ADVIS: A Scalable Formal Approach for Correctness-Assured Hardware/Software Co-Design

Jin Yang, Ph.D., Sr. Principal Engineer

Jeremy Casas, Zhenkun Yang

Intel Labs IDP Strategic CAD Labs

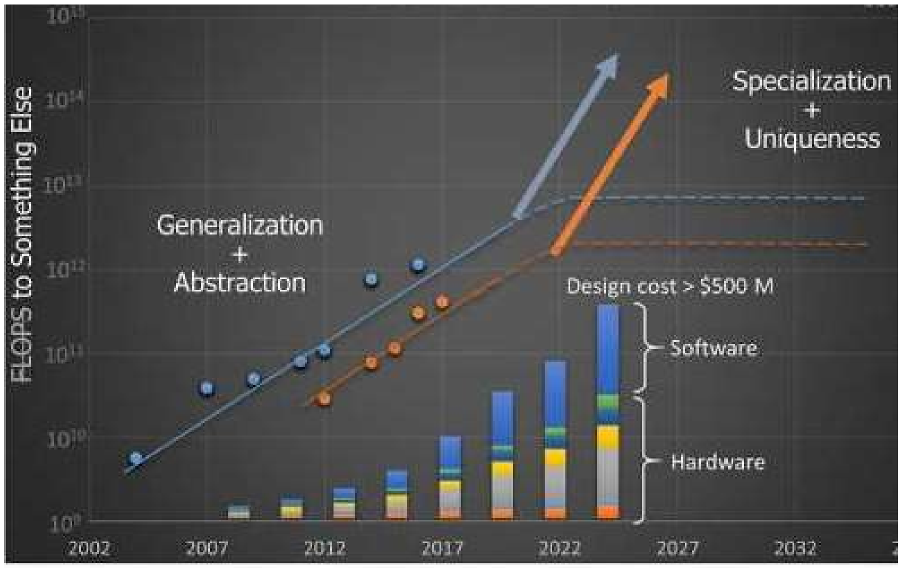
October 2023

intel®

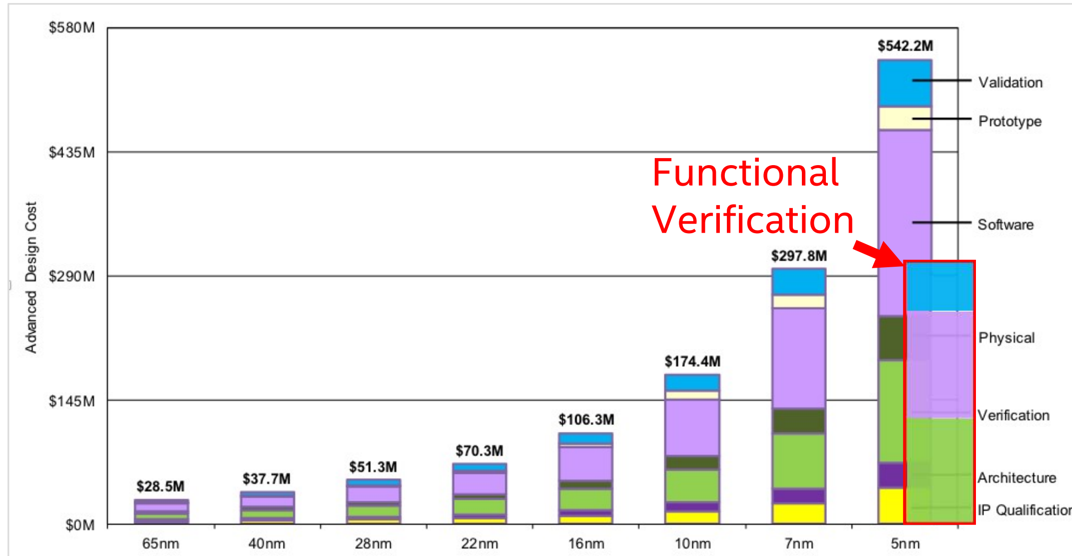
Outline

- Why Correctness Assurance Is Critical
- Scalable Correctness Assurance Approach with An Example
- Summary of Key Points

Moore's Inflection



DARPA 2018

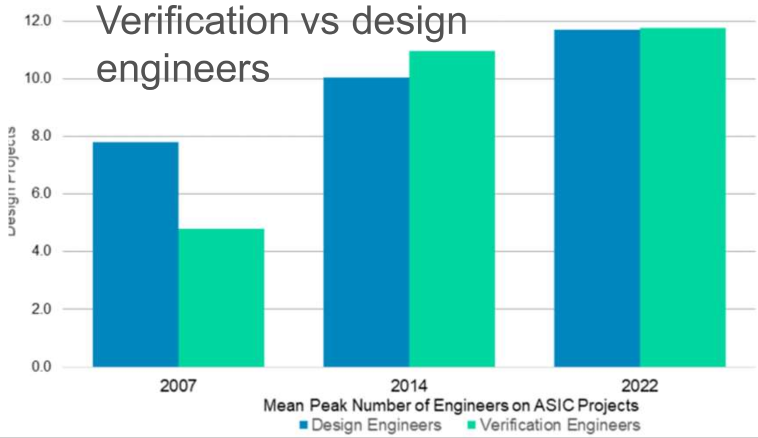


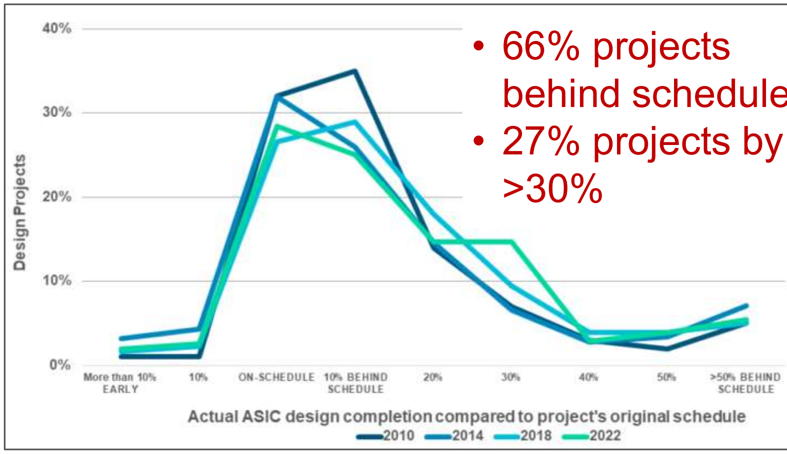
IBS 2018

- Exponential design cost impeding rapid pace of innovations
- Functional verification major cost (as high as 75%) w/ no solution in sight

Industry Trends (SIEMENS Wilson Research Group 2022)

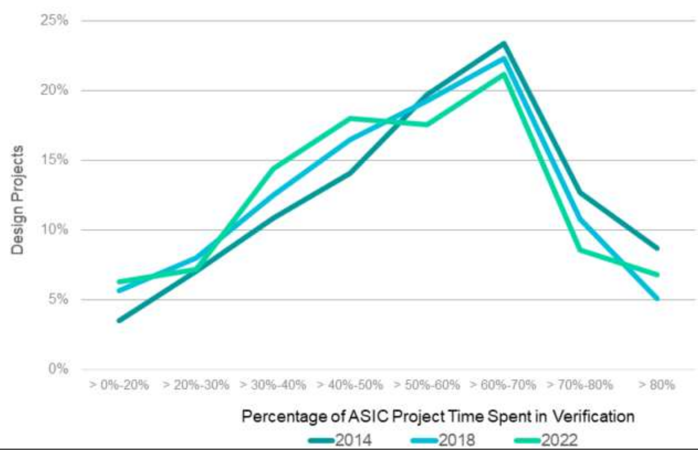
- 1:1 mean ratio on projects
- some projs 5:1
- 146% vs 50% in increase
- Designers 49% time on verif.

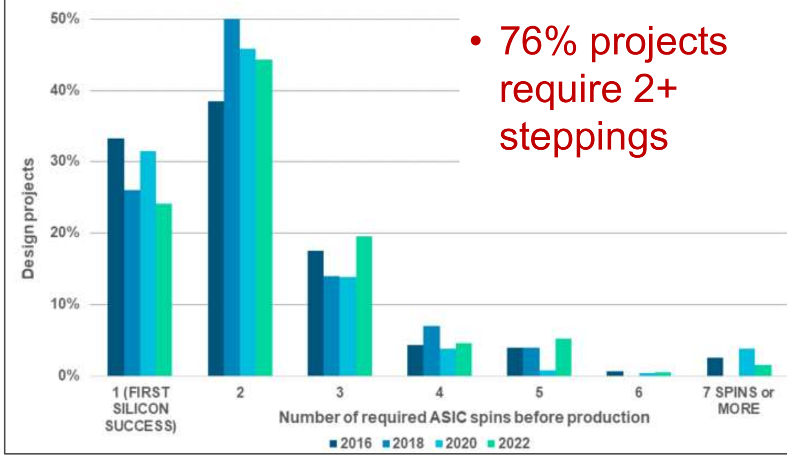




- 66% projects behind schedule
- 27% projects by >30%

- 50%-60% med. project time on verification
- 15% projects >70% time





- 76% projects require 2+ steppings

Computer Bugs Have Real Consequences

Sandy Bridge Bug 2X Costly as Pentium Math Bug

tom's HARDWARE
THE AUTHORITY ON TECH

By Douglas Perry published January 31, 2011

The FDIV bug was a catastrophic design flaw in the original Pentium (P5) … ended up paying about $475 million in total. … … more than 3000 engineers … testing new chip designs usually for at least 9 months before a chip is … released. … … With a $700 million bill and the stock market fallout still to be seen, Intel may be looking into ways on how to improve its quality control system

Intel's Sapphire Rapids Had 500 Bugs, Launch Window Moves Further

tom's HARDWARE
THE AUTHORITY ON TECH

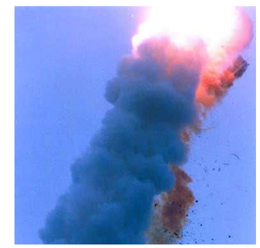
By Anton Shilov published August 01, 2022

Someone call pest control.

Intel has delayed the release of its 4th Generation Xeon Scalable "Sapphire Rapids" processor for a number of times … According to Igor's Lab, Sapphire Rapids had about 500 bugs that required the company 12 steppings to fix them.

Financial impact, safety, security, human lives

- 1985 - 1987 – Therac-25 medical accelerator malfunctions and delivers lethal radiation doses, … at least 5 patients die; others are seriously injured
- 1991 - Patriot Missile failed to intercept a missile due to miscalculation, … leading to 28 US soldier' deaths
- 1996 – Ariane 5 explored immediately after the lunch due to an error in the onboard computer, costing $7b dollars
- 2012 – Knight Capital lost $0.5b dollars in half an hour due to a computer glitch … stock plunged by ~70% in two days
- 2015 – A SW vulnerability in Boeing's 787 Dreamliner has the potential to cause pilots to lose control of the aircraft.
- 2019 – a debugging mode had an undocumented feature on Intel PCHs, …. which made the mode accessible with a normal motherboard possibly leading to a vulnerability
- 2022 – Intel's SGX has been breached yet again, … spills users' most sensitive secrets in seconds from SGX enclaves.
- 07/2021 – 10/2022, DoT reported 605 crashes involving vehicles equipped with advanced driver assistance systems
- … …



- Correctness assurance is increasingly critical in a ubiquitous computing world

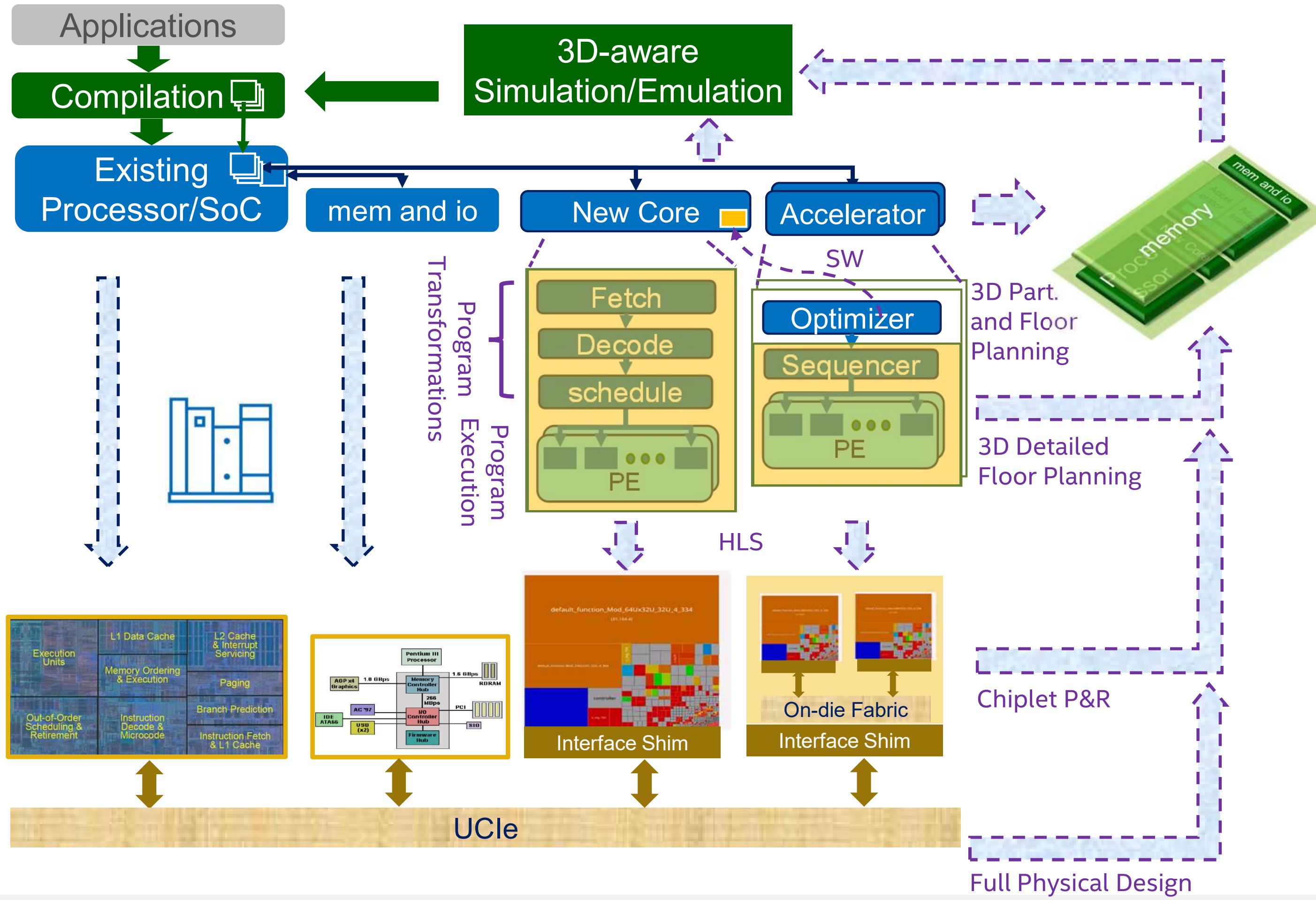
Design and Verification Paradigm Shift

- Raise levels of SW/HW co-design abstraction to applications and refine down to implementation
  - Reduce design complexity and improve design efficiency
- Incremental verification tightly integrated with principled correct-by-composition design
  - Ensure correctness as first-class requirement

End-to-End Design Flow w/ Correctness Assurance

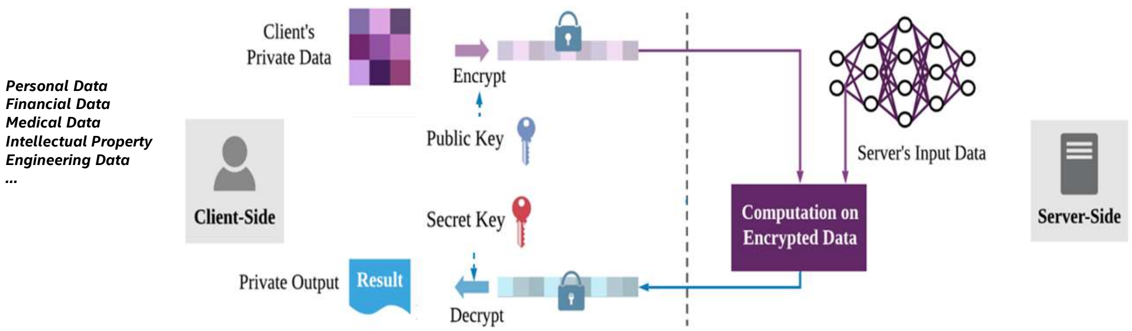
- Starting with applications
- Architecture spec, design and exploration with correctness assurance
- Incremental, composable micro-architecture spec, design and exploration with correctness assurance
- Seamless HW/SW co-design and exploration, using HLS to generate HW
- Verified parametric design libraries and correct-by-composition integration

- >>10x productivity improvement
- >>10x functional bug reduction

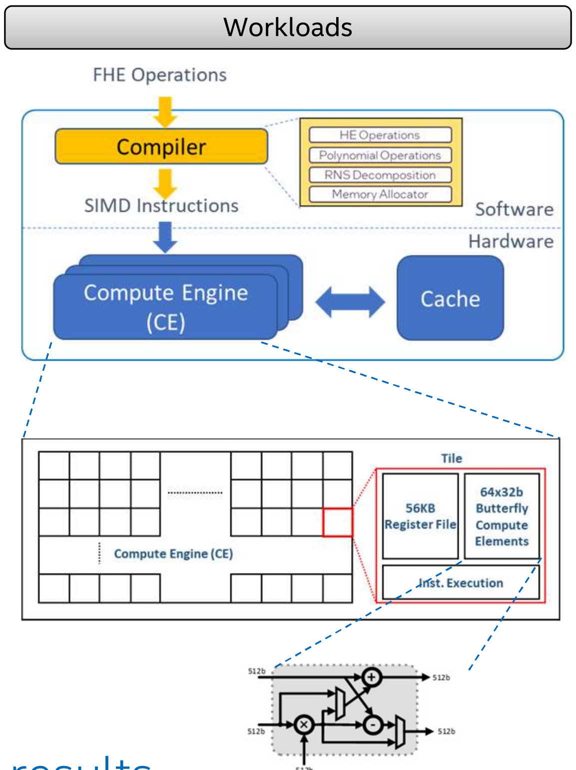


Opportunity: DARPA DPRIVE

A FHE HW accelerator co-designed and optimized across the full stack … that can be formally verified within minutes to hours … …



FHE Accelerator Design



Shadowed Research

- Paper and pencil conceptual design
- ADVIS research prototype development
- PoC of key ADVIS ideas on the accelerator
- Show impact on current design approach even with partial results

Conceptual SW/HW Co-Design: Application API

FHE workloads

Decision 1: A large word size accelerator to reduce FHE computational runtime

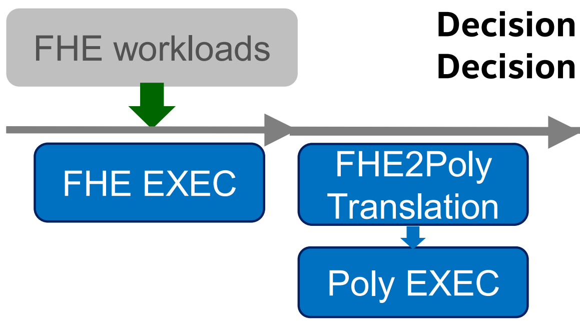
FHE EXEC

FHE Stream

```
FHE_EXEC(in stream)
 while stream:
  instr = stream.get()
   if (instr is “FHE_add q (x0,x1) (y0,y1) (z0,z1)”:
    (z0,z1) = (poly_add(q,x0,y0), poly_add(q,x1,y1))
   elif instr is “FHE_mul q (x0,x1) (y0,y1) (z0,z1,z2)”:
    (z0,z1,z2) =
     (poly_mul(q,x0,y0),
      poly_add(q,poly_mul(q,x0,y1),poly_mul(q,x1,y0)),
      poly_mul(q,x1,y1))
    ...
```

Conceptual SW/HW Co-Design: FHE to Poly Translation

Decision 1: A large word size accelerator to reduce FHE computational runtime
Decision 2: localize data to reduce latency, and accelerate polynomial operations



FHE stream

```
FHE2Poly_TRANS(in FHE_stream, out poly_stream)
 while FHE_stream:
  instr = FHE_stream.get()
  if (instr is “FHE_add q (x0,x1) (y0,y1) (z0,z1)”):
   poly_stream.send( “mload (x0,x1) (c_x0,c_x1)” )
   poly_stream.send( “mload (y0,y1) (c_y0,c_y1)” )
   poly_stream.send( “poly_add q c_x0 c_y0 c_z0” )
   poly_stream.send( “poly_add q c_x1 c_y1 c_z1” )
   poly_stream.send( “mstore (c_y0,c_y1) (z0, 1)” )
  ... ...
```

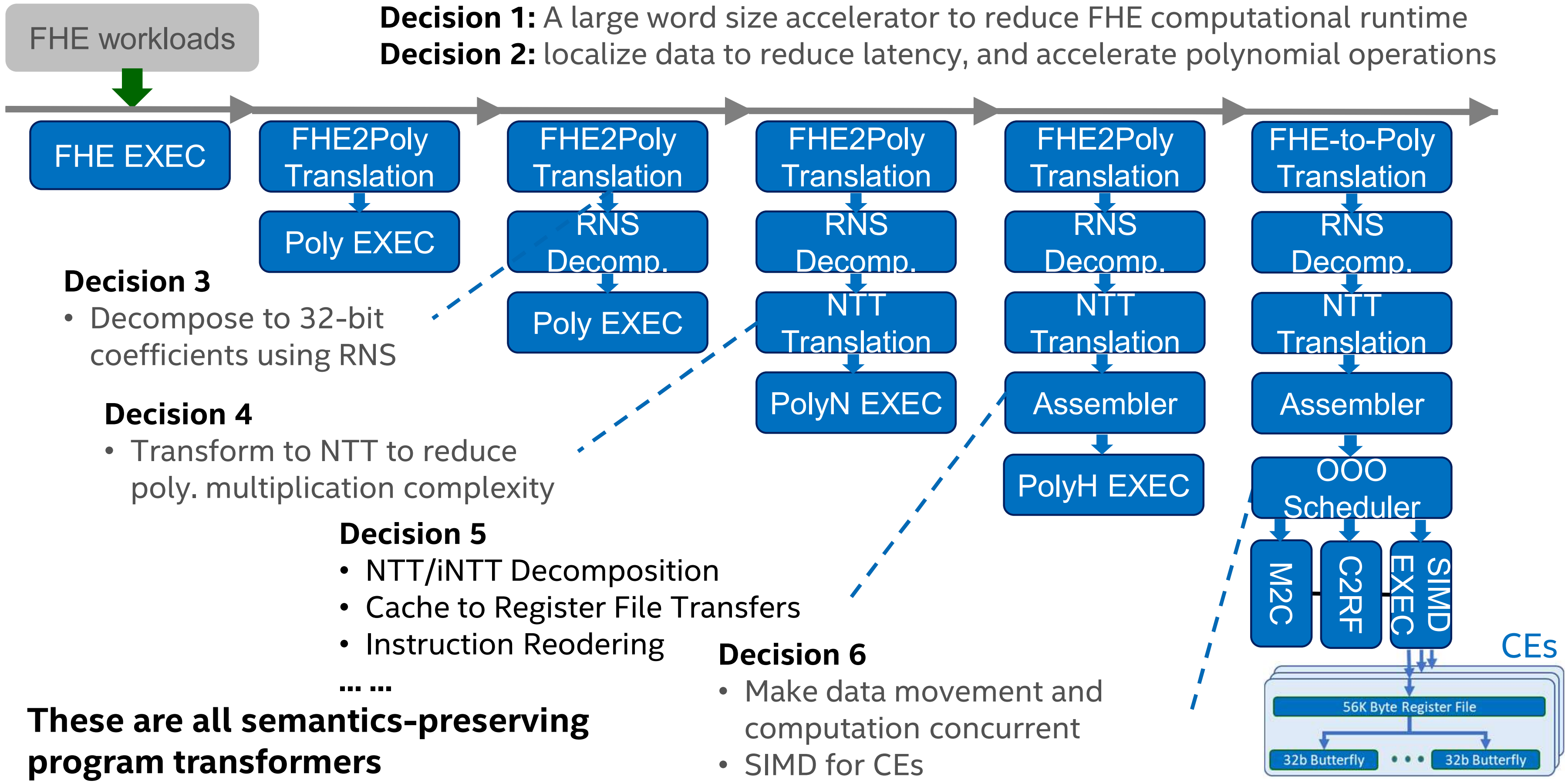
poly stream

```
Poly_EXEC(in poly_stream)
 while poly_stream:
  if (instr is “mload (x0,x1),(c_x0,c_x1)”):
   c_x0 = x0;
   c_x1 = x1;
  elif (instr is “poly_add q c0 c1 c2”):
   c2 = poly_add(q, c1, c2)
  elif (instr is “poly_mul q c0 c1 c2”):
   c2 = poly_mul(q, c1, c2)
  ... ...
```

Semantics-preserving program transformer

Conceptual SW/HW Co-Design: End-to-End

Decision 1: A large word size accelerator to reduce FHE computational runtime
Decision 2: localize data to reduce latency, and accelerate polynomial operations



Decision 3
- Decompose to 32-bit coefficients using RNS

Decision 4
- Transform to NTT to reduce poly. multiplication complexity

Decision 5
- NTT/iNTT Decomposition
- Cache to Register File Transfers
- Instruction Reodering

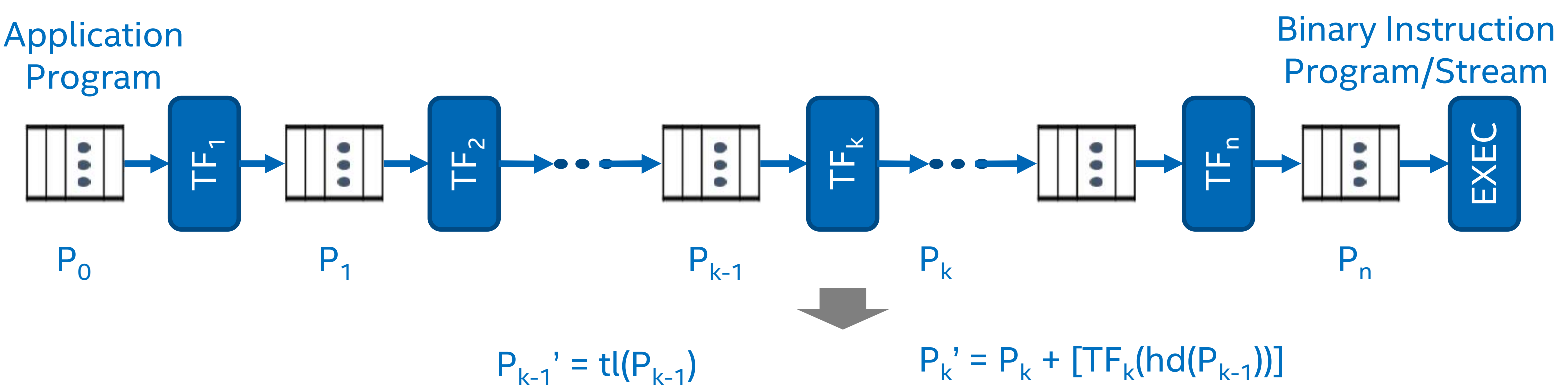
Decision 6
- Make data movement and computation concurrent
- SIMD for CEs

... ...

These are all semantics-preserving program transformers

Mathematical Foundation: Program Transformation

- Program: $P = P_0 \; @ \; P_1 \dots P_{k-1} \; @ \; P_k \dots @ \; P_n$

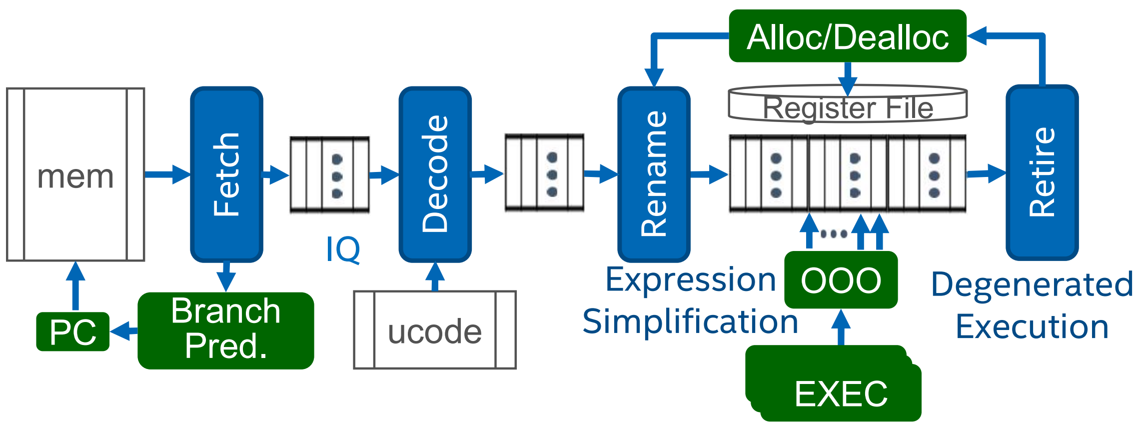


- Program Transformer: TF_k is a transformer, if $\forall$ P and $\forall$ state s, run(P, s) $\equiv$ run(P', s)
- **Theorem:** Fetch, Branch Prediction, Data Localization (cache, scratch buffers, RFs), OOO Scheduler, … are program transformers

Mathematical Foundation: Transformer Composition

Theorem: If TF_1 and TF_2 are transformers, then so are the following

- non-deterministic selection $TF = TF_1 \mid TF_2$
- sequential composition $TF = TF_1 ; TF_2$
- for any condition c, $TF = \text{if } (c)\ TF_1$ or $TF = \text{if } (c)\ TF_1 \text{ else } TF_2$
- parallel composition $TF = TF_1 \mid\mid TF_2$ if $range(TF_1) \cap range(TF_2) = \{\}$

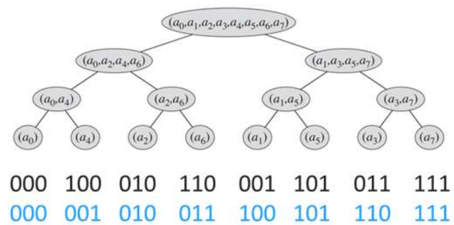



```
uProc():
  Repeat
    Fetch() n times   ||
    Decode() m times  ||
    Rename() k times  ||
    OOO()    p times  ||
    Retire() q times
```

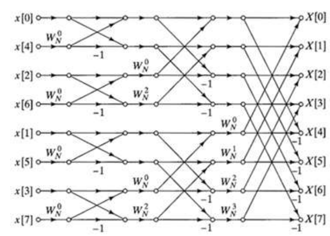
Correctness-Assured Algorithmic Refinement

- NTT Refinement in Assembler

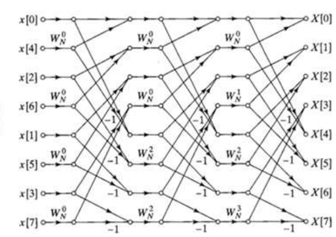
Recursive NTT → Iterative NTT → Constant Geometry NTT



≡



≡



- Montgomery Reduction in Butterfly

Baseline Montgomery → Multi-word Montgomery → Optimized Montgomery

```
method REDC(
    R: nat, R': nat, // R: auxiliary modulus, R' = (R⁻¹)
    N: nat, N': nat, // N: the modulus, N·N' ≡ -1 (mod R)
    T: nat)          // Input to be reduced
  returns (S: nat)   // Output after the reduction
  requires 0 < R ∧ 0 < R' ∧ 0 < N ∧ 0 < N' ∧ N < R ∧ T < N * R
  requires IsCoPrime(R, N)
  requires R * R' % N == 1 % N
  requires N * N' % R == -1 % R

  ensures S < N
  ensures S % N == T * R' % N
{
  var m: nat := T % R * N' % R;
  var t: nat := (T + m * N) / R;
  S := if t ≥ N then t - N else t;
}
```

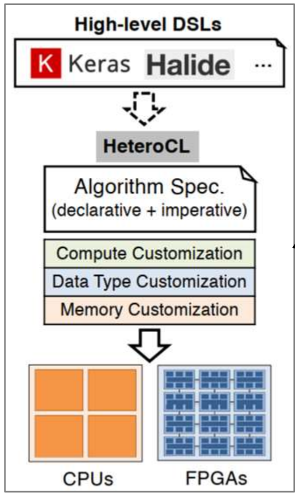
≡

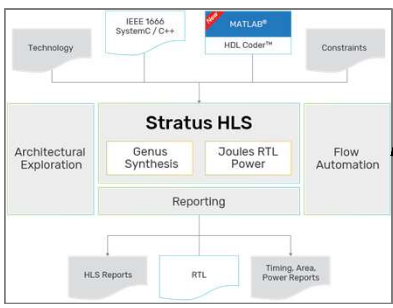
```
method REDC_Mult(
  R: nat, R': nat,    // R: auxiliary modulus, R' = (R⁻¹)
  N: nat, N': nat, n: nat, // modulus, N·N' ≡ -1 (mod R)
  T: nat, BASE: uint) // Input with BASE to be reduced
  returns (S: nat)    // Output after the reduction

  requires n > 0 ∧ N > 0 ∧ N' > 0 ∧ R' > 0 ∧ T > 0
  requires 0 < N < Pow(BASE, n)
  requires R == Pow(BASE, n)
  requires R * R' % N == 1 % N
  requires IsCoPrime(N, BASE)
  requires IsCoPrime(R, N)
  requires N * N' % BASE == -1 % BASE
  requires T < N * R

  ensures S < N
  ensures S % N == T * R' % N
{
  var a: nat := T;
  for i: nat := 0 to n {
    var ui: uint := a % BASE * N' % BASE;
    a := a + ui * N;
    a := a / BASE();
  }
  S := if a ≥ N then a - N else a;
}
```

≡

```
method REDC_Mult_Opt
  R: nat, R': nat,    // R: auxiliary modulus, R' = (R⁻¹)
  N: nat, N': nat, n: nat, // modulus, N·N' ≡ -1 (mod R)
  T: nat, BASE: uint) // Input with BASE to be reduced
  returns (S: nat)    // Output after the reduction

  requires n > 0 ∧ N > 0 ∧ N' > 0 ∧ R' > 0 ∧ T > 0
  requires 0 < N < Pow(BASE, n)
  requires R == Pow(BASE, n)
  requires R * R' % N == 1 % N
  requires IsCoPrime(N, BASE)
  requires IsCoPrime(R, N)
  requires N * N' % BASE() == -1 % BASE
  requires T < N * R

  // additional preconditions for the optimizations
  requires ∃ k: nat · N == k * BASE + 1

  ensures S < N
  ensures S % N == T * R' % N
{
  var k: nat :| k * BASE() + 1 == N;
  var a: nat := T;
  for i: nat := 0 to n {
    var ah, a0 := a / BASE(), a % BASE();
    var carry := if a0 == 0 then 0 else 1;
    a := ah + TwosCpl(a0) * k + carry;
  }
  S := if a ≥ N then a - N else a;
}
```

Research Prototype

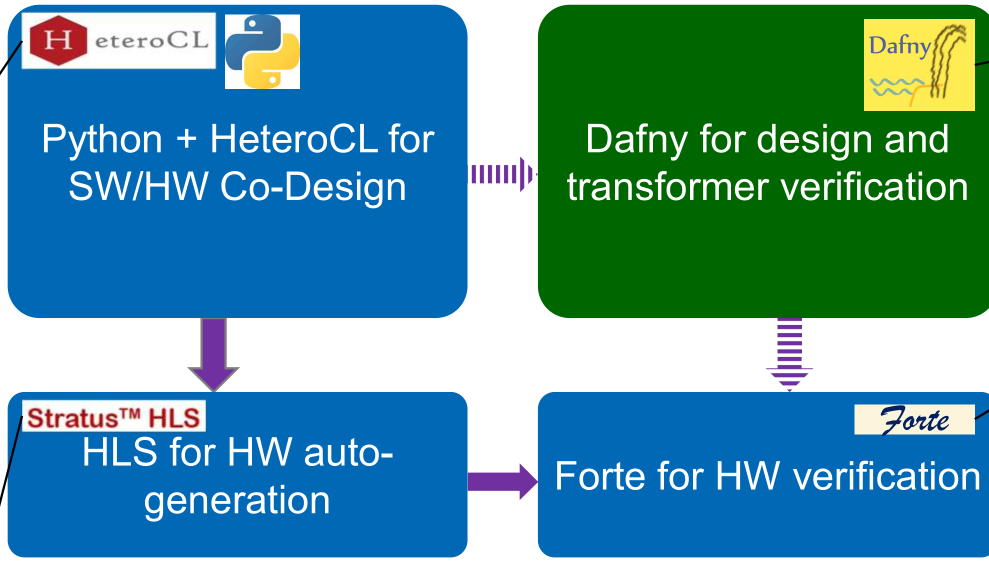


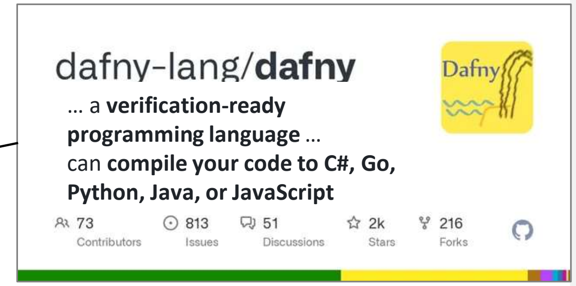


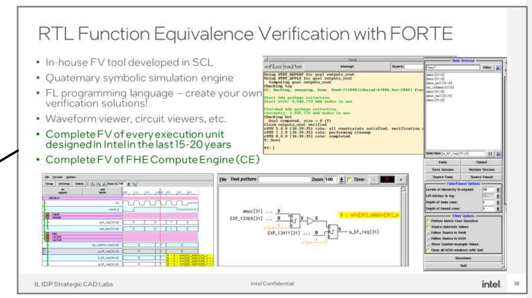
Cadence

Cornell + UCLA

MSR + AWS



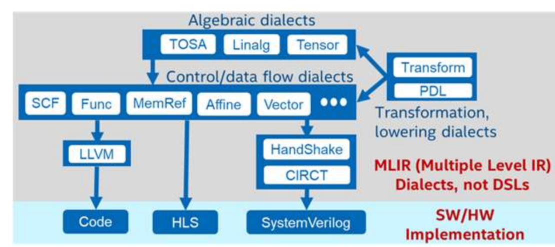




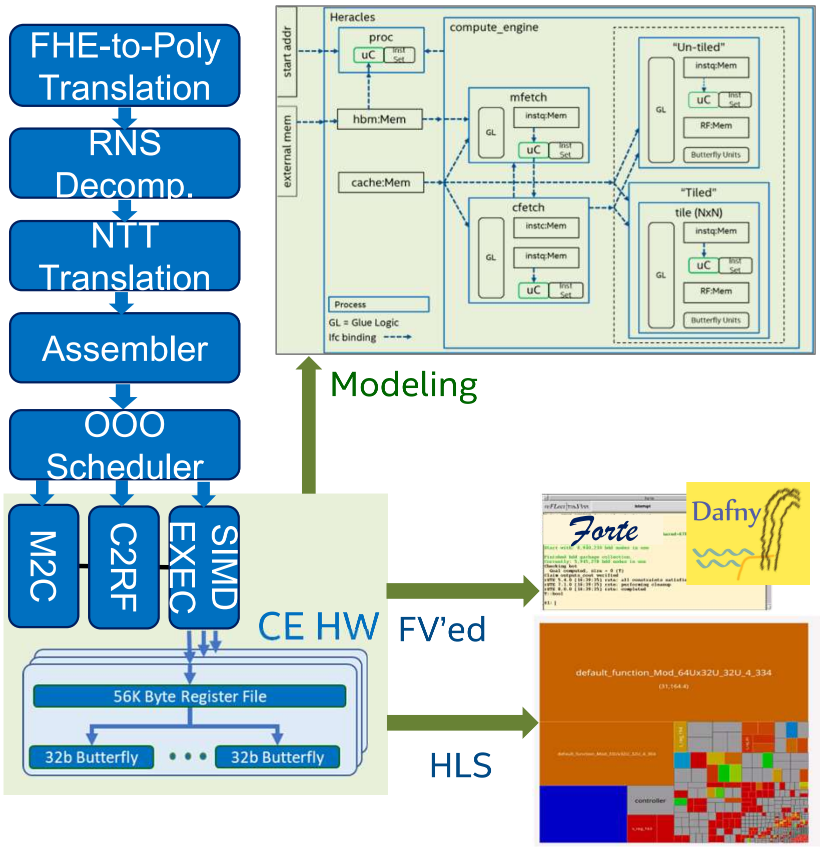
Intel

Moving towards Python/MLIR based system

- HeteroCL (Cornell, ISRA), ScaleHLS (UIUC)
- New dialects for specs and transformers (UIUC, SRC)
- EGG (UW, CAD SRS), CDAG based FEV (CSU, ISRA) and KLEE based FEV (PSU, ISRA)



FHE Accelerator Proof-of-Concept Results



- Micro-architecture modeling in Python + HeteroCL
  - 1 pers.-wk initial, < 1 pers.-day incremental, way ahead of RTL schedule
  - Only model to run workloads w/ cycle estimate within 4% of VCS
  - Caught functional failures and read-after-write hazards
  - Reference model for FV and accelerator emulation
- Manual CE RTL formally verified in Dafny + Forte
  - < 1 hr. runtime, TAT in a few days to weeks, 30+ issues found
- RTL generated with Cadence Stratus HLS
  - Monolithic model: 200 MHz ,100K nm^2 tile, ~25M gate
  - ~40 min initial run, ~5 min incr. run
- Ongoing and future work
  - Modeling and verification of SW/compiler transformers
  - Modular HLS and integration

Summary of Key Points

- Correctness must be a first principle
- Raise level of abstraction to SW, generating RTL
- Design for correctness with precisely defined intents
  - Transformation based design and composition
  - Language/IR for capturing intents
- Scalable integrated verification approach
  - Incremental intent verification
  - Correct-by-construction composition

References

- J. Yang, Z. Yang, J. Casas, S. Ray, “Correct-by-Construction Design of Custom Accelerator Microarchitectures”, IEEE Transactions on Computers
- J. Yang, Z. Yang, J. Casas, “A Scalable Formal Approach for Correctness-Assured Hardware Design”, invited paper, DAC 2023
- J. Casas, Z. Yang, W. Wang, J. Yang, A. Godbole, “Towards A Formally Verified Fully Homomorphic Encryption Compute Engine”, DAC 2023
- S. Ray, K. Hao, F. Xie, and J. Yang, “Formal Verification for High-Assurance Behavioral Synthesis”, ATVA 2009
- K. Hao, S. Ray, F. Xie, and J. Yang, “Optimizing Equivalence Checking for Behavioral Synthesis”, DATE 2012