Comparing Performance, Productivity and Scalability of the TILT Overlay Processor to OpenCL HLS

Rafat Rashid, J. Gregory Steffan, Vaughn Betz

Electrical and Computer Engineering

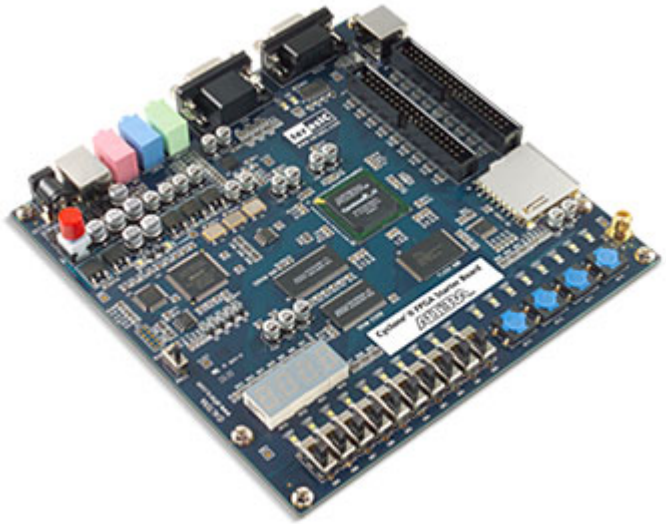
University of Toronto, Ontario, Canada

rafat.rashid@utoronto.ca, {steffan,vaughn}@eecg.toronto.edu

UNIVERSITY OF TORONTO

FPGA Programming

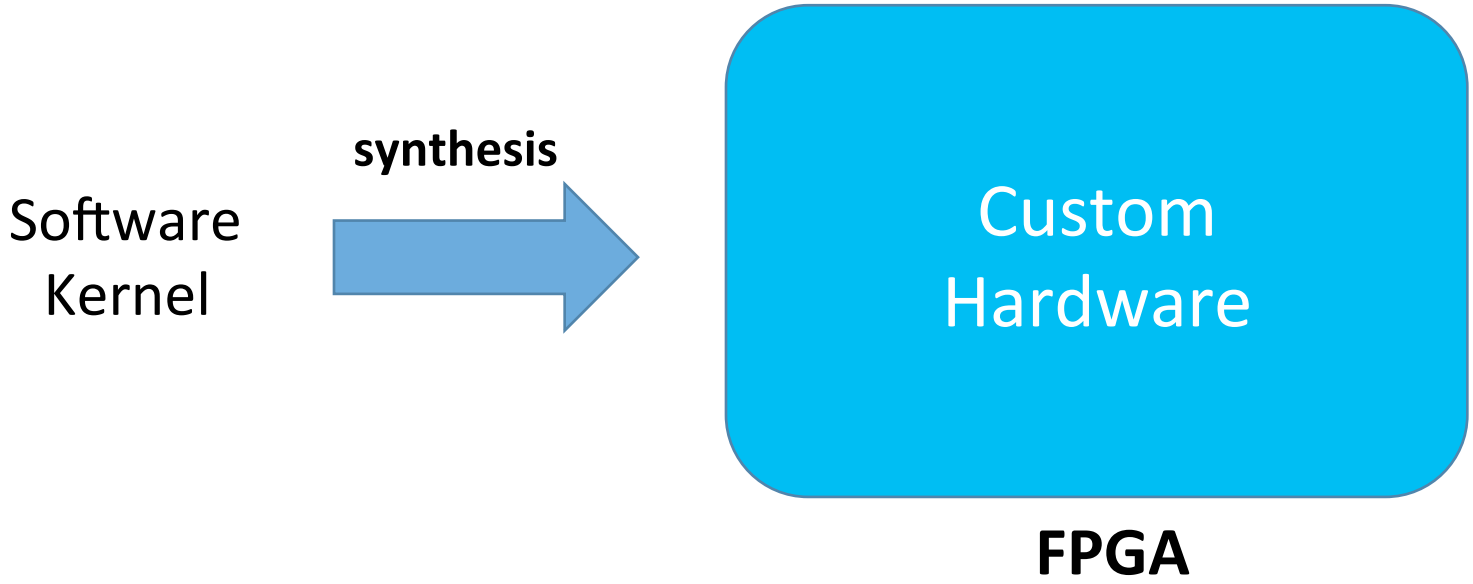
- Requires expert hardware designer
- Long hardware compile times
  - Up to a day for a large design



→ Options for programming with high-level languages?

Option 1: High-Level-Synthesis (HLS)

- Maps high-level languages to custom hardware
  - Eg. FCUDA, LegUp, Vivado, Altera OpenCL
- **OpenCL**: increasingly popular acceleration language

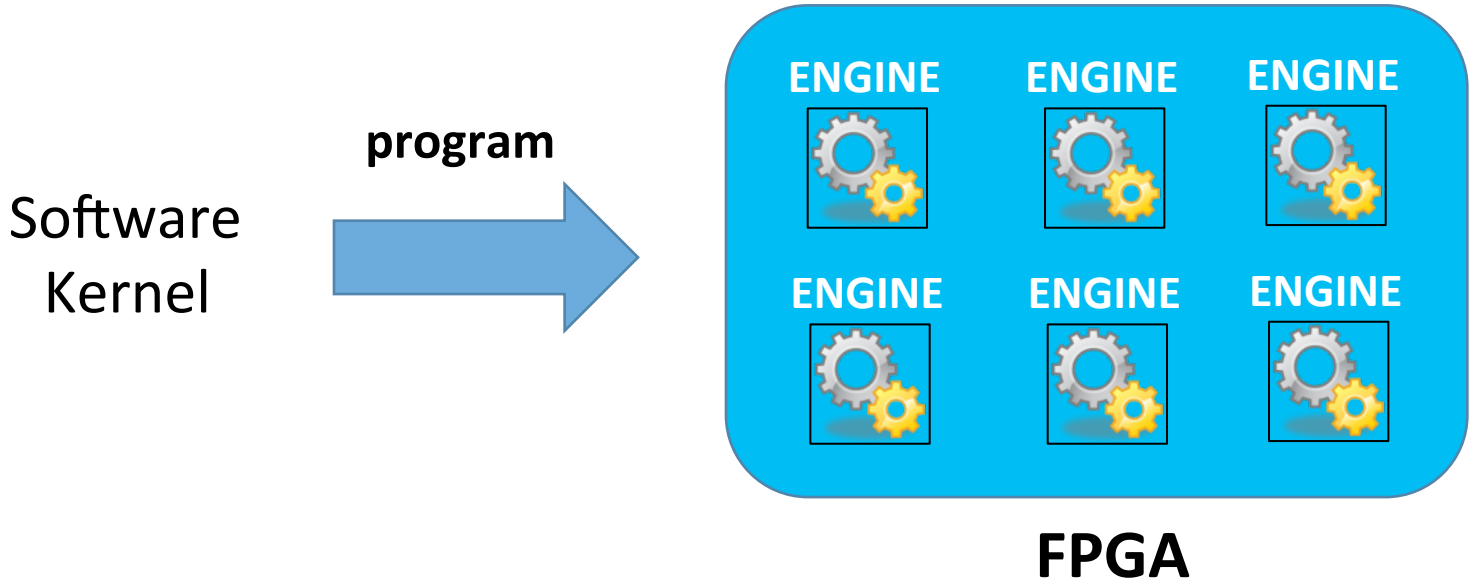


→ Suffers long compile times

Option 2 – Overlays

- Maps high-level languages to software-programmable hardware

➢ Quickly reprogrammed (vs. regenerating hardware)
➢ Versatile (multiple software functions per engine)



→ **Area overhead**

Our Focus

- Application-customized but software-programmable datapath
- High throughput-per-area (computationally dense)

➢ Assume threaded data parallelism
➢ Target floating-point applications

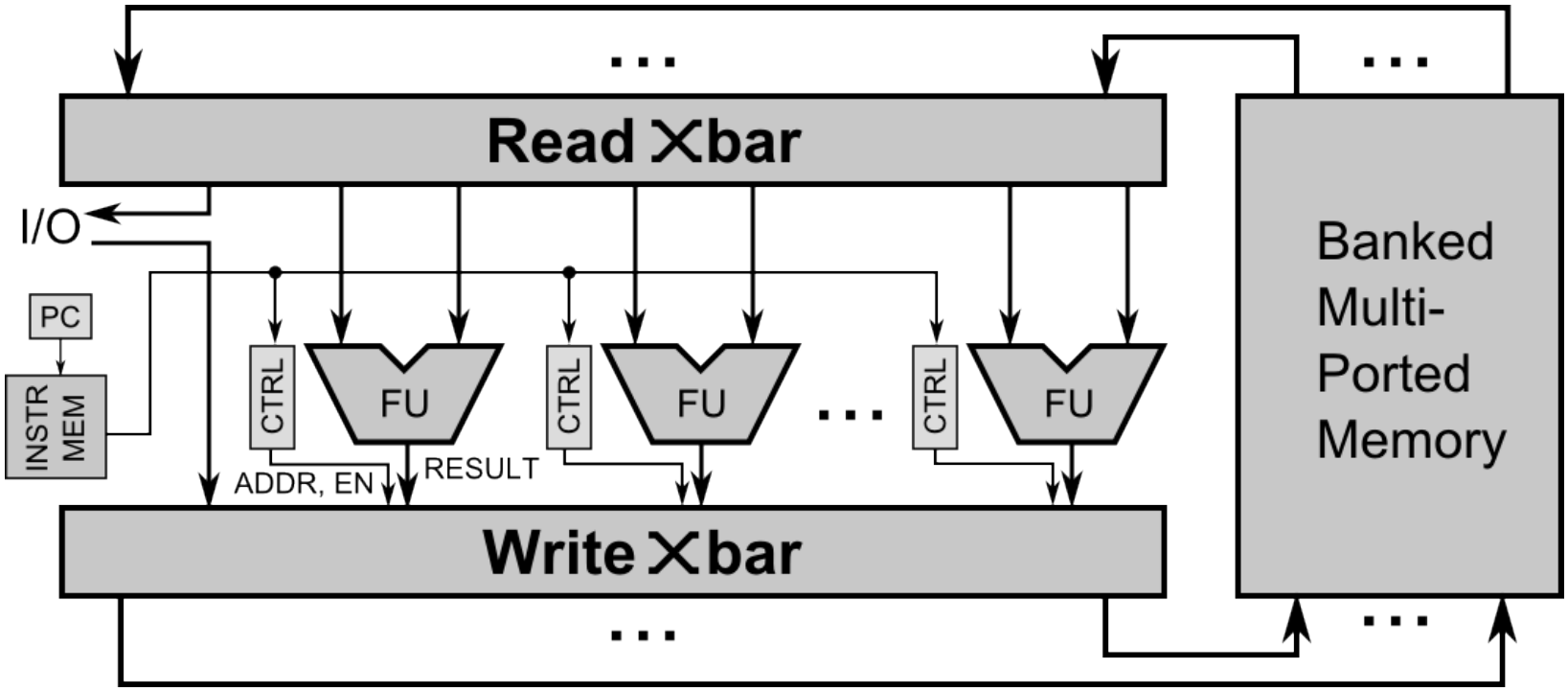
→ **Result:**

➢ **TILT Overlay Processor**

- Prior work by Ovtcharov and Tili

K. Ovtcharov, I. Tili, and G. Steffan, "TILT: A Multithreaded VLIW Soft Processor Family," FPL, Aug 2013.

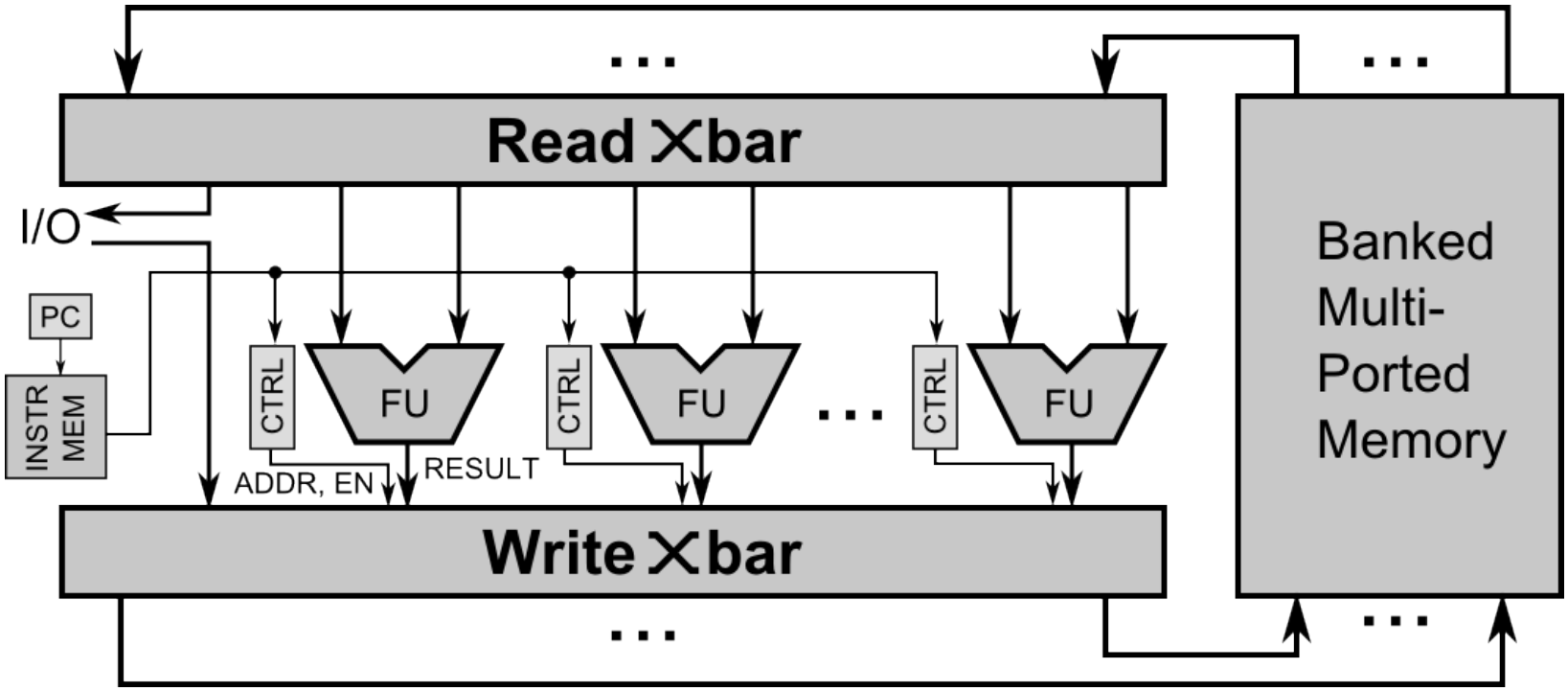
TILT – Overlay Processor



K. Ovtcharov, I. Tili, and G. Steffan, "TILT: A Multithreaded VLIW Soft Processor Family," FPL, Aug 2013.

TILT – Overlay Processor

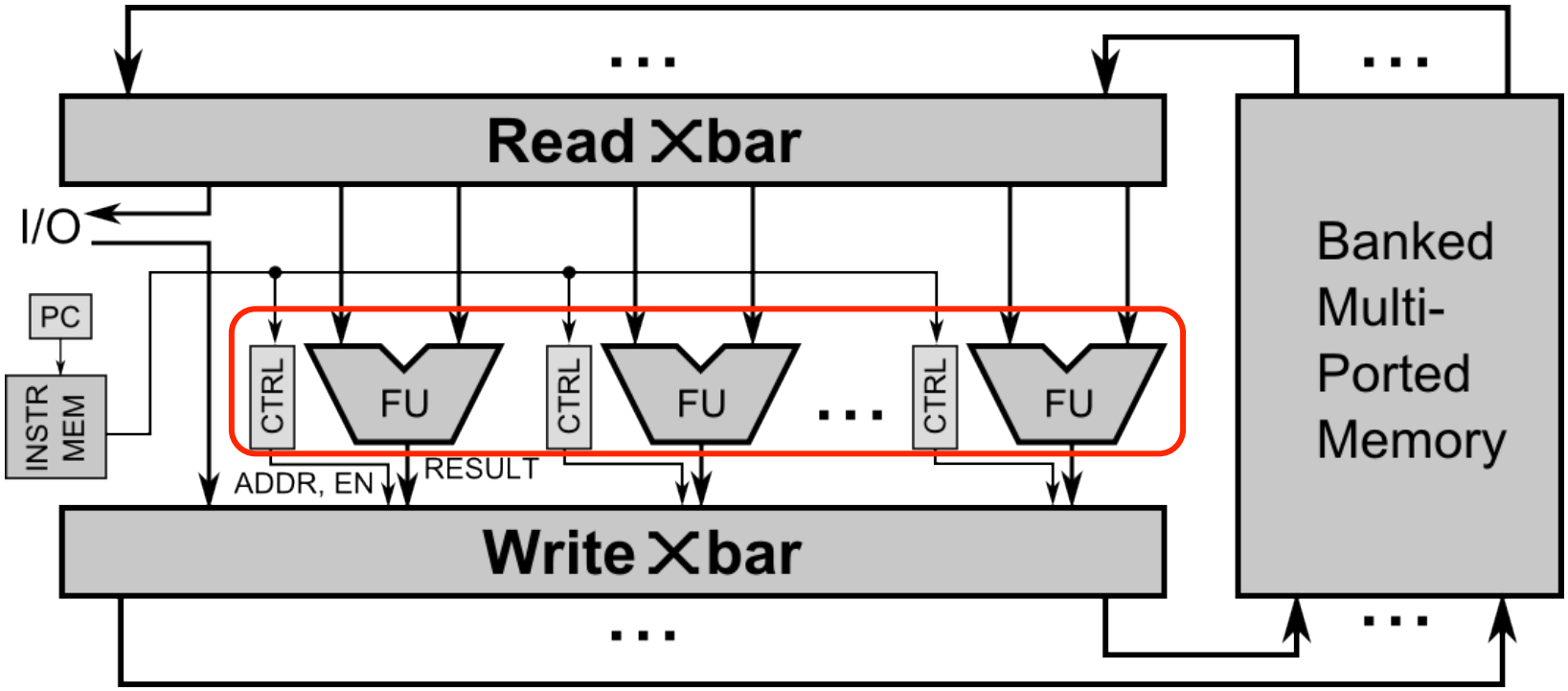
- Supports thread and instruction level parallelism
- Highly configurable
  - Customizable FU mix and memory architecture



K. Ovtcharov, I. Tili, and G. Steffan, "TILT: A Multithreaded VLIW Soft Processor Family," FPL, Aug 2013.

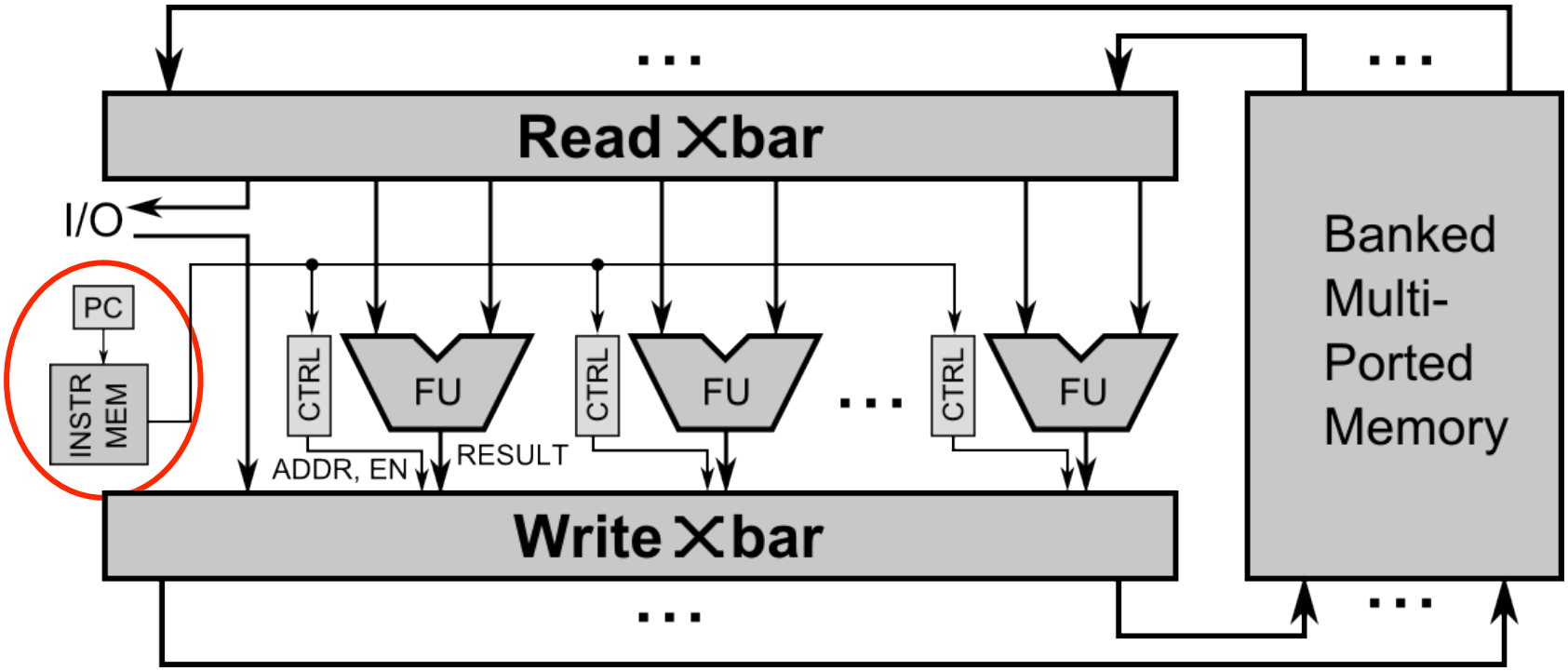
TILT – Overlay Processor

- Multiple floating-point FUs
  - Add/Sub, Multiply, Sqrt, Log, …
  - Deeply pipelined, variable latency: 1 to 28 cycles

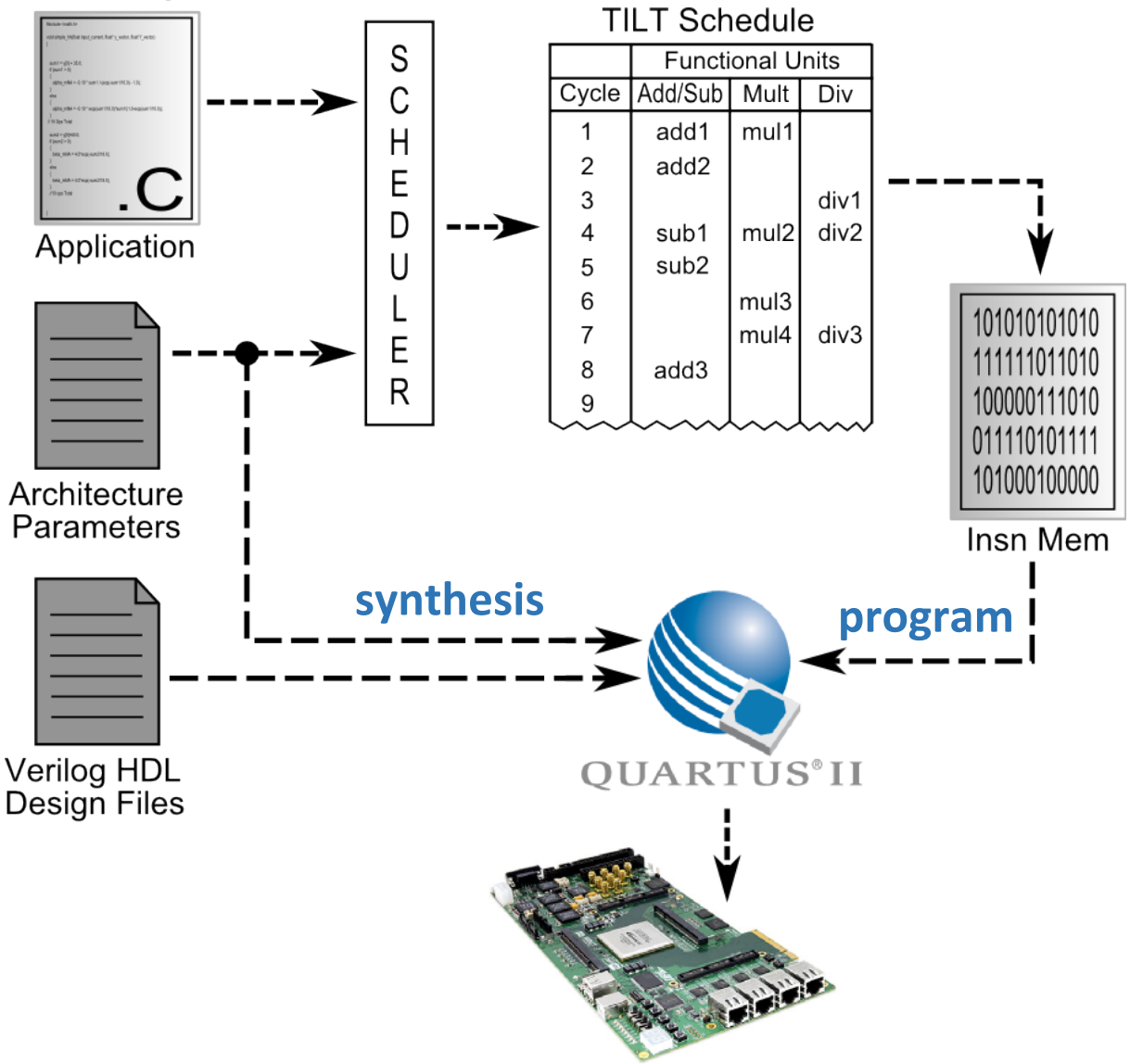


TILT – Overlay Processor

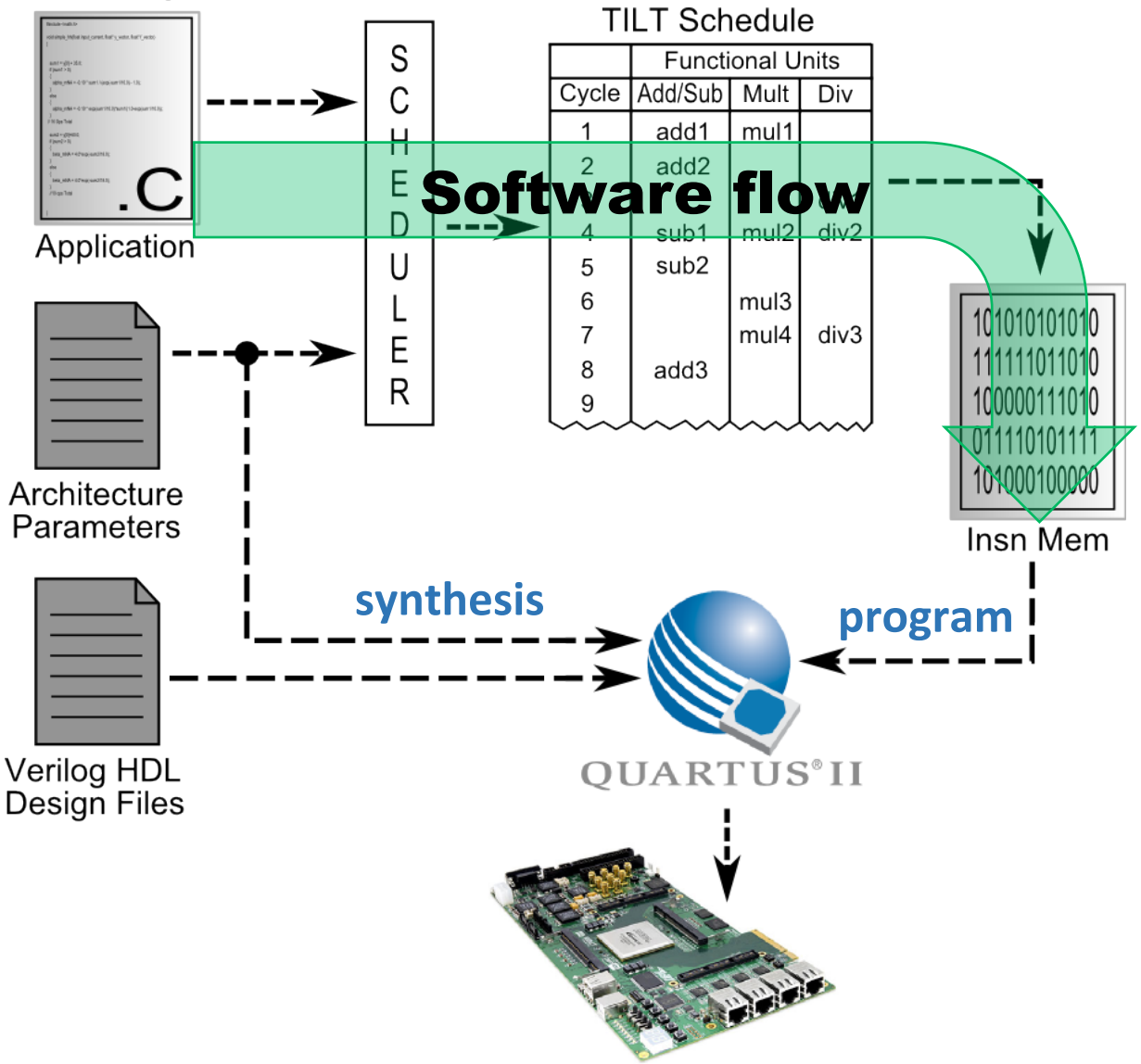
- Minimal hardware
  - Compiler produces dense (VLIW) instruction schedules (stored in a small on-chip memory)



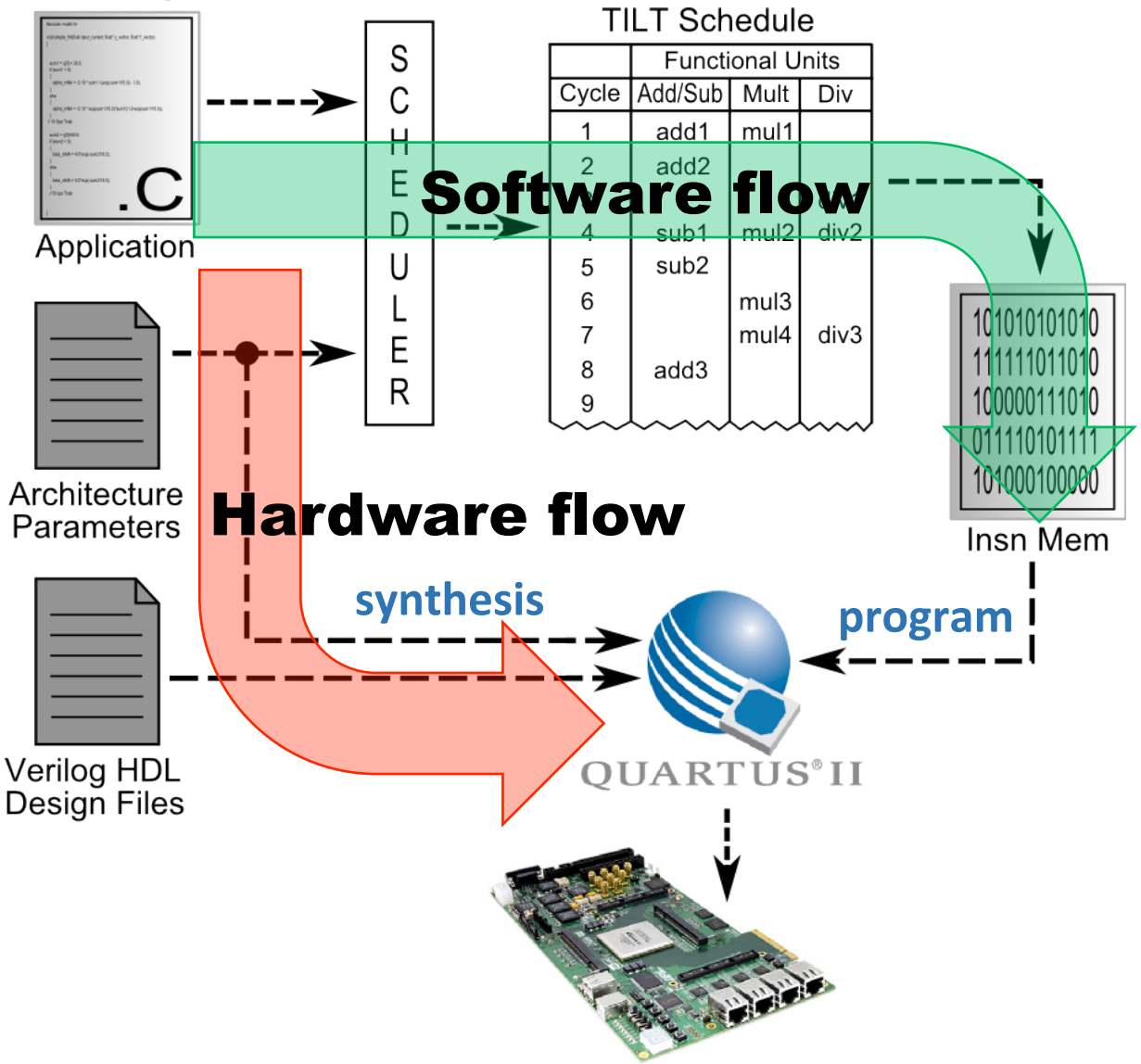
TILT – Compiler Flow



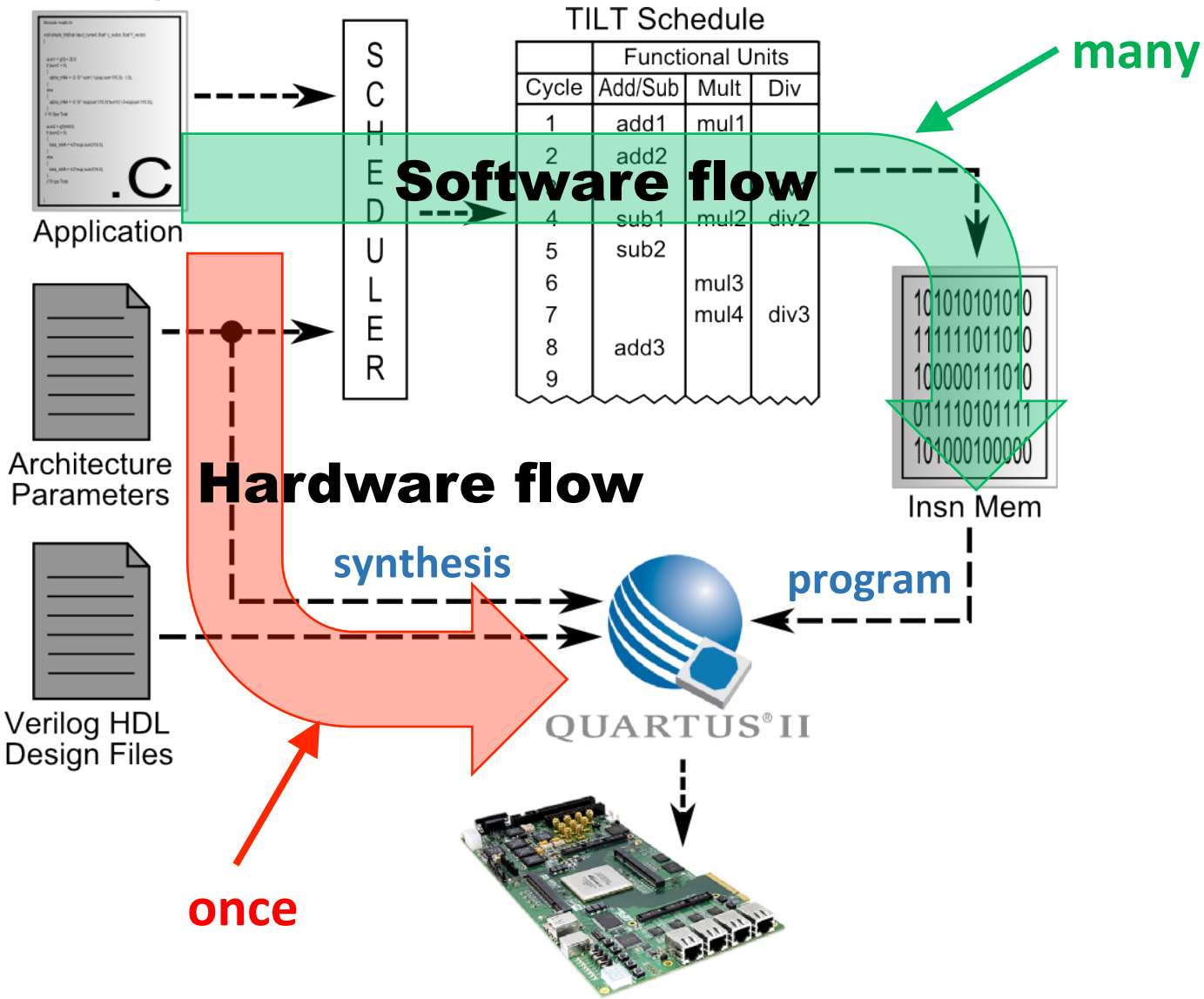
TILT – Compiler Flow



TILT – Compiler Flow



TILT – Compiler Flow



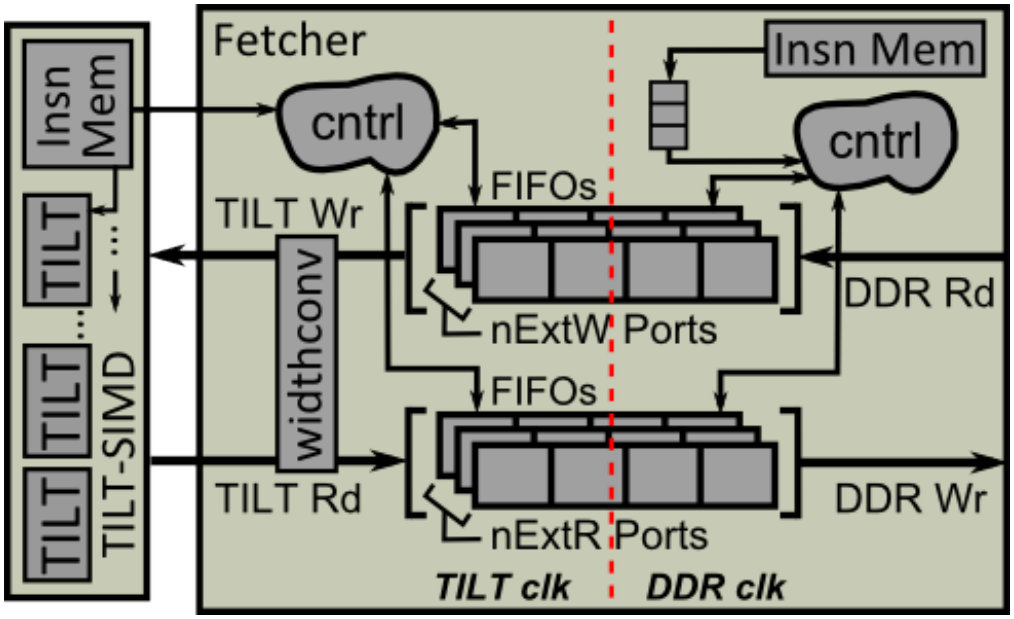
Our Contributions

- **Memory Fetcher Unit**
  - Interfacing TILT with off-chip memory
- **Software Predictor Tool**
  - Rapid selection of best architecture for an application
- **TILT Architectural Enhancements**
  - To improve efficiency of large loops and of indirect memory access
- **Comparison with Altera's OpenCL HLS**
  - Performance / Area / Productivity / Scalability

Memory Fetcher

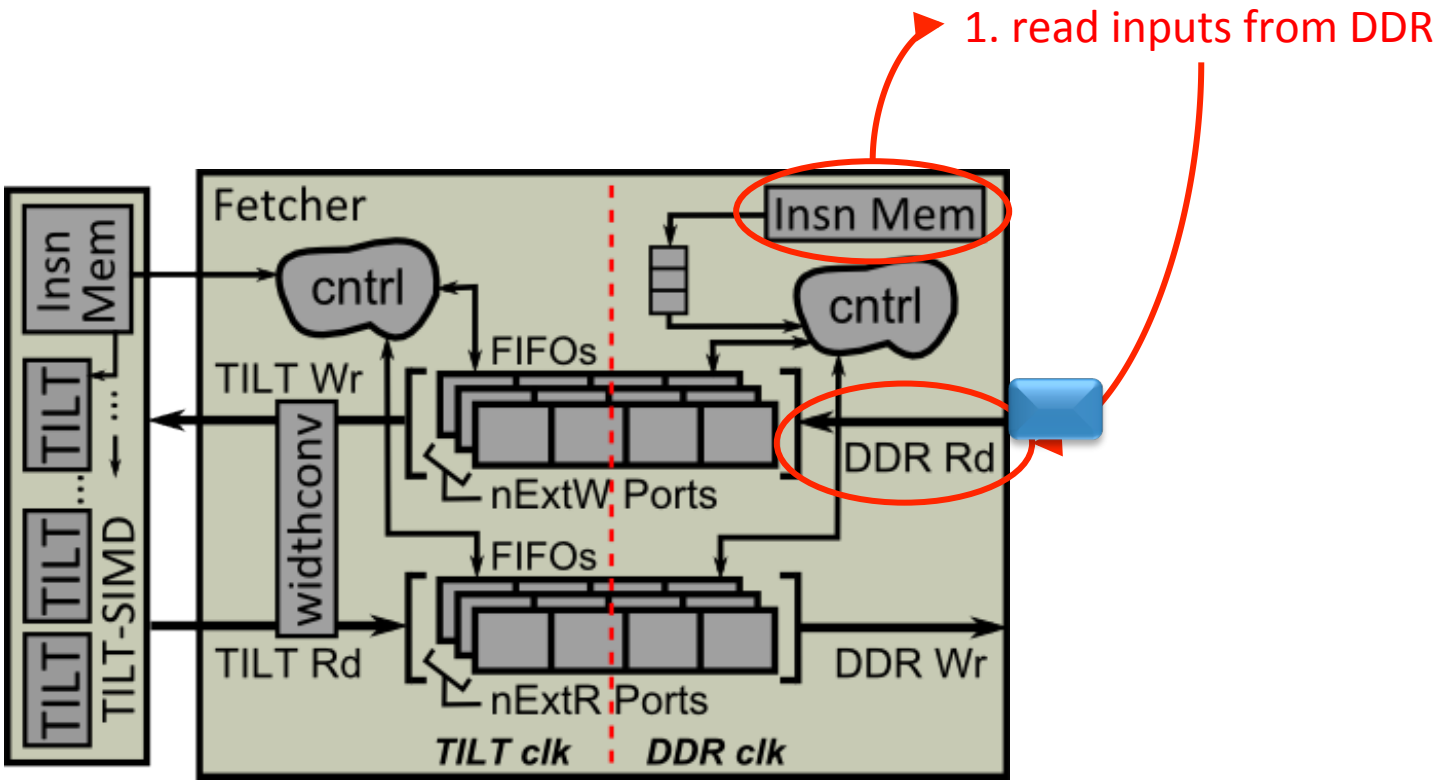
Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams



Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

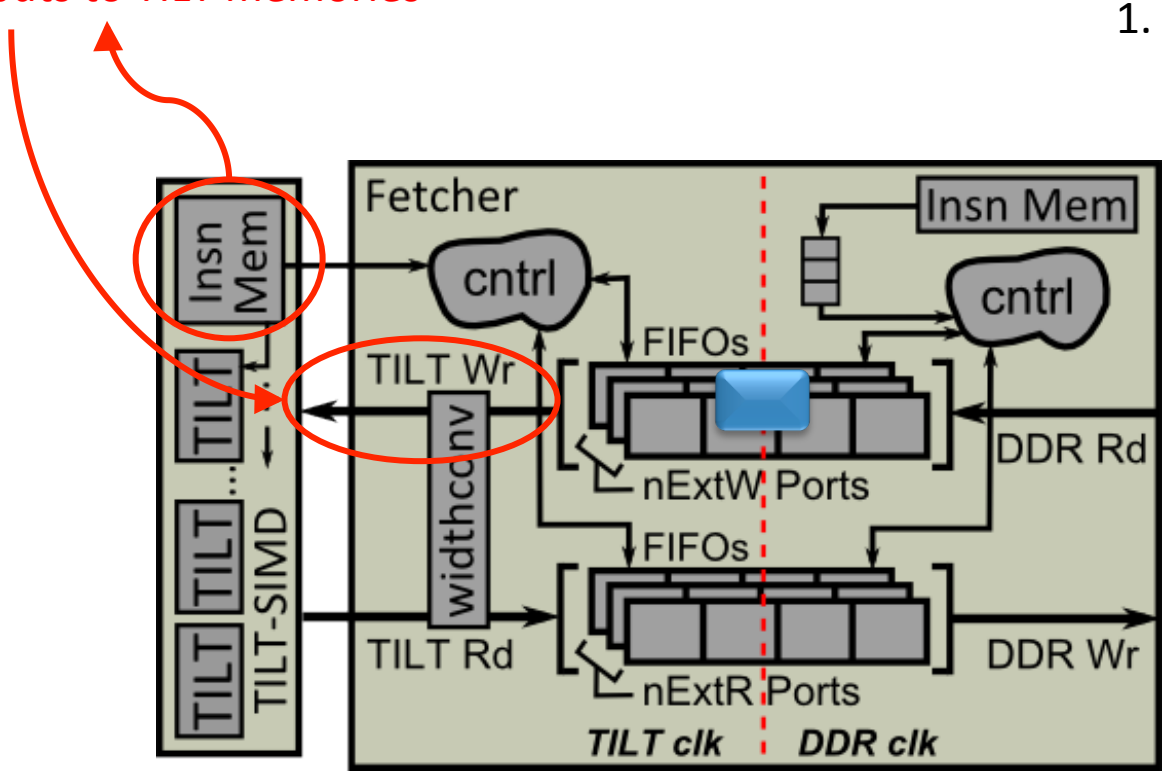


Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

2. write inputs to TILT memories

1. read inputs from DDR



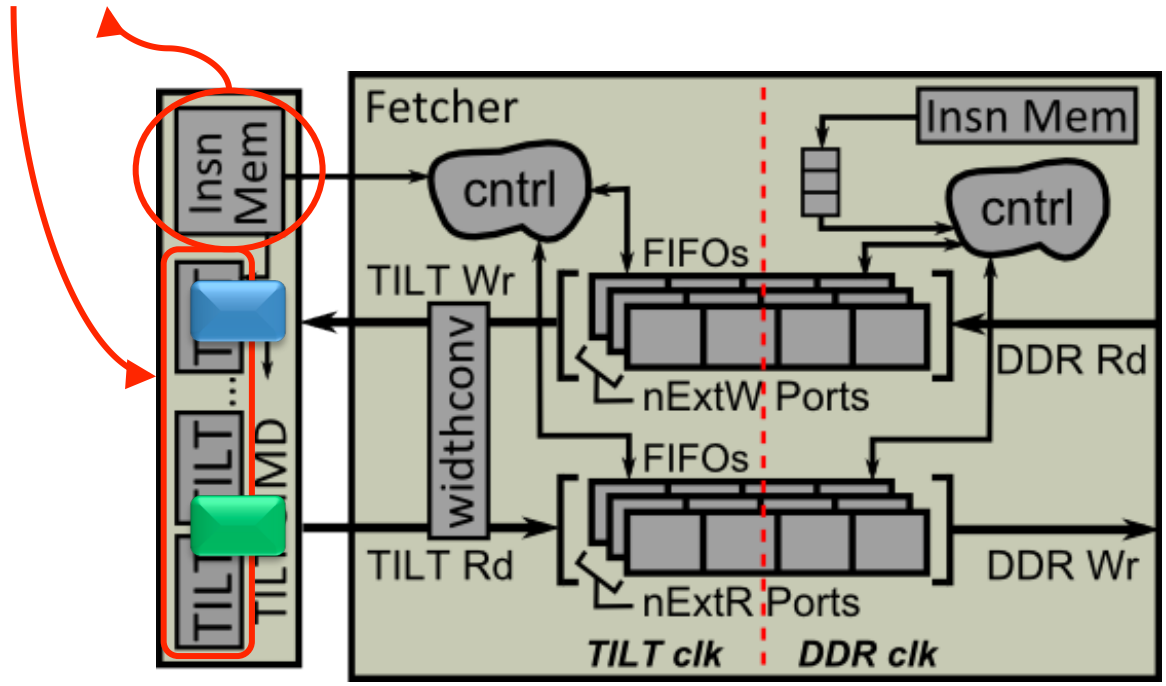
Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

2. write inputs to TILT memories

3. compute outputs (in SIMD)

1. read inputs from DDR



Memory Fetcher

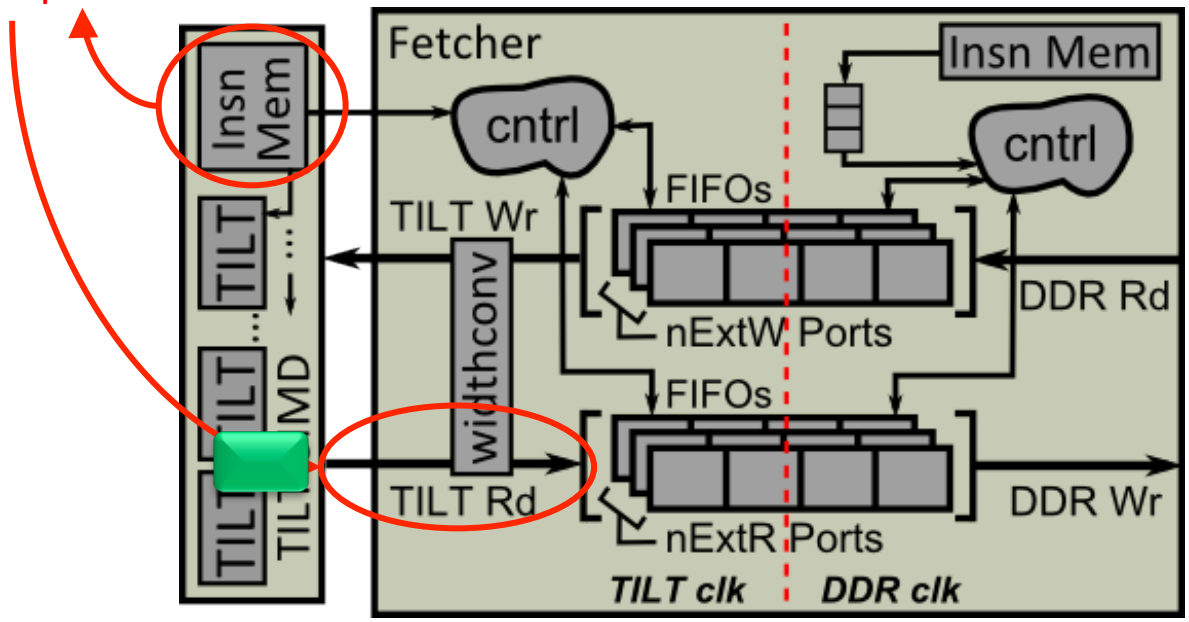
- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

2. write inputs to TILT memories

3. compute outputs (in SIMD)

4. write outputs to FIFOs

1. read inputs from DDR

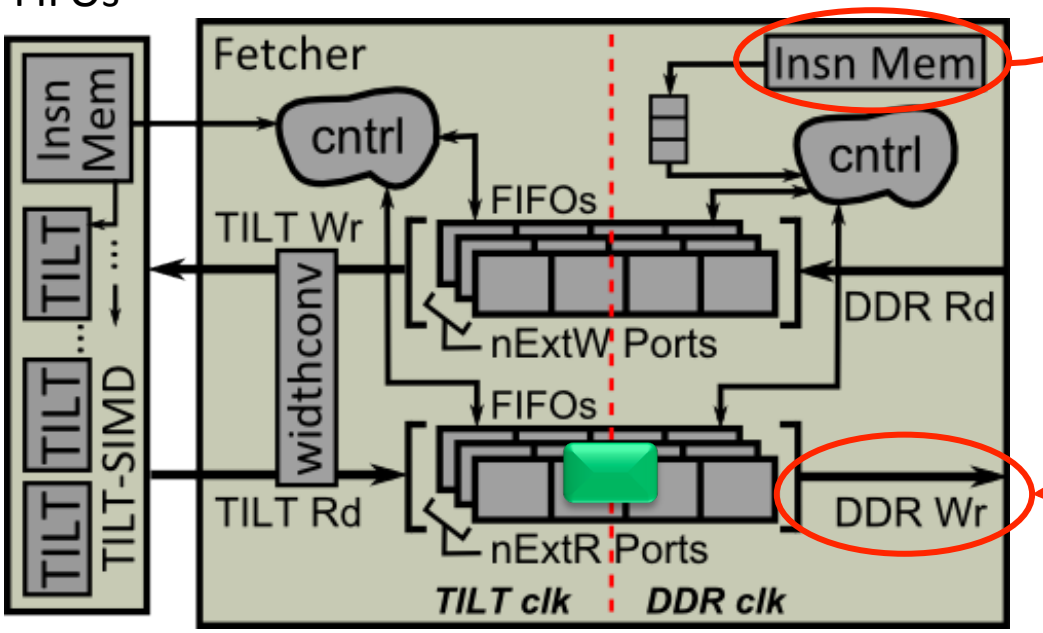


Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

2. write inputs to TILT memories
3. compute outputs (in SIMD)
4. write outputs to FIFOs

1. read inputs from DDR
5. write outputs to DDR

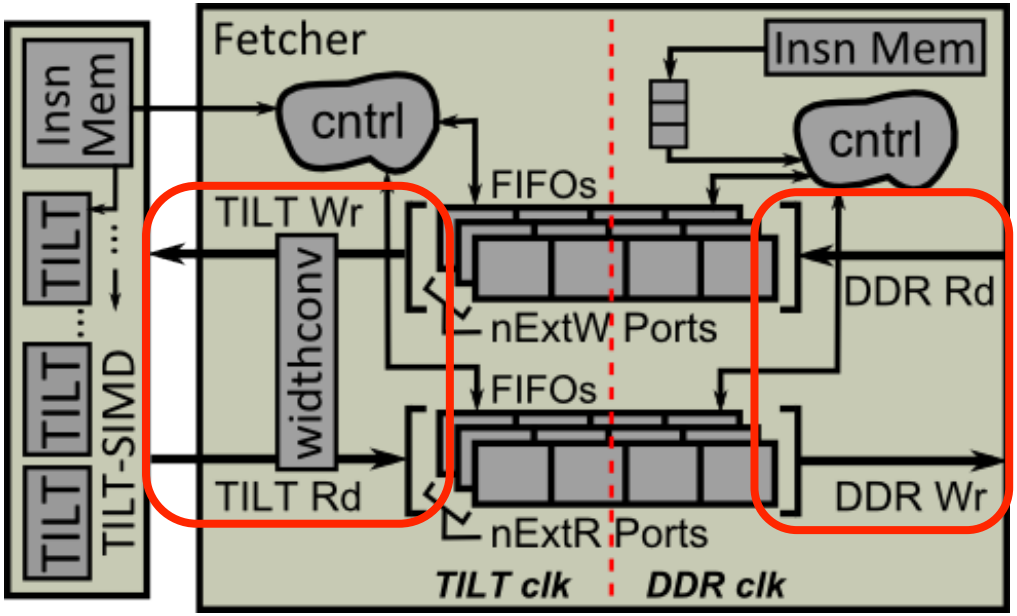


Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams

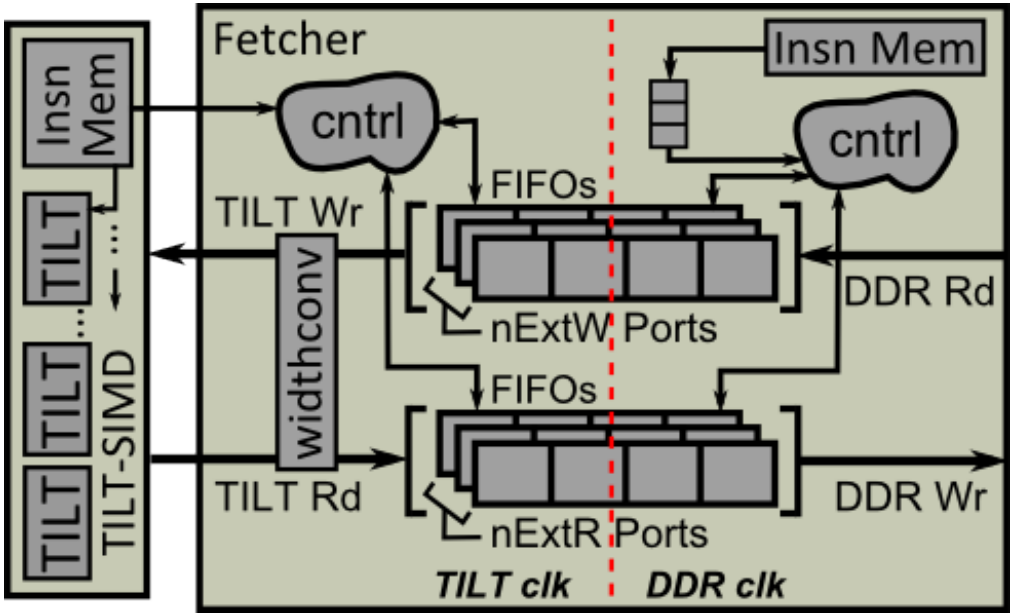
→ Constant, fast access

→ Variable, long access latency
→ Burst access



Memory Fetcher

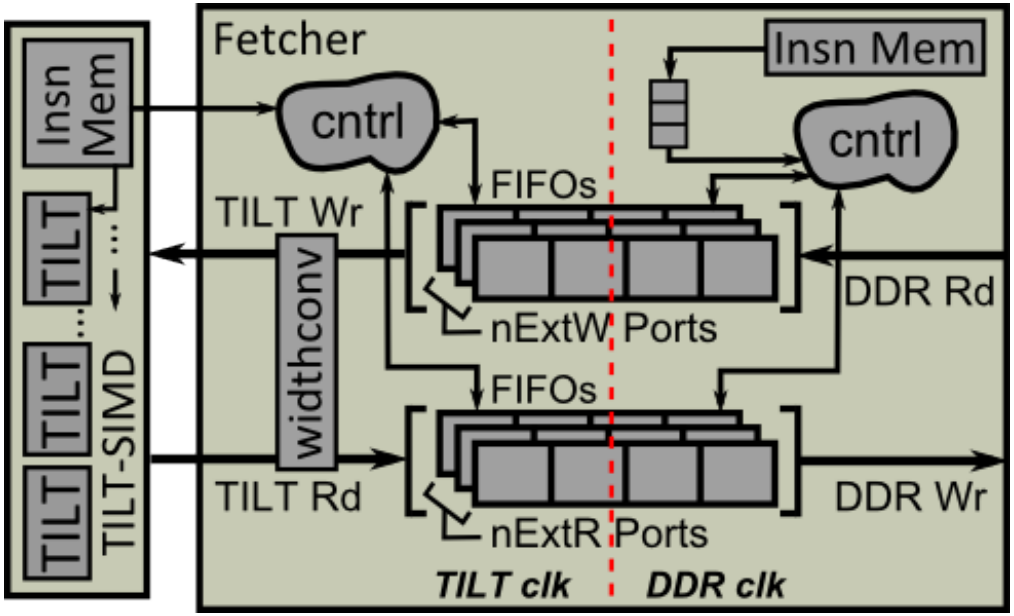
- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams
  - **FIFOs**:
    - ➢ Decouple TILT execution from off-chip communication



Memory Fetcher

- Configurable and scalable off-chip data prefetcher
- Minimal hardware → statically scheduled
  - **Data movement** : Two parallel instruction streams
  - **FIFOs**:
    - ➢ Decouple TILT execution from off-chip communication

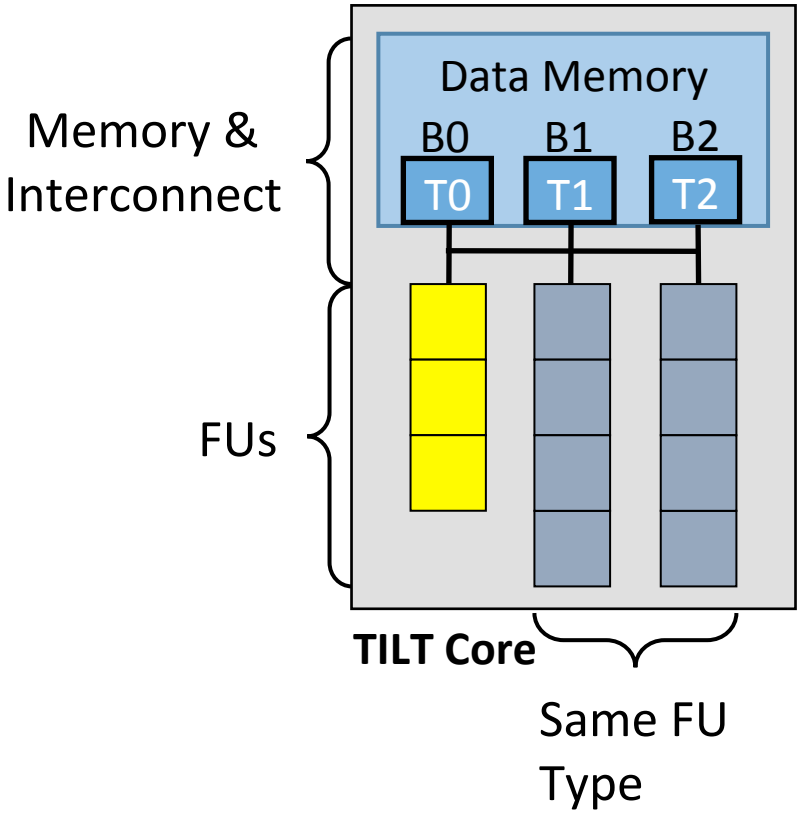
TILT-System



Predictor Tool

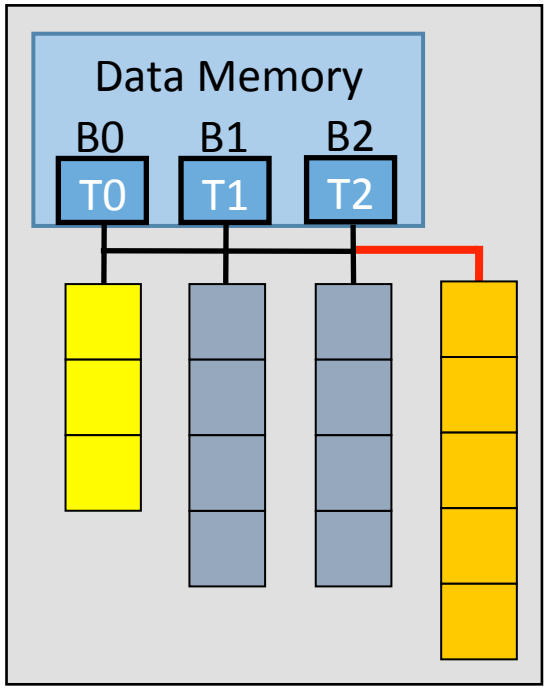
TILT / Fetcher Design Space

- **TILT Compute Cores**



TILT / Fetcher Design Space

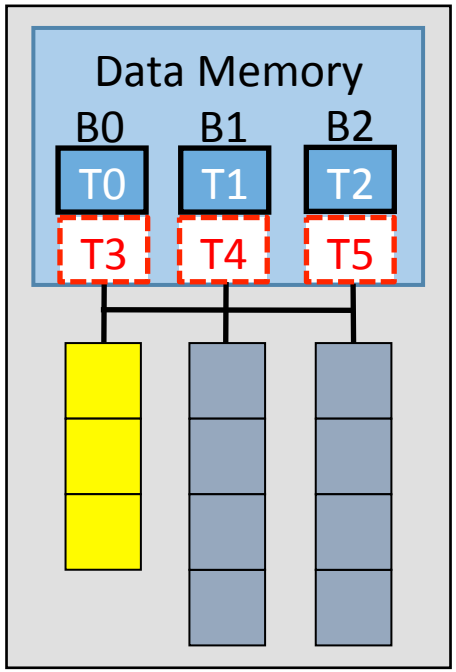
- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?



TILT Core

TILT / Fetcher Design Space

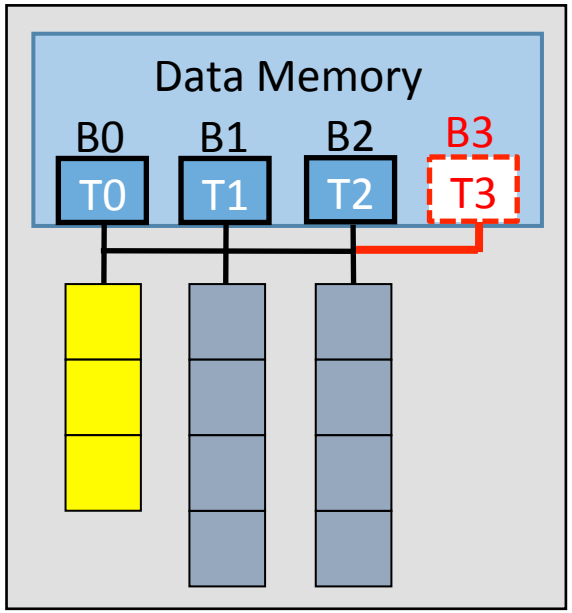
- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads



TILT Core

TILT / Fetcher Design Space

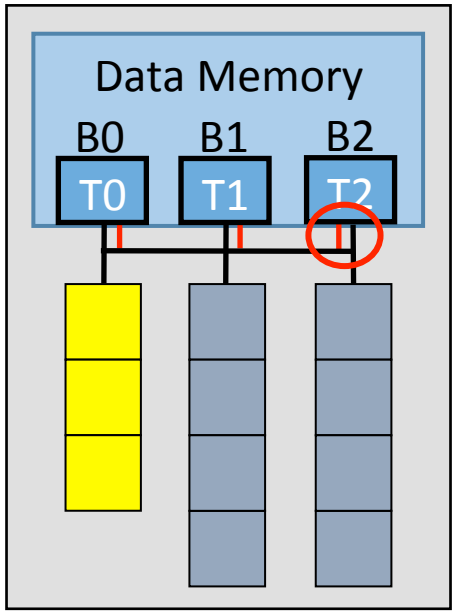
- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads and memory banks



TILT Core

TILT / Fetcher Design Space

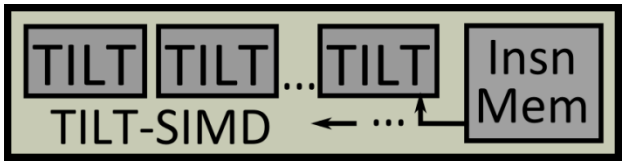
- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads and memory banks
    - Vary read / write bank port mixes



TILT Core

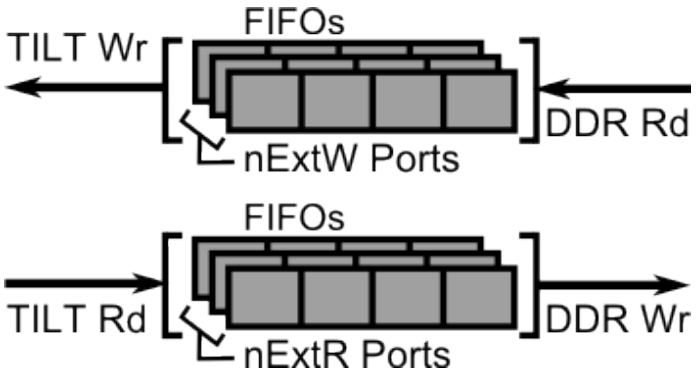
TILT / Fetcher Design Space

- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads and memory banks
    - Vary read / write bank port mixes
- **Fetcher**
  - Talks to *multiple* TILT cores in parallel



TILT / Fetcher Design Space

- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads and memory banks
    - Vary read / write bank port mixes
- **Fetcher**
  - Talks to *multiple* TILT cores in parallel
  - Vary data FIFO depths
  - Vary external read / write ports

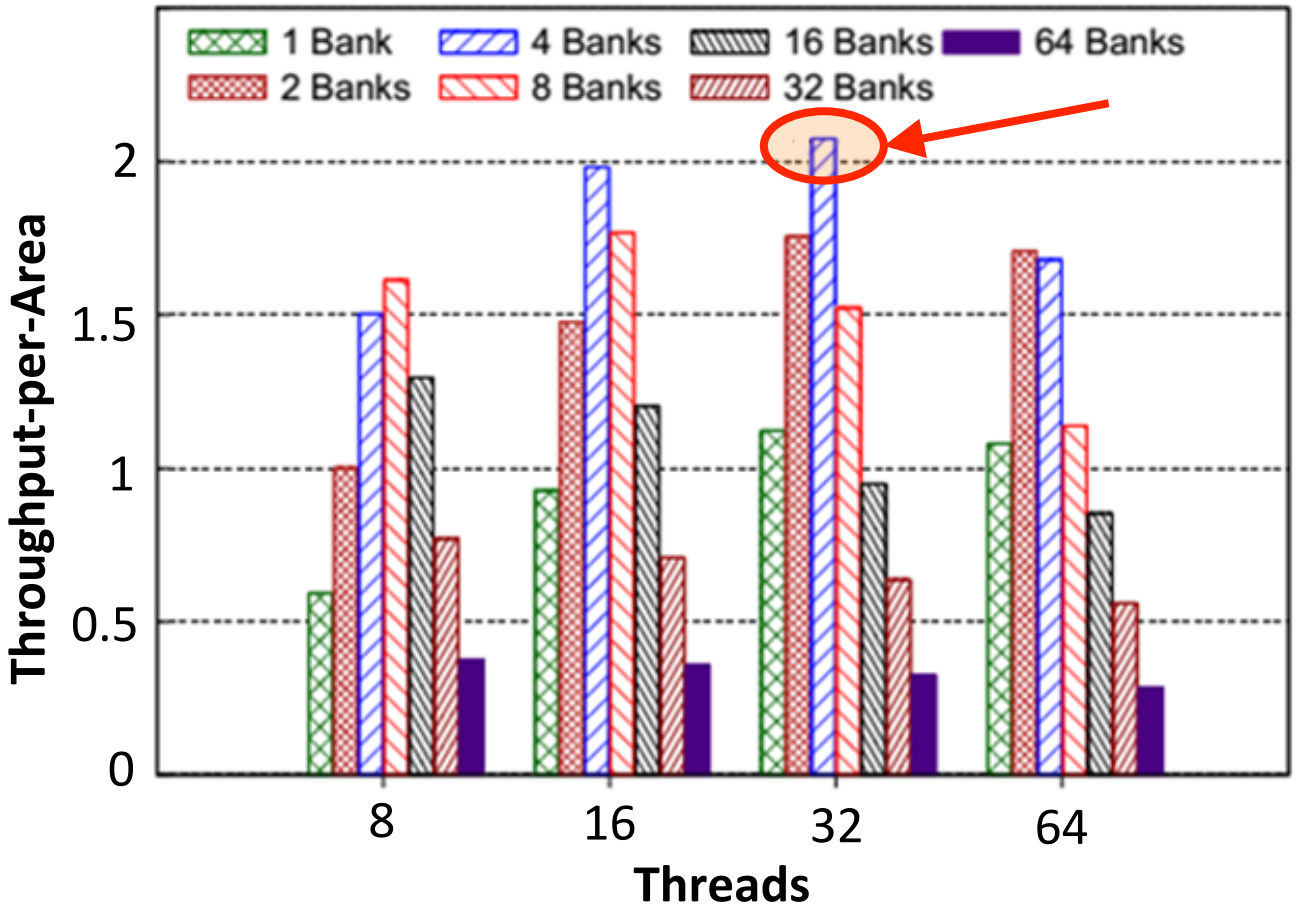


TILT / Fetcher Design Space

- **TILT Compute Cores**
  - Functional Units (FUs)
    - 9 FU types – FU mix?
  - Data Memory
    - Vary # of threads and memory banks
    - Vary read / write bank port mixes
- **Fetcher**
  - Talks to *multiple* TILT cores in parallel
  - Vary data FIFO depths
  - Vary external read / write ports

→ Many possible design solutions!

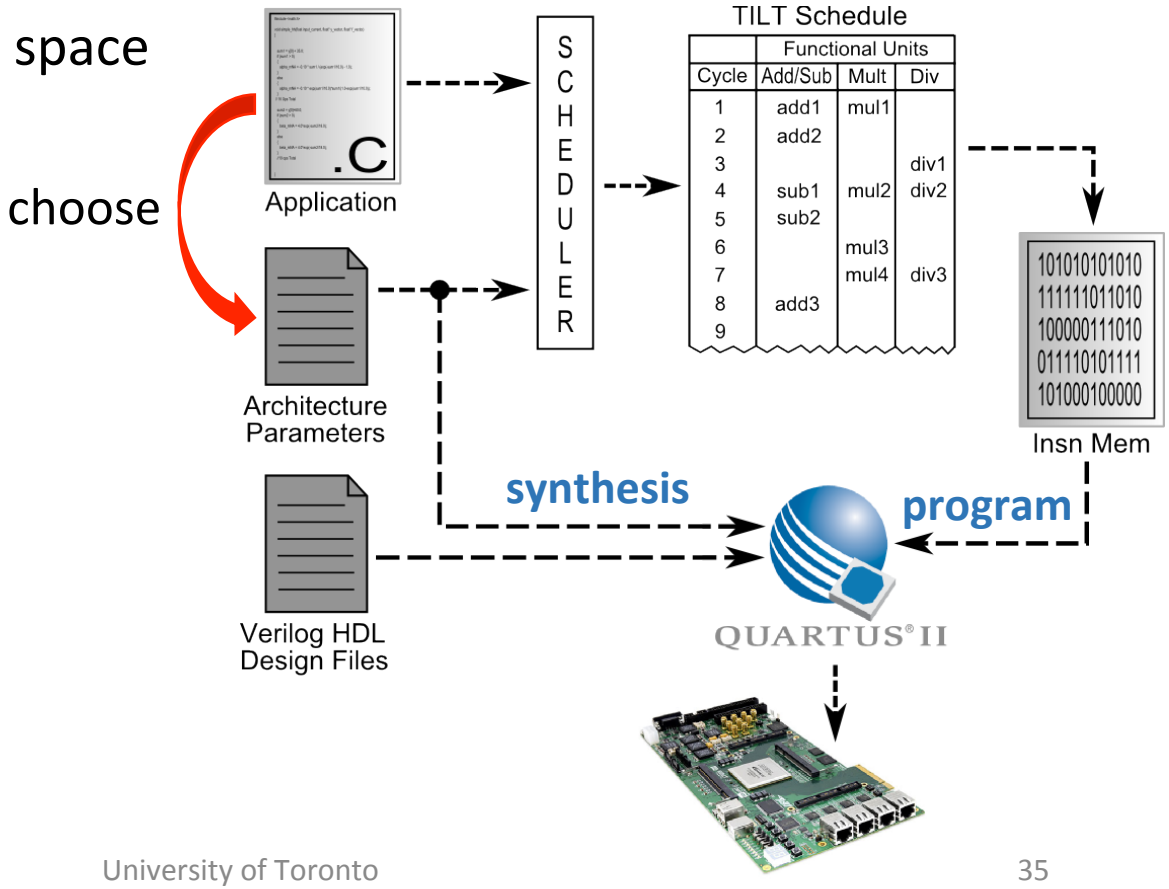
TILT / Fetcher Design Space



→ **Obtaining best design for an application is non-trivial.**

Predictor Tool

- Predicts densest (highest throughput-per-area) TILT and Fetcher design for a target application
- Uses software performance and area models
- Prune explored design space using heuristics



Evaluation

Comparison with Altera's OpenCL HLS

Altera's OpenCL HLS

- Generates high throughput, power efficient designs on FPGAs compared to CPUs and GPUs [chen13]

[chen13] D. Chen and D. P. Singh, "Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms." in ASP-DAC, 2013.

Altera's OpenCL HLS

→ **Compare Performance / Area / Productivity / Scalability**

- **Our TILT processor** (enhanced, application-customized)
- **with Altera's HLS tool** (custom designs)

Benchmarks

- Data-parallel, compute-intensive, floating-point applications
  - **Black-Scholes** Option Pricing (BSc)
  - **High Dynamic Range** Image Processing (HDR)
  - **Mandelbrot** Fractal Rendering (MBrot)
  - **Hodgkin-Huxley** Neuron Simulation (HH)
  - 64-tap Time-Domain **Finite Impulse Response** (FIR)

Platform

- Nallatech 385 D5 board
  - Stratix V GS FPGA
  - DDR3, 2 banks of 4 GB memory each

- Quartus 13.1

Metrics

- Task: completion of single compute kernel instance

Metrics	Units
Performance Compute Throughput	millions of tasks/sec (M tps)
FPGA Area	ALMs, BRAMs and DSPs → equivalent ALMs (eALMs)
Ranking Designs Compute Density	Throughput-per-Area

Comparison Methodology

Both

- Assume all input data resides in DDR3
- Implement tuned compute kernels (for either approach)

TILT with Fetcher

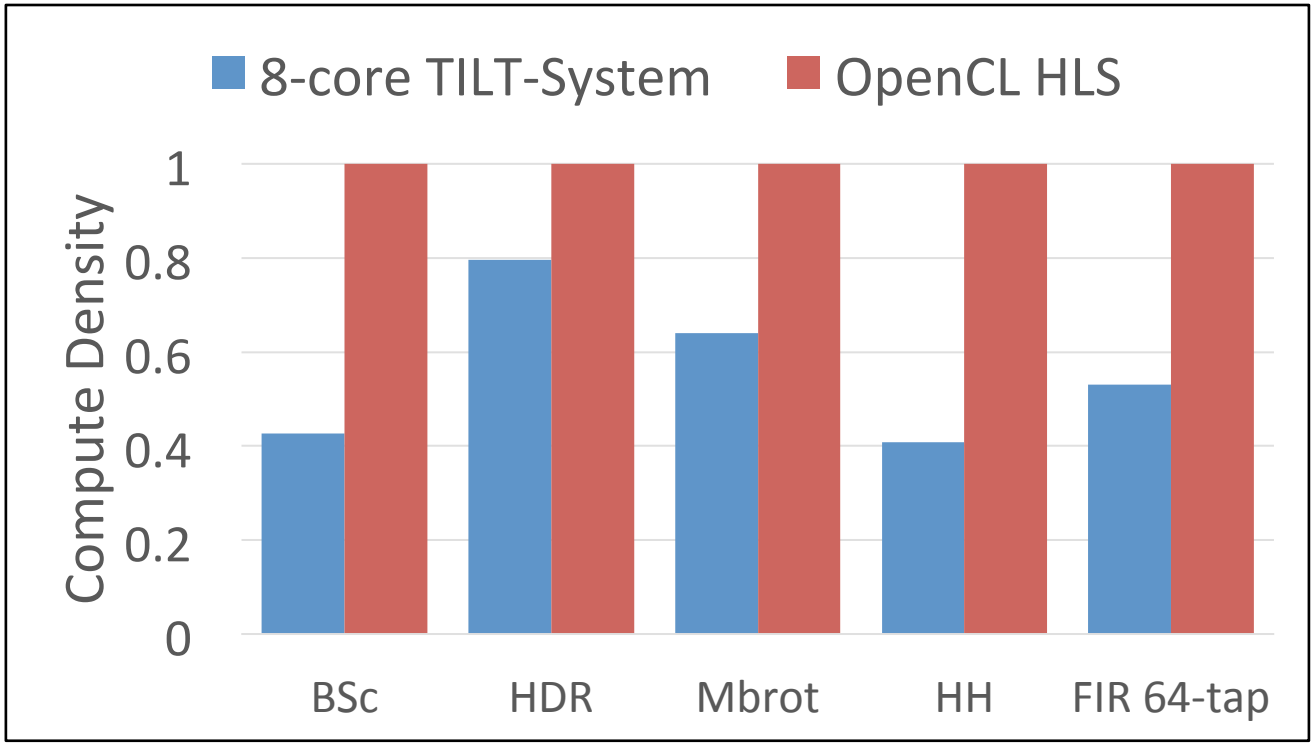
- Use Predictor to select best configuration
- Throughput – cycle-accurate ModelSim simulations

OpenCL HLS

- Exclude CPU-to-DDR3 transfer time
- As many threads as necessary to saturate throughput

TILT with Fetcher vs. OpenCL HLS

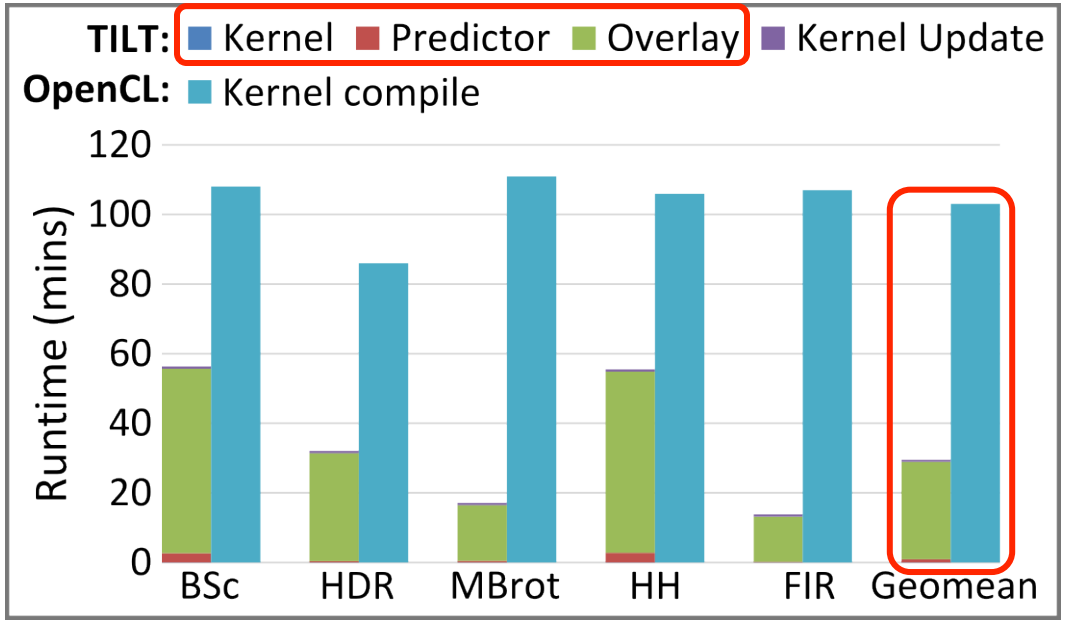
- **Good Performance**
  - Achieve **41% (HH)** to **80% (HDR)** of the compute density of equivalent OpenCL designs



TILT with Fetcher vs. OpenCL HLS

- **Compelling Gain in Productivity**

Runtime of tools – 8-core TILT-System vs. OpenCL HLS

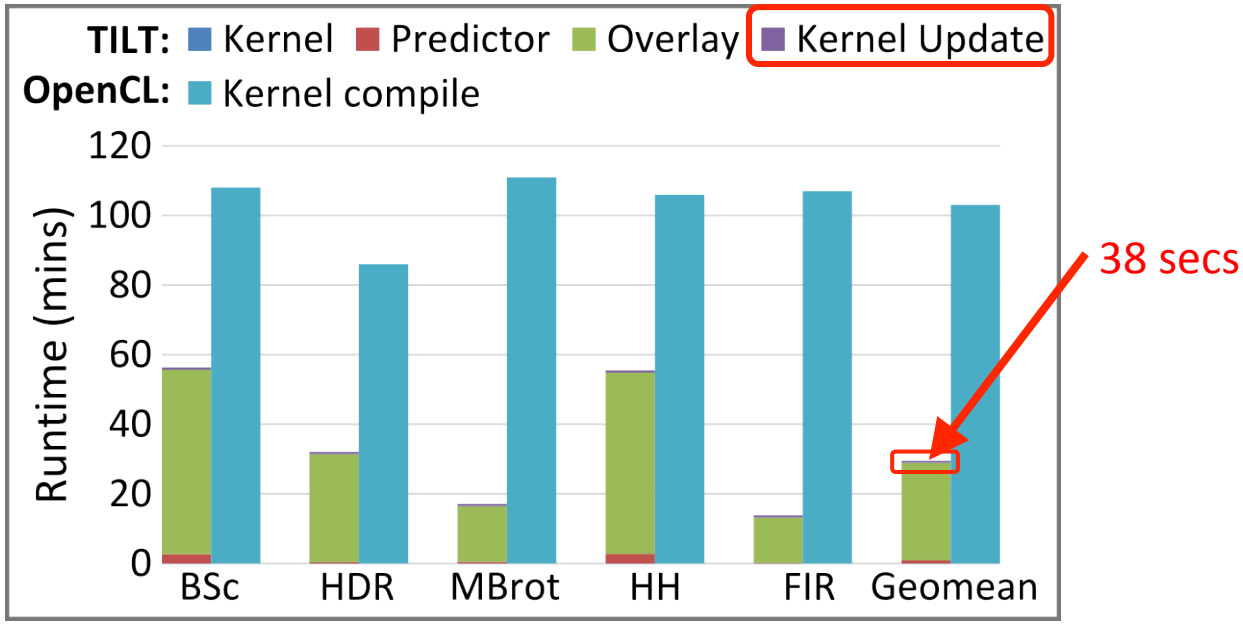


➢ Initial Setup: **29 mins** (TILT-System) vs. **103 mins** (OpenCL HLS)

TILT with Fetcher vs. OpenCL HLS

- **Compelling Gain in Productivity**

Runtime of tools – 8-core TILT-System vs. OpenCL HLS

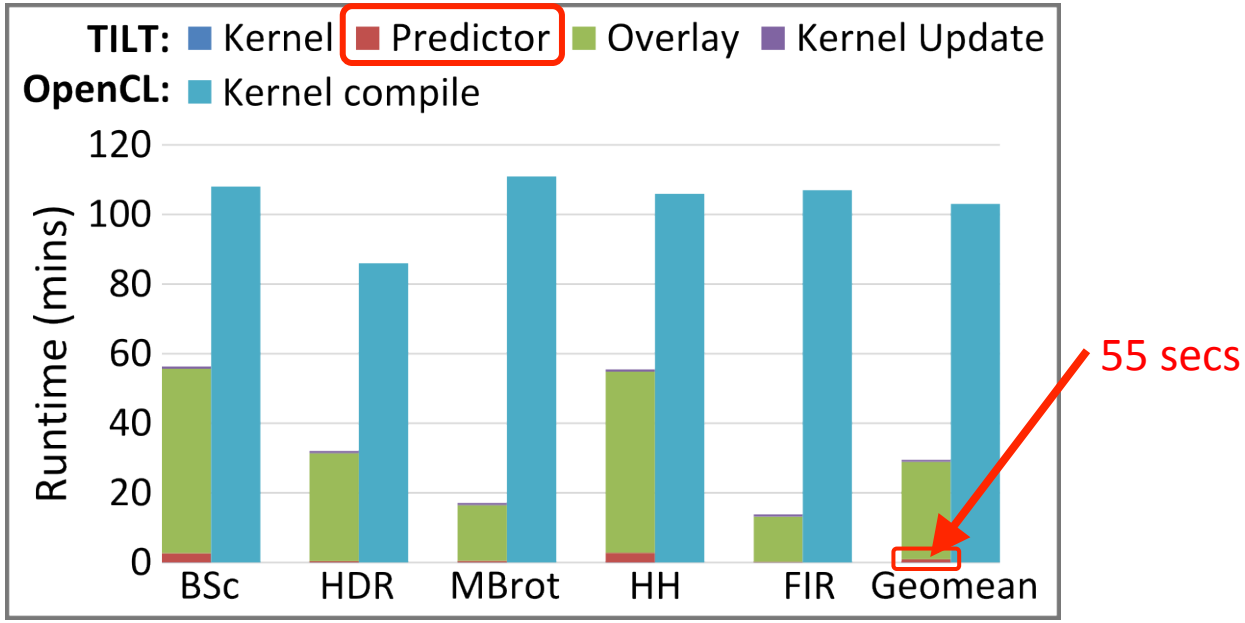


- Initial Setup: **29 mins** (TILT-System) vs. **103 mins** (OpenCL HLS)
- Fast overlay reconfig after kernel code change
  - **163x** faster than OpenCL synthesis

TILT with Fetcher vs. OpenCL HLS

- **Compelling Gain in Productivity**

Runtime of tools – 8-core TILT-System vs. OpenCL HLS



- Initial Setup: **29 mins** (TILT-System) vs. **103 mins** (OpenCL HLS)
- Fast overlay reconfig after kernel code change
  - **163x** faster than OpenCL synthesis
- Predictor enables fast selection of tuned TILT-System architecture

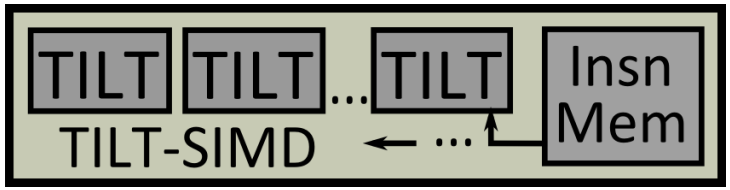
Scalability

- To meet throughput requirements / area budget

Scalability

- To meet throughput requirements / area budget

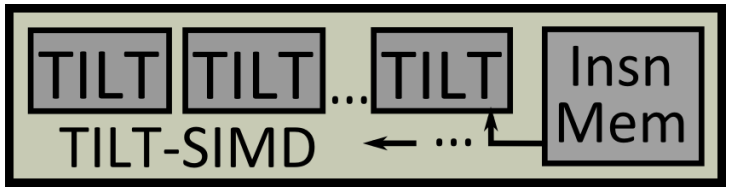
➢ TILT-System –
  - Execution of multiple TILT cores in parallel



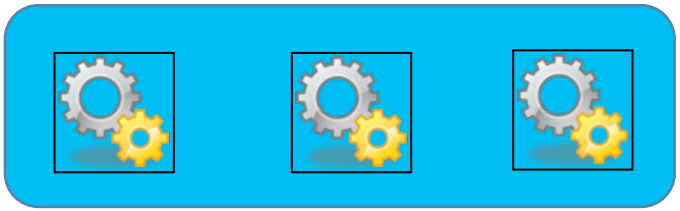
Scalability

- To match throughput requirements / area budget

➢ TILT-System –
  - Execution of multiple TILT cores in parallel

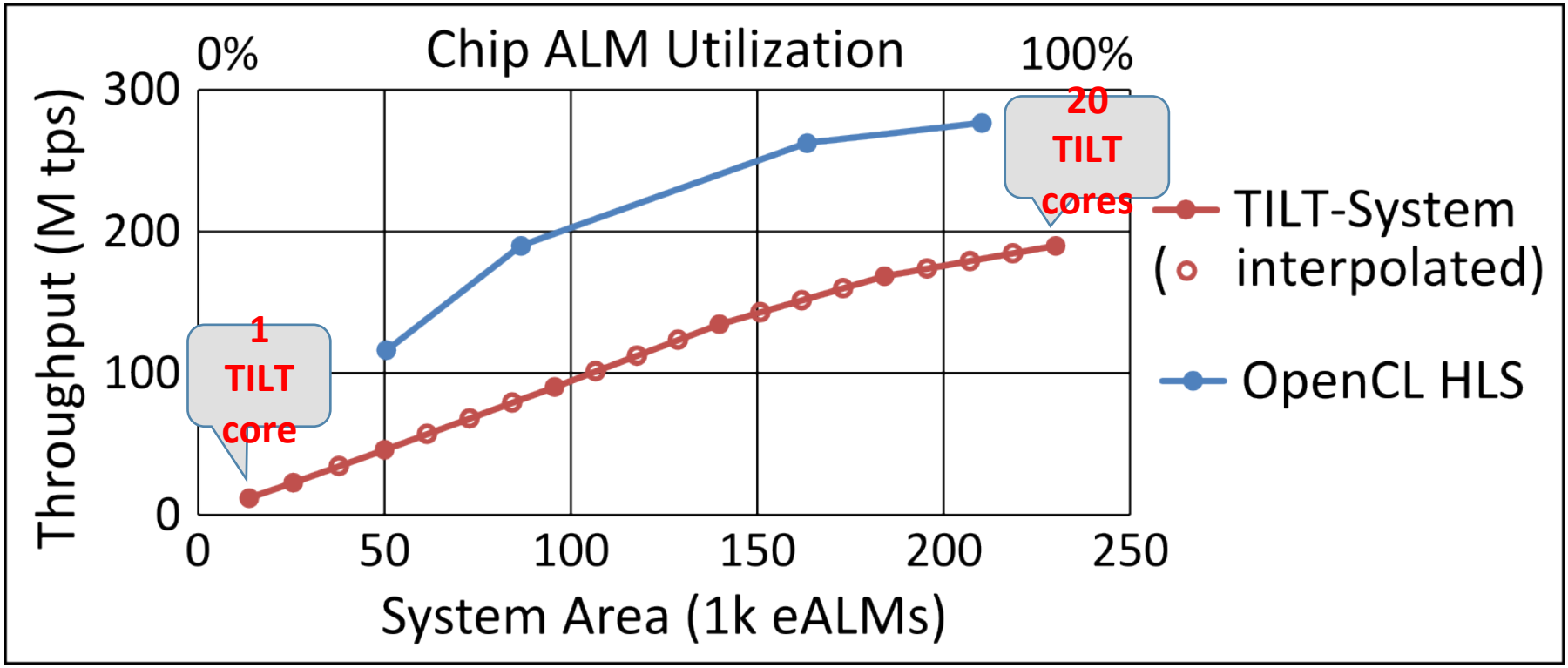


- OpenCL HLS –
  - Replicate compute pipeline



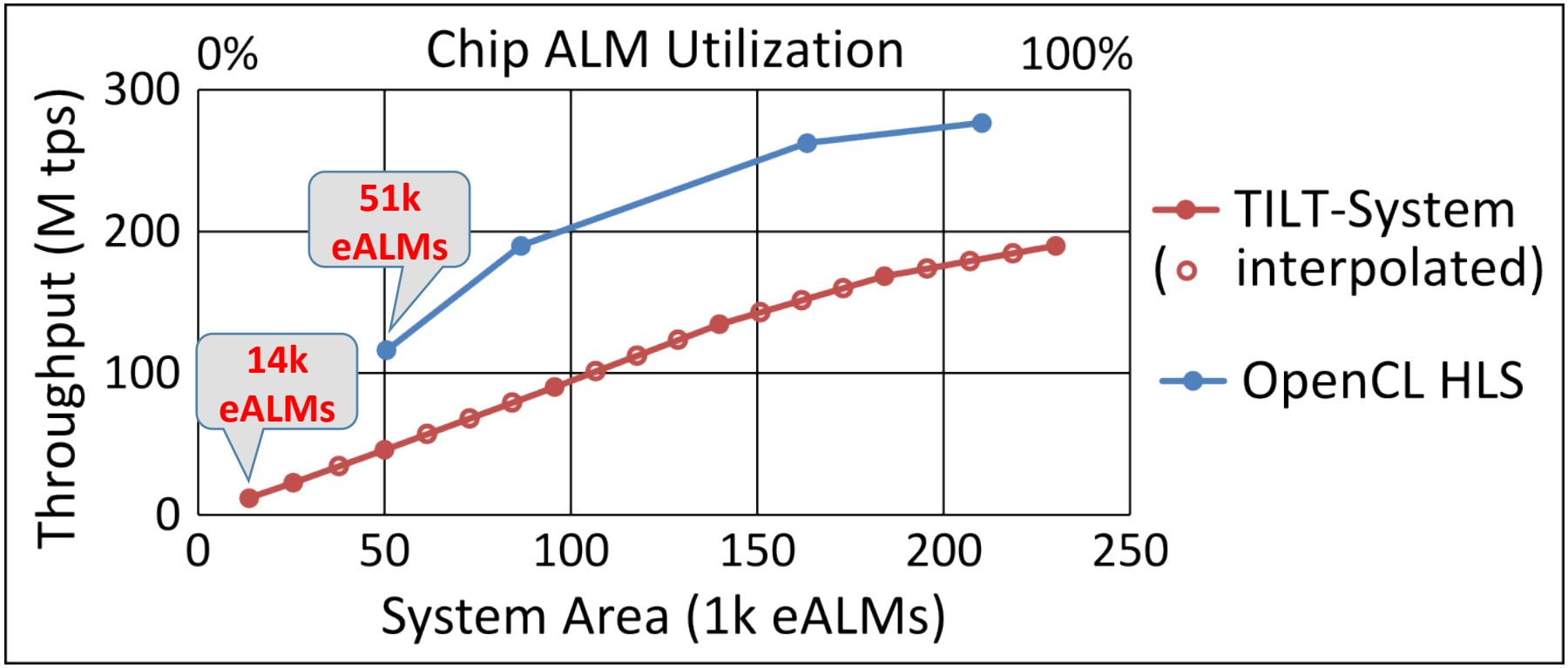
Scaling Hodgkin-Huxley

- Near-linear growth in throughput on TILT-System
  - But lower than OpenCL HLS
- Limited by chip capacity



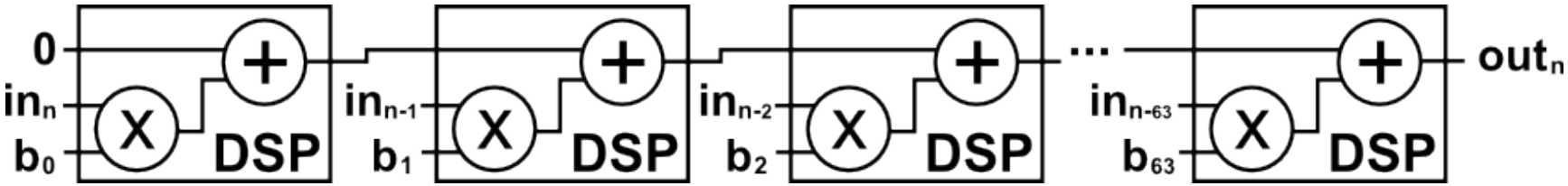
Scaling Hodgkin-Huxley

- Greater range of area-efficient throughput and area solutions on TILT-System



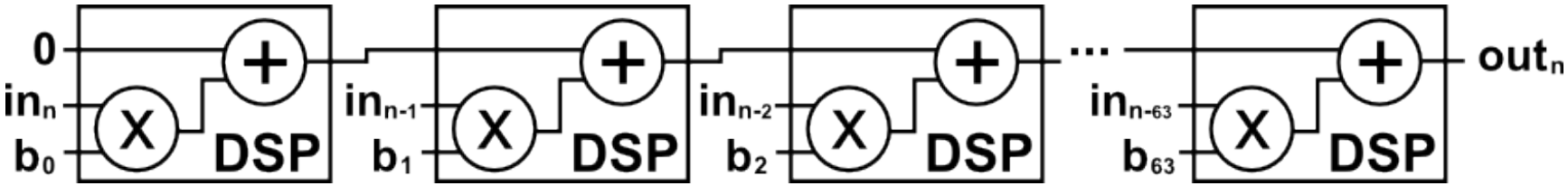
Scaling 64-tap FIR

- **OpenCL HLS –**
  - Spatially pipelined FIR with 64 stages

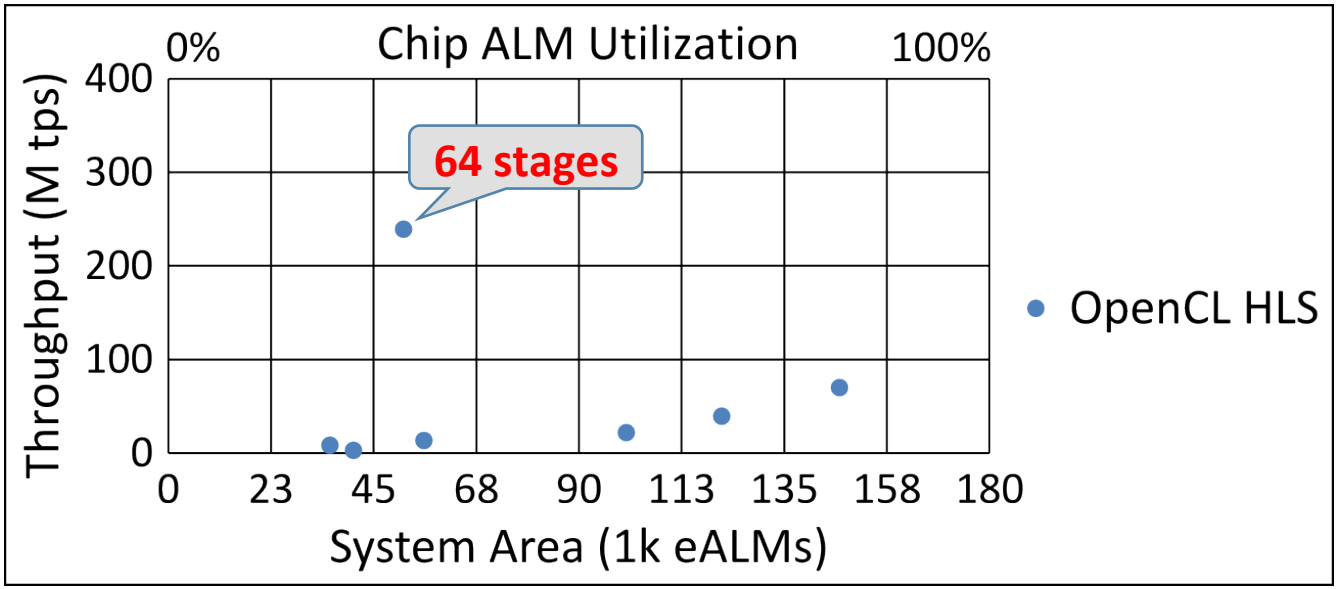


Scaling 64-tap FIR

- **OpenCL HLS –**
  - Spatially pipelined FIR with 64 stages

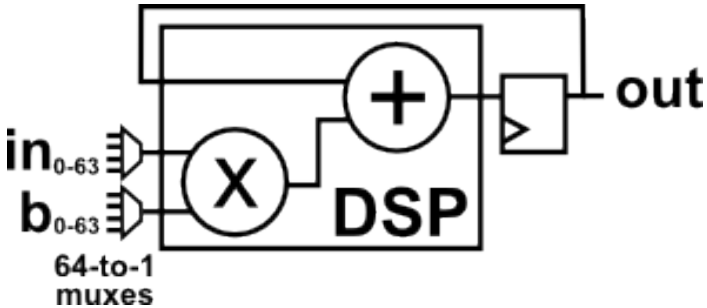


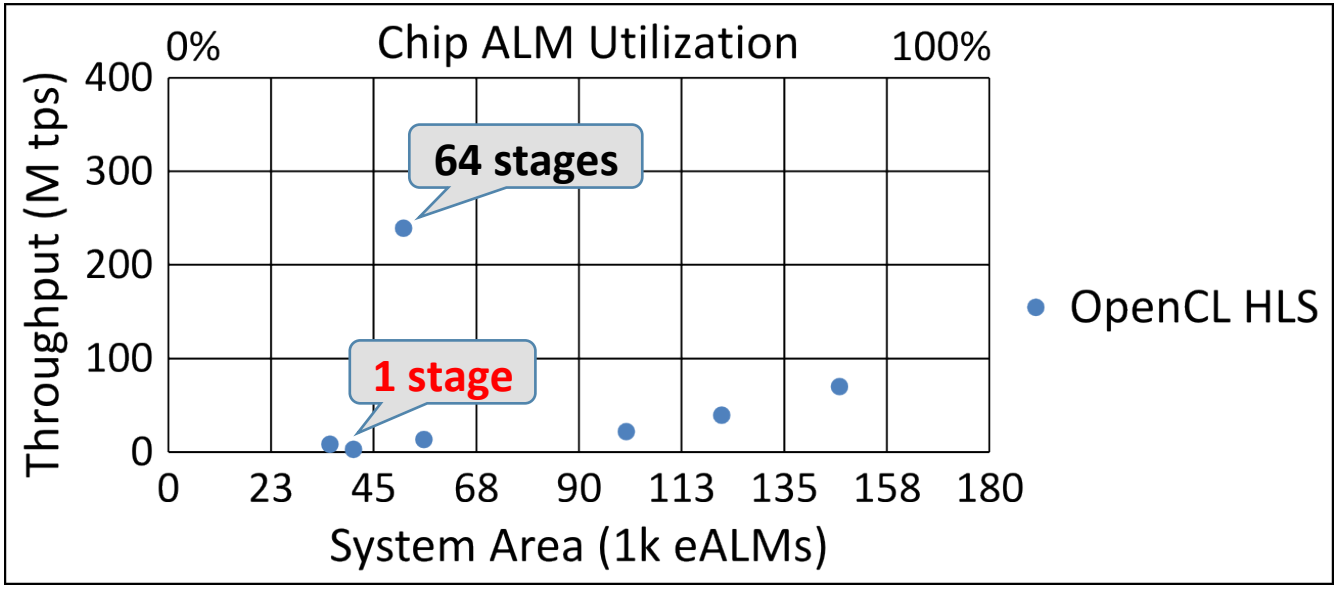
→ High throughput for relatively low area



Scaling 64-tap FIR

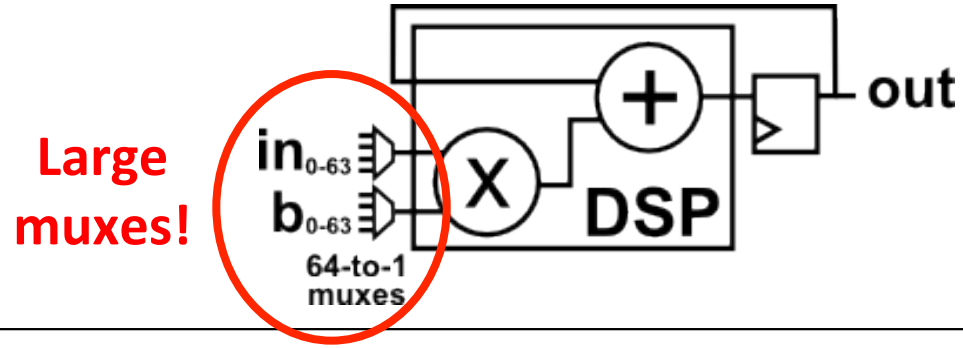
- **OpenCL HLS –**
  - 64-tap FIR with 1 Multiply-Add unit (DSP)

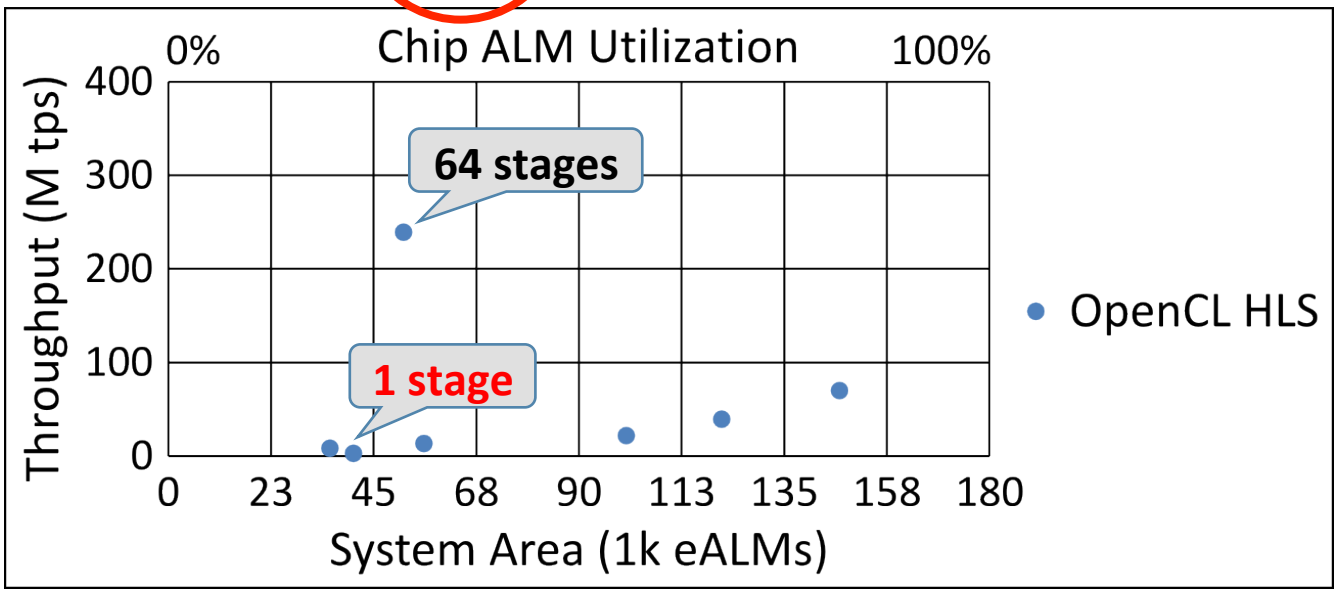




Scaling 64-tap FIR

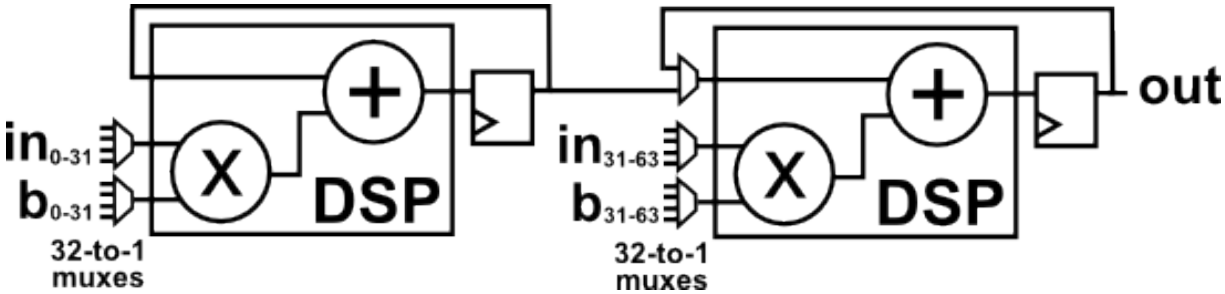
- **OpenCL HLS –**
  - 64-tap FIR with 1 Multiply-Add unit (DSP)

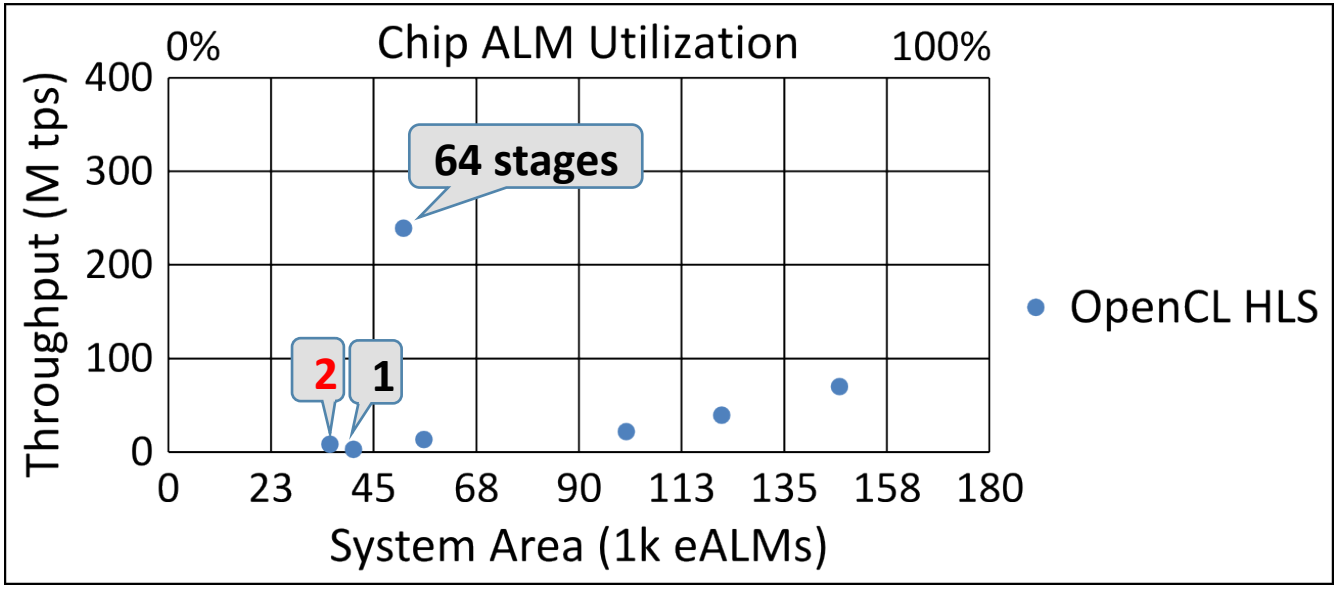




Scaling 64-tap FIR

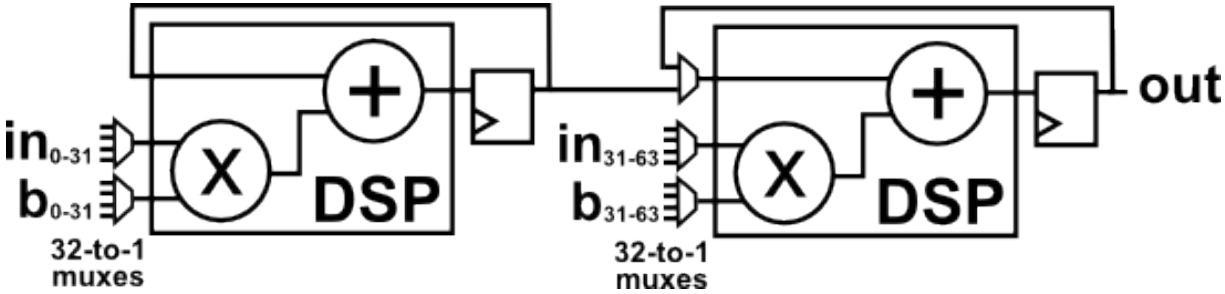
- **OpenCL HLS –**
  - 64-tap FIR with 2 Multiply-Add units (DSPs)



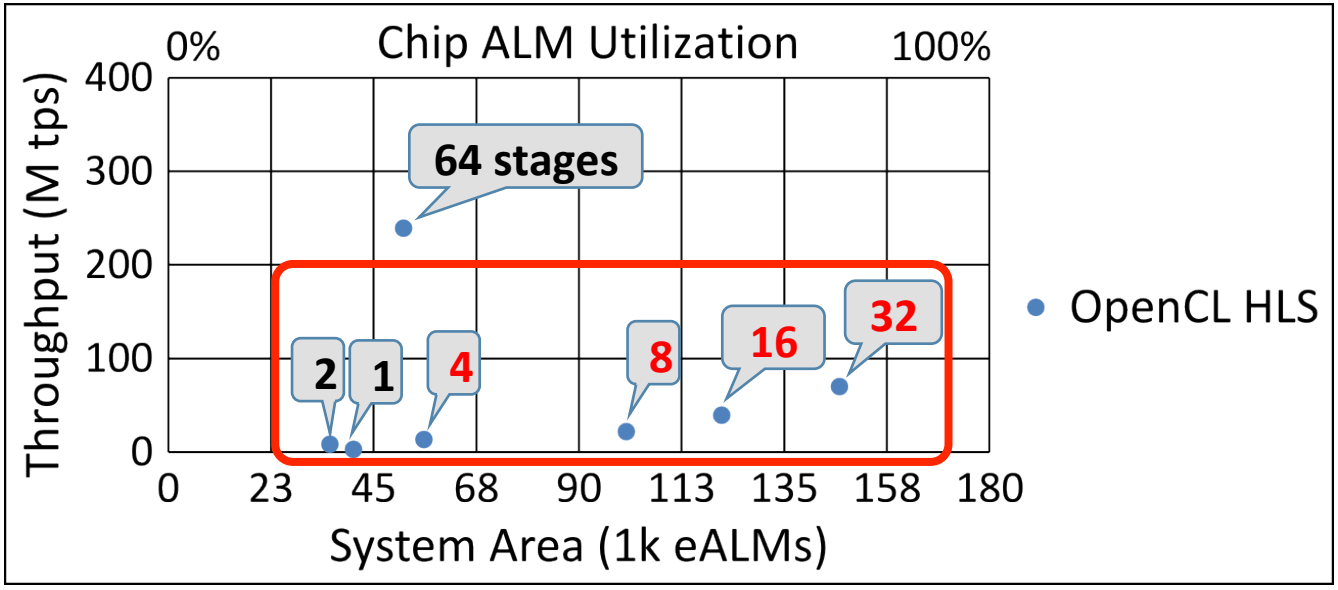


Scaling 64-tap FIR

- **OpenCL HLS –**

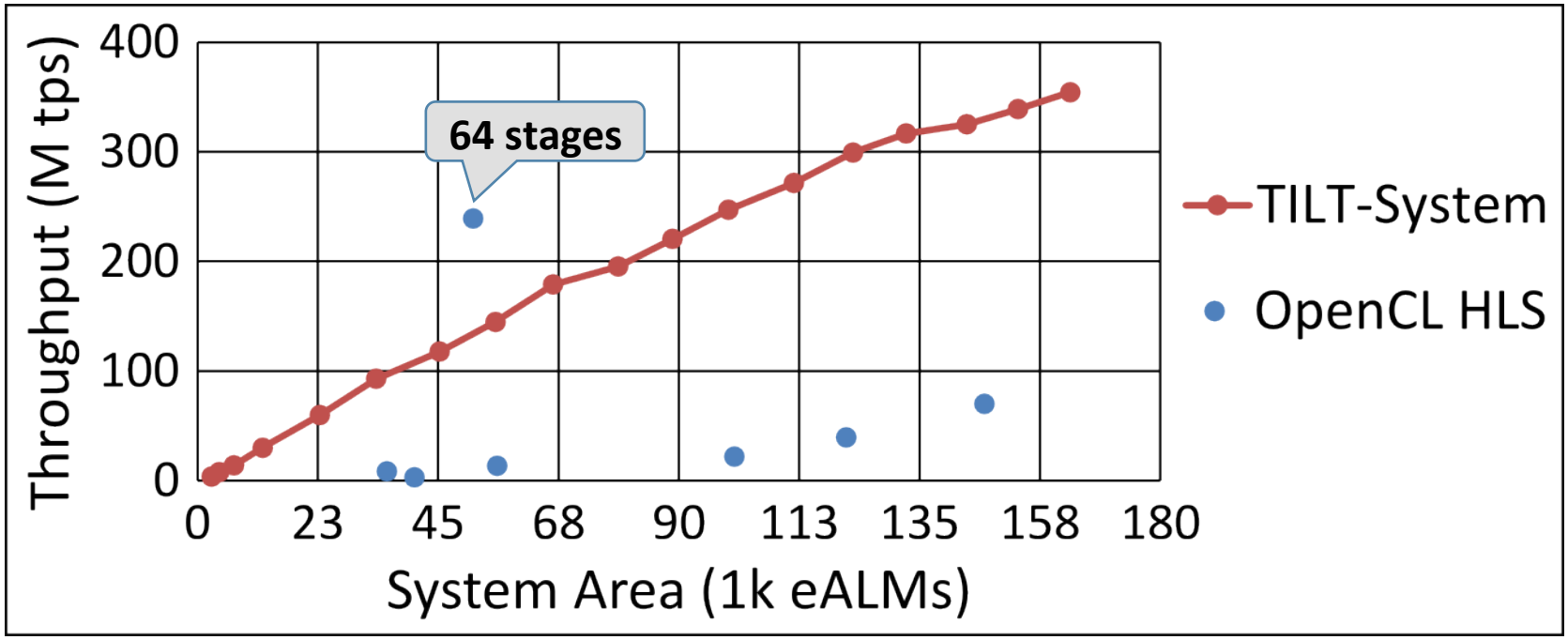


→ Difficulty generating area-efficient lower throughput systems!



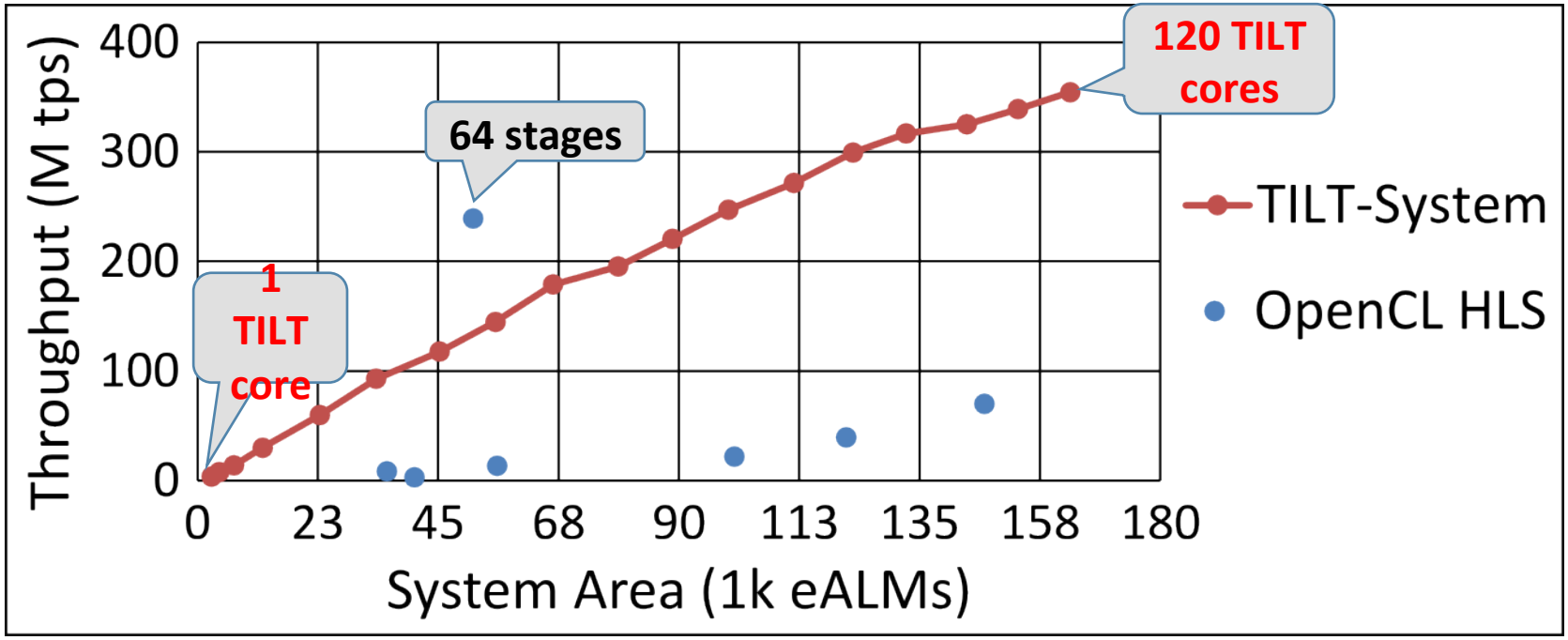
Scaling 64-tap FIR

- **TILT-System –** scaled up by increasing TILT core count



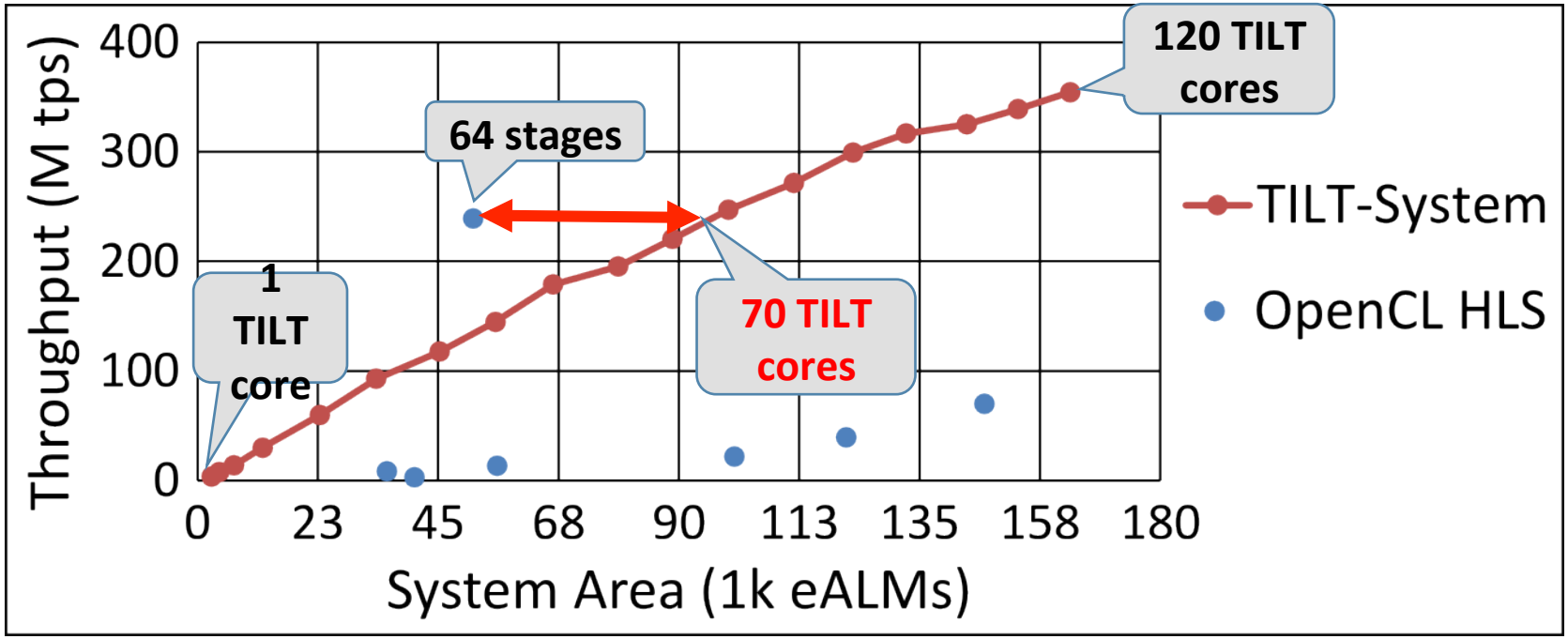
Scaling 64-tap FIR

- **TILT-System –** scaled up by increasing TILT core count
  - Near-linear scaling in throughput and area



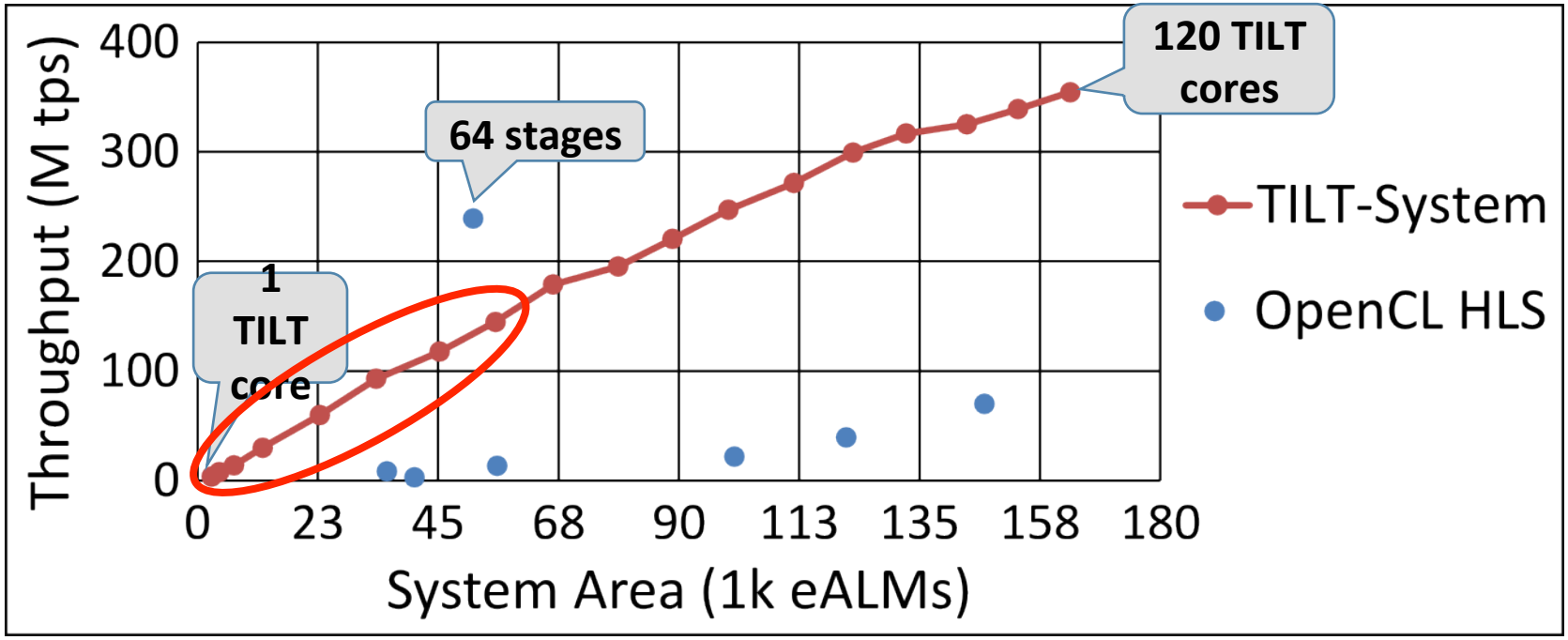
Scaling 64-tap FIR

- **TILT-System –** scaled up by increasing TILT core count
  - Near-linear scaling in throughput and area
  - Exceed 239 M inputs/sec (of spatial OpenCL FIR) with ~70 TILT cores
    - ➢ But at 2x more area



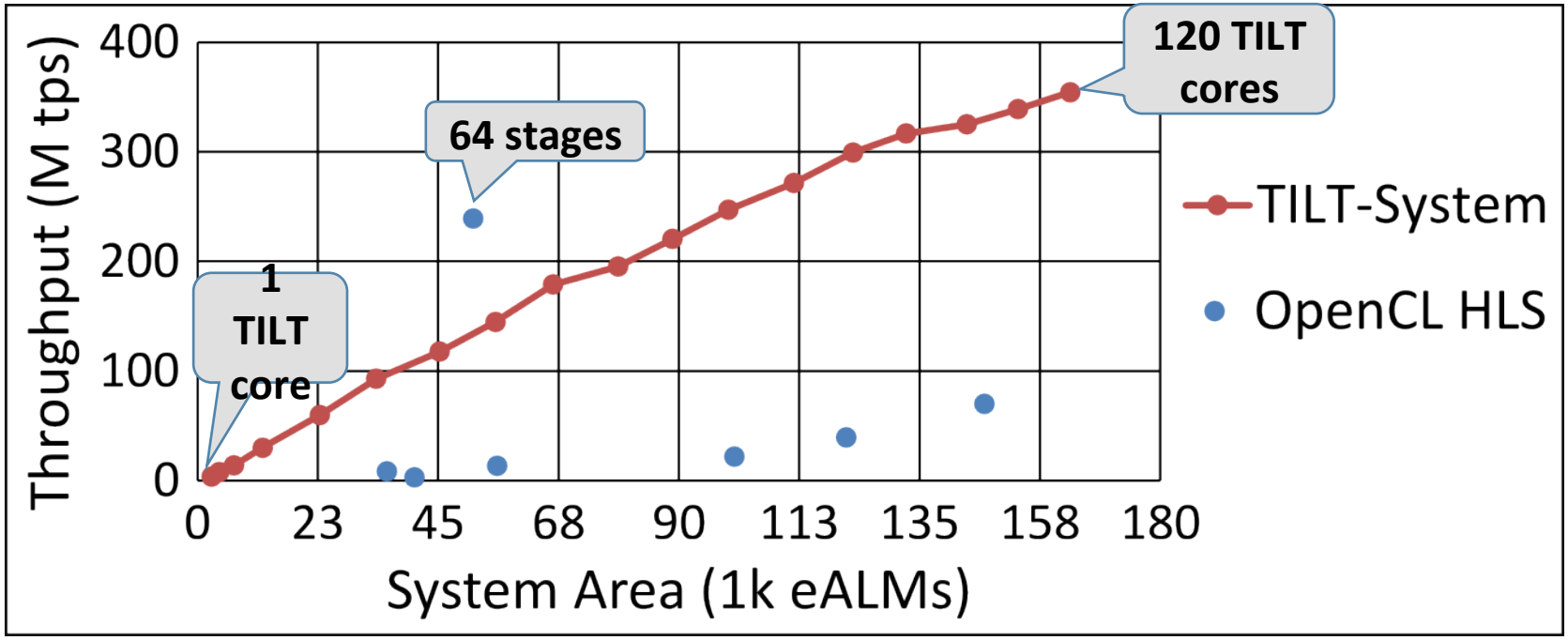
Scaling 64-tap FIR

- **TILT-System –**
  - Enables small, area-efficient designs with modest throughput



Scaling 64-tap FIR

- **TILT-System –**
  - Enables small, area-efficient designs with modest throughput
- **OpenCL HLS –**
  - Cannot efficiently scale down spatial computations



Contributions Summary

- **Memory Fetcher Unit**
  - Configurable, scalable and loosely coupled memory unit
- **Software Predictor Tool**
  - Rapidly and accurately predicts best TILT-System for an application
- **TILT Architectural Enhancements**
  - To improve efficiency of large loops and of indirect memory access
- **Comparison of TILT-System with Altera's OpenCL HLS**
  - Fast application recompile
  - Reasonable (but lower) throughput vs. OpenCL "sweet spot"
  - Wider trade-off range of throughput vs. area

➢ Useful complement to HLS!

Thank you!

Questions?