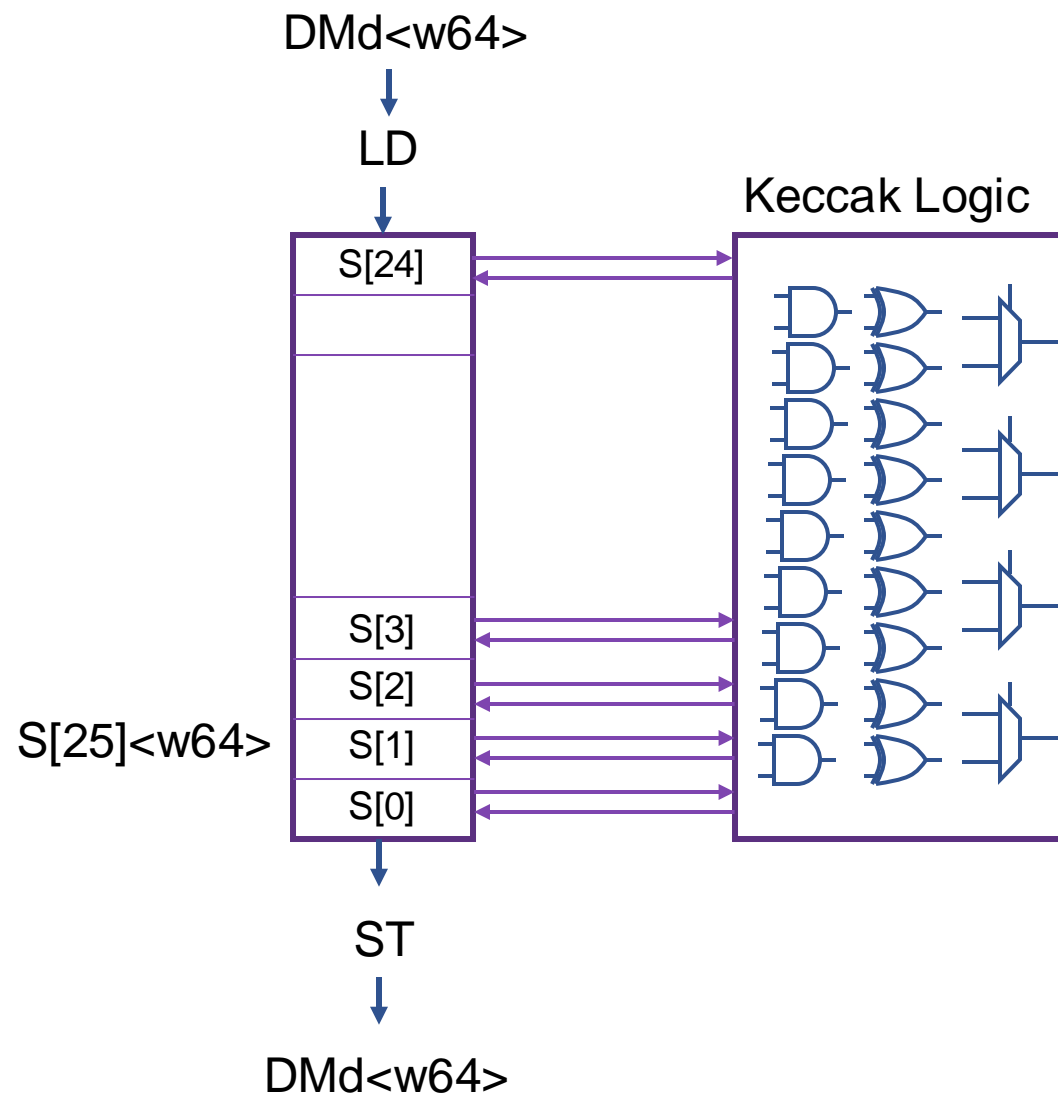
# Accelerating Keccak Permutation

C code (extract)

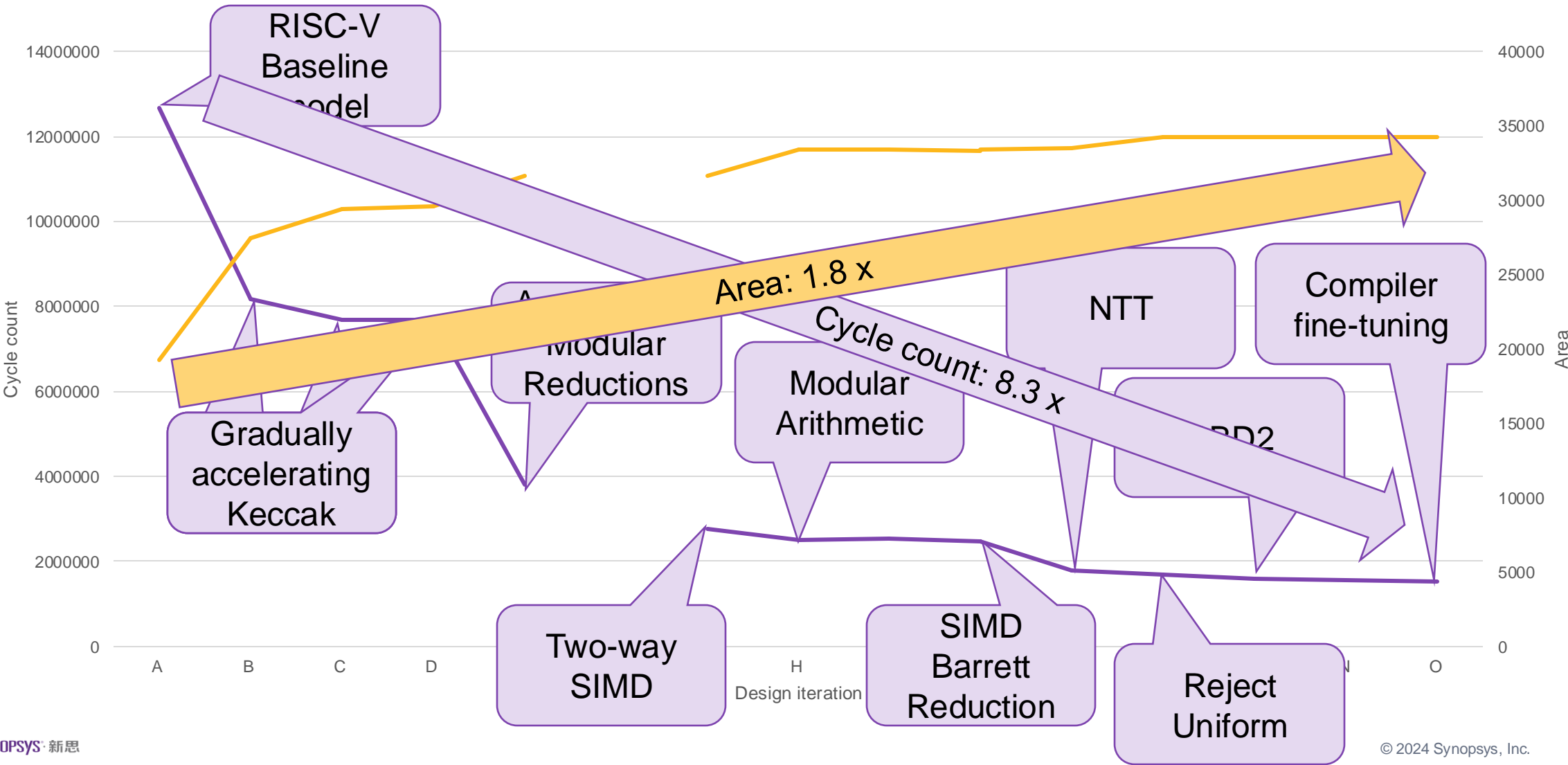
```
static void KeccakF1600_StatePermute(uint64_t state[25])
{
    int round;

    uint64_t Aba, Abe, Abi, Abo, Abu;
    uint64_t Aga, Age, Agi, Ago, Agu;
    uint64_t Aka, Ake, Aki, Ako, Aku;
    uint64_t Ama, Ame, Ami, Amo, Amu;
    uint64_t Asa, Ase, Asi, Aso, Asu;
    uint64_t BCa, BCe, BCi, BCo, BCu;
    uint64_t Da, De, Di, Do, Du;
    uint64_t Eba, Ebe, Ebi, Ebo, Ebu;
    uint64_t Ega, Ege, Egi, Ego, Egu;
    uint64_t Eka, Eke, Eki, Eko, Eku;
    uint64_t Ema, Eme, Emi, Emo, Emu;
    uint64_t Esa, Ese, Esi, Eso, Esu;
```

```
    BCe = ROL(Age, 44);
    Aki ^= Di;
    BCi = ROL(Aki, 43);
    Amo ^= Do;
    BCo = ROL(Amo, 21);
    Asu ^= Du;
    BCu = ROL(Asu, 14);
    Eba = BCa ^ ((~BCe) & BCi);
    Eba ^= (uint64_t)KeccakF_RoundConstants[round];
    Ebe = BCe ^ ((~BCi) & BCo);
```

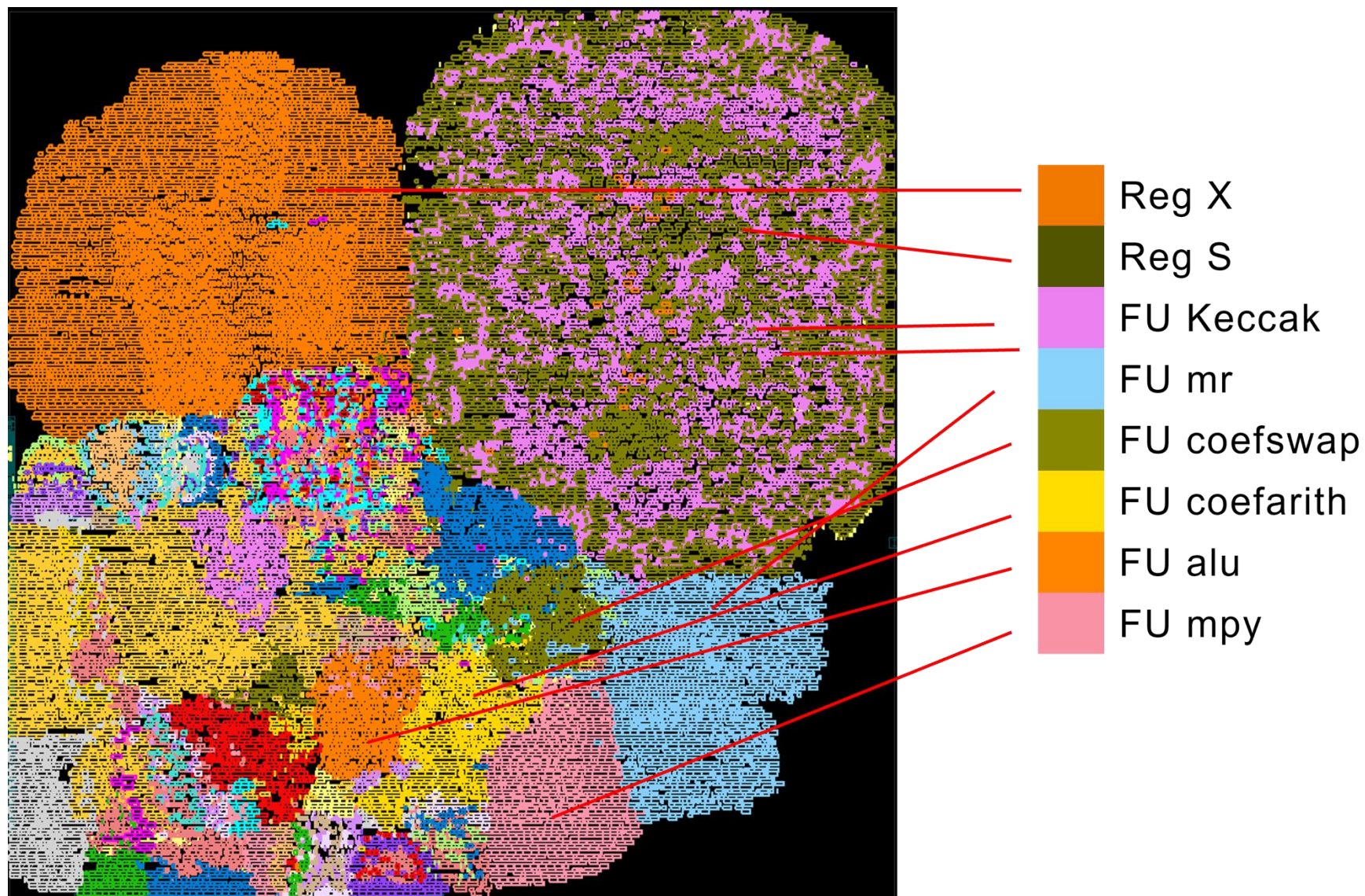


# Architectural Exploration: Results

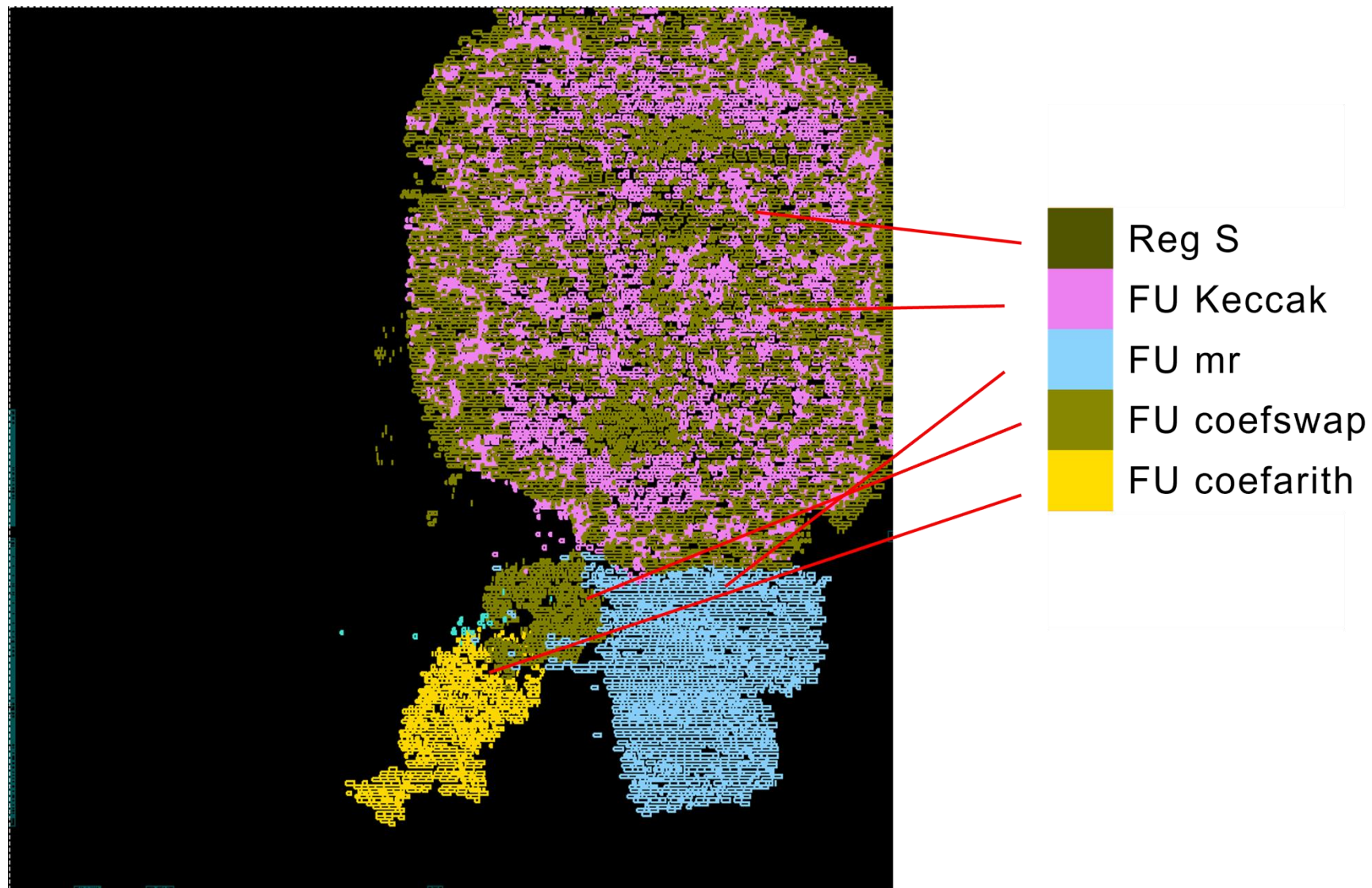




# Tsec ASIP: Layout



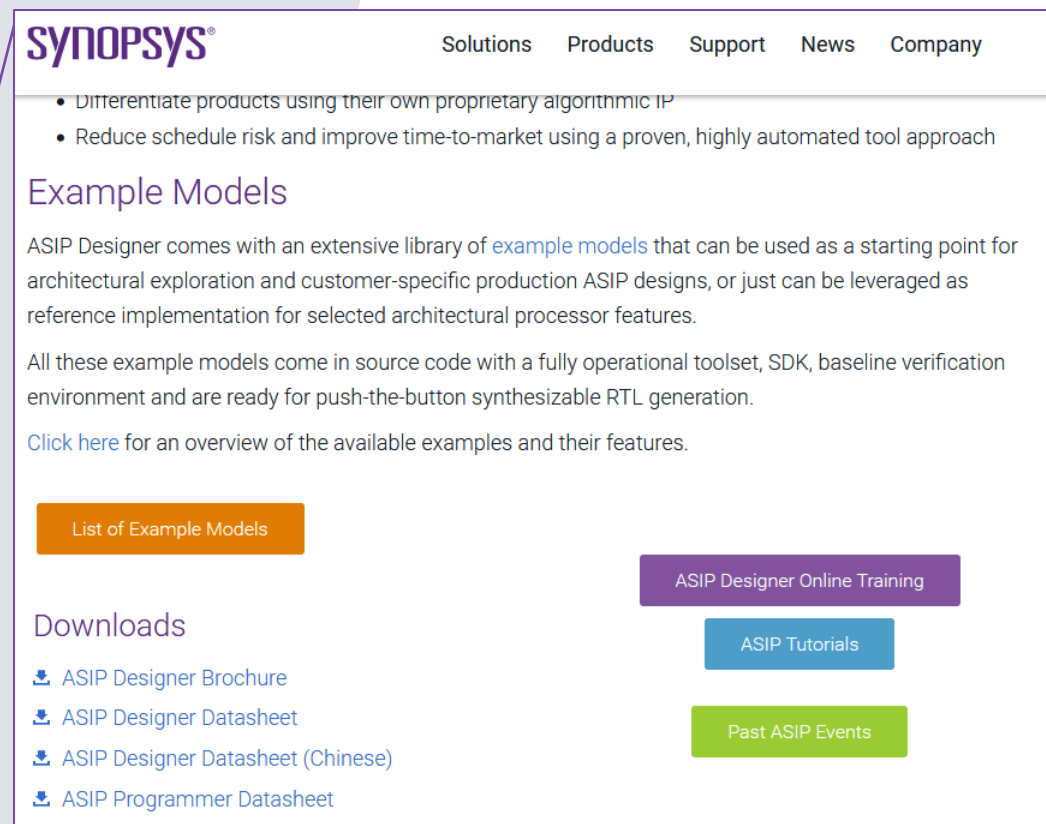
# Tsec ASIP: Layout (Kyber-specific units only)





# Summary

- Trends:
  - ISA customization & extensibility drive RISC-V adoption
  - RISC-V baseline ISA offers extensibility and existing ecosystem
  - ASIPs address the optimal solution to maintain programmability without compromising PPA, for applications with daunting throughput and latency requirements
  - resulting in demand for ASIPs based on RISC-V
- ASIP Designer is a unique tool suite
  - ASIP architectural exploration with Compiler-in-the-loop and Synthesis-in-the-loop
  - Automatic generation of an efficient SDK
  - Automatic generation of efficient RTL
- ASIP Designer supports RISC-V extension flow
  - rich set of ready-to-go RISC-V example models to start from
  - easy design flow for simple data-path extensions
  - unlimited large-scale extensions
  - Case Study: Tsec example model -> shipped with release
- More Material: Visit [www.synopsys.com/asip](https://www.synopsys.com/asip)



The screenshot shows the Synopsys ASIP Designer website. The header includes the Synopsys logo and navigation links for Solutions, Products, Support, News, and Company. The main content area features a list of bullet points: 'Differentiate products using their own proprietary algorithmic IP' and 'Reduce schedule risk and improve time-to-market using a proven, highly automated tool approach'. Below this is a section titled 'Example Models' which describes the ASIP Designer's library of example models and their use in architectural exploration and production designs. It mentions that these models come with source code, a fully operational toolset, SDK, baseline verification environment, and are ready for push-the-button synthesizable RTL generation. A link 'Click here' is provided for an overview of the available examples and their features. There are three buttons: 'List of Example Models' (orange), 'ASIP Designer Online Training' (purple), and 'ASIP Tutorials' (blue). Below the 'List of Example Models' button is a 'Downloads' section with four links: 'ASIP Designer Brochure', 'ASIP Designer Datasheet', 'ASIP Designer Datasheet (Chinese)', and 'ASIP Programmer Datasheet'. There is also a 'Past ASIP Events' button (green).

**SYNOPSYS®** Solutions Products Support News Company

- Differentiate products using their own proprietary algorithmic IP
- Reduce schedule risk and improve time-to-market using a proven, highly automated tool approach

### Example Models

ASIP Designer comes with an extensive library of [example models](#) that can be used as a starting point for architectural exploration and customer-specific production ASIP designs, or just can be leveraged as reference implementation for selected architectural processor features.

All these example models come in source code with a fully operational toolset, SDK, baseline verification environment and are ready for push-the-button synthesizable RTL generation.

[Click here](#) for an overview of the available examples and their features.

[List of Example Models](#)

### Downloads

- [ASIP Designer Brochure](#)
- [ASIP Designer Datasheet](#)
- [ASIP Designer Datasheet \(Chinese\)](#)
- [ASIP Programmer Datasheet](#)

[ASIP Designer Online Training](#)

[ASIP Tutorials](#)

[Past ASIP Events](#)



# Thank You