2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Next Gen System Design and Verification for Transportation

David Aerne

Jacob Wiltgen

Richard Pugh

Mentor®
A Siemens Business

accellera SYSTEMS INITIATIVE

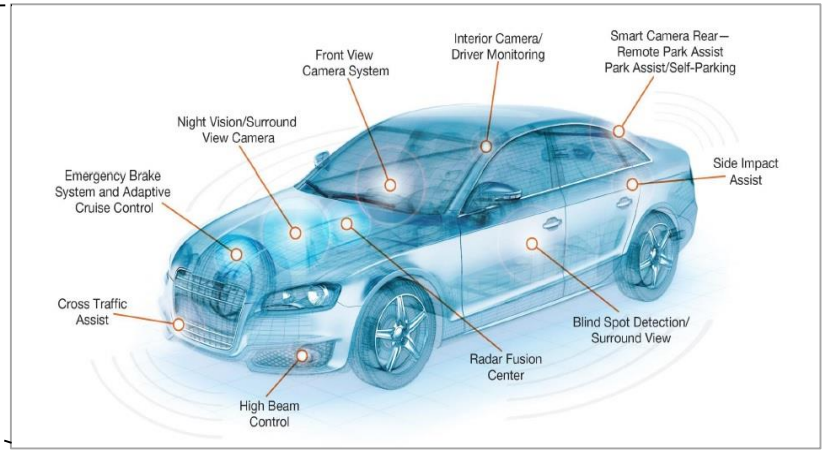
2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

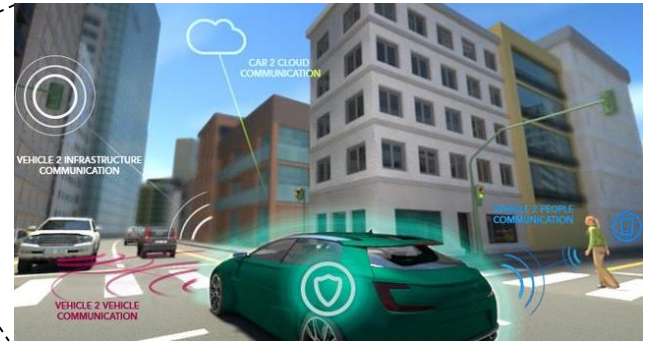
Changes in Automotive

Consumer demands are reshaping the industry

- Electrification
- ADAS/Autonomous Drive
- Smart Sensors
- Infotainment
- Connectivity

No Automation	Driver Assistance	Partial Automation	Conditional Automation	High Automation	Full Automation
Level 0	Level 1	Level 2	Level 3	Level 4	Level 5
No system	"Feet-off"	"Hands-off"	"Eyes-off"	"Brain-off"	No driver



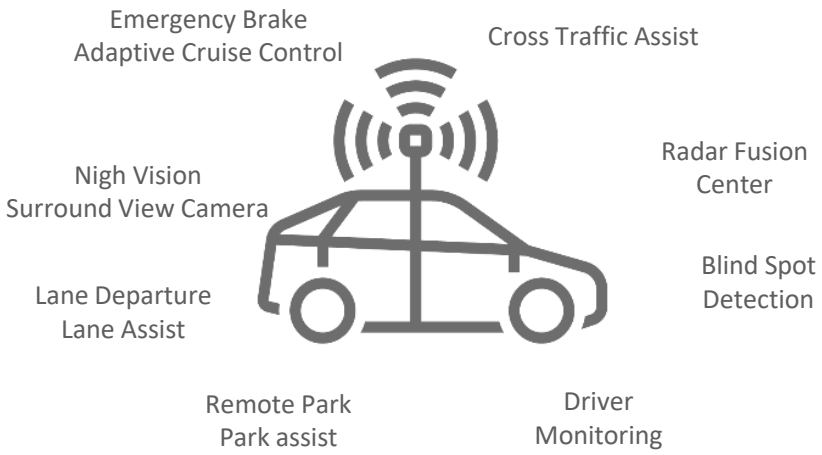


accellera SYSTEMS INITIATIVE

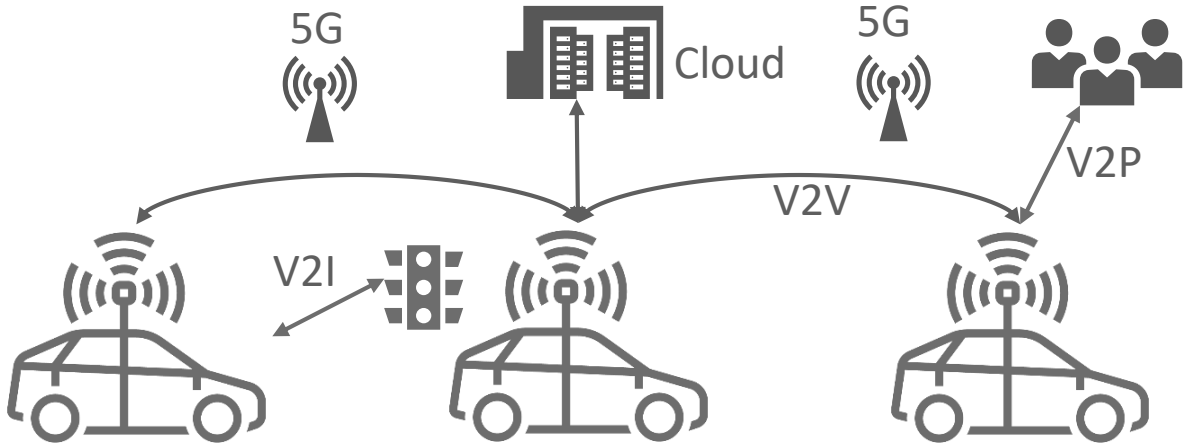
Impact on Tier1/Tier2 suppliers

To realize level 4/5, design and verification must evolve

Intelligence moving to the edge



High Level Synthesis Design
HW accelerated verification

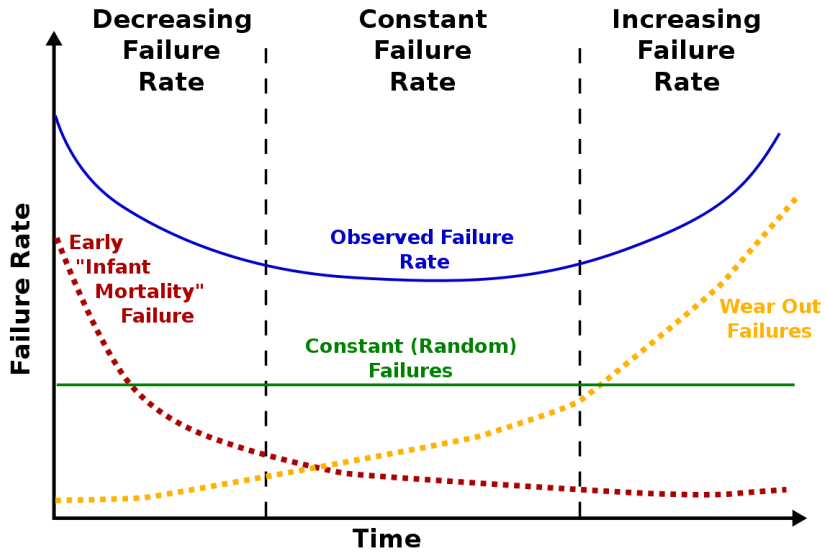


Systems level verification
Systems of Systems verification

Challenges facing Tier1/Tier2

To realize level 4/5, design and verification must evolve

- Explosion in design complexity and size to address automotive market needs
- Continually changing algorithms and sensors
- Satisfying functional requirements and ensuring functional safety
- Volume of testing required to reach Level 5 autonomy
  - Millions of tests and billions of road miles
- System and Systems of Systems testing
  - V2X
  - Sense-Compute-Actuate



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

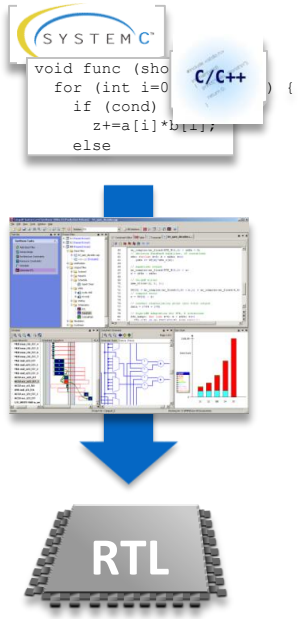
Tutorial Overview

A workflow demonstrating the use of HLS and emulation in safety-critical application

1 → 2 → 3 → 4

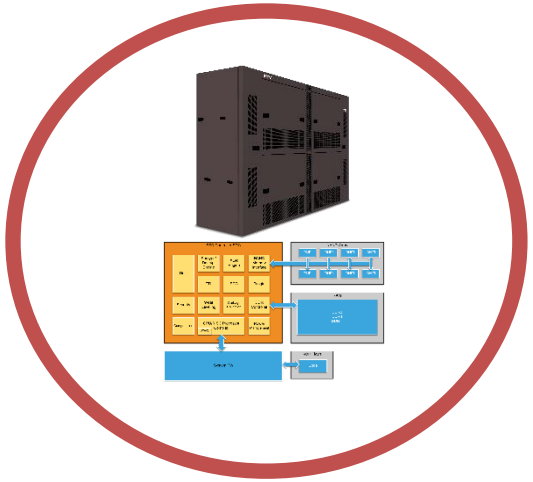
1

Accelerated design using HLS



2

Functional verification of an HLS design in emulation

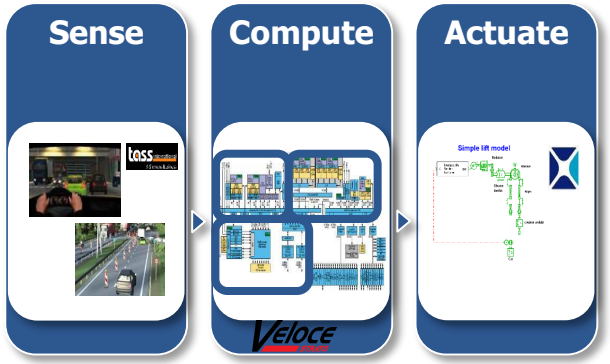


3

Seamless integration of the safety workflow

4

Accelerating system verification using emulation



accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

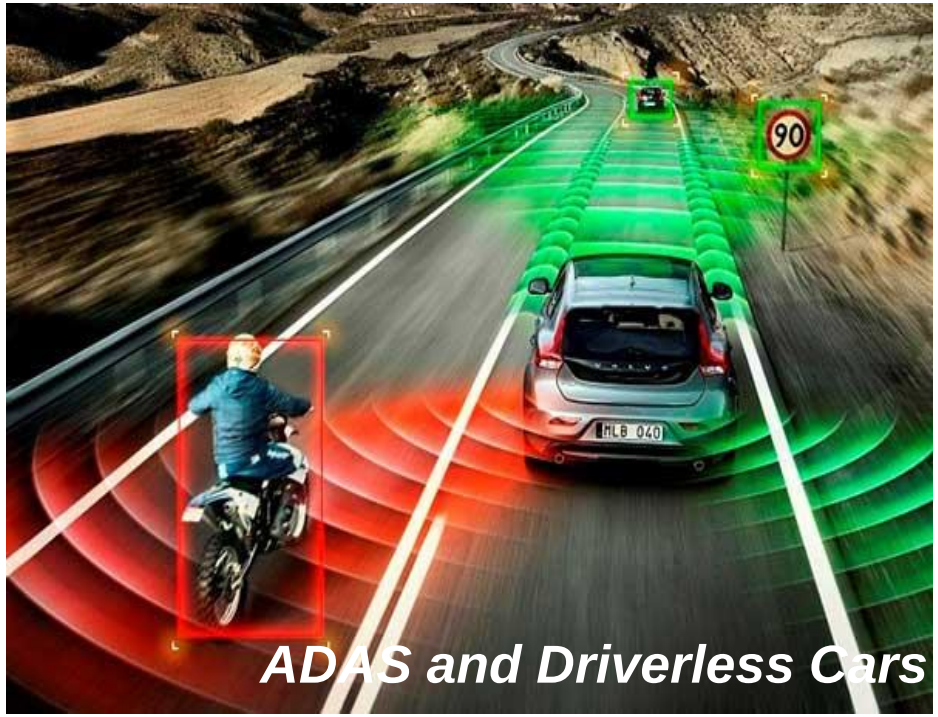
Using HLS to rapidly develop AI Algorithms

accellera SYSTEMS INITIATIVE

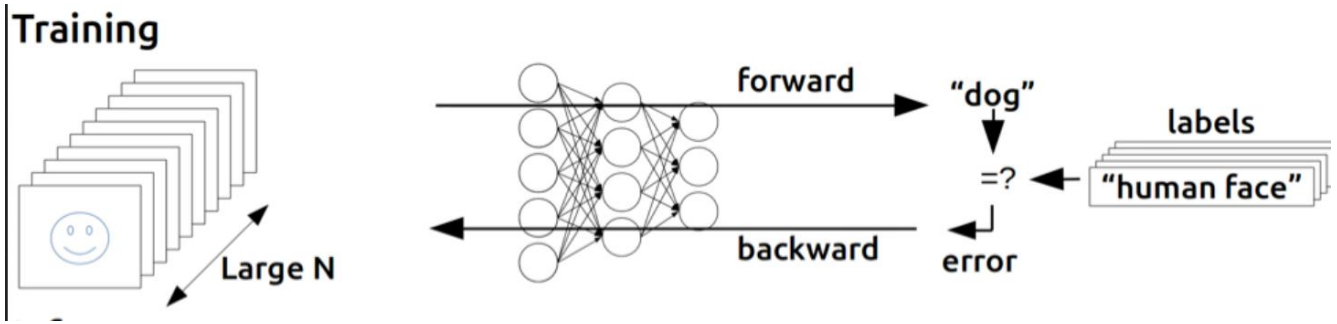
Computer Vision/AI Application Challenges

Automotive and other "real-time" application especially challenging

- Computationally very expensive
  - Billions of operations/second
- High responsiveness required
  - High-bandwidth and low-latency
  - Real-time processing of data required
- ADAS solution required to be < 100W
- Continually evolving algorithms and sensors
- Each provider wants to add their "secret sauce"

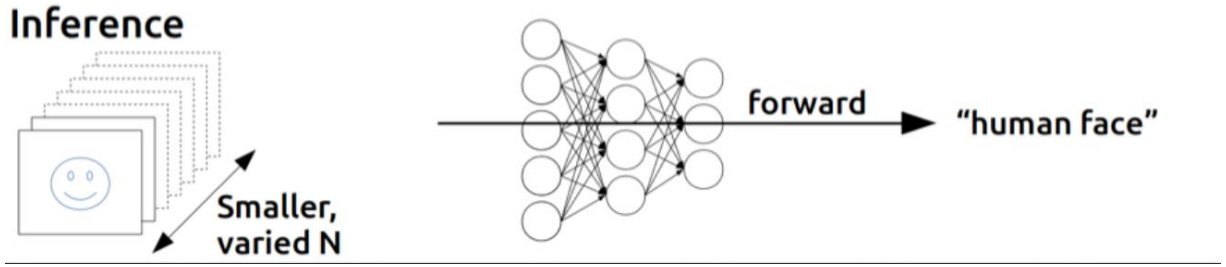


Convolutional Neural Networks: Training vs Inference (Embedded AI)



- Compute intensive, very large datasets & memory, CPU/GPU farms, floating point required

Today's focus

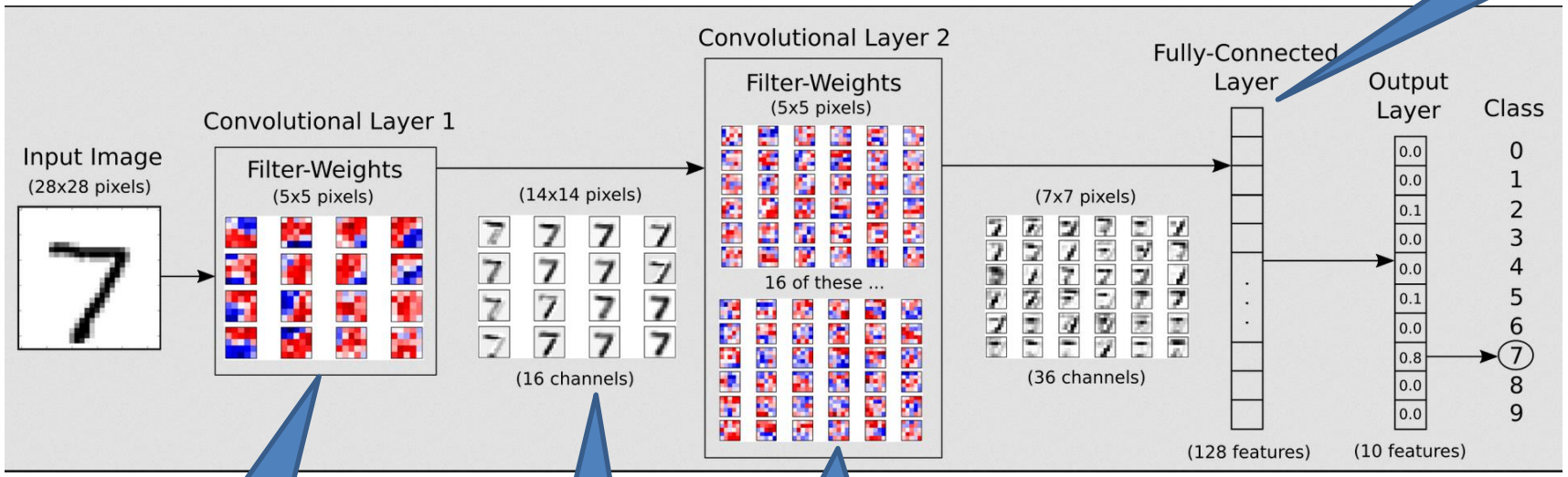


- Uses data from trained network, end system often has real-time requirements, mapping to FPGA/ASIC and/or dedicated HW, fixed point, power often a consideration

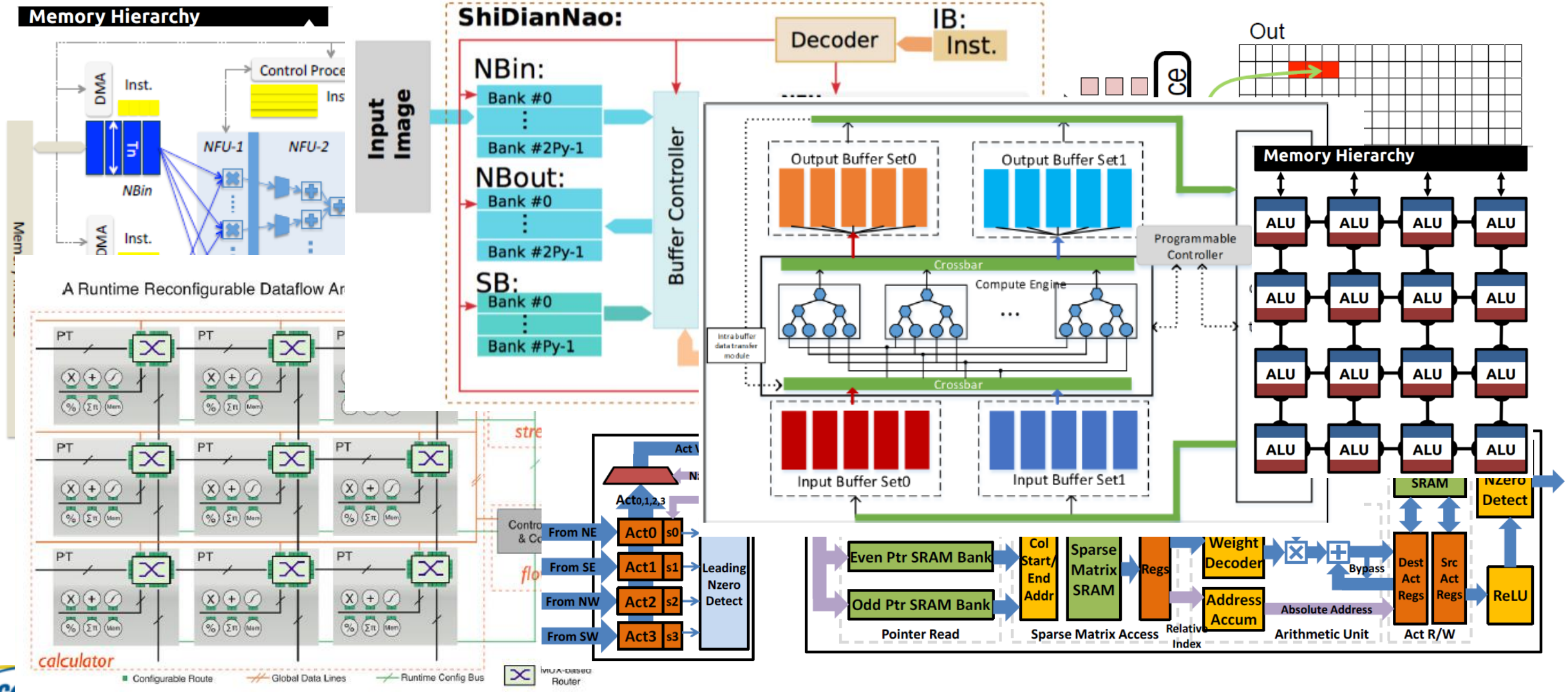
Next-generation CV Designs Require Parallelism

- Convolutional Neural Networks use lots of 2-d convolutional filters
  - Billions of multiply-accumulate operations per second
- Multiple convolutional layers
- Networks are constant evolving
  - Data rates, number of layers, image size, etc..

Simple 2-layer CNN for Digit Recognition



Numerous Hardware/Memory CNN Architectures



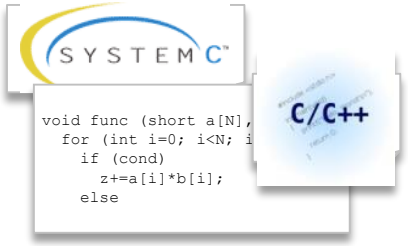
What are the Choices for Hardware Platform?

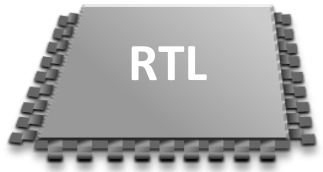
There is no clear winner today as this market is emerging

- CPU
  - Not fast or efficient enough
- DSP
  - Good at image processing but not enough performance for Deep AI
- GPU
  - Good at training but too power hungry for long term inference solution
- FPGA
  - Mostly meets performance/latency, not the lowest power, eventually cost for volume a problem, RTL flow not practical
- ASIC
  - Lowest power, meets performance/latency, lowest volume cost, high NRE and no field modifications/upgrades, Algorithms still changing, RTL flow not practical
- Dedicated AI and CV processors or accelerators in IP and ASIC
  - Popping up like weeds – high performance, locks customer in, many server target
- Some scalable combination of the above

High-Level Synthesis – Design at a Higher Level

- HLS generates high quality RTL from C++/SystemC level descriptions
  - Micro-architecture exploration is accelerated
    - Parallelism, Throughput, Latency, Area (loop unrolling and pipelining)
    - Memories vs. Registers (resource allocation)
    - Integrated Power estimation
  - Library of IP components, e.g. math, DSP, video algorithms
  - ASIC and FPGA target support



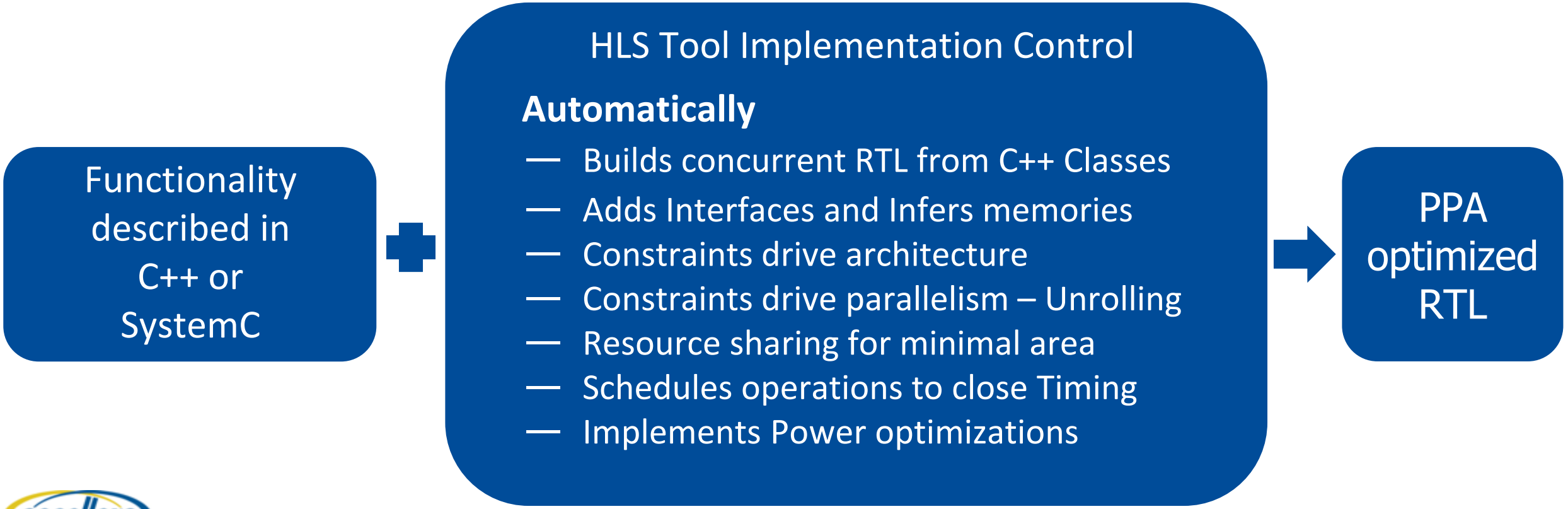


- Verify at a higher level of abstraction (HLS-ready C++ source)
  - Perform Formal checks prior to synthesis
  - Simulate 50-1000x faster
  - Achieve Code and Functional coverage goals

- Post RTL generation verification and optimization
  - SLEC HLS to formally prove C++ model and RTL equivalency
  - Integrated Power Optimization

Why HLS is So Much More Productive than RTL

- HLS separates functionality from implementation with powerful tool capabilities for controlling implementation



YOLO Tiny* *progressive refinement*

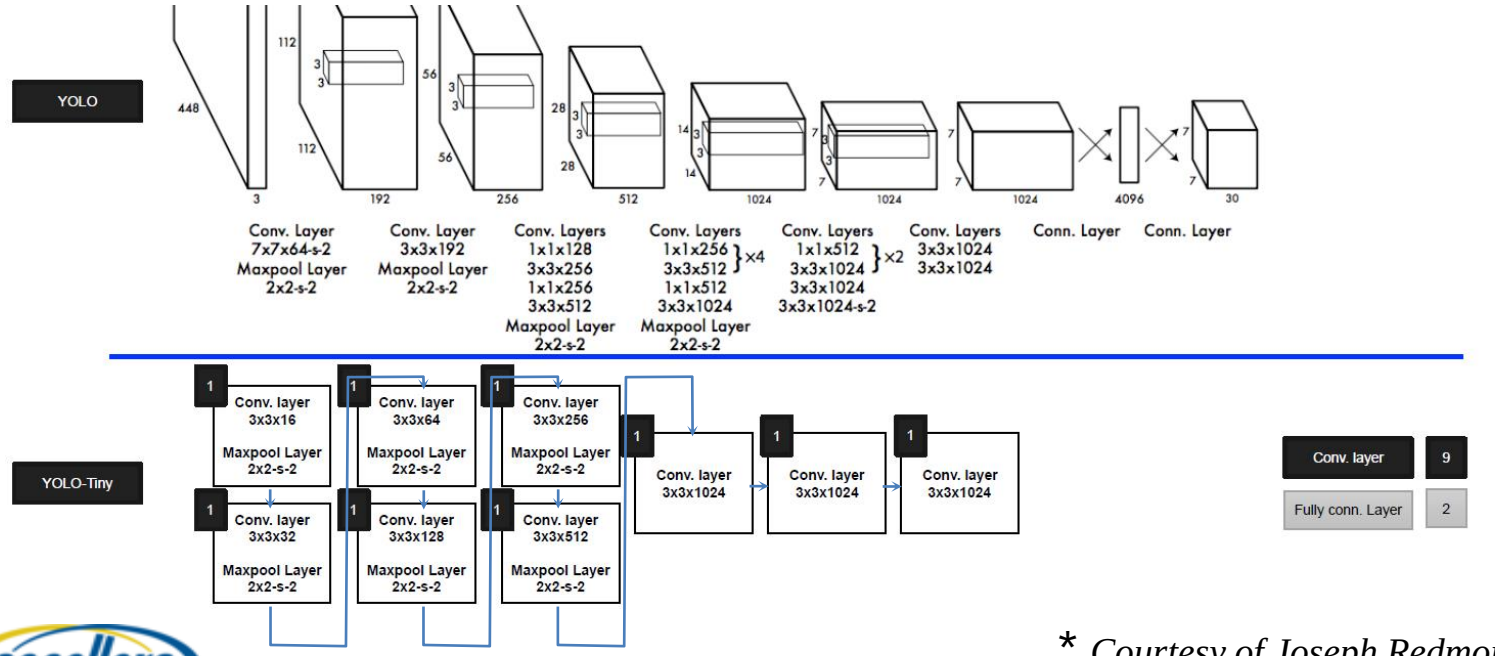
- YOLO Tiny: Real-time object detection and classification CNN
- This YOLO Tiny demo is based on the Google *TensorFlow* open-source machine learning technology based in Python
- The intent of the demo is to show techniques for progressive refinement from high-level abstracted TensorFlow CNN layer models written in Python down to HLS-synthesized RTL, i.e.,

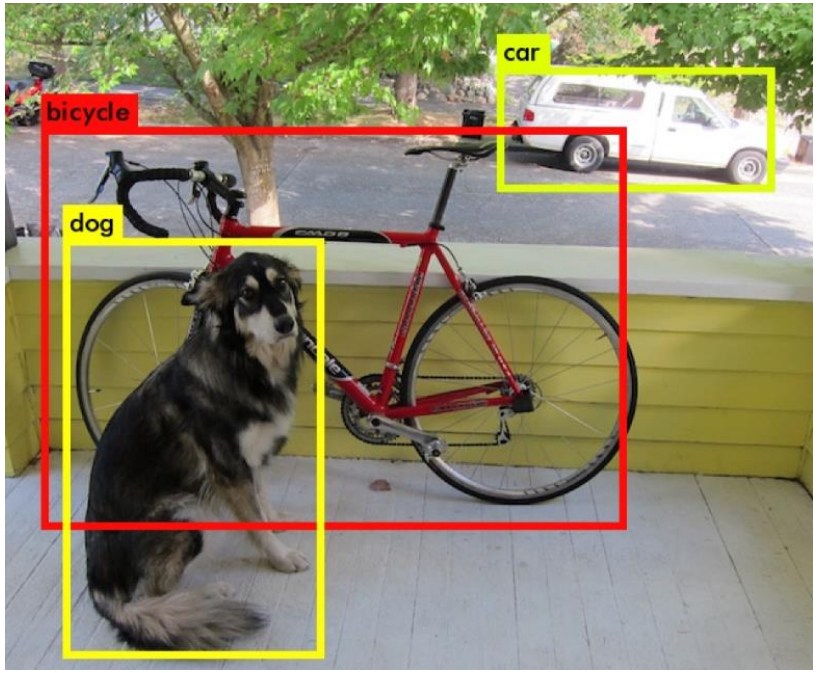
Original TensorFlow code ➡ HLS ready C++ blocks ➡ Synthesized RTL blocks

* *Courtesy of Joseph Redmon, https://pjreddie.com/darknet/YOLO*

YOLO Tiny*

- Real-time object detection and classification
  - Detects over 20 different objects
  - Yet even this “small” CNN is computationally intensive
  - Over 70 Billion MAC/s using over 25 million weights
- Made up of mostly 2-d convolution and pooling layers

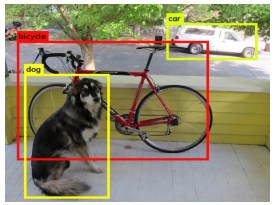




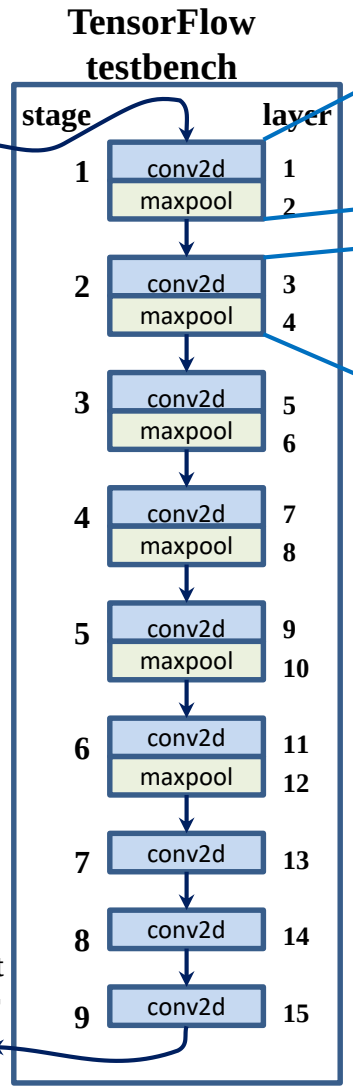
* *Courtesy of Joseph Redmon, https://pjreddie.com/darknet/yolo*

YOLOTiny: Original Python/TensorFlow testbench

input tensor **x**



output tensor **o9**



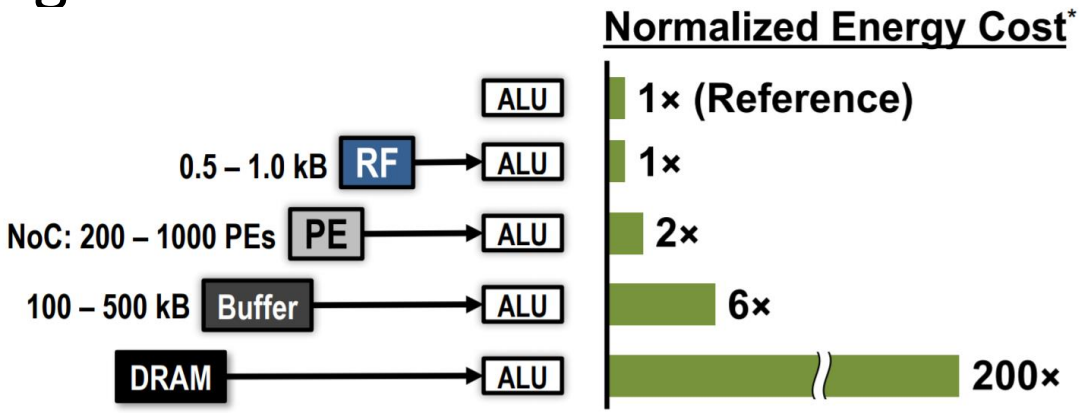

```
#1 conv1     16  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  16
w1 = weight_variable([3,3,3,16])
b1 = bias_variable([16])
h1 = tf.nn.conv2d(x, w1, strides=[1, 1, 1, 1], padding='SAME') + b1
#2 max1          2 x 2 / 2   416 x 416 x  16   ->   208 x 208 x  16
max1 = max_pool_layer(o1,kernel_size=2,stride=2,padding='VALID')
```

```
#3 conv2     32  3 x 3 / 1   208 x 208 x  16   ->   208 x 208 x  32
w2 = weight_variable([3,3,16,32])
b2 = bias_variable([32])
h2 = tf.nn.conv2d(max1, w2, strides=[1, 1, 1, 1], padding='SAME') + b2
o2 = leaky_relu(h2, relu_alpha)
n_params = n_params + 3*3*16*32 + 32*4
#4 max2          2 x 2 / 2   208 x 208 x  32   ->   104 x 104 x  32
max2 = max_pool_layer(o2,kernel_size=2,stride=2,padding='VALID')
```

- 9 stage CNN with 9 *conv2d* layers the first 6 of which are separated by *maxpool* layers which then feed densely connected *conv2d* layers
- First *conv2d* layer is fed an input tensor *'x'* which is the 2-dimensional `preprocessed_image` from the top level python *test.py* testbench
- 9th stage provides recognized images in the output tensor *'o9'* which is fed back up to top the level *test.py* for post processing of the output image, with classification and bounding box info included
- Each *conv2d* image is fed learned weights and biases for that stage
- Where preceded by a *maxpool* layer, it is fed by the output of that layer, otherwise simply the output of the preceding *conv2d* layer

Memory Architecture and Power Considerations

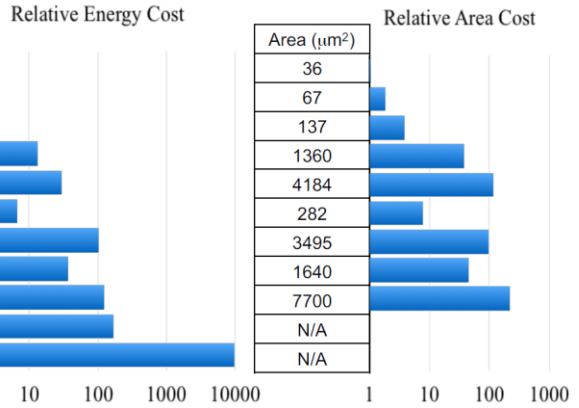
- Keeping data local is key to minimizing power consumption
  - Very important for ASIC
- Floating-point is costly
  - Used in training of networks
  - Not needed in inference engine
- Fixed-point doesn't need to be power-of-two



*NVIDIA 2017

Cost of Operations

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



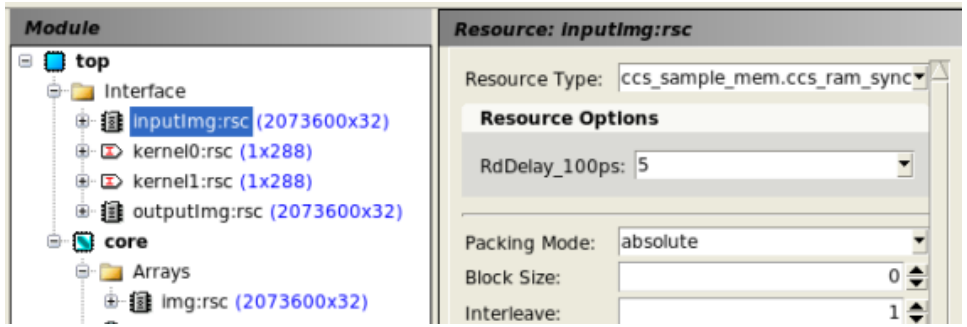
Area (μm²)
36
67
137
1360
4184
282
3495
1640
7700
N/A
N/A

Energy numbers are from Mark Horowitz "Computing's Energy Problem (and what we can do about it)", ISSCC 2014

Memory Inference and 0-time Back-Door Memory Accesses

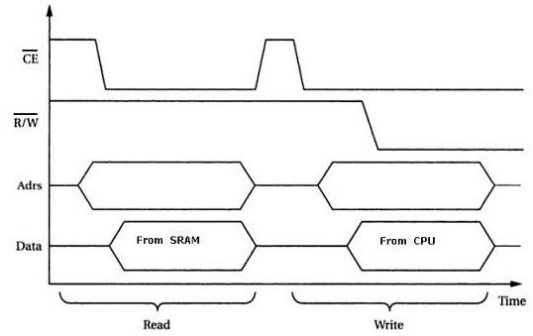
- Large C++ arrays automatically mapped to ASIC or FPGA memories
- Arrays on the design interface can be synthesized as memory interfaces
- Internal arrays synthesized to instantiated (black boxed) memories

Catapult Architectural Constraints View



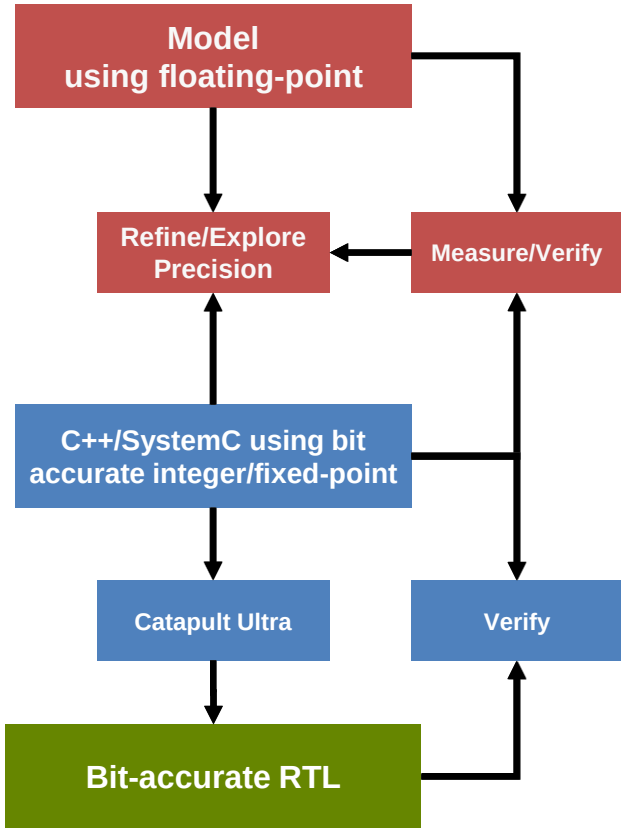

```
void top(int inputImg[1080][1920], int kernel0[3][3],
int kernel1[3][3], int outputImg[1080][1920]){
int img[1080][1920];

conv2d(inputImg,kernel0,img);
conv2d(img,kernel1,outputImg);
}
```

Memory interface protocol



Precise Modeling of Bit-accuracy

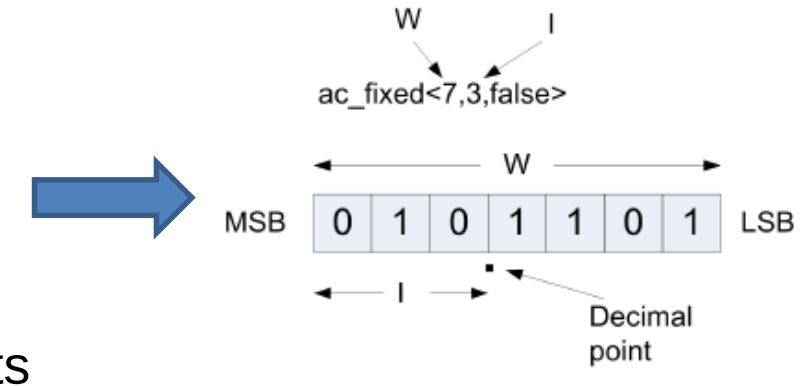
- HLS uses exact bit-widths to meet specification and save power/area
  - bit-widths are not always pow2 (4, 8, 16, 32, 64 bits)
- Rapid simulation of true hardware behavior
- RTL is correct by construction
  - Precise consistency of representation and simulation results between C++ algorithm and synthesized RTL



The Algorithmic C fixed point data types are declared as:

```
ac_fixed<W,I,S> x;
```

width #integer bits



2-D Convolution with Windowing IP

```
ac_window2d<uint8,3,1080,1920,1,AC_MIRROW_101> window;
ac_flags_gen<1080,1920> flags;

FRAME:do{
  if(window.canRead())
    data_in = input.read();

  flags.generate(data_in.TUSER, data_in.TLAST,sof,eof,sol,eol);

  window.slide_window(data_in,sof,eof,sol,eol);

  if(window.isValid()){
    KERNEL_Y:for(int i=0;i<3;i++){
      KERNEL_X:for(int j=0;j<3;j++){
        acc += window[i][j] * kernel[i][j];
    data_out.write(acc);
  }
}while(!window.eof)
```

Sliding window class

Framing signal generator class (sof,eof,sol,eol)

Sliding window class controls data reads

AXI4 streaming interface class read

Flag generator uses sof and eol from AXI4 video stream

Advance the window

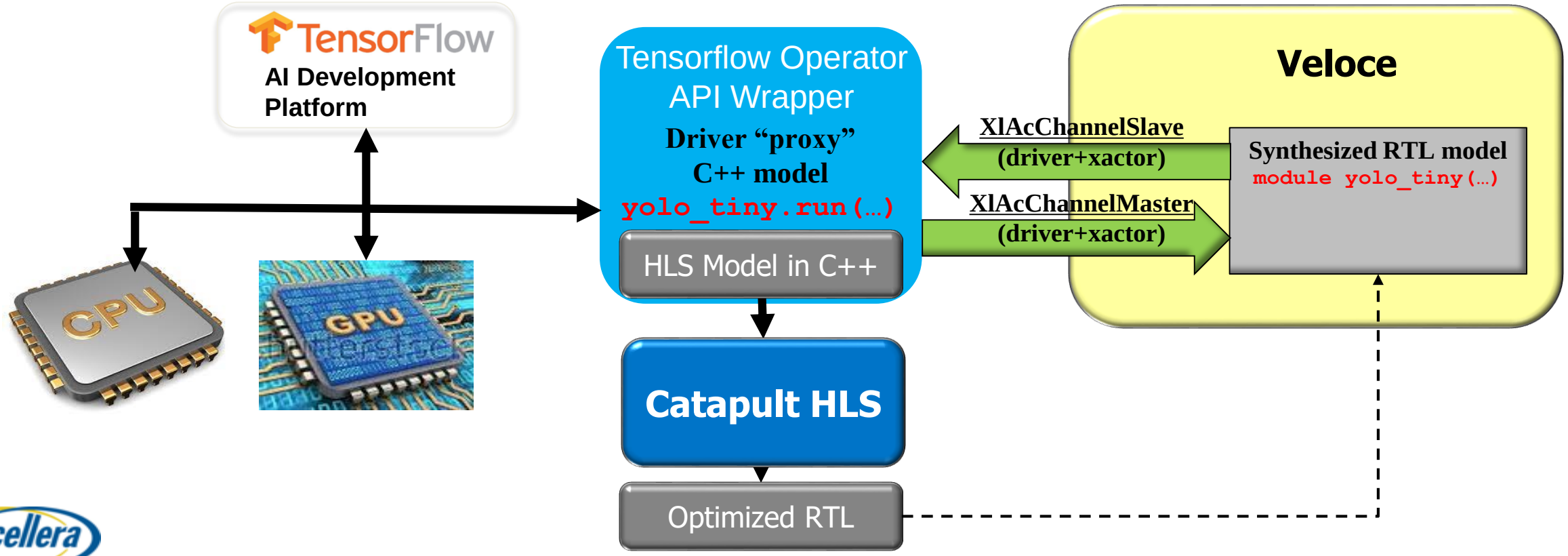
Sliding window indicates when valid data available

Convolution using window data

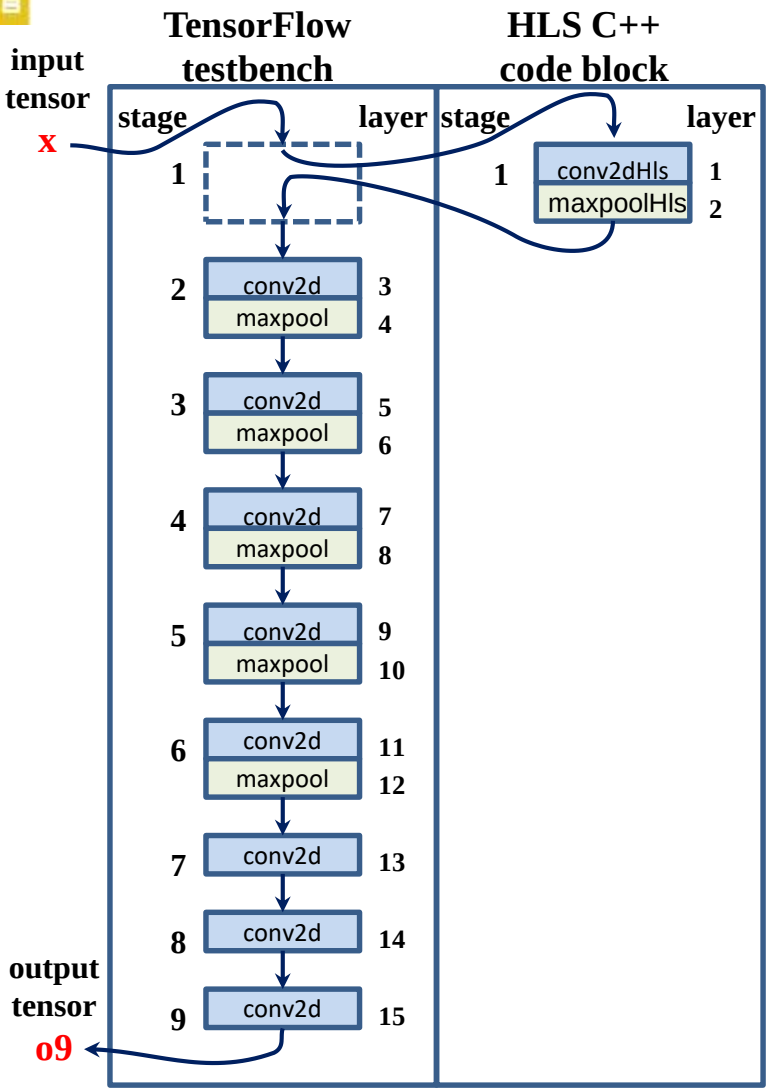
Run until end-of-frame detected

Catapult and Veloce Solve the Verification Bottleneck

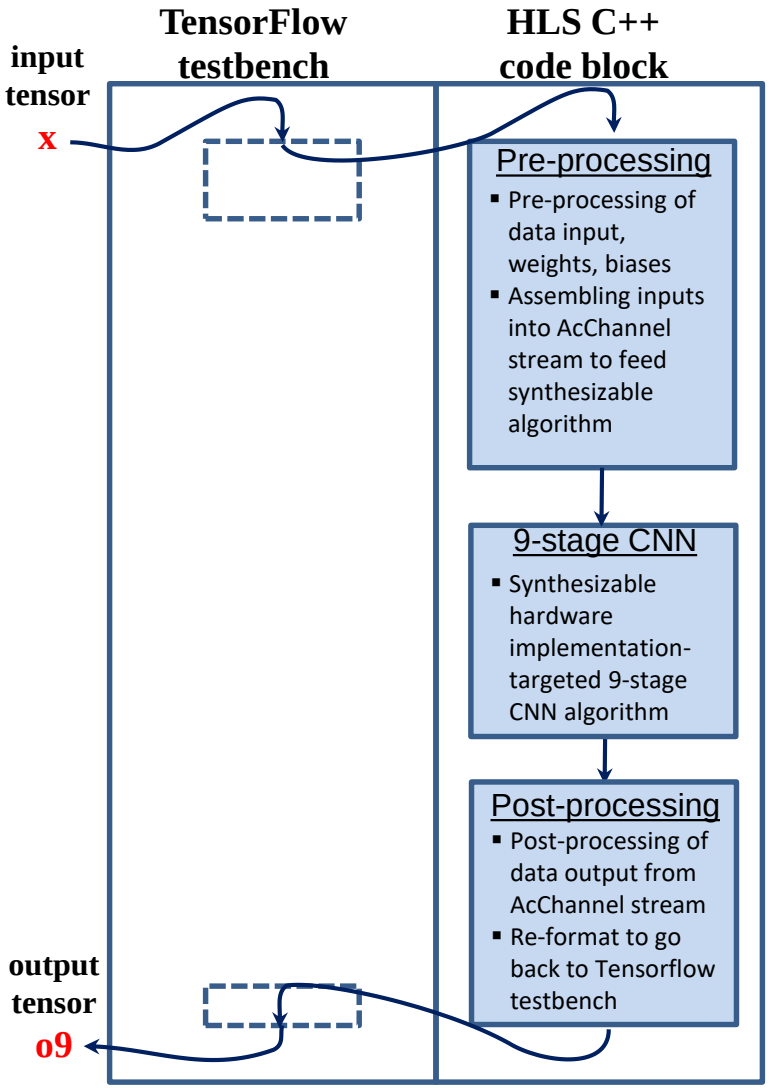
- Quickly verify synthesizable HLS C++ and RTL in the Tensorflow environment
  - Test the quantized HLS against the floating point model in tensorflow
- Reduce RTL verification from hours to minutes



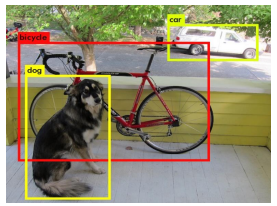
YOLOTiny: Selected layers of TensorFlow testbench broken out to HLS-ready C++ implementation-targeted algorithms



1-stage breakout

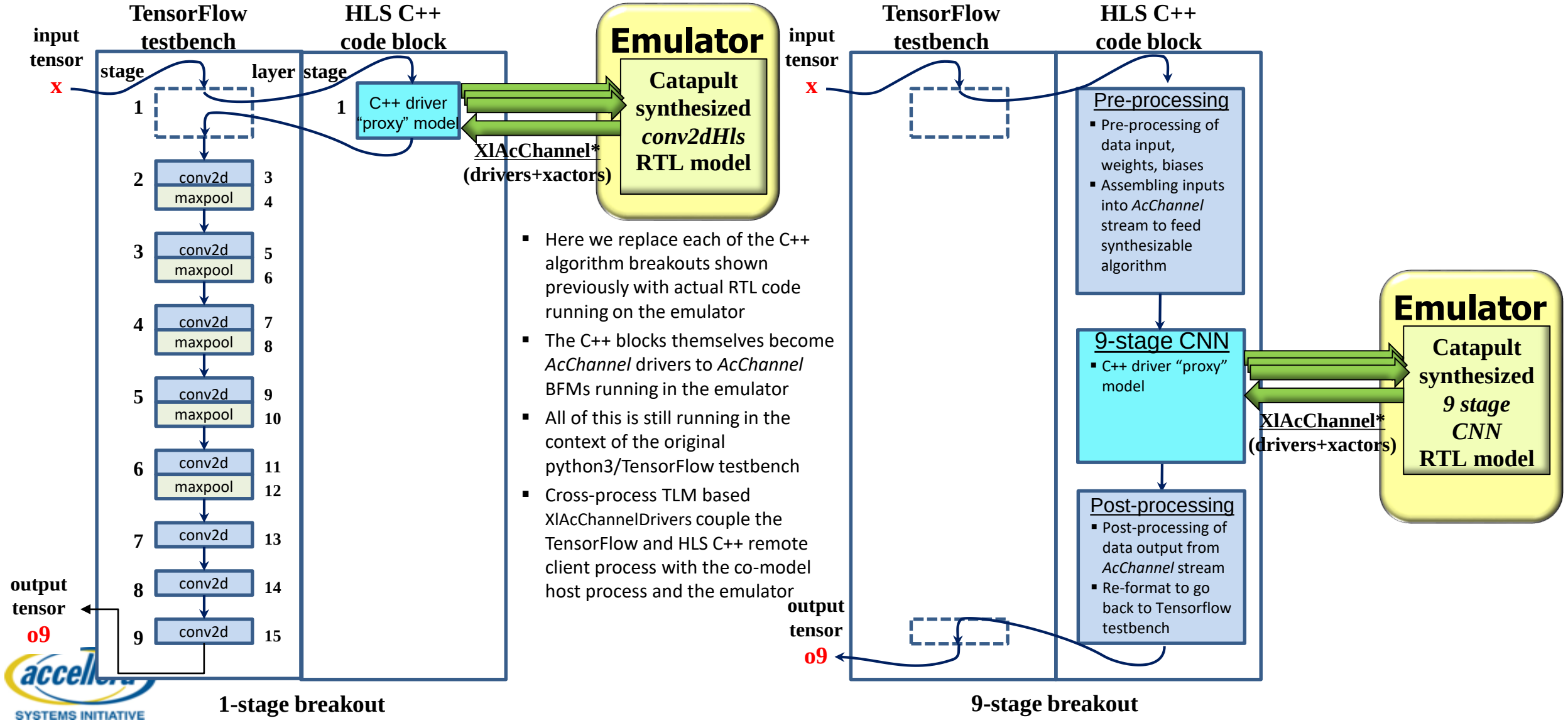


9-stage breakout



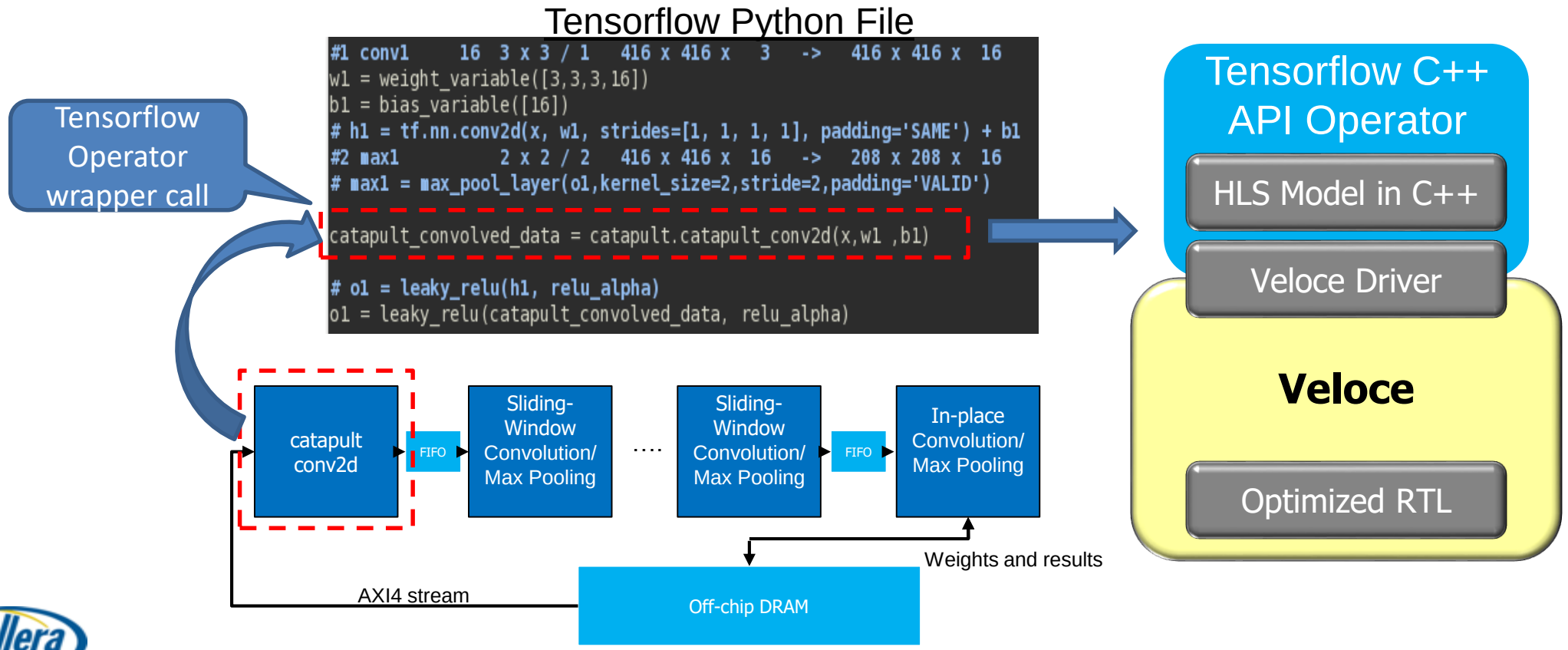
- Here we break out 1 or more of the original TensorFlow layers to experiment with implementation synthesis
- We still run the new C++ code prototypes in the context of the original TensorFlow testbench
- We pre-verify the synthesizable code even before we generate RTL

YOLOTiny: C++ implementations of CNNs replaced with synthesized RTL blocks



Easily Test HW Models and RTL Quickly

- Swap any layer or the entire design
  - HLS C++ executable or RTL running on Veloce is a python function call in tensorflow



Memory Access Bottleneck Changes Between Layers

- Some CNN Layers have millions of coefficients
  - Not possible to store everything locally for all layers
- Every layer is different
  - Weight vs feature map storage

YOLO Tiny CNN

Layer	Input Channels	Output Channels	Feature Map Size	Weight Mem(bytes)	Feature Mem(bytes)	Line Buffer Memory	MAC/sec	Unroll Factor Needed for 30 frames/sec
1	3	16	448	432	602112	4032	2,601,123,840.00	8.6704128
2	16	32	224	4608	802816	10752	6,936,330,240.00	23.1211008
3	32	64	112	18432	401408	10752	6,936,330,240.00	23.1211008
4	64	128	56	73728	200704	10752	6,936,330,240.00	23.1211008
5	128	256	28	294912	100352	10752	6,936,330,240.00	23.1211008
6	256	512	14	1179648	50176	10752	6,936,330,240.00	23.1211008
7	512	1024	7	4718592	25088	10752	6,936,330,240.00	23.1211008
8	1024	1024	7	9437184	50176	21504	13,872,660,480.00	46.2422016
9	1024	1024	7	9437184	50176	21504	13,872,660,480.00	46.2422016
				97200	275968	36288		

Store weights locally for first 4 layers

Store feature maps locally for last 5 layers

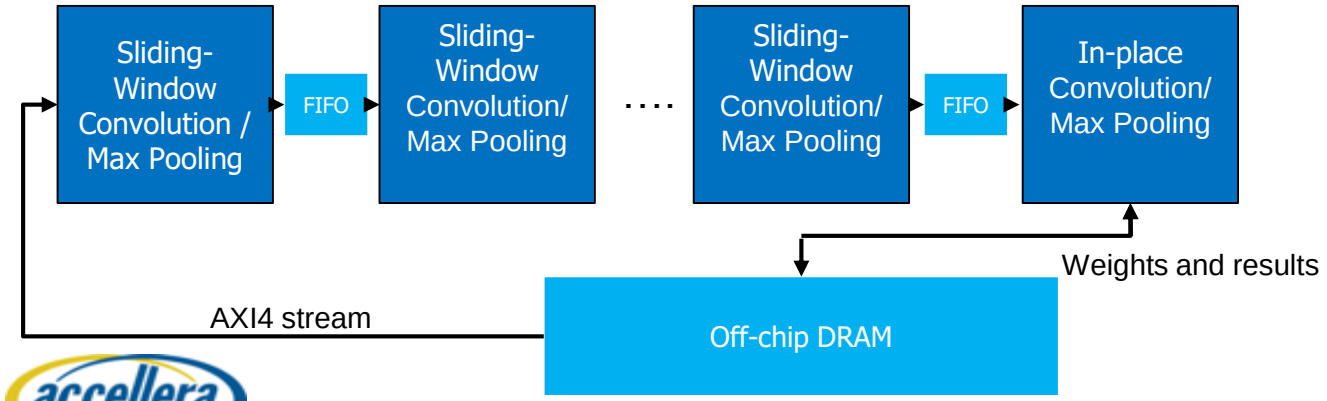
9M coefficients in final 2 layers each

Total Mem 409456

Quickly Explore CNN Architectures Using HLS

- HLS constraints allow architectural exploration
  - Massive parallelism is possible (if fed efficiently)
  - Evaluate PPA across multiple architectures
- Easily code multiple architectures in C++
  - Sliding-window architecture processes fmap data in raster order
  - In-place architecture reads weights once

YOLO Tiny



Original Algorithm

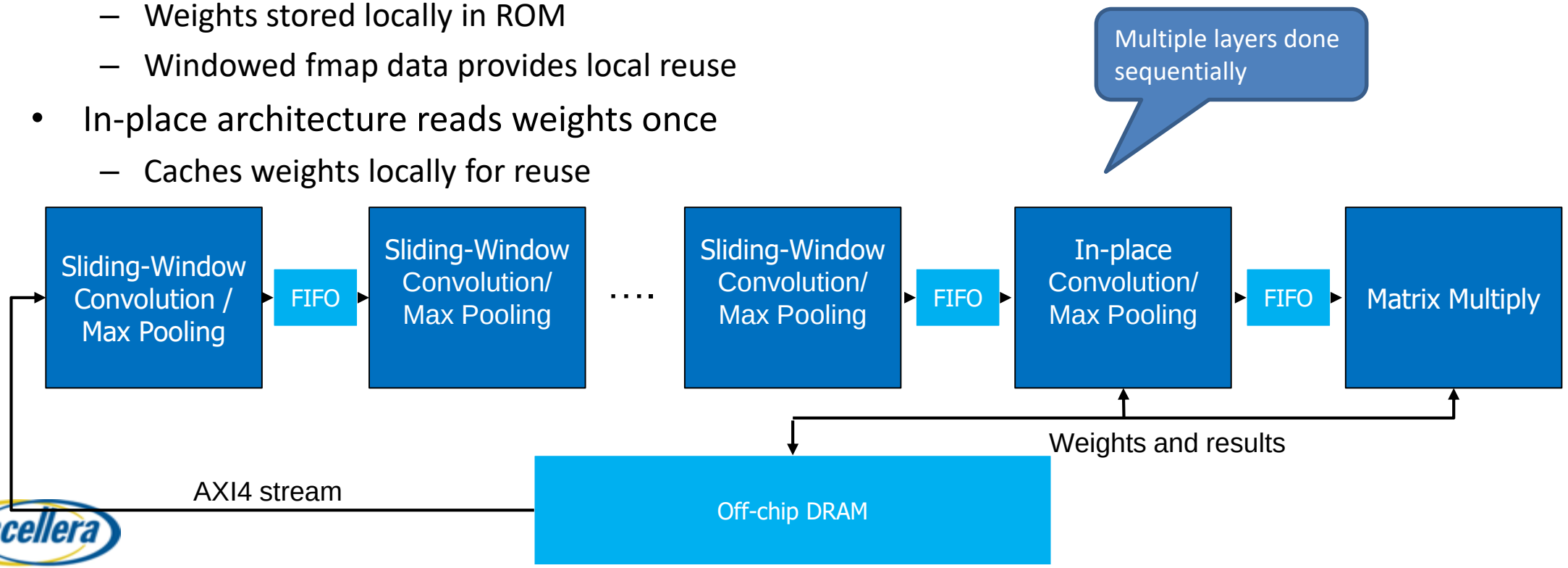
```
OUT_CHAN:for(int oc=0;oc<OUT_CHANNELS;oc++){
  FMAP_HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
    FMAP_WIDTH:for(int c=0;c<IN_WIDTH+1;c++){
      IN_CHAN:for(int ic=0;ic<IN_CHANNELS;ic++){
        KERNEL_Y:for(int i=0;i<3;i++){
          KERNEL_X:for(int j=0;j<3;j++){
            acc+=fmap[ic][r-i/2][c-j/2]*kernel[ic][oc][i][j];
          }
        }
      }
      fmap_out[d][r][c] = acc;
    } } }
```

Reordered Loops (Layers 1-4)

```
FMAP_HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
  IN_CHAN:for(int ic=0;ic<IN_CHANNELS;ic++){
    FMAP_WIDTH:for(int c=0;c<IN_WIDTH+1;c++){
      < Read feature map data stream >
      < Sliding window of feature map data >
      OUT_CHAN:for(int oc=0;oc<OUT_CHANNELS;oc++){
        < Read kernel weights from SRAM >
        KERNEL_Y:for(int i=0;i<3;i++){
          KERNEL_X:for(int j=0;j<3;j++){
            acc += fmap_window[r+i][c+j] * kernel[i*3+j];
          }
        }
        < Write out partial output channel sums >
      } } }
```

Hybrid Architecture for Minimizing RAM Access

- Dual-layer architecture
  - Based of feature map and weight storage requirements
  - DRAM traffic is minimized
- Sliding-window architecture processes fmap data in raster order
  - Weights stored locally in ROM
  - Windowed fmap data provides local reuse
- In-place architecture reads weights once
  - Caches weights locally for reuse



Sliding Window Architecture (Layers 1-4)

- Reordering Loops Facilitates Loop Unrolling
- Use “sliding window” to store fmap data locally
- Small number of weights, store locally

Original Algorithm

```
OUT_CHAN:for(int oc=0;oc<OUT_CHANNELS;oc++){
  FMAP_HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
    FMAP_WIDTH:for(int c=0;c<IN_WIDTH+1;c++){
      IN_CHAN:for(int ic=0;ic<IN_CHANNELS;ic++){
        KERNEL_Y:for(int i=0;i<3;i++){
          KERNEL_X:for(int j=0;j<3;j++){
            acc+=fmap[ic][r-i/2][c-j/2]*kernel[ic][oc][i][j]
          }
        }
      }
      fmap_out[d][r][c] = acc;
    }
  }
}
```

Reordered Loops

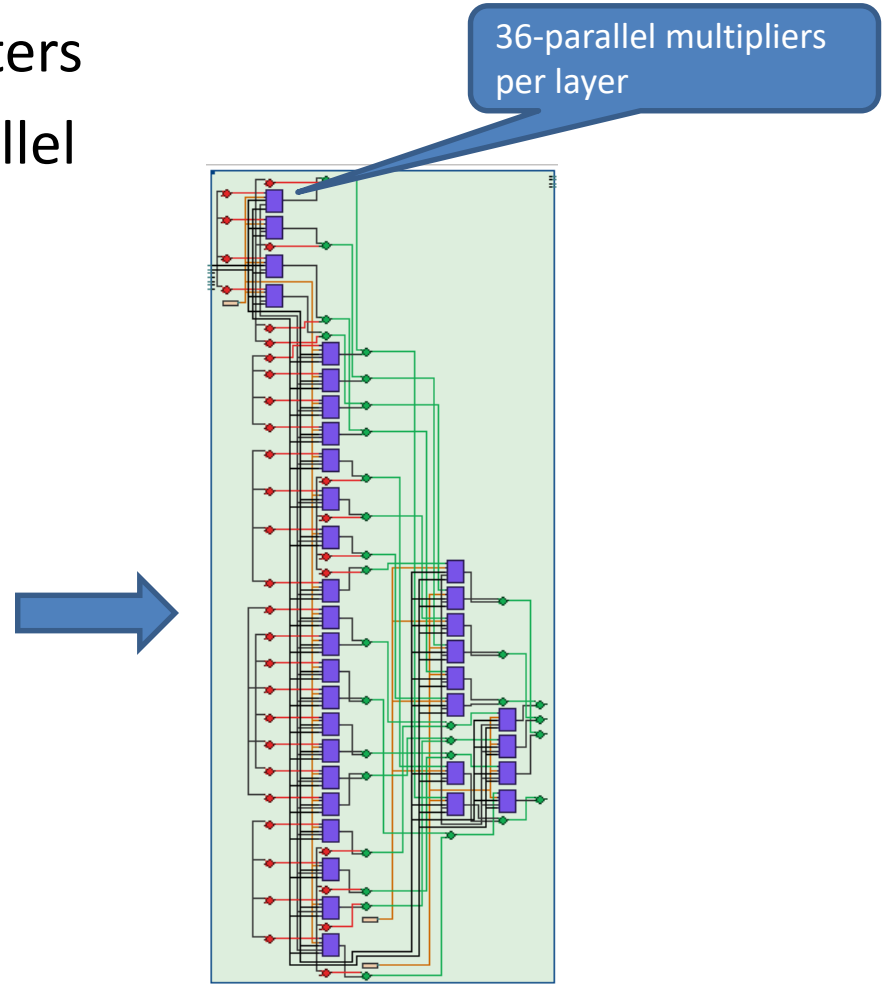
```
FMAP_HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
  IN_CHAN:for(int ic=0;ic<IN_CHANNELS;ic++){
    FMAP_WIDTH:for(int c=0;c<IN_WIDTH+1;c++){
      < Read feature map data stream >
      < Sliding window of feature map data >
      < stationary data over output channels >
      OUT_CHAN:for(int oc=0;oc<OUT_CHANNELS;oc++){
        < Read kernel weights from SRAM >
        KERNEL_Y:for(int i=0;i<3;i++){
          KERNEL_X:for(int j=0;j<3;j++){
            acc += fmap_window[i][j] * kernel[i*3+j];
          }
        }
        < Write out partial output channel sums >
      }
```

Sliding Window Architecture

- "windowed" fmap data and kernel data stored in registers
- Multiple output channel data can be computed in parallel

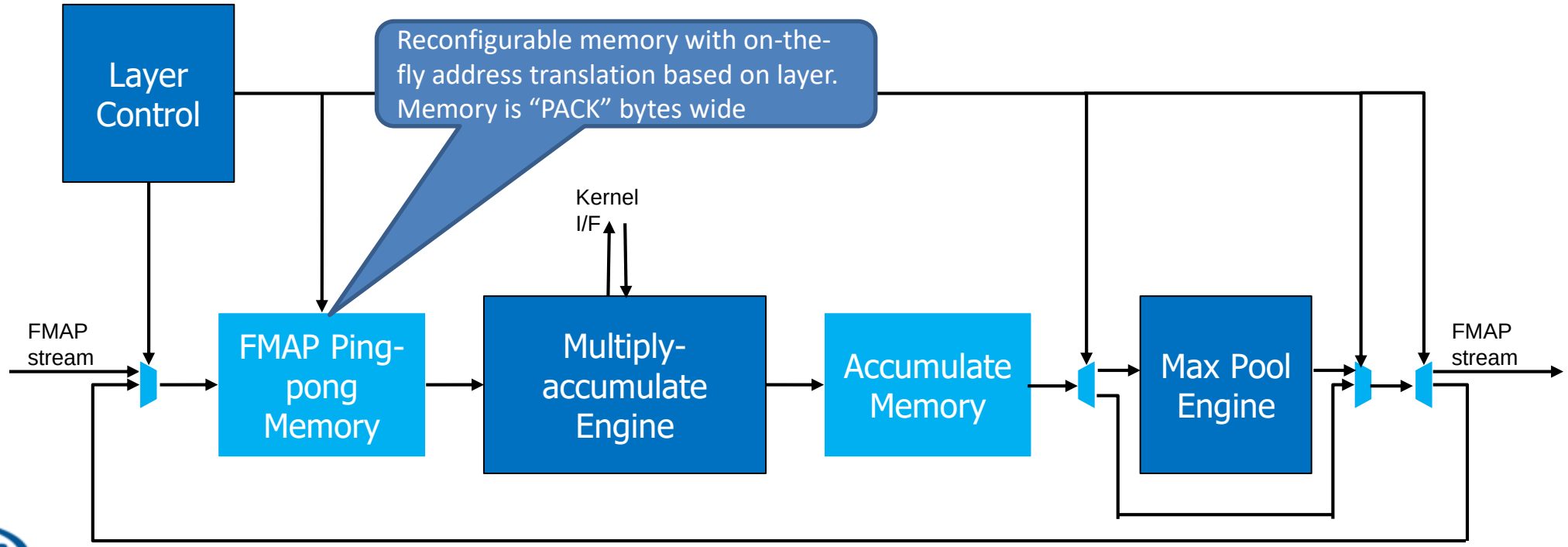
```
FMAP_HEIGHT:for(int r=0;r<IN_HEIGHT;r++){
  IN_CHAN:for(int ic=0;ic<IN_CHANNELS;ic++){
    FMAP_WIDTH:for(int c=0;c<IN_WIDTH+1;c++){
      < Read feature map data stream >
      < Sliding window of feature map data >
      OUT_CHAN:for(int oc=0;oc<OUT_CHANNELS;oc++){
        < Read kernel weights from SRAM >
        KERNEL_Y:for(int i=0;i<3;i++){
          KERNEL_X:for(int j=0;j<3;j++){
            acc += fmap_window[r+i][c+j] * kernel[i*3+j];
          }
        }
        < Write out partial output channel sums >
      }
    }
  }
}
```

Loops can be unrolled



In-place Architecture (Layers 5-9)

- Layers processed one after another
- Feature maps stored locally in SRAM
- Weights read from system memory



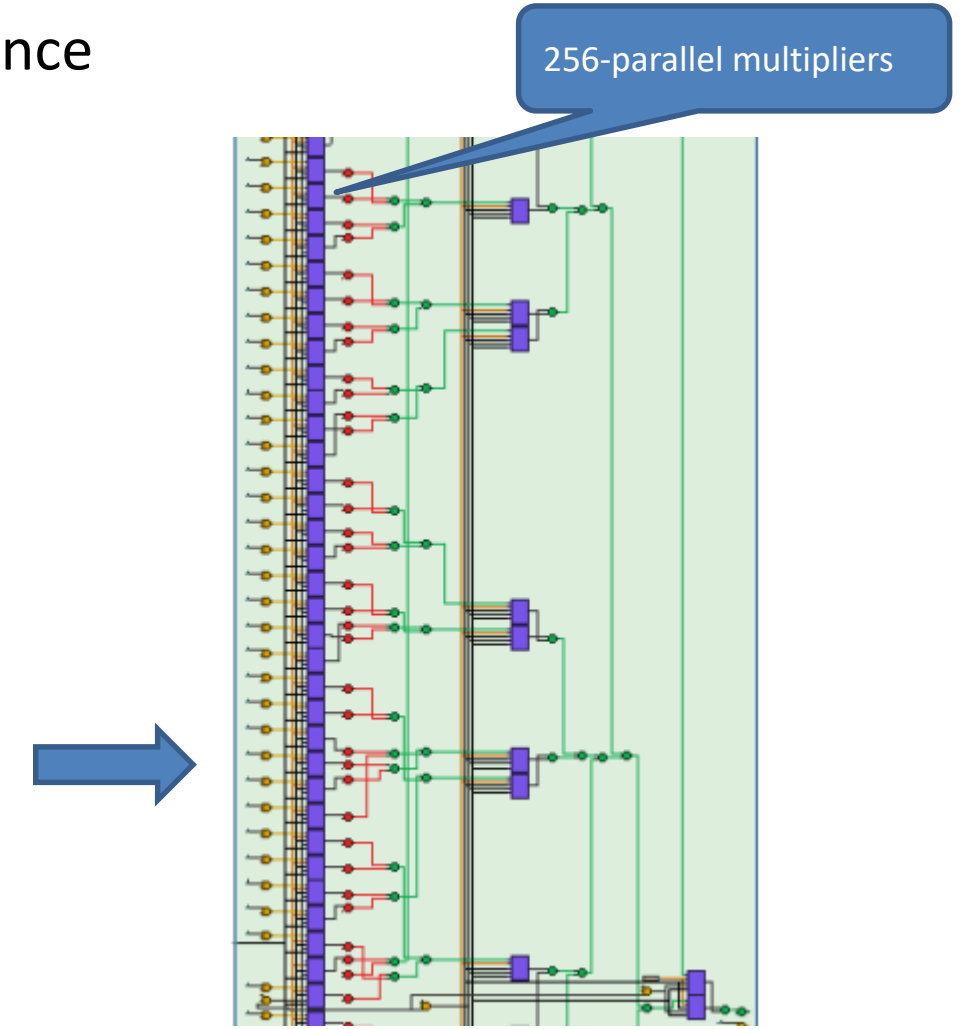
Reordering Loops to Keep Feature Map Data Local

- Loops are organized so that weights are only read once from system DRAM
- Weights are held stationary across feature maps
- Feature maps are computed in order and stored in local SRAM

Reordered Loops

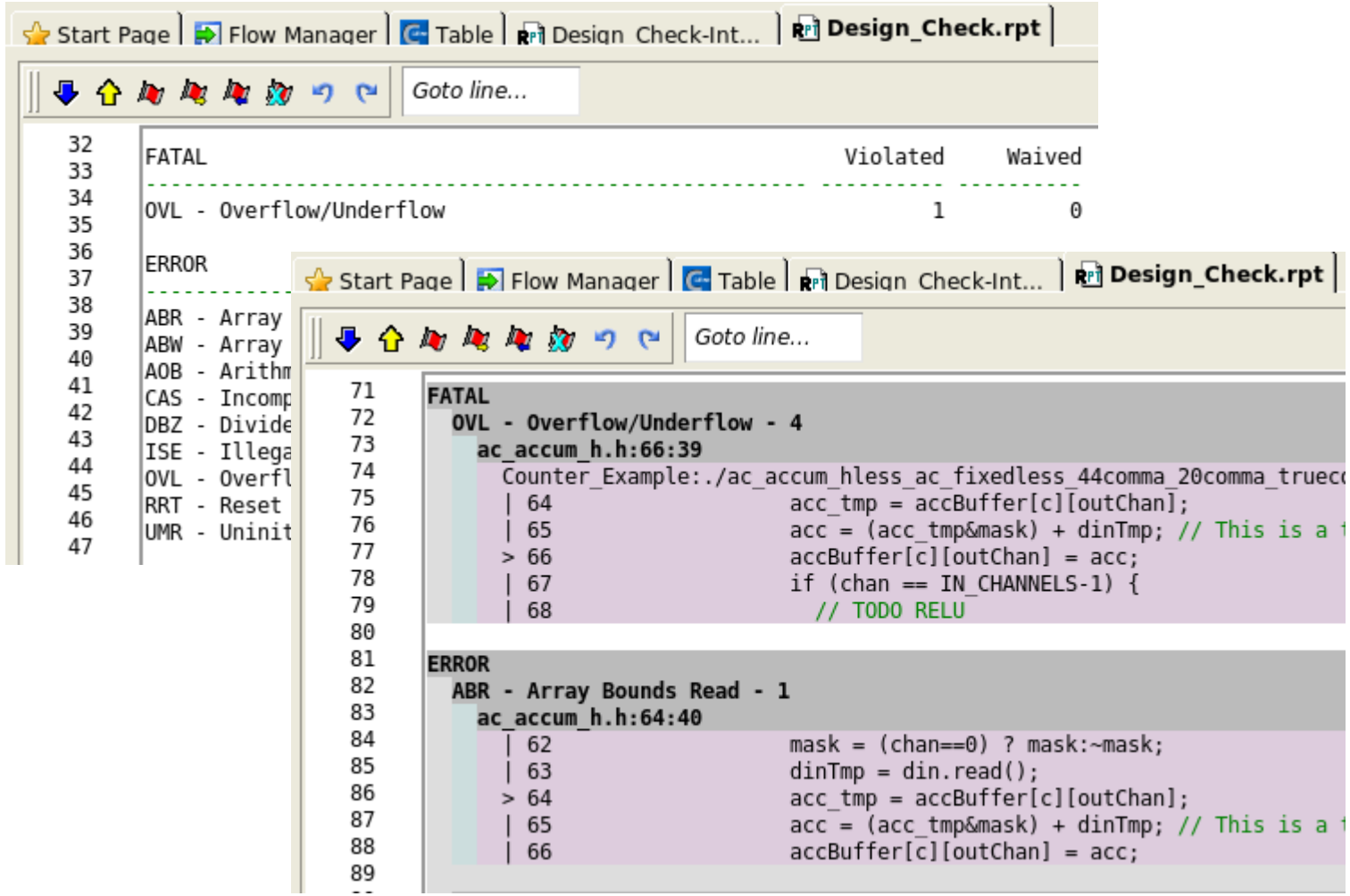
```
OUT_CHAN:for(int oc=0;oc<MAX_OUT_CHANNELS;oc++){
  KERNEL_Y:for(int i=-1;i<2;i++){
    KERNEL_X:for(int j=-1;j<2;j++){
      FMAP_HEIGHT:for(int r=0;r<MAX_HEIGHT;r++){
        FMAP_WIDTH:for(int c=0;c<MAX_WIDTH;c++){
          IN_CHAN:for(int ic=0;ic!=MAX_IN_CHANNELS;ic+=PACK){
            < FMAP ping-pong memory read PACK values >
            < Read and cache weights once for reuse >
            MAC:for(int p=0;p<PACK;p++)
              acc += fmap_data[p] * kernel_data[p];
          }
          acc_mem[r][c] += acc;
    } } } }
```

Loop can be unrolled



Flush out issues via formal analysis of HLS design source

- Quickly and easily find coding bugs and errors before synthesis or simulation
- Certain C++ language behavior is ambiguous for hardware
  - Lead to mismatches between C++ and RTL simulation
  - Difficult to debug
- Combination of static “lint” and formal based checks plus QoR checks, e.g.
  - Un-initialized memory read
  - Out of bounds reads and writes
  - Accumulator of native C type



Achieve Coverage Closure on HLS design source

- Bringing RTL coverage to HLS
  - C++ and SystemC design source
- Match coverage concepts from RTL
  - Statement, Branch, FEC and Toggle
  - Functional Coverage including covergroups, coverpoints, bins, crosses
- Synthesis Aware Coverage
  - Function inlining/instances
  - Loop unrolling
- Coverage Data saved within UCDB
  - Test plan integration and merging
  - Track progress towards goals
  - Prevent Systematic Faults

Coverage Summary By Instance (78.11%)

Page Size: 10

Instance ↑	Branches	Expressions	Statements	Total
Search...	Search...	Search...	Search...	Search...
Total	80.48%	55.31%	98.54%	78.11%
\simpleCNN::run_73_3	-	-	100%	100%
\arb_resp::run[with T = a...	95.83%	100%	100%	98.61%
\conv1::run[with Tfdataln ...	87.17%	0%	100%	62.39%
\conv2::run[with Tfdataln ...	87.17%	0%	100%	62.39%

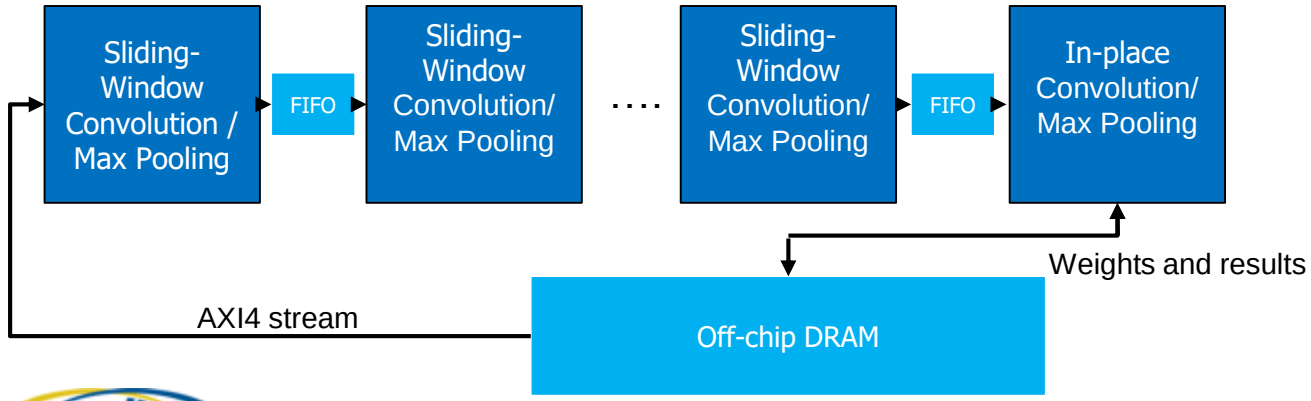
Covergroups Coverage (62.5%)

Covergroups	Bins	Hits	Goal	Coverage
main_23::MyCCoverGroup...	8	5	100	62.5%
MyCCoverGroup_1_inst	8	5	0	62.5%

Quickly Implement CNN Architectures Using HLS

- Easily code multiple architectures in C++
  - Sliding-window architecture processes fmap data in raster order
  - In-place architecture reads weights once
- HLS constraints allow architectural exploration
  - Massive parallelism is possible
  - Evaluate power, performance, and area PPA across multiple architectures and microarchitectures

YOLO Tiny

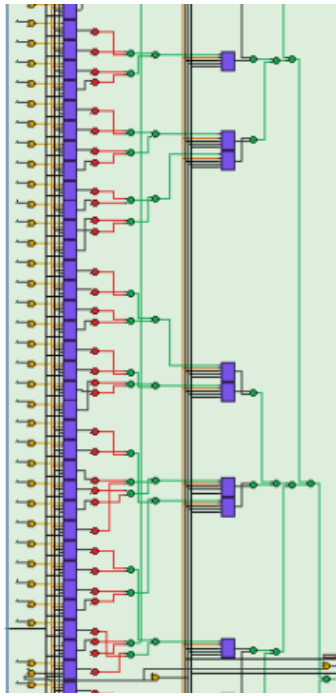


Reordered Loops (Layers 5-9)

```
OUT_CHAN:for(int oc=0;oc<MAX_OUT_CHANNELS;oc++){
  KERNEL_Y:for(int i=-1;i<2;i++){
    KERNEL_X:for(int j=-1;j<2;j++){
      FMAP_HEIGHT:for(int r=0;r<MAX_HEIGHT;r++){
        FMAP_WIDTH:for(int c=0;c<MAX_WIDTH;c++){
          IN_CHAN:for(int ic=0;ic!=MAX_IN_CHANNELS;ic+=PACK){
            < FMAP ping-pong memory read PACK values >
            < Read and cache weights once for reuse >
            MAC:for(int p=0;p<PACK;p++)
              acc += fmap_data[p] * kernel_data[p];
          }
          acc_mem[r][c] += acc;
        }
      }
    }
  }
}
```

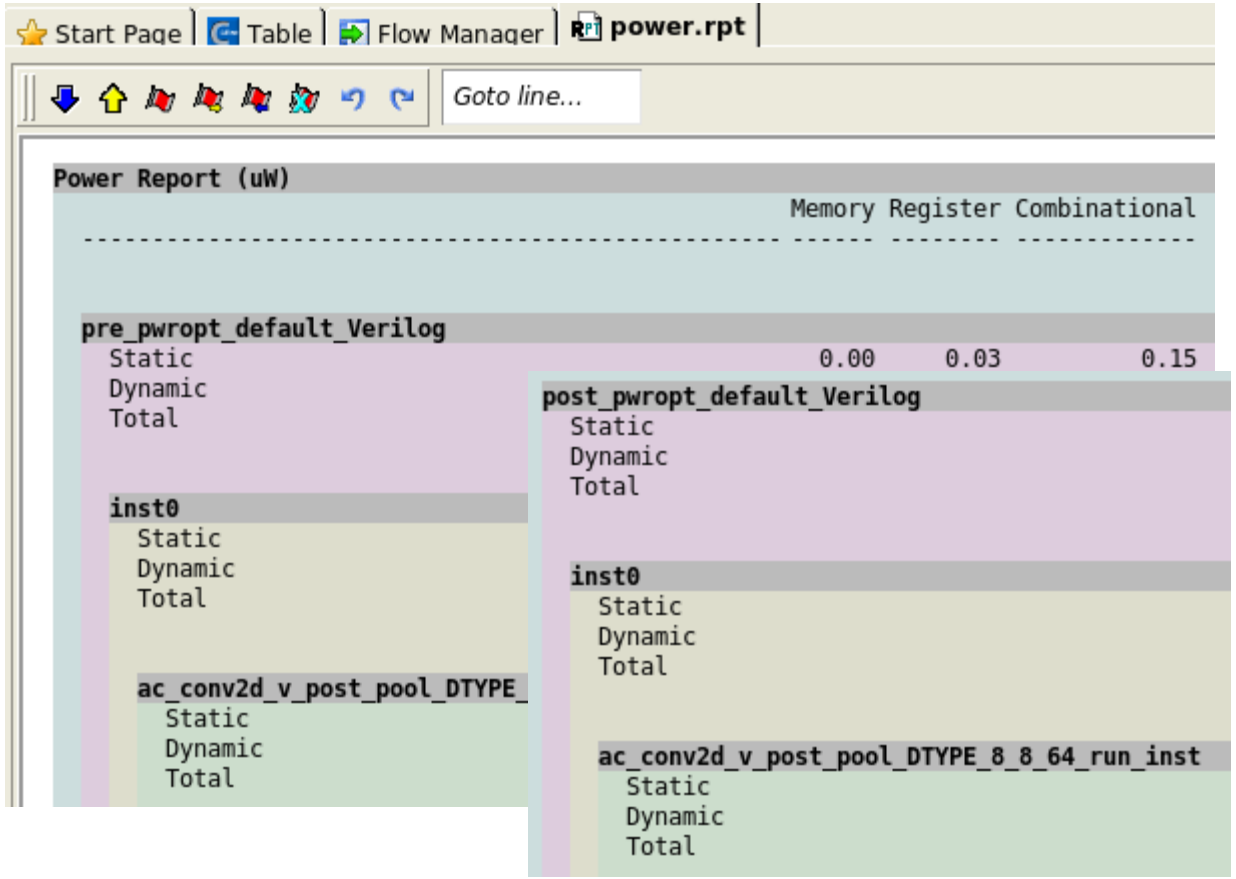
Loops can be unrolled

256-parallel multipliers



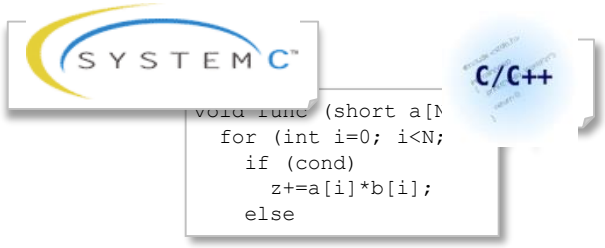
Meet power requirements via automatic power estimation and optimization

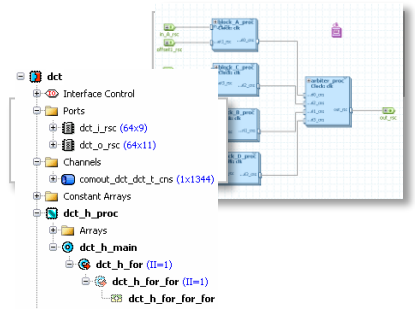
- Fast and accurate power estimation
  - Integrated solution
  - Power analysis and exploration
  - Guidance on how to reduce power
- Best Power Optimization
  - Gating of clocks, flops, memories and data
- Automatic flow
  - SLEC for formal verification
  - API Integration with emulation

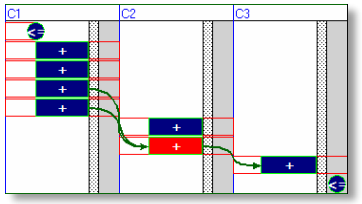


Catapult HLS is the Only Solution for Rapid Algorithm to RTL

- Accelerate design time with higher level of abstraction
  - 5x less code than RTL
  - New features added in days not weeks
- Quickly evaluate power and performance of algorithms
  - Rapidly explore multiple options for optimal PPA
- Enable late functional changes without impacting schedule
  - Algorithms can be easily modified and regenerated
  - New technology nodes are easy

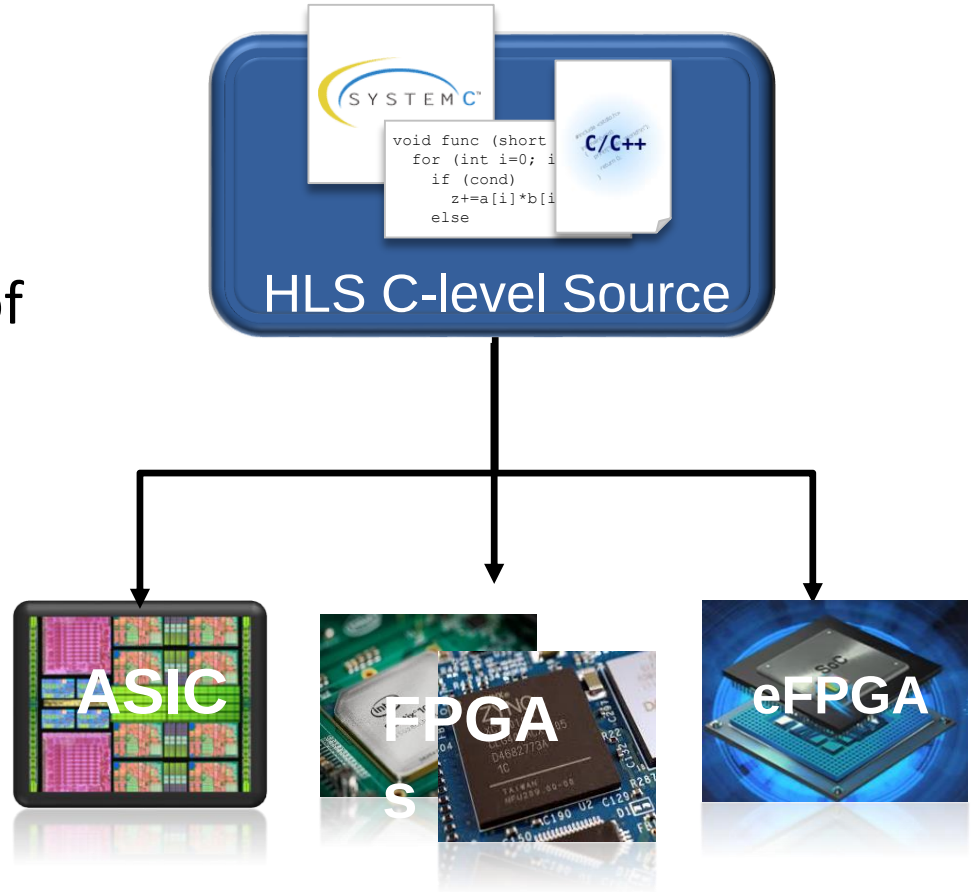






Catapult Enables Re-Use between FPGA and ASIC

- Enable designers to bring algorithms into high-speed HW/FPGA for fast Proof of Concept or demonstrator
- Key IP blocks reused from FPGA to ASIC to save months of redevelopment
  - Any ASIC library can be characterized to HLS
- Easy move between eFPGA and ASIC
- Same C code can be retargeted for different market/application within days



High speed FPGA and ASIC

Summary

- Next generation CV/AI algorithms are massively complex
- Delivering optimized RTL with the best PPA on time is difficult
  - Achieving the most optimal architecture is hard to do in hand-code RTL
  - Going from CV/AI development platform to RTL is not well understood
  - Verifying the RTL is time consuming
    - Billions of computations
    - Massively parallel hardware
- Catapult provides a complete methodology from high-level model to PPA optimized and rapidly verified RTL
- <Catapult_install>/shared/examples/ml/tinyYOLO_v2

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

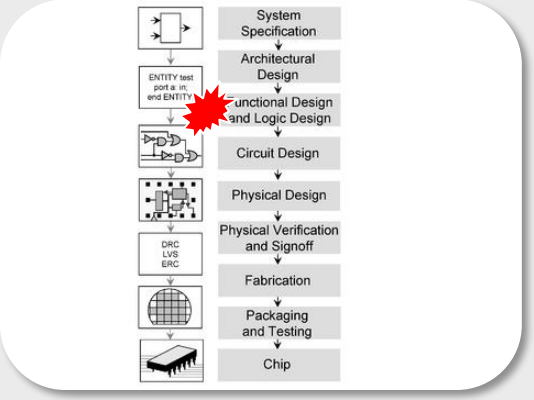
Functional Safety workflow of a High Level Synthesis Design

accellera SYSTEMS INITIATIVE

What is Functional Safety?

Driving down risk of Electrical and Electronics malfunctioning due to failures

Systematic Faults

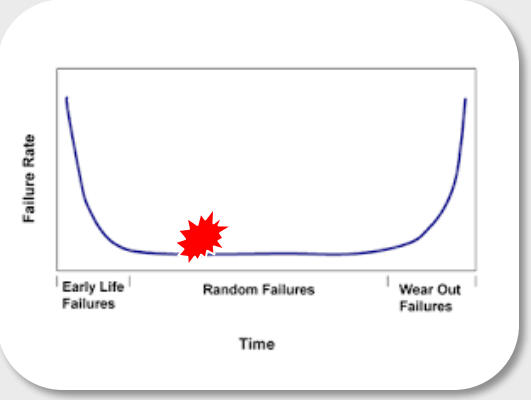


- Incomplete Specs
- Misinterpreted Specs
- Bad RTL
- HW/SW Interface Problems

Challenges

- Process & requirements
- IC complexity
- Exhaustive & efficient

Random Faults

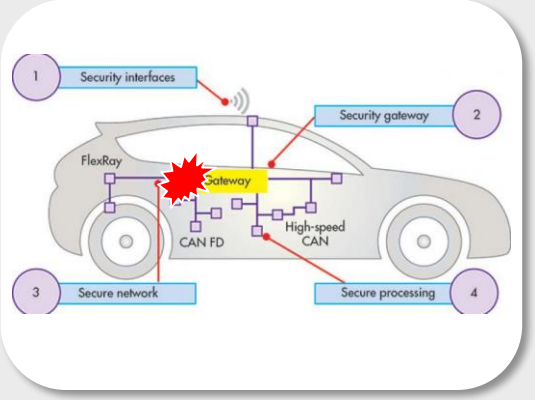


- EMI
- Electro-migration
- Permanent or transient
- Latent

Challenges

- Manual -> automation
- Scale with IC complexity

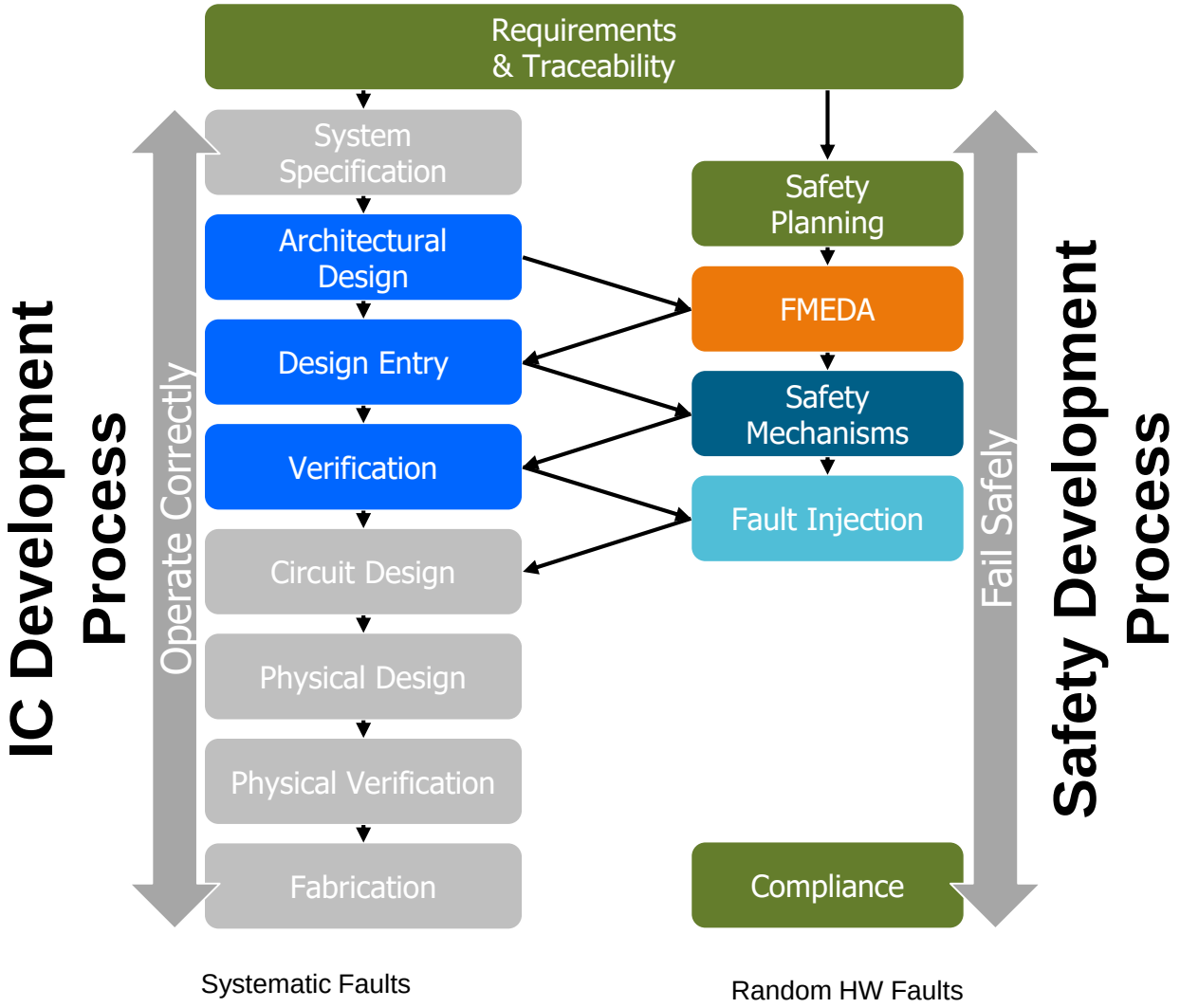
Malicious Faults



- Encryption Vulnerabilities
- Denial of Service
- Untrusted IC
- Hardware Trojan

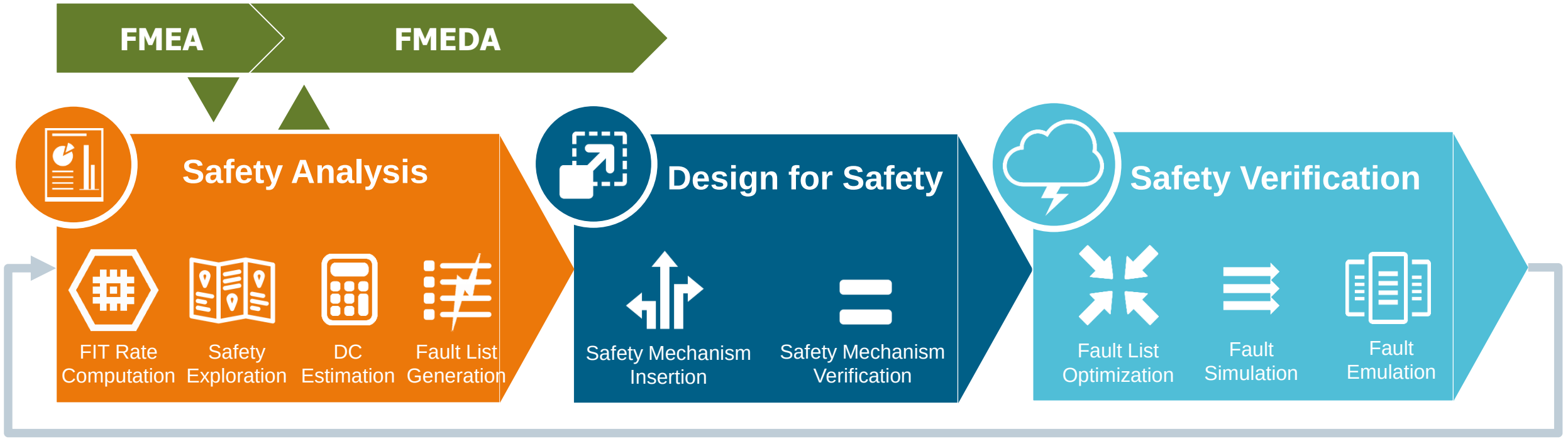
ISO 26262

Functional Safety in an HLS Workflow



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

First Time Right Safe IC Workflow



Calculation of FMEDA metrics and providing early safety architectural guidance

Creation of safe designs to mitigate the effects random hardware faults

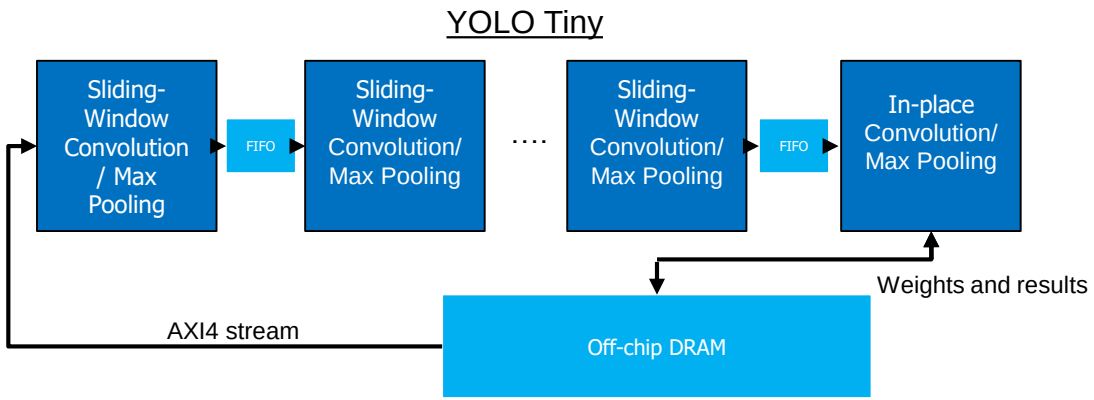
Proving design safeness that achieves target ASIL levels through fault campaigns

accellera SYSTEMS INITIATIVE

Making YOLO Tiny Safe

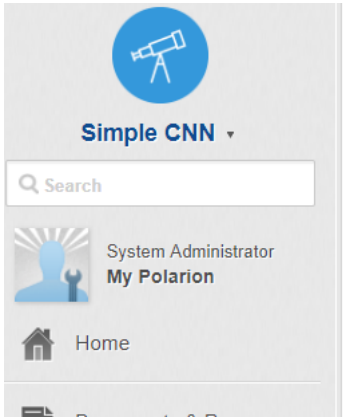
- Tutorial Goals
  - Evaluate the safeness of the YOLO Tiny design (a.k.a Simple CNN design)
  - Achieve ASIL B metric targets through minimal design enhancement
  - Proof the design achieves ASIL B targets through a fault injection campaign
  - Create FMEDA work product required for certification

- Assumptions
  - All logic in YOLO Tiny is safety critical
  - All single point faults must be protected by HW based safety mechanisms (no BIST)
  - ASIL B targets



FMEDA Work Product

- Evidence the design has sufficient protection from permanent and transient random HW failures



Simple CNN FMEDA Worksheet

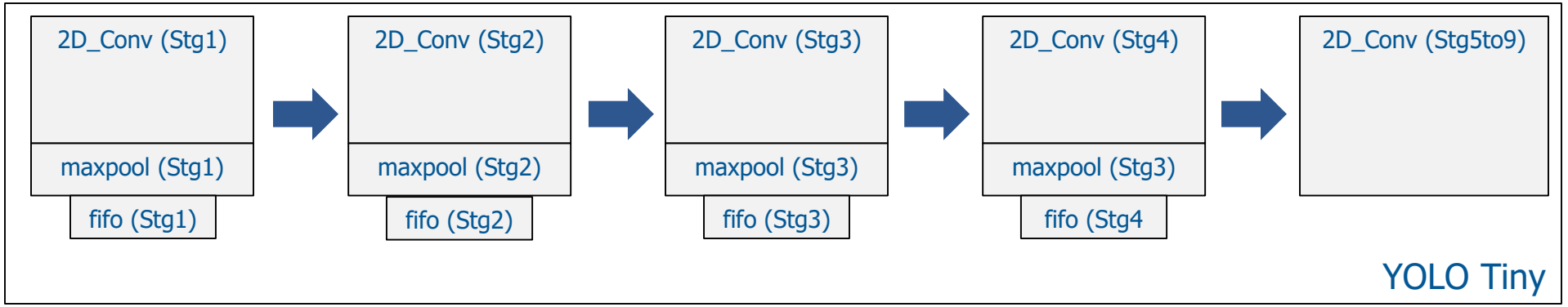
Component Name	Safety Related?	Violates SG	Register Count	Number of Gates	PERM FIT (λ)	TRAN FIT (λ)	Analysis (Estimated) Primary SM DC (PERM) Estimated %	Analysis (Estimated) Primary SM DC (TRAN) Estimated %	Verification (Proven) Primary SM DC (PERM) %	Verification (Proven) Primary SM DC (TRAN) %
2D_Conv_Stg1	yes	yes	21347	130027						
2D_Conv_Stg2	yes	yes	5074	67626						
2D_Conv_Stg3	yes	yes	4492	65585						
2D_Conv_Stg4	yes	yes	4533	66051						
2D_Conv_Stg5_to_stg9	yes	yes	60190	1105385						

Simplified FMEDA worksheet

Required metrics per ISO26262

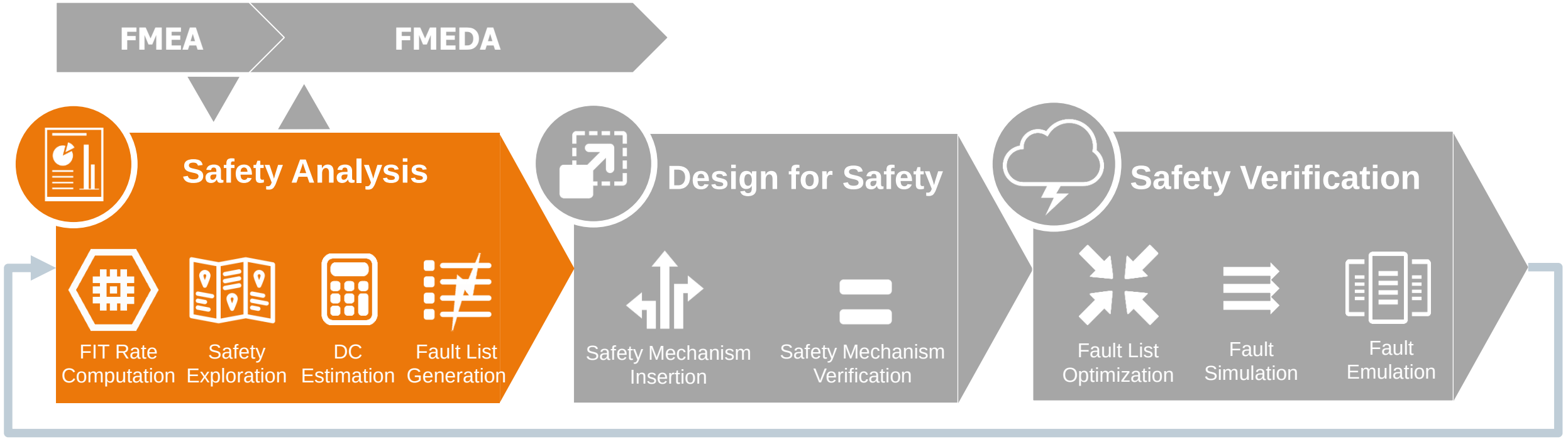
YOLO Tiny Architecture Overview

- First 4 stages are independent
- Stages 5 – 9 use shared logic and memory
- High degree of combinatorial logic due to Mult/Adds



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

First Time Right Safe IC Workflow



Calculation of FMEDA metrics and providing early safety architectural guidance

Creation of safe designs to mitigate the effects random hardware faults

Proving design safeness that achieves target ASIL levels through fault campaigns

accellera SYSTEMS INITIATIVE

FIT Computation

Mission Profile

Design Structural Analysis

Package Specification

$$\lambda = \left\{ \underbrace{\left\{ \lambda_1 \times N \times e^{-0.35 \times a} + \lambda_2 \right\} \times \left\{ \frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on} + \tau_{off}} \right\}}_{\lambda_{die}} + \underbrace{\left\{ 2.75 \times 10^{-3} \times \pi_\alpha \times \left(\sum_{i=1}^{z} (\pi_n)_i \times (\Delta T_i)^{0.68} \right) \times \lambda_3 \right\}}_{\lambda_{package}} + \left\{ \underbrace{\pi_I \times \lambda_{EOS}}_{\lambda_{overstress}} \right\} \right\} \times 10^{-9} / h$$

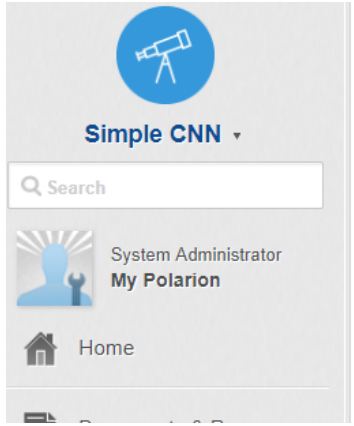
IEC 62380 FIT Model

Target technology

Package Materials

YOLO Tiny FIT Computation

- Perform FIT calculation on each sub-block/filter
- Sub-blocks rolled up to compute total YOLO Tiny FIT

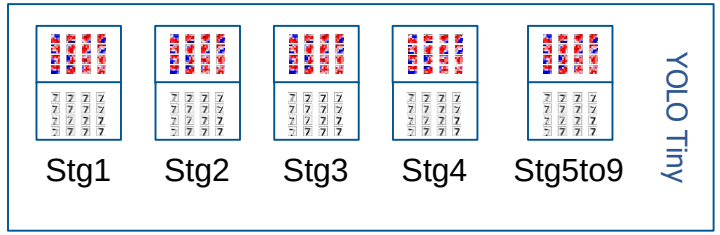


Simple CNN FMEDA Worksheet

Component Name	Safety Related?	Violates SG	Register Count	Number of Gates	PERM FIT (λ)	TRAN FIT (λ)
2D_Conv_Stg1	yes	yes	21347	130027	0.15	0.25
2D_Conv_Stg2	yes	yes	5074	67626	0.09	0.59
2D_Conv_Stg3	yes	yes	4492	65585	0.09	0.84
2D_Conv_Stg4	yes	yes	4533	66051	0.09	2.06
2D_Conv_Stg5_to_stg9	yes	yes	60190	1105385	0.99	2.07

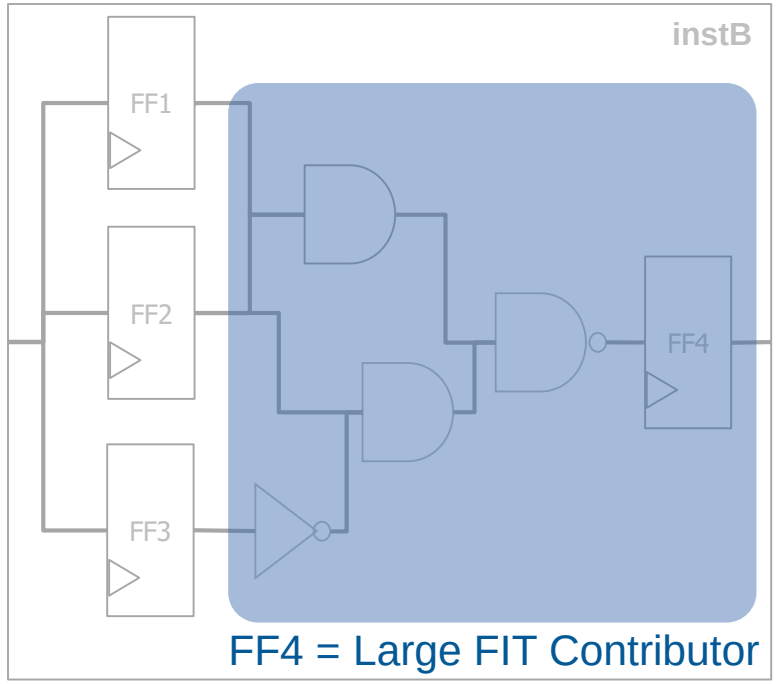
Simplified FMEDA worksheet

1.93 FIT (Perm)
8.09 FIT (Trans)

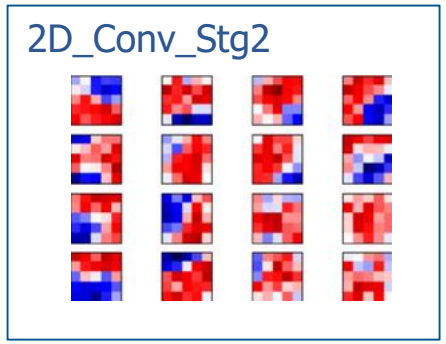


Identifying the safety holes

- Gain understanding of underlying architecture and RTL that was created via HLS synthesis
- Identify FIT contribution at the micro level to systematically guide safety mechanism exploration



FF4 = Large FIT Contributor



Subset of contributors

Instance	Perm %	Trans %
2D_Conv_Stg2.inst0	9.06588	8.00971
2D_Conv_Stg2.inst1	64.2728	44.1694
2D_Conv_Stg2.inst2	26.4449	47.4349

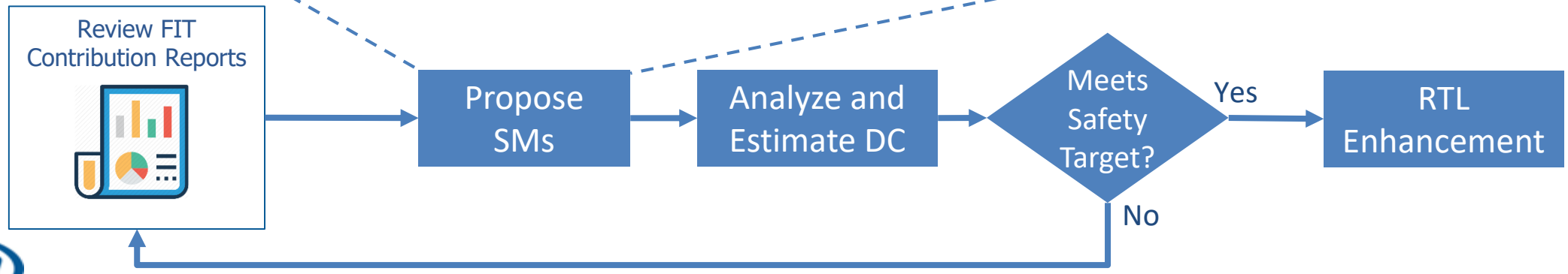
Large FIT contributors – Address first

Safety Exploration

- Estimate achievable diagnostic coverage using design analysis to measure the effectiveness of proposed safety mechanisms

SM Estimated Diagnostic Coverage

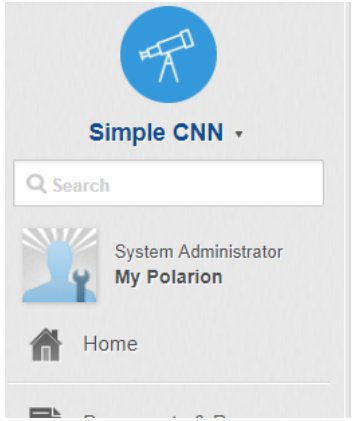
DC Mechanism	Diagnostic Coverage		Resource Utilization	Description
	Permanent	Transient		
Endpoint Parity	99	99	↑	Parity added to Flip Flops
Endpoint/Cone Duplication	99	99	↑↑	Logic Cone and EP Replication
Endpoint/Cone Triplication	99	99	↑↑↑	Logic Cone and EP Replication
Endpoint ECC	99	99	↑	ECC For Registers
Logic BIST	99	0	↑	LBIST : RunTime
Memory ECC	99	99	↑	Memory ECC with control coverage



Exploration Results

- Exploration performed on sub blocks and rolled up to attain top level **estimated** Diagnostic Coverage

Without memory protection



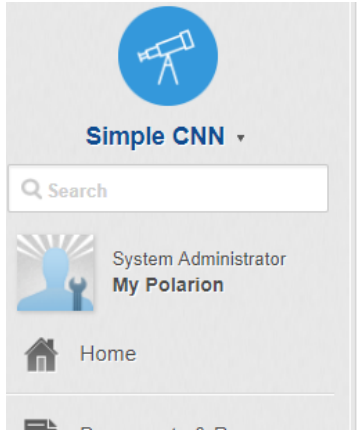
Simple CNN FMEDA Worksheet

Analysis (Estimated)

Low transient coverage due to missing SMs on memories

Component Name	Safety Related?	Violates SG	Register Count	Number of Gates	PERM FIT (λ)	TRAN FIT (λ)	Primary SM DC (PERM) Estimated %	Primary SM DC (TRAN) Estimated %
2D_Conv_Stg1	yes	yes	21347	130027	0.15	0.25	92	25
2D_Conv_Stg2	yes	yes	5074	67626	0.09	0.59	92	1
2D_Conv_Stg3	yes	yes	4492	65585	0.09	0.84	90	8
2D_Conv_Stg4	yes	yes	4533	66051	0.09	2.06	87	3
2D_Conv_Stg5_to_stg9	yes	yes	60190	1105385	0.99	2.07	92	31

With memory protection



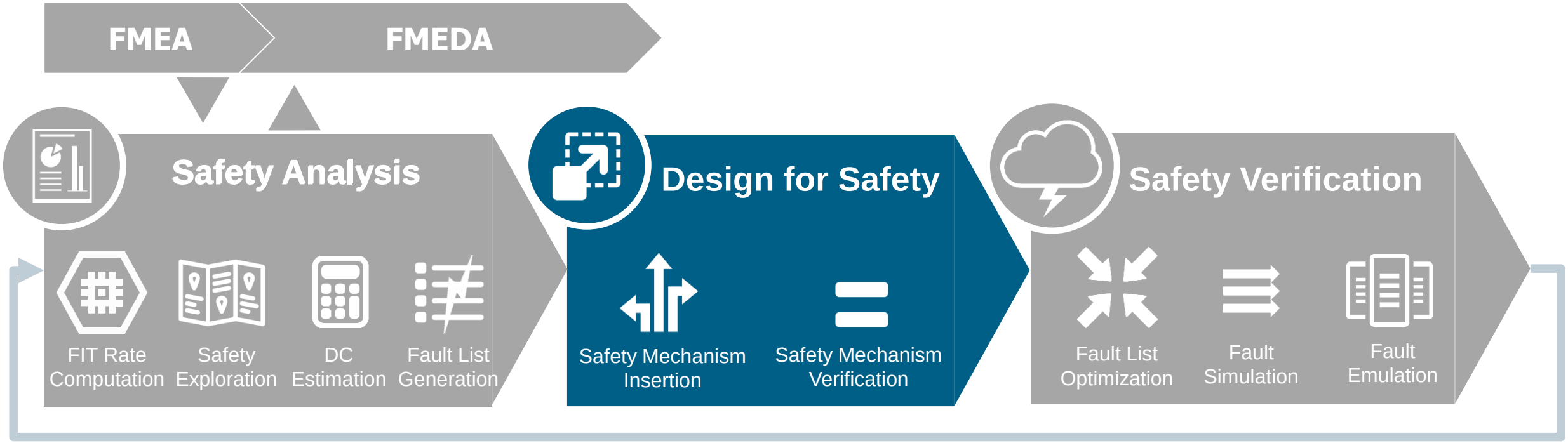
Simple CNN FMEDA Worksheet

Analysis (Estimated)

Component Name	Safety Related?	Violates SG	Register Count	Number of Gates	PERM FIT (λ)	TRAN FIT (λ)	Primary SM DC (PERM) Estimated %	Primary SM DC (TRAN) Estimated %
2D_Conv_Stg1	yes	yes	21347	130027	0.15	0.25	92	95
2D_Conv_Stg2	yes	yes	5074	67626	0.09	0.59	92	97
2D_Conv_Stg3	yes	yes	4492	65585	0.09	0.84	90	97
2D_Conv_Stg4	yes	yes	4533	66051	0.09	2.06	87	98
2D_Conv_Stg5_to_stg9	yes	yes	60190	1105385	0.99	2.07	92	94

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

First Time Right Safe IC Workflow



Calculation of FMEDA metrics and providing early safety architectural guidance

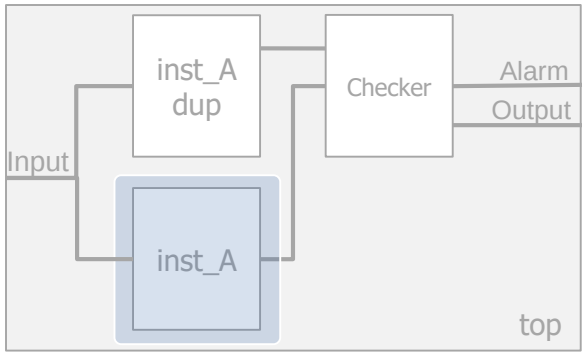
Creation of safe designs to mitigate the effects random hardware faults

Proving design safeness that achieves target ASIL levels through fault campaigns

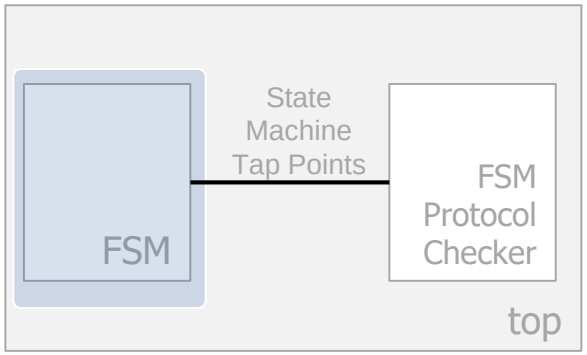
accellera SYSTEMS INITIATIVE

Safety Mechanism Overview

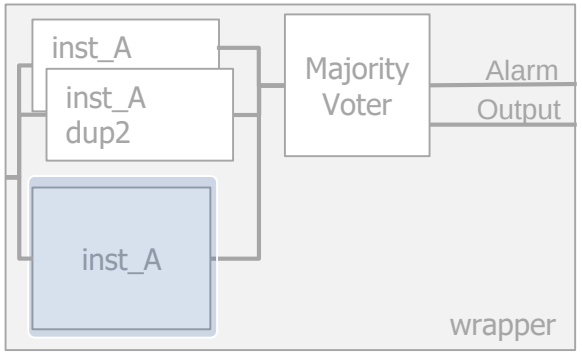
- Wide variety of safety mechanism approaches available
- PPA requirements + use model + Safety target => the optimal safety mechanism



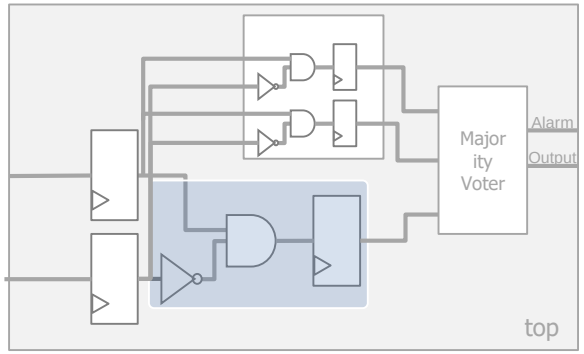
Lockstep duplication



FSM Monitor



Triple Modular Redundancy

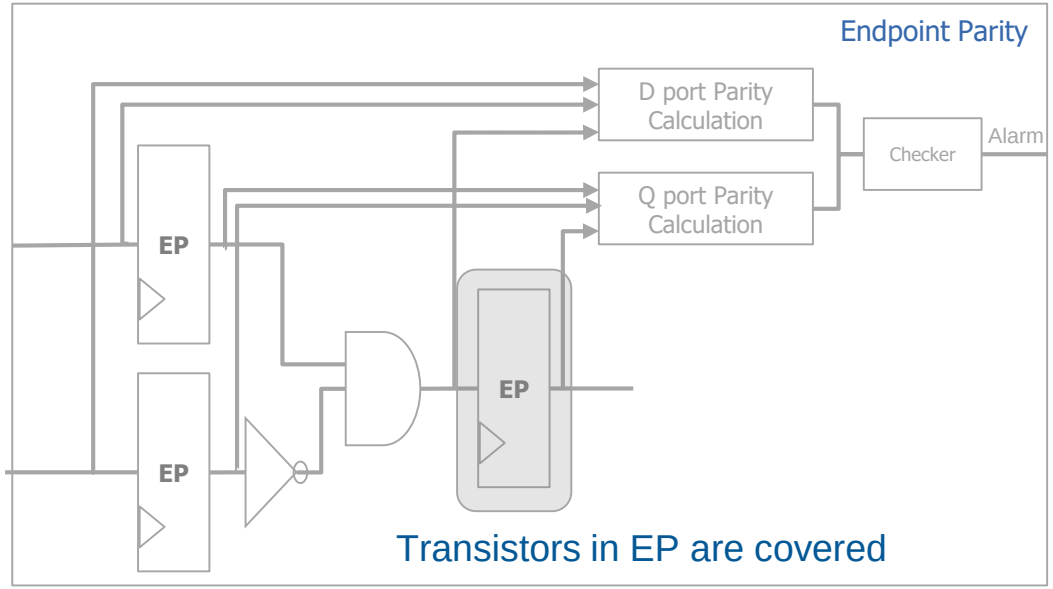


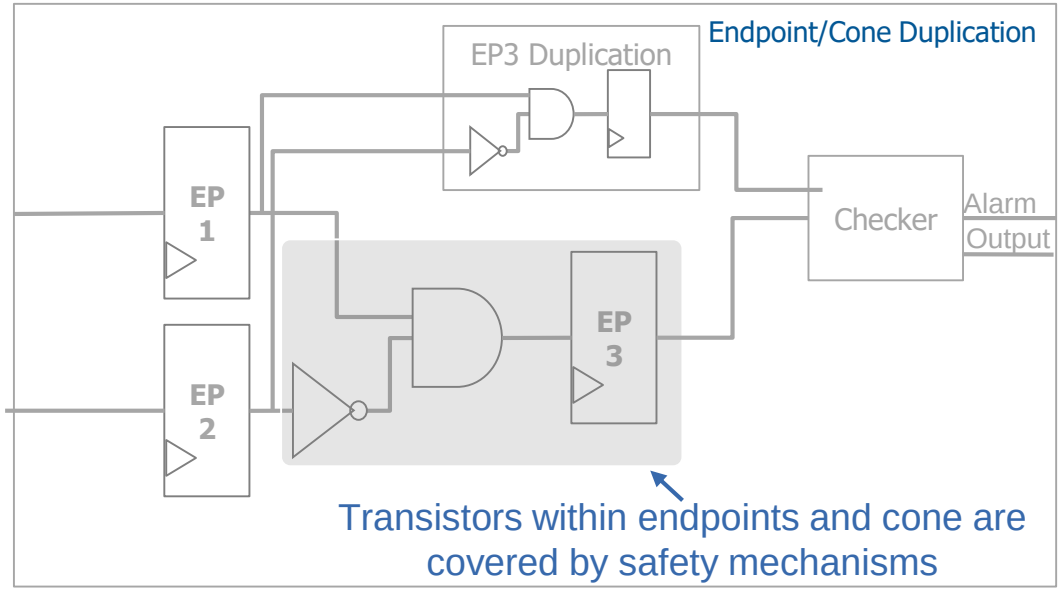
Endpoint Triplication

Indicates transistor/logic coverage of Safety Mechanism

Safety Enhancement of YOLO Tiny

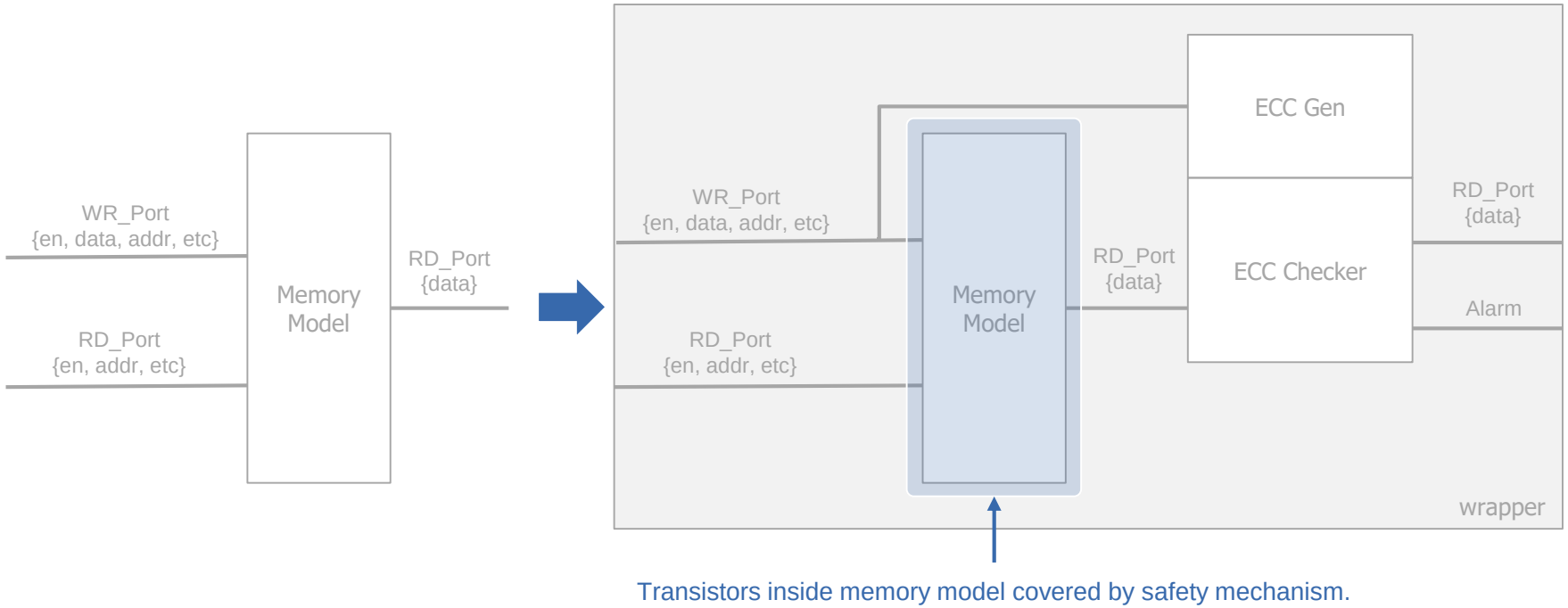
- Automated safety mechanism insertion performed on sub blocks
- Safety Exploration recommended a mix of register parity and duplication to hit safety target
  - 2D filters were duplication heavy due to large fan-in cones





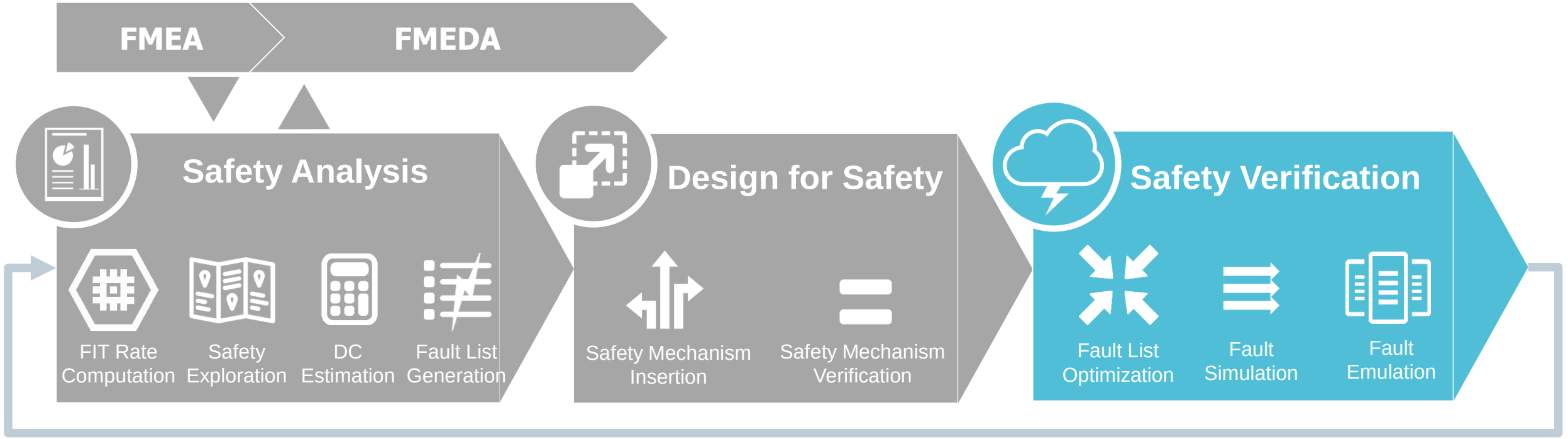
Memory ECC to protect from transients

- Memory ECC to increase the transient fault diagnostic coverage



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

First Time Right Safe IC Workflow



Calculation of FMEDA metrics and providing early safety architectural guidance

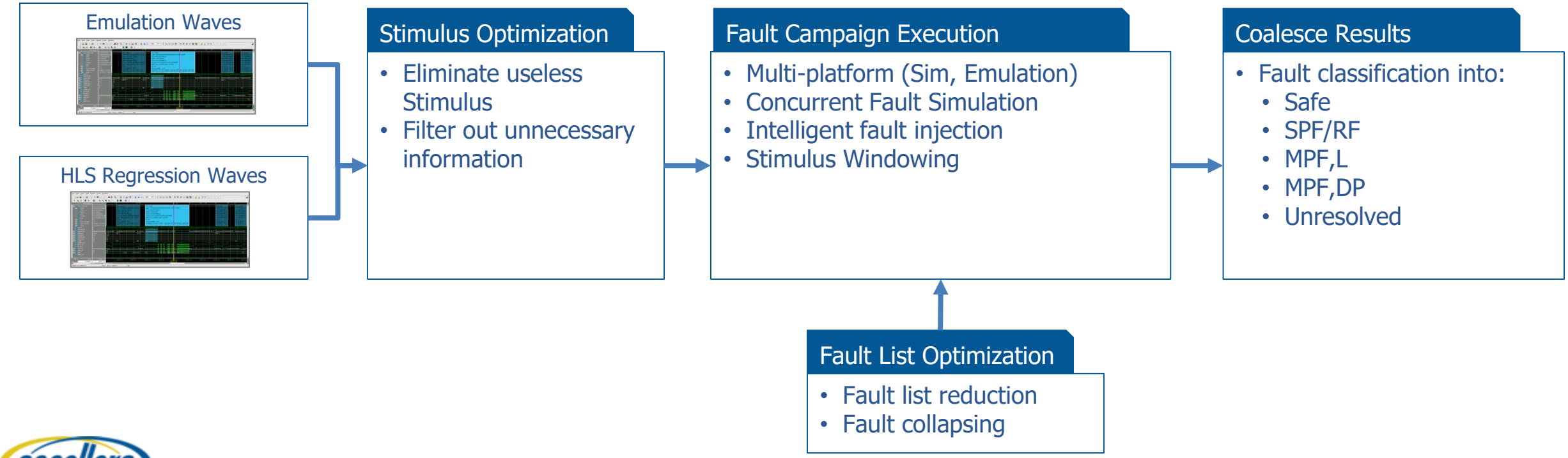
Creation of safe designs to mitigate the effects random hardware faults

Proving design safeness that achieves target ASIL levels through fault campaigns

accellera SYSTEMS INITIATIVE

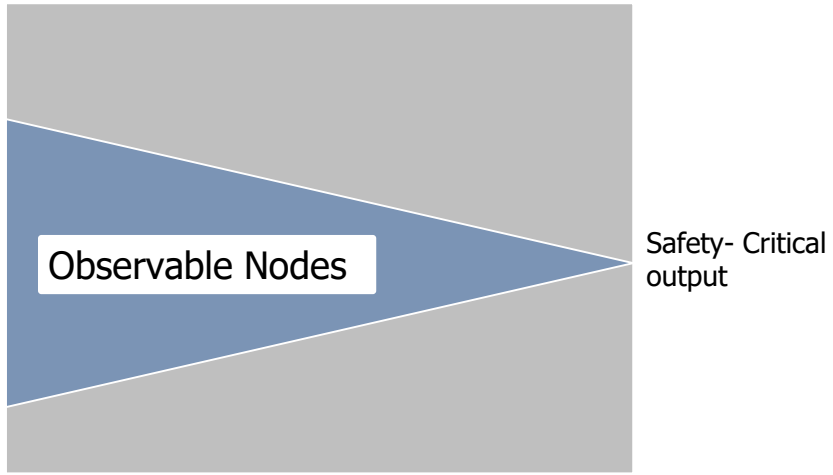
Fault Campaign Overview

- Fault campaign consists of fault injection of permanent and transient faults
- Goal: Classify all faults after reducing scope of fault campaign through fault optimization and collapsing

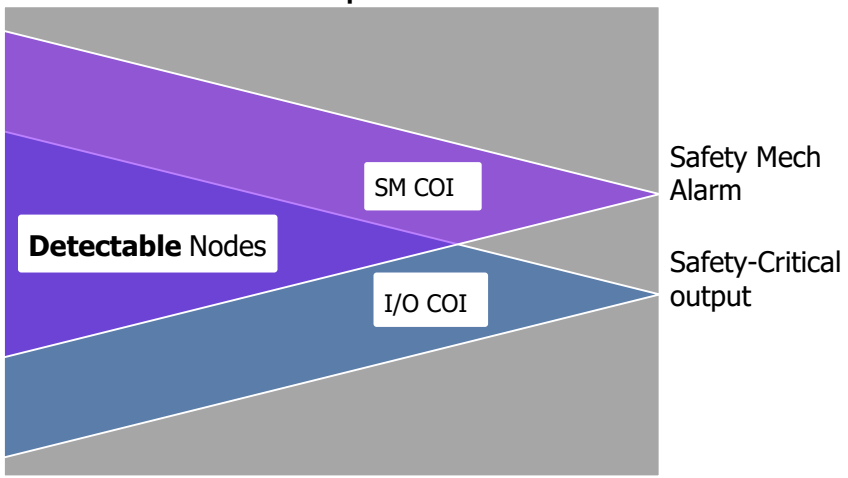


Fault List Optimization

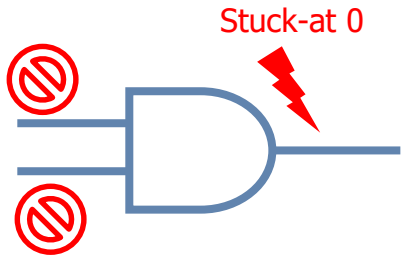
1
Safety Critical Optimization



2
SM Aware Optimization



3
Fault collapsing



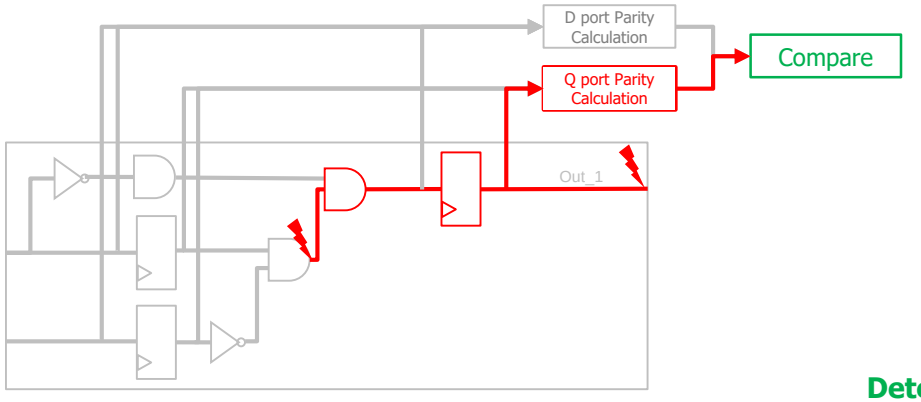
Stuck-at 0 on inputs removed from fault list

Module	DC %	Fault Count (before)	Fault Count (after)	% Reduction
2D_Conv_Stg1	92%	130660	106750	19%
2D_Conv_Stg2	90%	69712	49427	30%
2D_Conv_Stg3	92%	67845	47115	31%
2D_Conv_Stg4	87%	68316	47504	31%
2D_Conv_Stg5_to_Stg9	92%	1130651	616150	45%

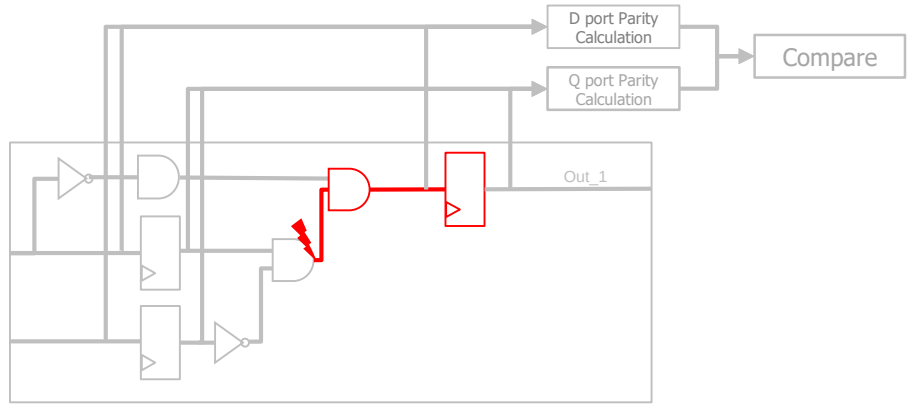
YOLO Tiny Optimization

Fault Classification

- The goal of a fault campaign is to classify all faults in the fault list

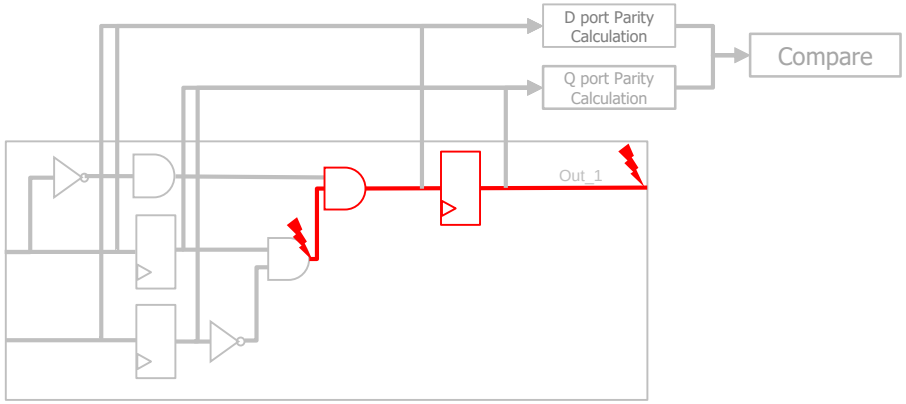


Detected

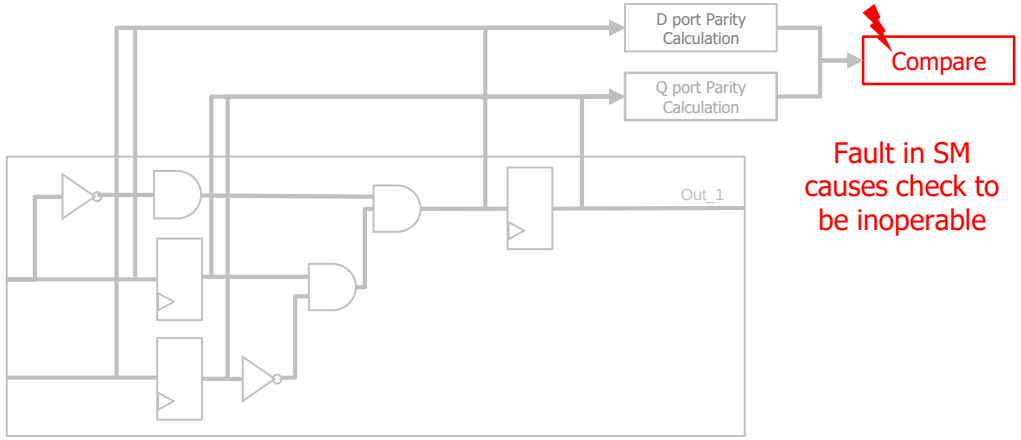


Unresolved

Undetected SPF or RF



Multi-Point Latent (MPF,L)



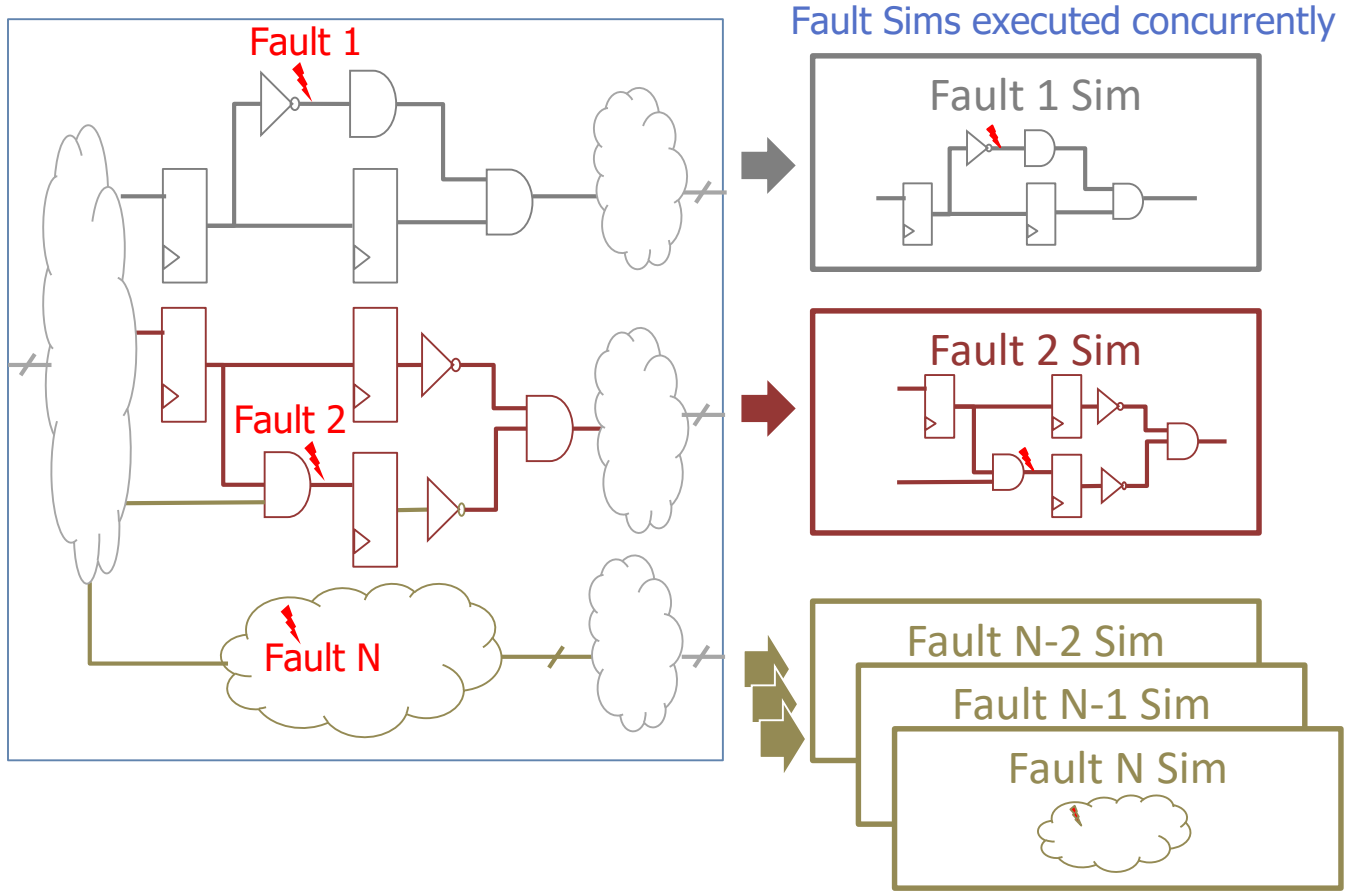
Fault in SM causes check to be inoperable

Stimulus Optimization

- Grade stimulus so only stimulus which has the potential for fault classification is used
- Filter stimulus to increase efficiency of fault injection simulation

Concurrent Fault Simulation

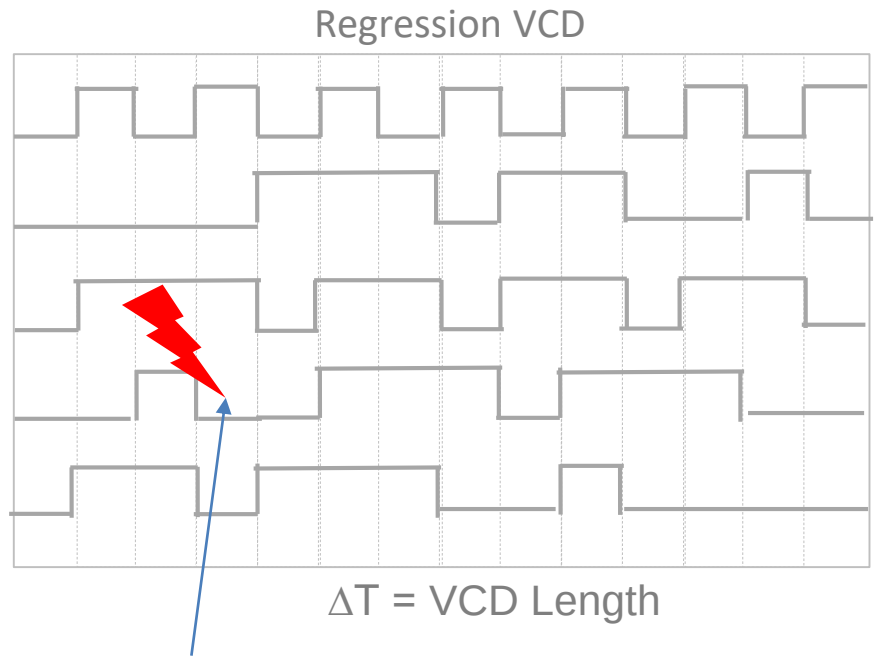
- Fault list can contain 10s of thousands of faults
  - Must execute faults in parallel to close on fault list
- Each fault executed concurrently in a single simulation
- Deviation checked against golden model



Intelligent Fault injection

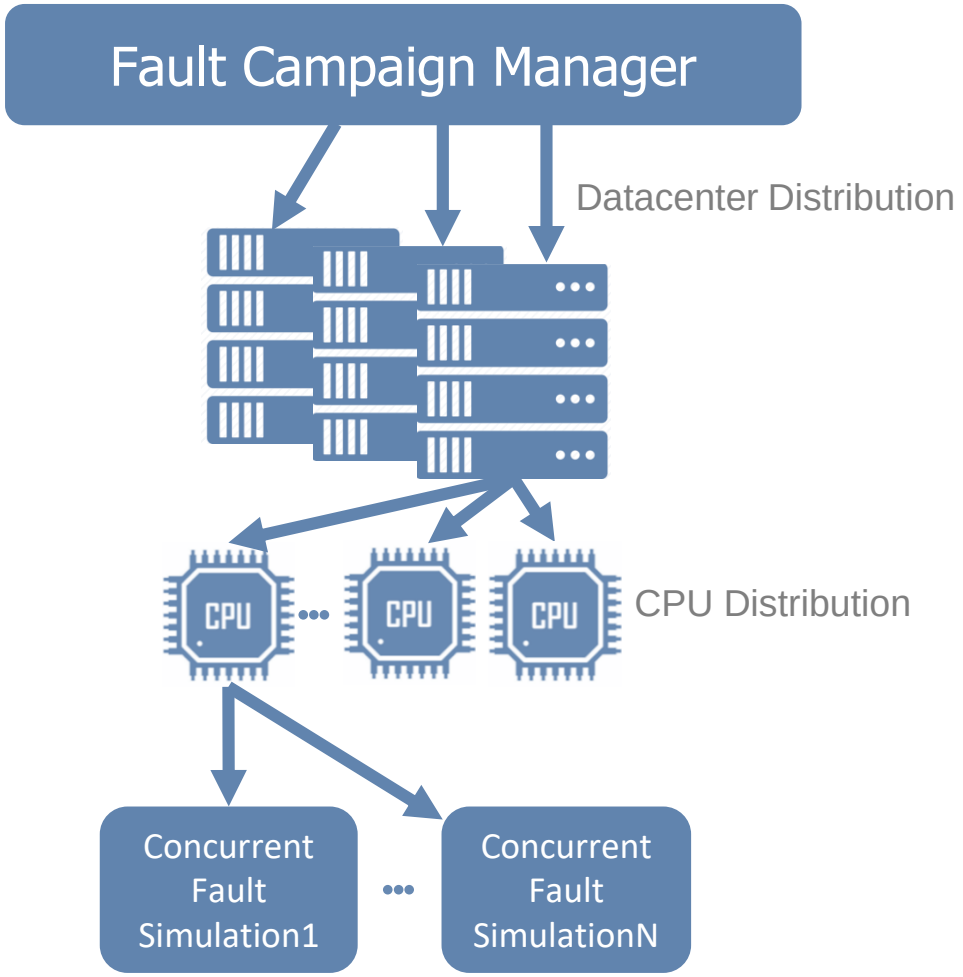
- Find the right point to inject a fault
- Ensure there is propagation potential
- Identification of high cost fault nodes

ac_accum_h_ac_fixed_..._true_FTYPE_DTYPE_208_208_16_32_run_inst.add_1274.n4



Fault Campaign Manager

- Parallelize window fault injection across compute cores
- Distribute jobs across machine grid



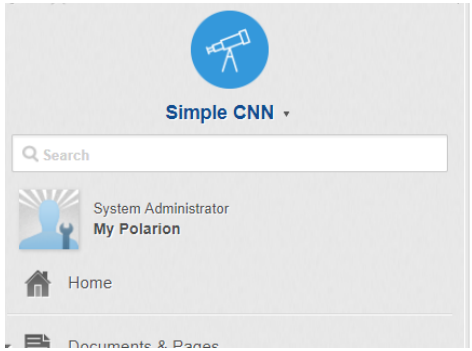
YOLO Tiny Fault Classification

Fault Node	Fault Type	InjectTime	AlarmTime	Resolution	Stimulus
ac_conv2d_coeff_store_DTYPE_FTYPE_FTYPE_DTYPE_208_208_16_32.inst2.ac_accum_h_ac_fixed_45_21_true_AC_TRN_AC_WRAP_FTYPE_DTYPE_208_208_16_32_run_inst.add_1274.n4	SA0	1040	1044	Detected	vsim.vcd
ac_conv2d_coeff_store_DTYPE_FTYPE_FTYPE_DTYPE_208_208_16_32.inst2.ac_accum_h_ac_fixed_45_21_true_AC_TRN_AC_WRAP_FTYPE_DTYPE_208_208_16_32_run_inst.add_1274.n4	SA1	1080	1084	Detected	vsim.vcd
…					
ac_conv2d_coeff_store_DTYPE_FTYPE_FTYPE_DTYPE_208_208_16_32.inst2.ac_accum_h_ac_fixed_45_21_true_AC_TRN_AC_WRAP_FTYPE_DTYPE_208_208_16_32_run_inst.FMAP_OCHAN_outChan_5_2_lpi_3_dfm_2_0_1[0]	SA0	2010	2014	Detected	vsim.vcd
ac_conv2d_coeff_store_DTYPE_FTYPE_FTYPE_DTYPE_208_208_16_32.inst2.ac_accum_h_ac_fixed_45_21_true_AC_TRN_AC_WRAP_FTYPE_DTYPE_208_208_16_32_run_inst.FMAP_OCHAN_outChan_5_2_lpi_3_dfm_2_0_1[0]	SA1	2040	2044	Detected	vsim.vcd

2D_Conv_Stg2 Fault Classification

Certification Work Products

- Close the loop linking estimated metrics to proven metrics

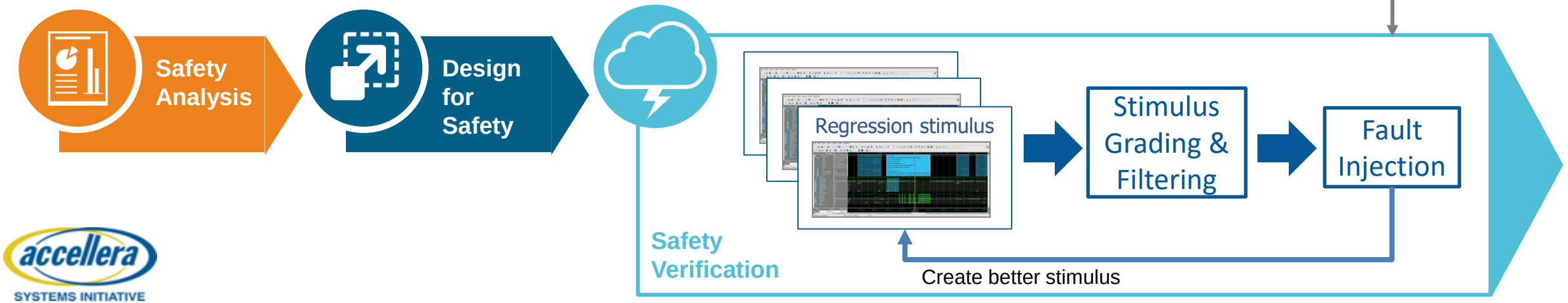


Simple CNN FMEDA Worksheet

							Analysis (Estimated)		Verification (Proven)	
Component Name	Safety Related?	Violates SG	Register Count	Number of Gates	PERM FIT (λ)	TRAN FIT (λ)	Primary SM DC (PERM) Estimated %	Primary SM DC (TRAN) Estimated %	Primary SM DC (PERM) %	Primary SM DC (TRAN) %
2D_Conv_Stg1	yes	yes	21347	130027	0.15	0.25	92	95	78	82
2D_Conv_Stg2	yes	yes	5074	67626	0.09	0.59	92	97	84	88
2D_Conv_Stg3	yes	yes	4492	65585	0.09	0.84	90	97	66	82
2D_Conv_Stg4	yes	yes	4533	66051	0.09	2.06	87	98	75	88
2D_Conv_Stg5_to_stg9	yes	yes	60190	1105385	0.99	2.07	92	94	80	82

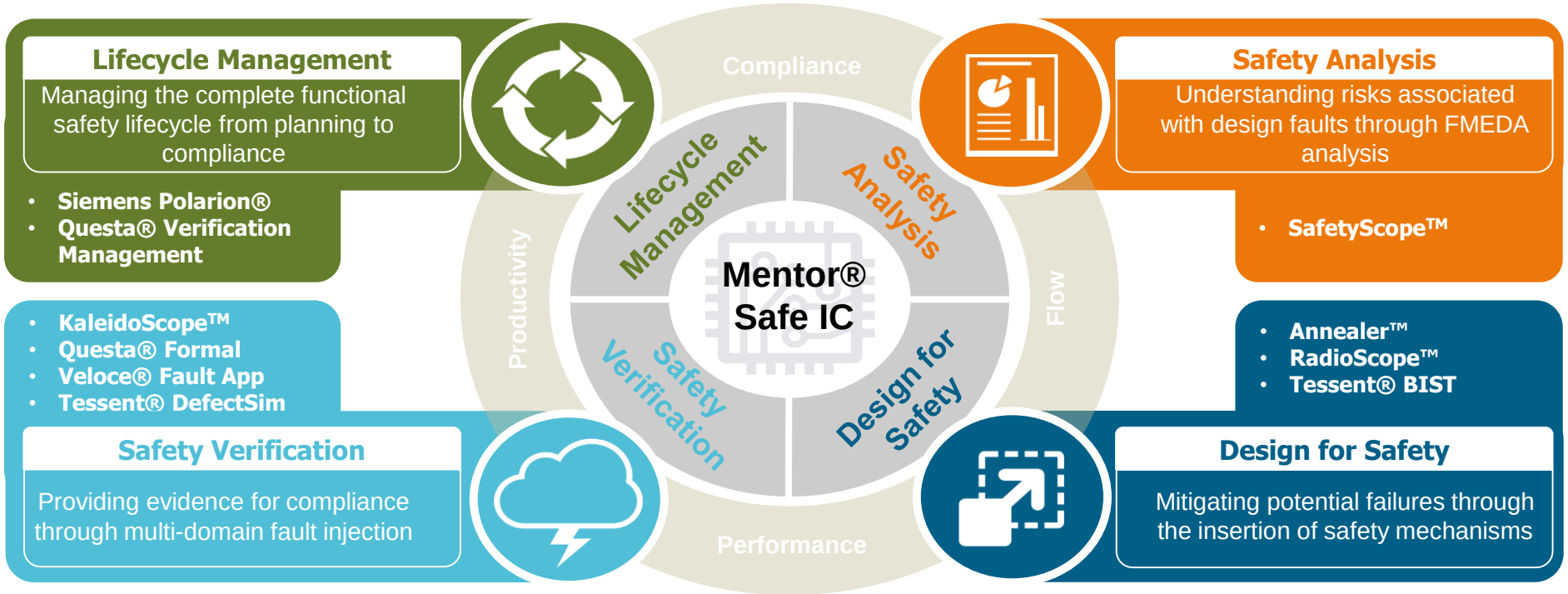
Simplified FMEDA worksheet

DC not met => Insufficient stimulus



Summary

- Demonstrated a functional safety workflow on a real HLS design
- If you want to learn more, here is how Mentor® tools mapped to this workflow



Break – 15 minutes

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Emulation Hardware-in-the-Loop for System-of-Systems Verification

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

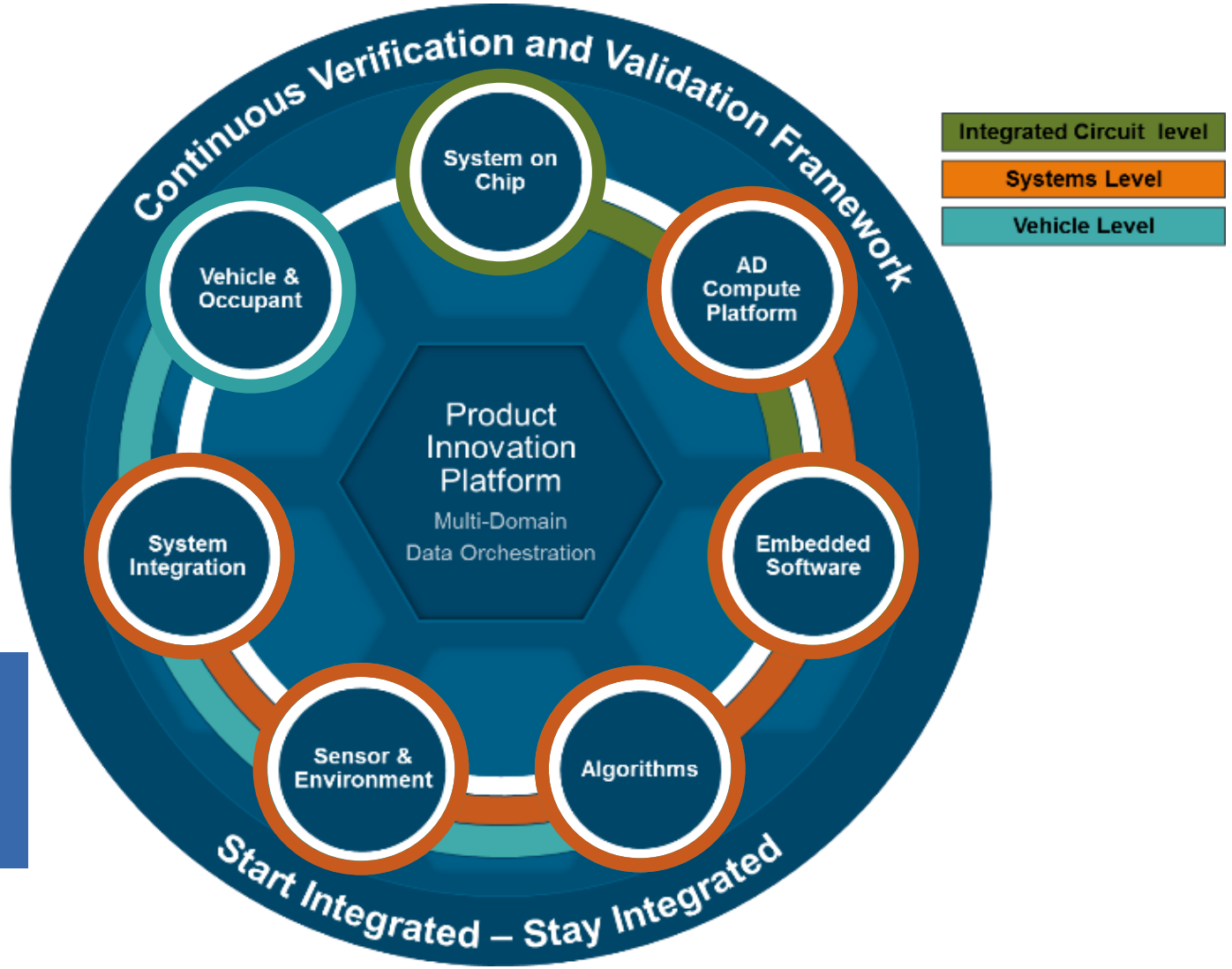
Autonomous vehicle development relies on converging technologies

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

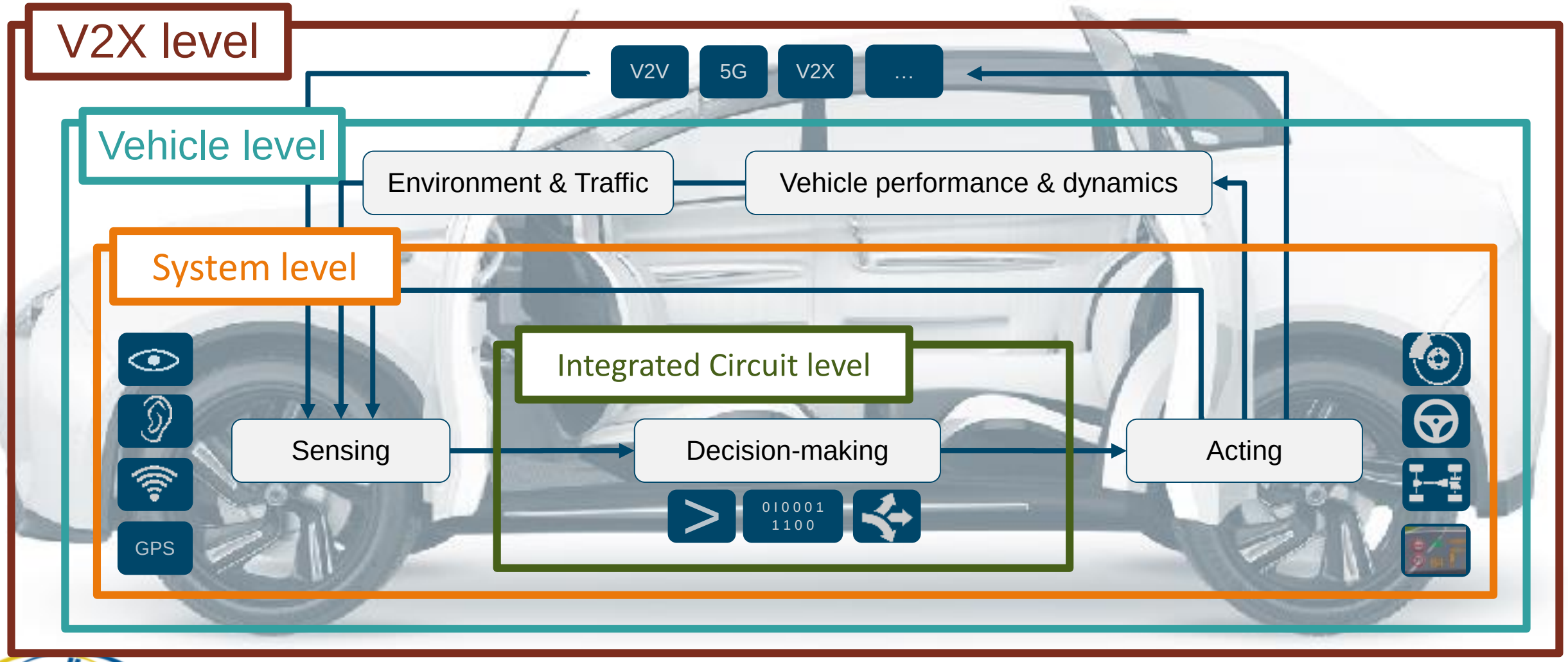
A broad portfolio of solutions is needed for autonomous vehicle development

Ensuring digital continuity, multi-domain traceability and functional safety of autonomous systems



accellera SYSTEMS INITIATIVE

Electronic system of system challenges for AV verification and validation



Self-driving technology requires massive verification cycles to reach safety for “Level 5”

“14.2 billion miles of testing is needed”

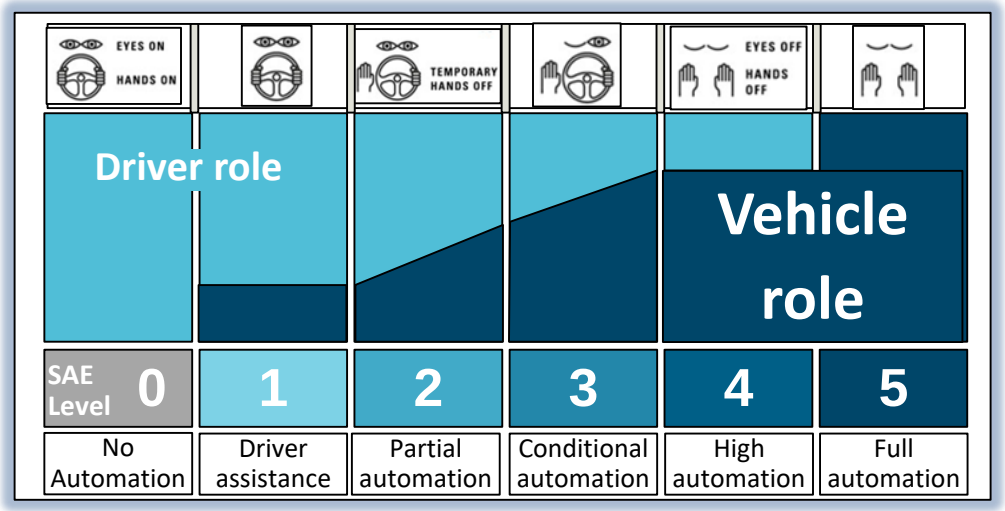
Akio Toyoda, CEO of Toyota
Paris Auto Show 2016

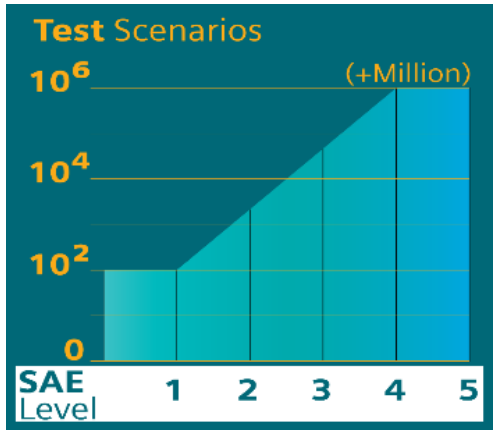
“Design validation will be a major – if not the largest – cost component”

Roland Berger
“Autonomous Driving” 2014

“While hardware innovations will deliver - software will remain a critical bottleneck”

McKinsey
“When will the robots hit the road?





2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Multiple variants of the same scenario are part of the verification process

These scenarios and the multiple variants can be tested real-time when using a high-performance computing environment

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Safety and Security key challenges are addressed early in the design cycle by virtualizing the system

Reinvent the vehicle development processes to address:

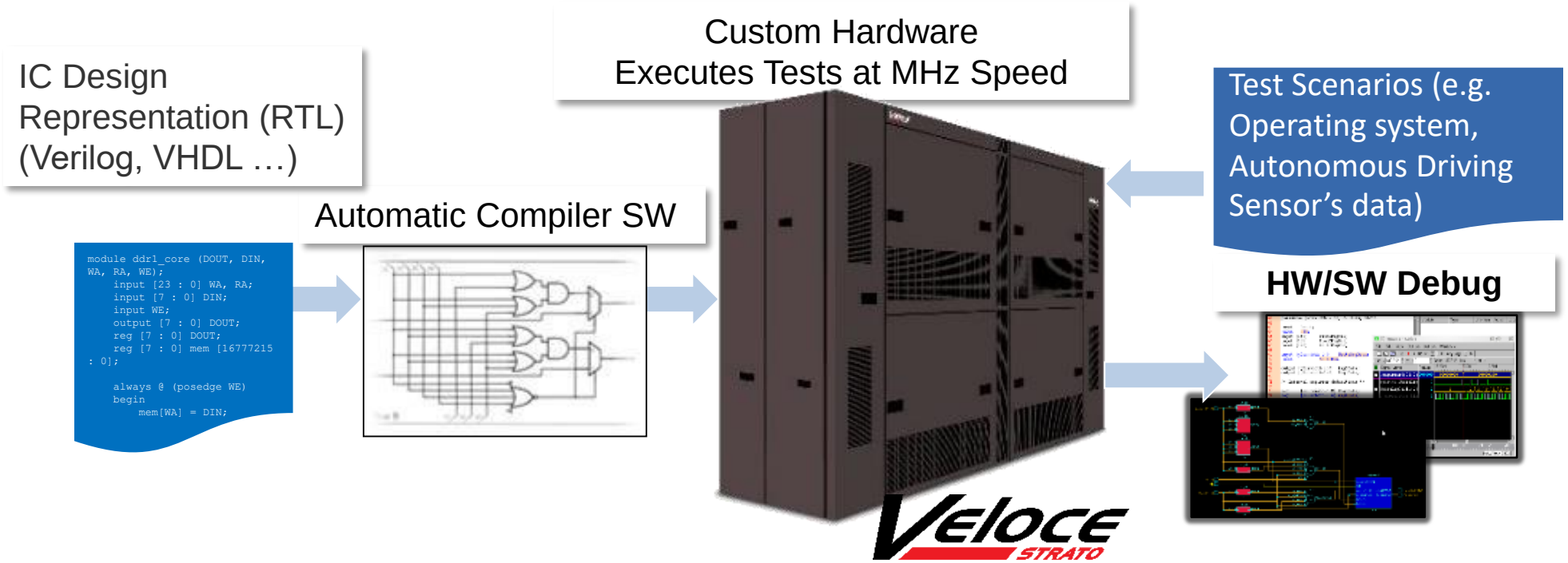
- Increasing software and hardware complexity
- Massive validation and verification cycles
- Growing number and variety of sensors
- Reconciling agility with better traceability

accellera SYSTEMS INITIATIVE

Hardware emulation is the ideal platform for system of systems verification and validation

- **Emulators typically run 1000's times faster compared to a SW simulator running on general purpose computer**
  - **An emulator is a special purpose supercomputer for modeling digital integrated circuits**
- **An emulator includes a HW system, OS SW and SW applications**
- **Emulation technology enables new design and verification methodologies from chips to systems**

How a hardware emulator works…



- **Virtual system for pre-silicon software verification**
- **Full visibility into hardware design for efficient debug**
  - **Cannot have with FPGA prototype**
- **Fault injection, monitoring, results analysis for safety-critical applications**

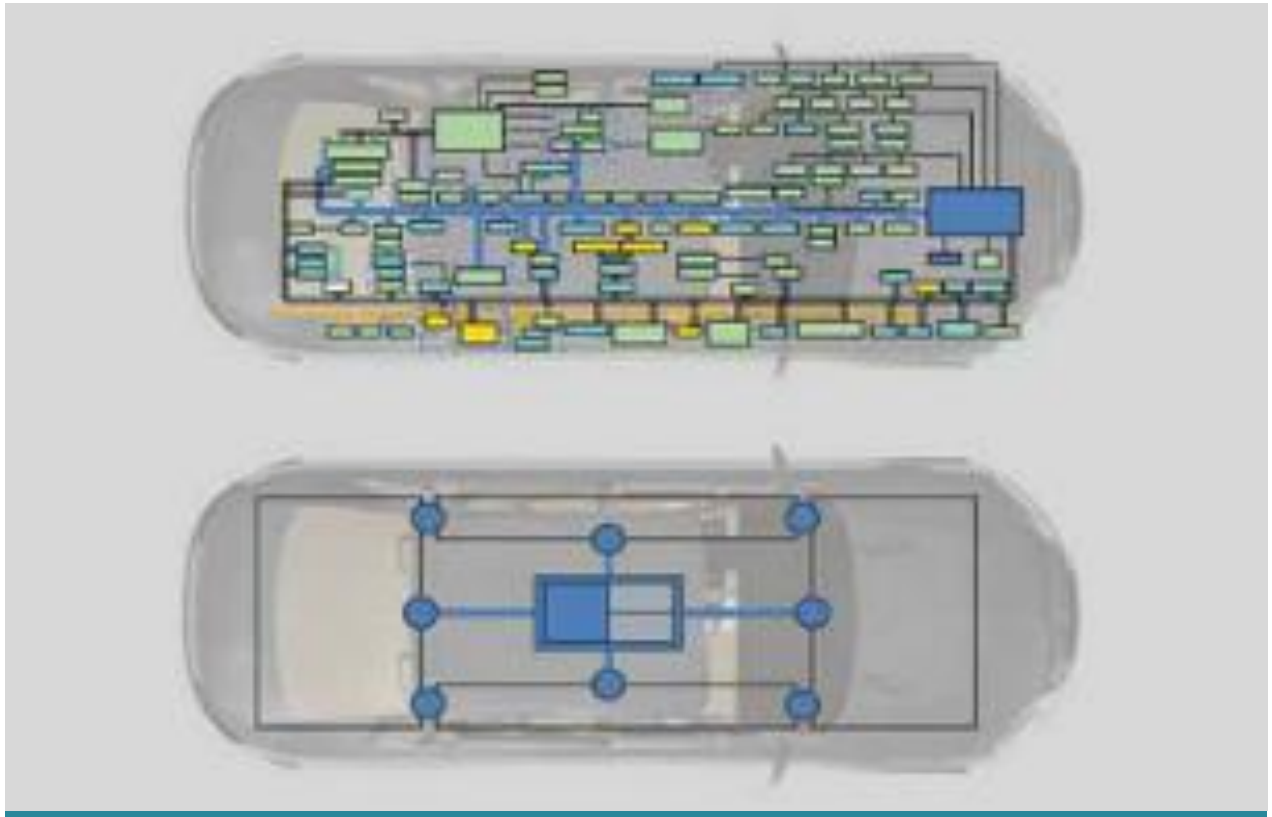
Veloce Automotive Solution — four pillars

Veloce VIP
VTL transactors

Functional
Safety

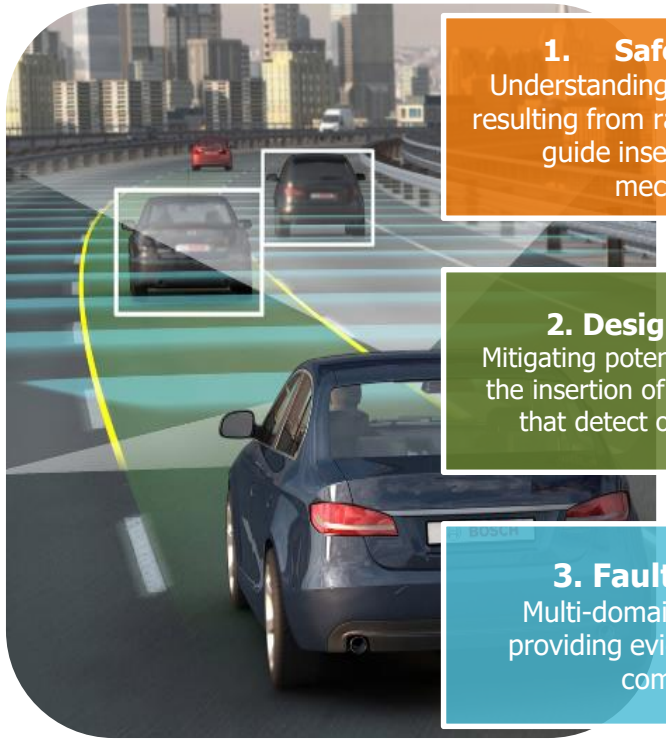
Automotive
Digital Twin

Security



System of Systems

Functional safety – failure analysis & higher reliability



1. Safety Analysis
Understanding the failure modes resulting from random HW faults to guide insertion of safety mechanisms

2. Design for Safety
Mitigating potential failures through the insertion of safety mechanisms that detect or correct failures

3. Fault Injection
Multi-domain fault injection providing evidence to achieve compliance

Veloce Fault App

Run Fault Campaigns

- **Perform mission-critical safety circuit verification**
- **Analyze effectiveness of safety mechanisms in the design**
- **Mimic the effects of transient and hard faults on the design**
- **Targeting safety critical industries (automotive, aerospace, military)**

Value: Optimize and accelerate Fault campaigns

Digital twin technology

Faster TTM

Reduce development times and increase quality while shortening time to market by shifting left

Greater Efficiency

More efficient and reliable software by providing high-speed virtual platforms long before silicon

Collaboration

Supports geographically dispersed teams collaborating on pre-silicon development and pre- and post-silicon debug

Real-time Insights

Track progress to requirements and schedule through incremental metrics for safety, security, power, performance and benchmarks pre-silicon

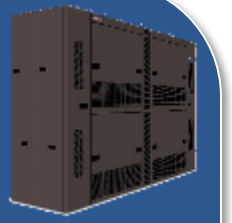
A complete autonomous vehicle verification and validation environment at system level

Sense

Siemens

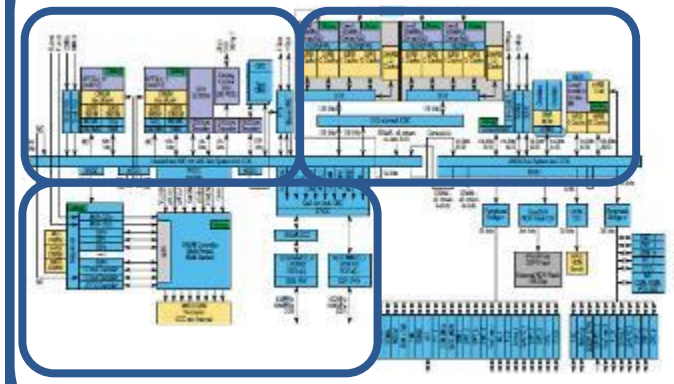
Tass' PreScan

Compute



Siemens Mentor

Hardware Verification Platform

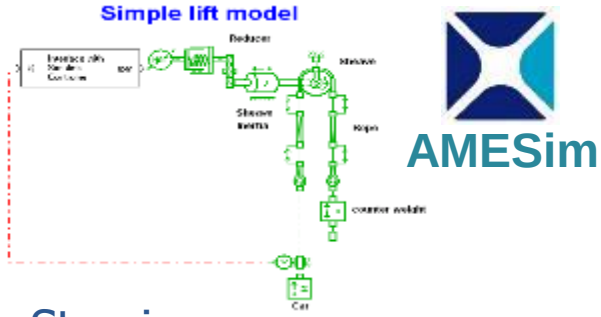


Veloce Strato

Actuate

Siemens

AMESim

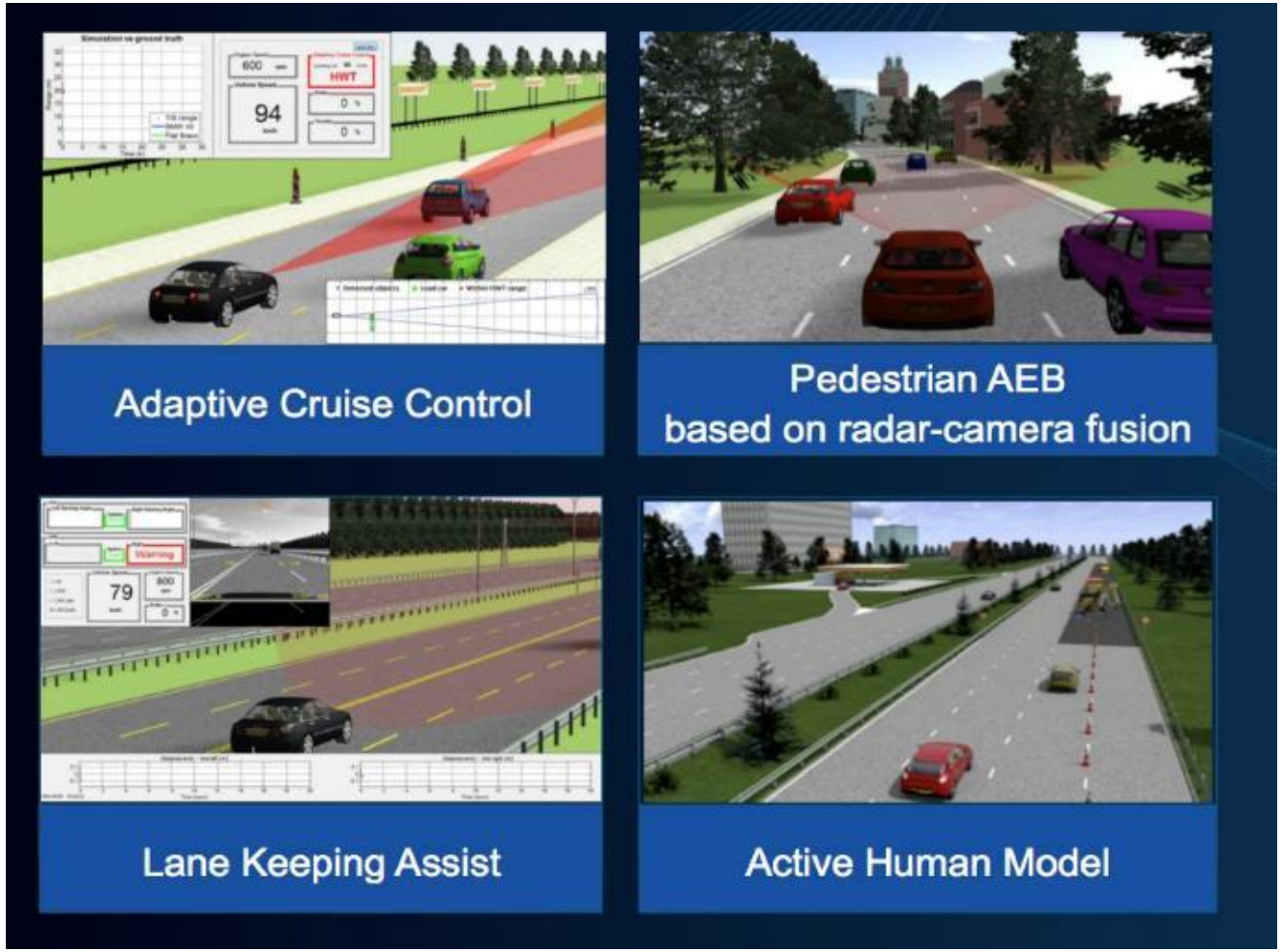


Steering

Braking

Drivetrain

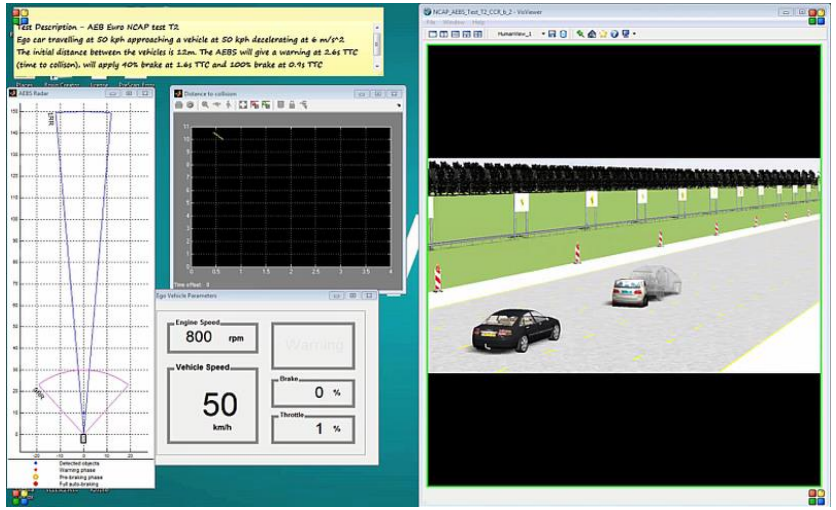
Virtual testing of autonomous driving functions accelerates time to safety goals



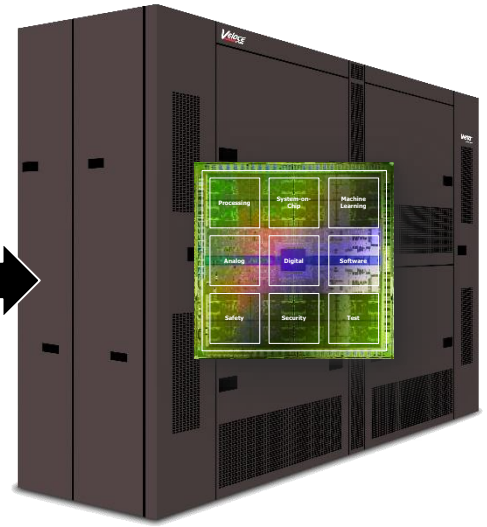
- Prescan (TASS International)
  - World modeling and scenario building
    - Road sections, bridges, etc.
    - Trees, buildings, traffic signs
    - Cars, trucks, pedestrians
    - Weather conditions
  - Sensor model library
    - Camera
    - Radar
    - Lidar
    - Ultrasonic
    - Infrared
    - V2X
    - GPS

High Performance Solution: PreScan with Veloce emulation

PreScan generates virtual driving scenarios and sensor data



Veloce verifies the most complex chip designs



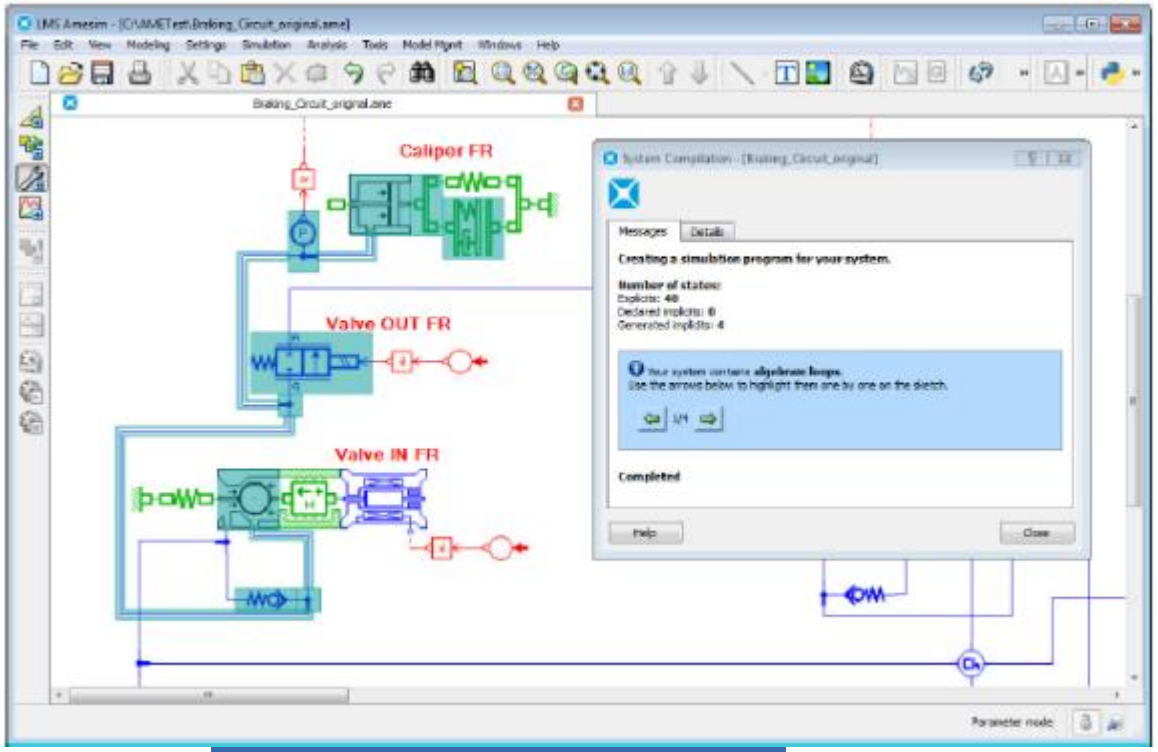
VELOCE STRATO

- Verification of ADAS chips in the context of many different traffic scenarios
- Full design visibility for comprehensive debug of SW and HW and SW/HW interactions

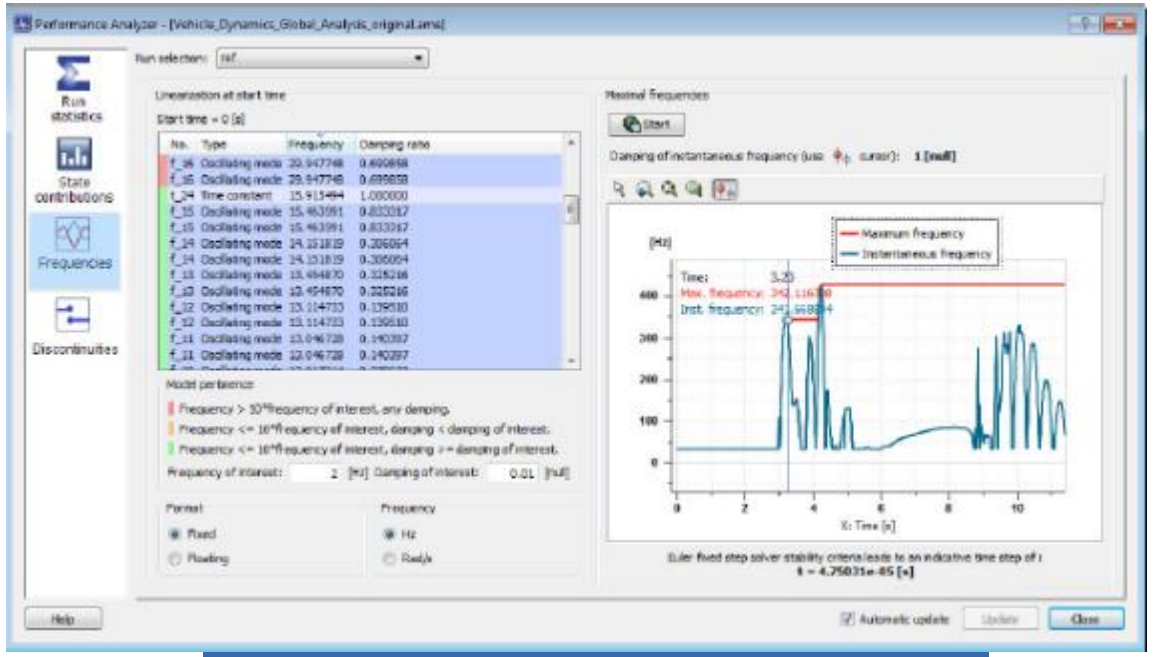
SENSE + COMPUTE = VALUE

LMS Amesim
Platform for multi-domain system simulation

- Unique capabilities to create real-time ready models preserving physical relevance



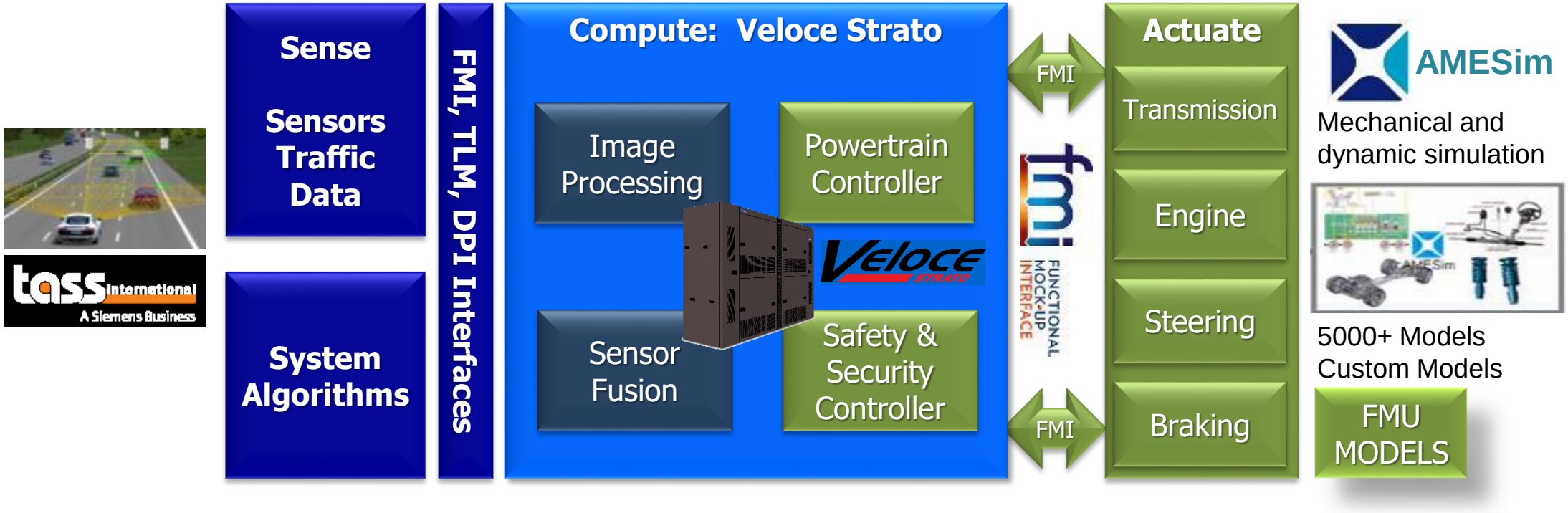
Highlighted algebraic loops



Performance analyzer - Frequency analysis of the chassis model

Comprehensive verification to shift left the development cycle

- Integrated heterogeneous system of systems framework to simulate and verify multiple ECUs



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Verification of System of Systems with Multiple ECUs using Digital Twin Virtual Environment

1 Dashboard ECU

- **Modeled using Vector CANoe**
- Packet Exerciser & Monitor:
  - Get Different driving inputs
  - Display speed/RPM

2 Engine Control ECU

- **PowerPC Virtual Platform (Mentor Vista)**
- AUTOSAR Stack + Application:
  - Reads Input Combination from Dashboard & gives Command/Pedal/Brake Angle to Transmission Controller ECU

3 Transmission Controller ECU

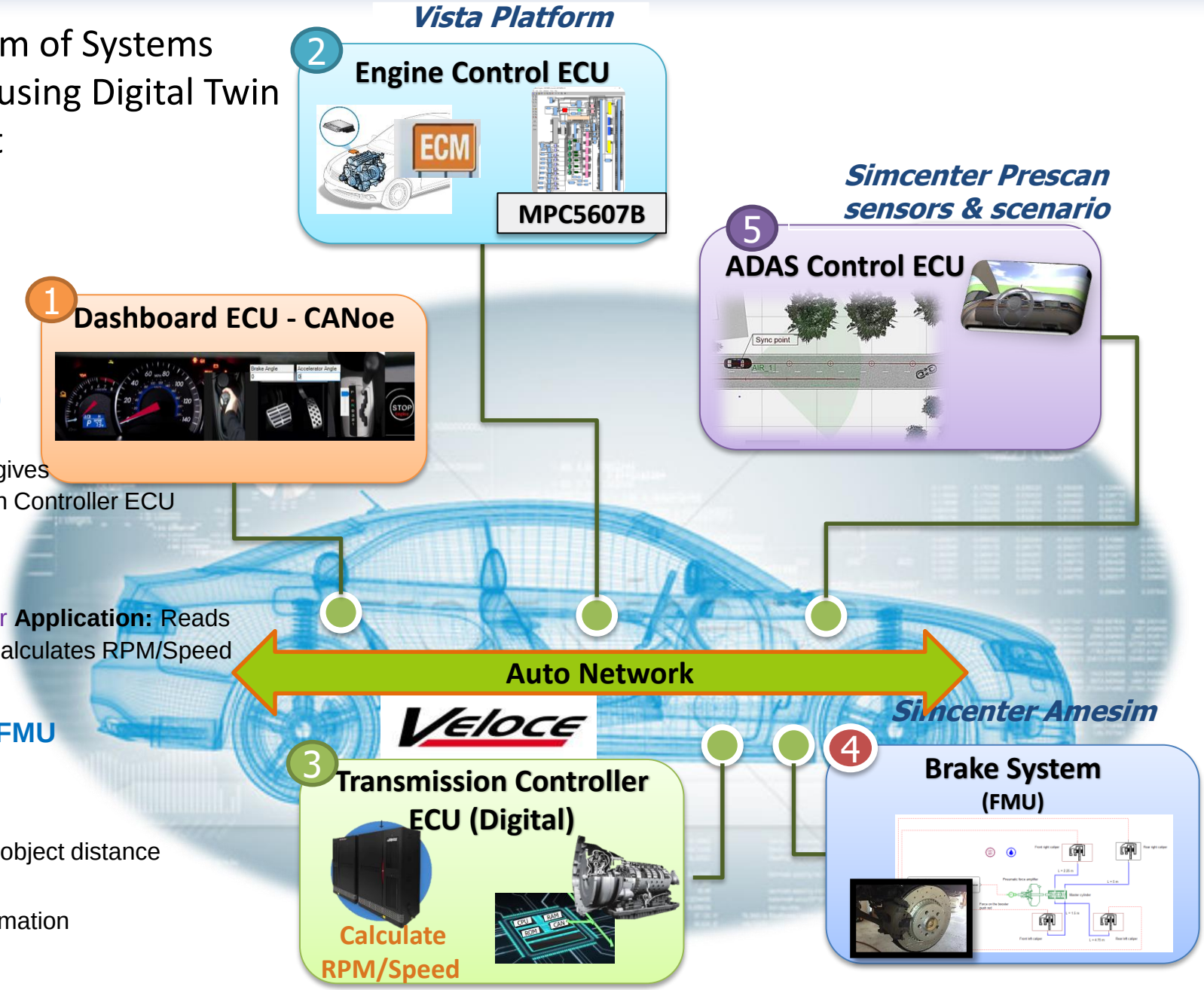
- **Hardware Model (Mentor Veloce Strato)**
  - RISC-V + Memory Subsystem + CAN Controller **Application:** Reads Pedal Angle from Engine Controller ECU and Calculates RPM/Speed

4 Brake System

- **Simcenter Amesim Co-simulation Slave FMU**

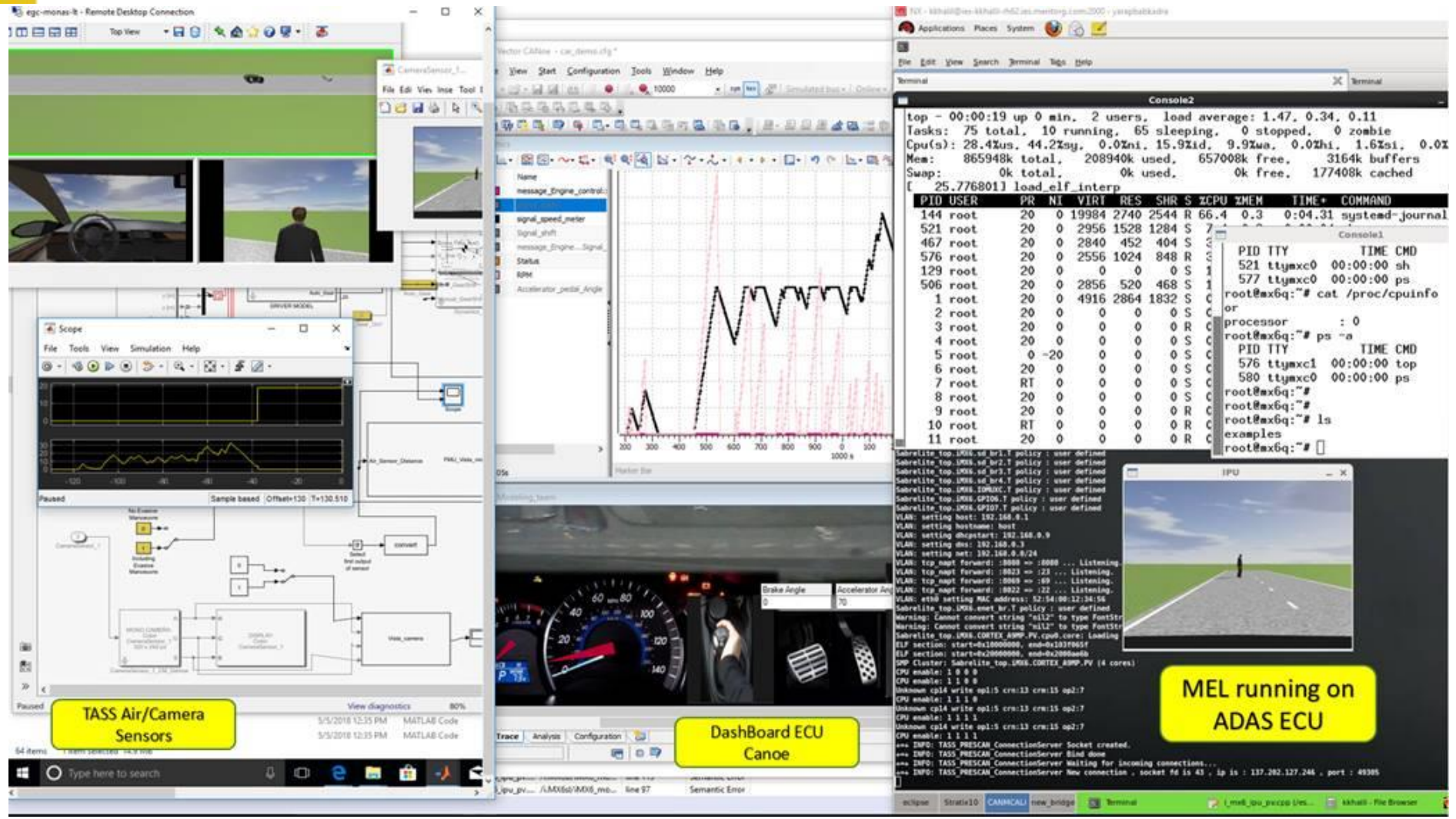
5 ADAS Control ECU

- **Mentor Veloce HYCON**
  - Read camera stream, detect object and supply object distance
- **Simcenter Prescan**
  - Perform evasive maneuver using distance information



accellera SYSTEMS INITIATIVE

DEMO : Advanced Emergency Braking System



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Summary: Veloce Transforms Automotive Design

Secure rapid achievement of safety requirements

- **Addresses complexity, size, and accurate timing needs of automotive electronic systems**
  - Full visibility and debug for automotive designs
- **Removes Risk associated with Safety Critical Systems**
  - Functional verification for systematic failure analysis
  - Safety verification for random failure analysis
  - Fault tolerance and coverage
  - ISO 26262 compliance
- **Digital Twin Functionality**
  - Full ECU verification to shift left the development cycle
  - Delivering on time-to-market

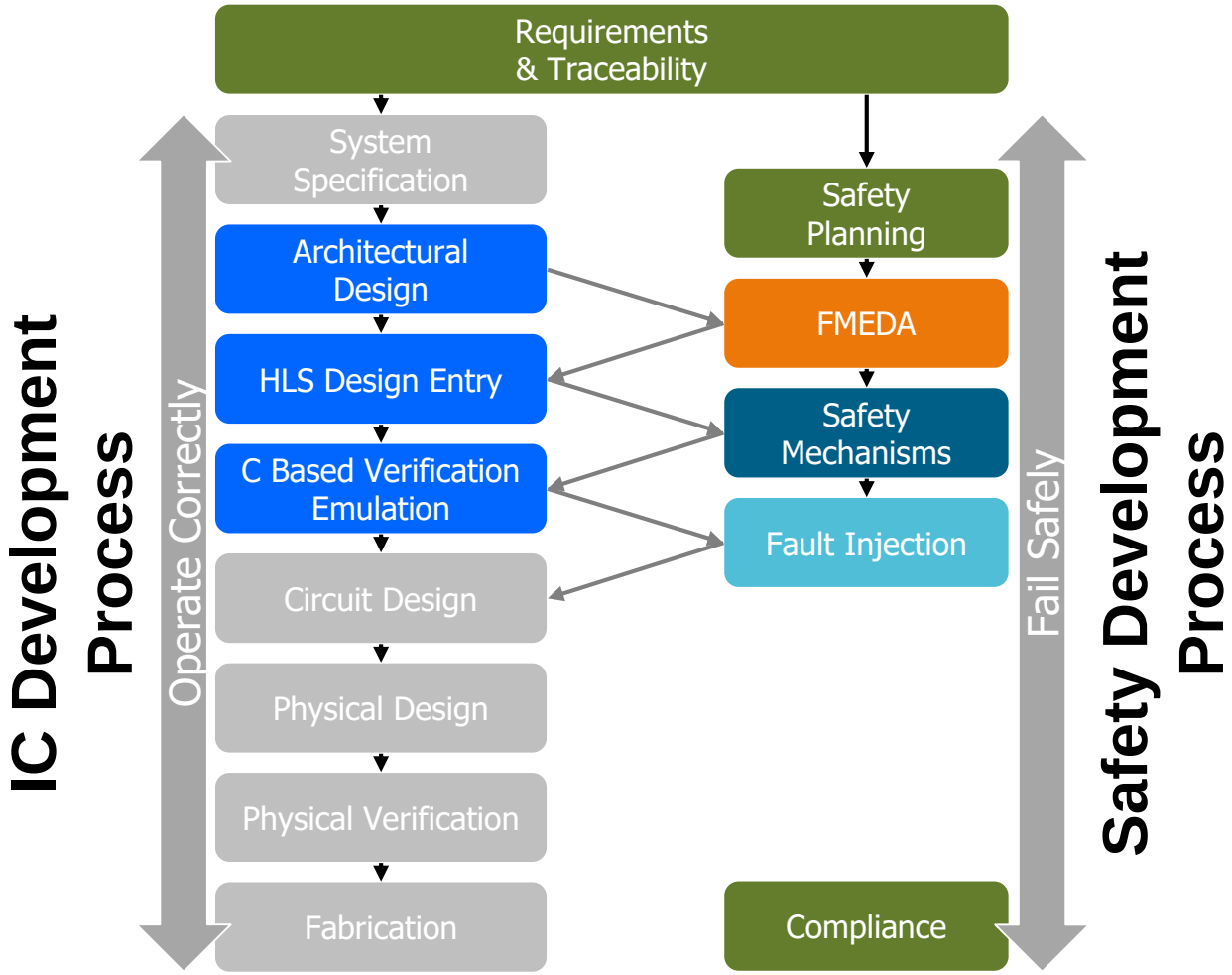
Veloce STRATO

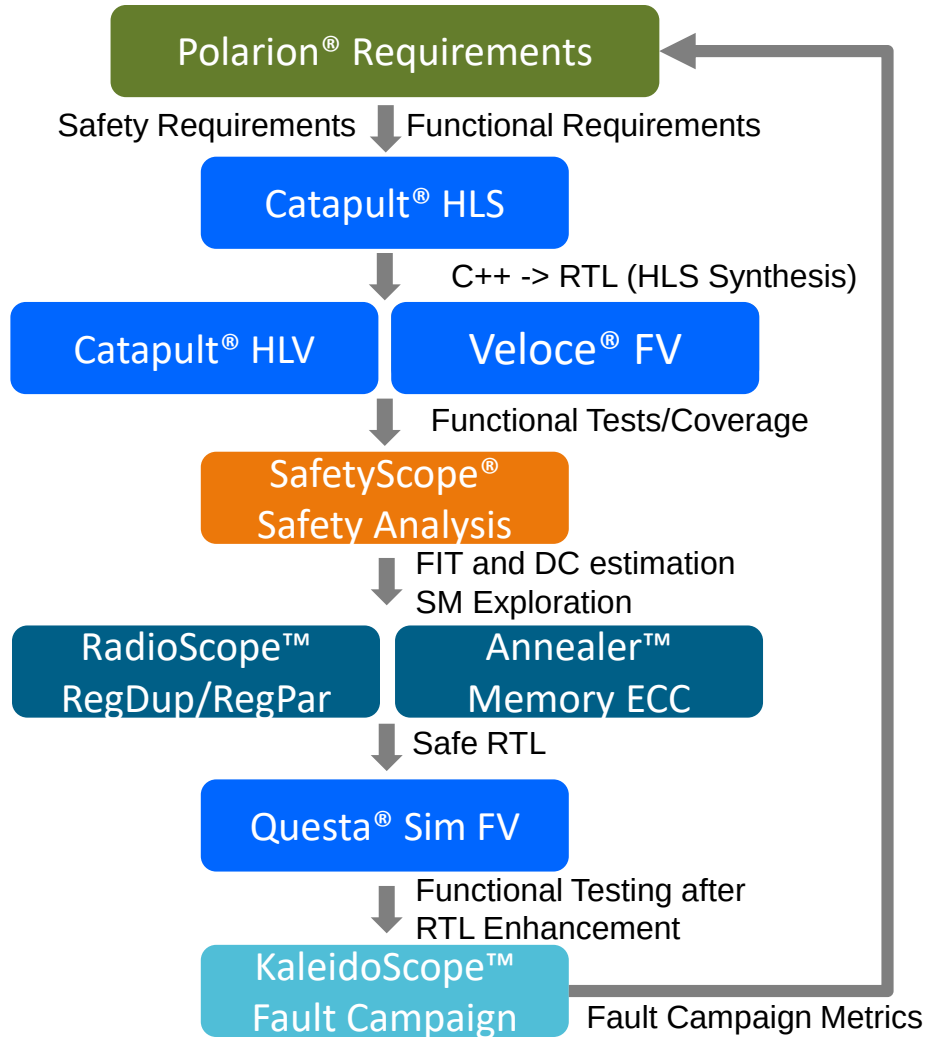
ISO 26262

accellera SYSTEMS INITIATIVE

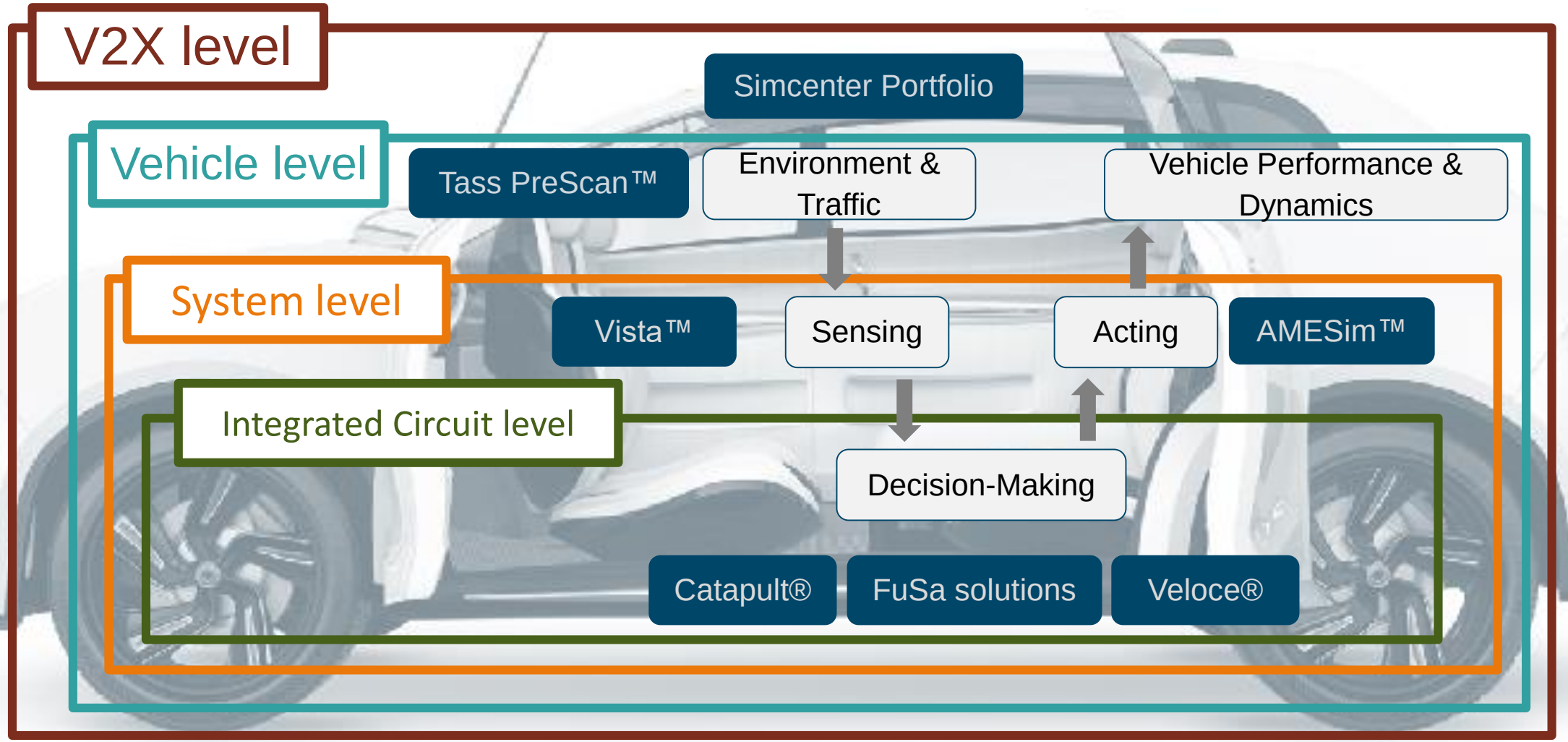
Conclusion

Mentor® HLS, Emulation and FuSa Workflow





Siemens and Mentor provide comprehensive transportation solutions!



2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Questions?

accellera SYSTEMS INITIATIVE