XDF XILINX DEVELOPER FORUM

Accelerating ADAS Computer Vision Application Development at Ford using SDSoC

Presented By

Vijay Nagasamy
Research Scientist – Autonomous Vehicles
October 2, 2018

XILINX

Accelerating ADAS Computer Vision Application Development at Ford using SDSoC

Vijay Nagasamy

Research Scientist – Autonomous Vehicles

Ford Greenfield Labs, Palo Alto

Ford Motor Company

Ford

Go Further

Agenda

- Background
- Project Challenges
- Design Approach
- Project Results
- What we liked/what could be improved

Go Further

Driver Assistance Trends

- Warnings -> active control -> fully auton.
- Cameras are important
  - 1 camera -> 10+ cameras
- Machine vision technology is critical
- Vision application -> deep learning

My Background

- 25+ years of industry experience
  - Top-Down Design Methodology, High Level Synthesis (HLS)
- Developed early HLS tools and environment for ASIC design at LSI Logic
  - Co-inventor of several Patents in this field
- Managed ASIC/FPGA teams at Fortune 500 companies and startups
- Expertise in several domains
  - Computer Vision, Image Processing, Machine Learning, IoT, Telecom
- Consulted initially at Ford Research & Innovation Center in 2017
- Currently, Research Scientist in the Autonomous Vehicles group at Ford

Ford Driver Assistance System Project

Development Team

- Diverse group of research engineers and scientists with software background; limited FPGA development expertise

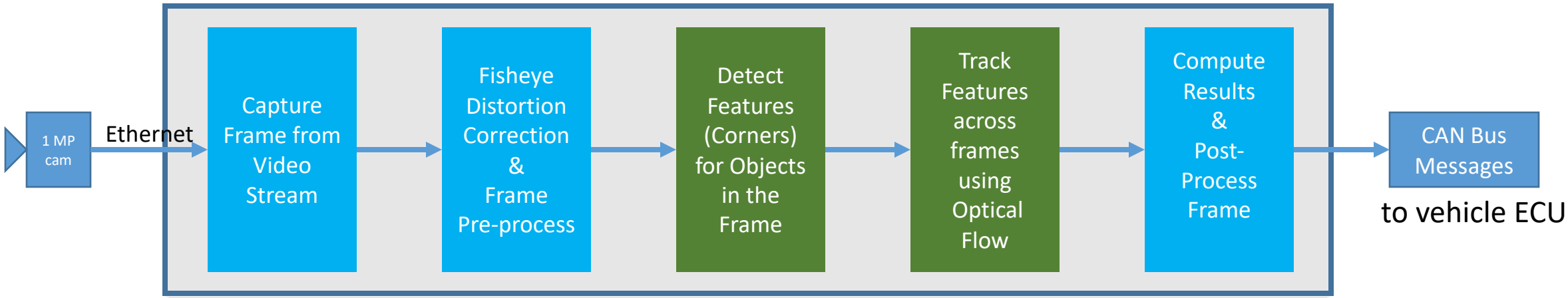
Initial implementation

- Team developed Machine Vision algorithms for a Driver Assist application
- Running at 30 frames/sec on a Linux PC platform

Hardware Platform Selection

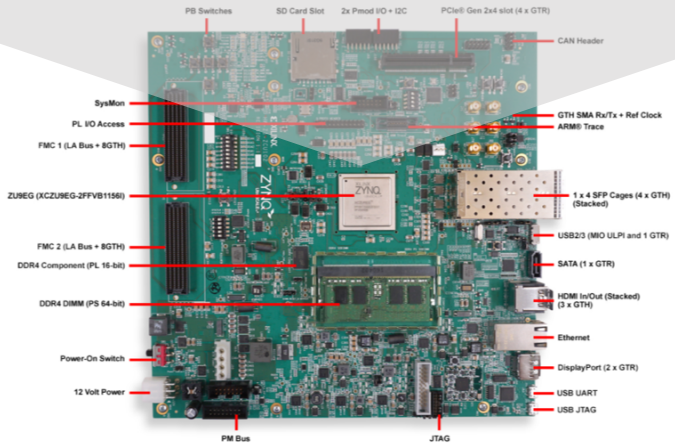
- Team selected Xilinx but had not yet selected a device or design approach

System Project Block Diagram



<u>FUNCTIONS</u>

- Capture image from camera
- Correct for lens distortion
- Detect and track objects in view
- Compute and communicate results to the vehicle ECU



<u>GOAL</u>

Implement application on a Zynq MPSoC device

Accelerate OpenCV library blocks

- Corner Detection
- Optical Flow

Ford Go Further

Project Goals and Challenges

Speed/Performance

- Meet performance of at least 10-15 fps on the hardware

Flexible and Versatile

- Hardware needs to run and accelerate other Driver Assist functions as well

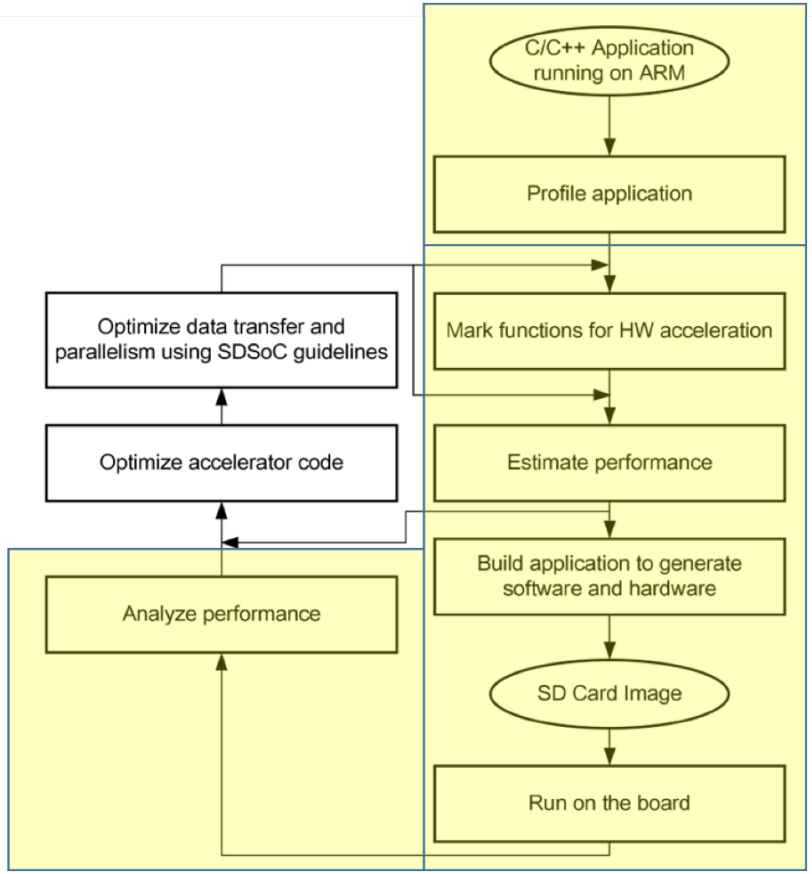
Hardware Platform Selection

- Determine hardware implementation feasibility within 2 months
- Develop overall “Concept Ready” solution within 10 months
- Train research engineer to write efficient C++ code targeting Zynq MPSoC

Design Approach

- Selected Xilinx ZU9EG device and ZCU-102 board for development
- Selected SDSoC as the best tool/environment for the project
- Used OpenCV libraries and high level design methodology
- Used a top-down design implementation approach

Top-Down Iterative Design Methodology



Source: Xilinx SDSoC Environment User Guide, 2017

Existing C++ Algorithm

Algorithm Developer

Tune algorithm on a PC platform

Optimize Application

Analyze Performance

Modify C++ for ARM using OpenCV Library
Replace Identified OpenCV SW functions with Xilinx accelerated HW functions

Cross-compile for target ARM platform using SDSoC Environment & Libraries on a Windows/Linux PC

Embedded SW developer

SD Card Image

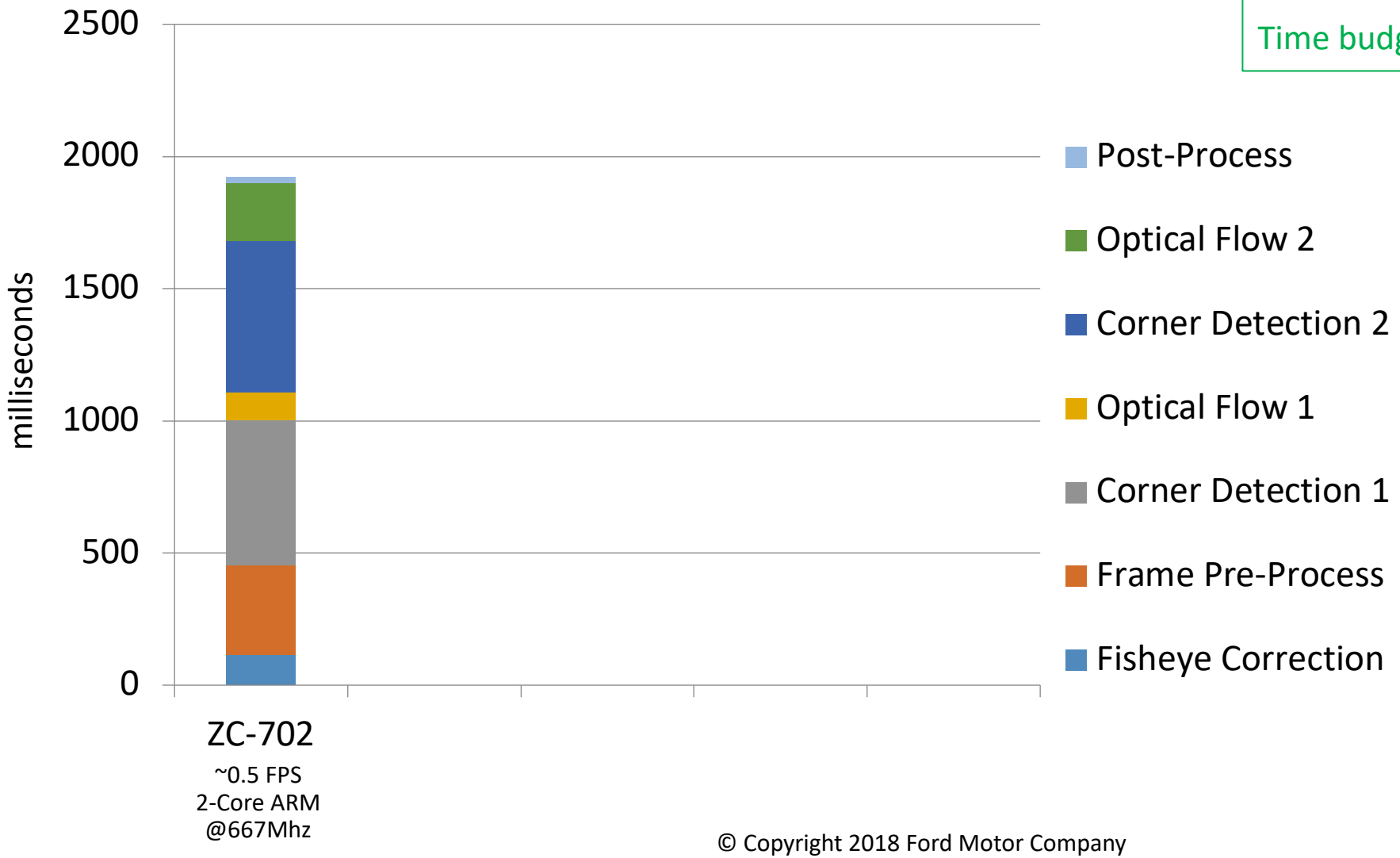
Run on Xilinx ZCU-102 development platform

Analyze Performance Profile

Results: Time to Process a Frame

Time budget per frame for 15 FPS = 66ms

Time budget per frame for 10FPS = 100ms



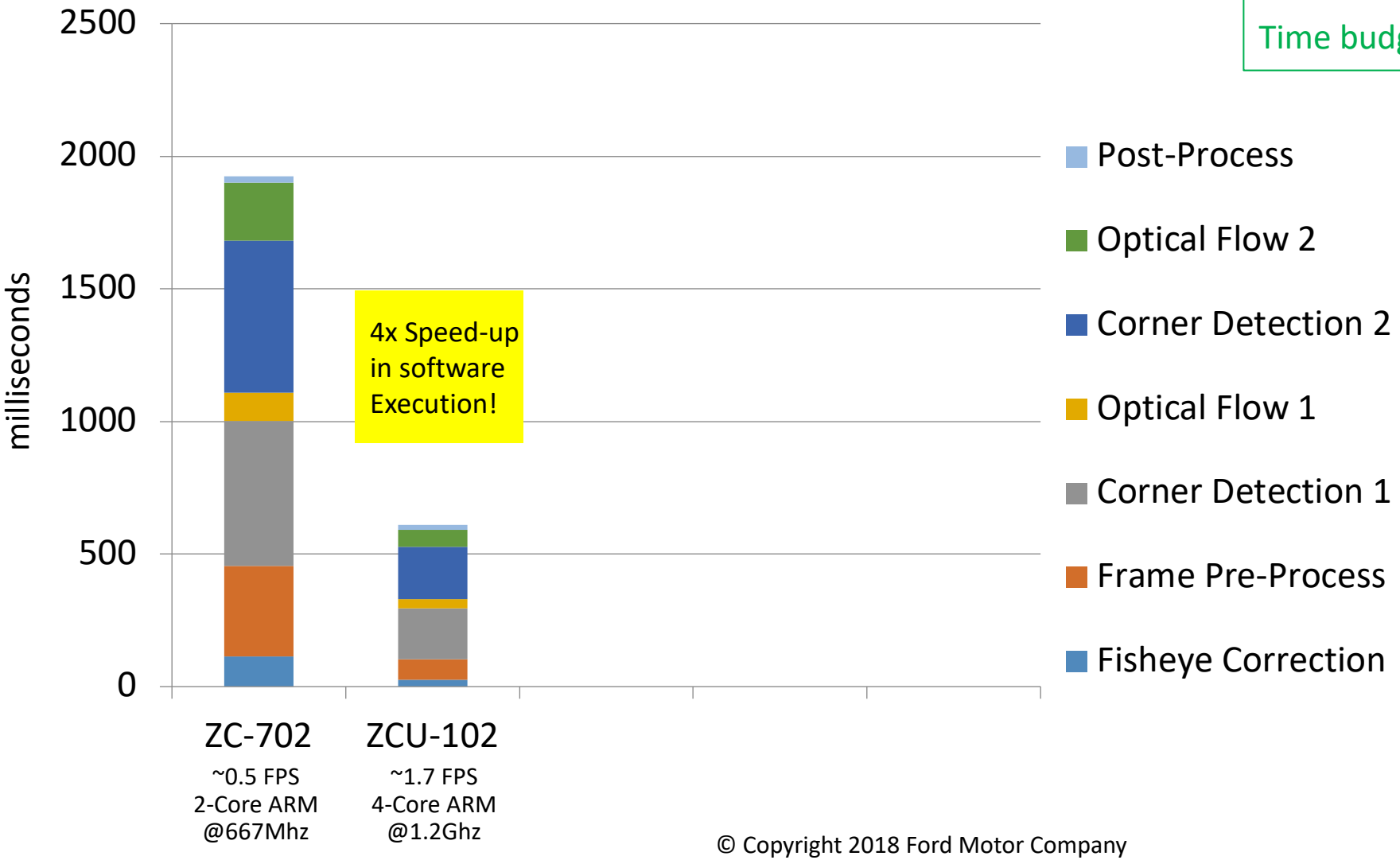
- Initially ran application on Zynq Ultrascale device
- Too far from performance target
- Insufficient PL resources to accelerate critical functions

Ford
Go Further

Results: Time to Process a Frame

Time budget per frame for 15 FPS = 66ms

Time budget per frame for 10FPS = 100ms



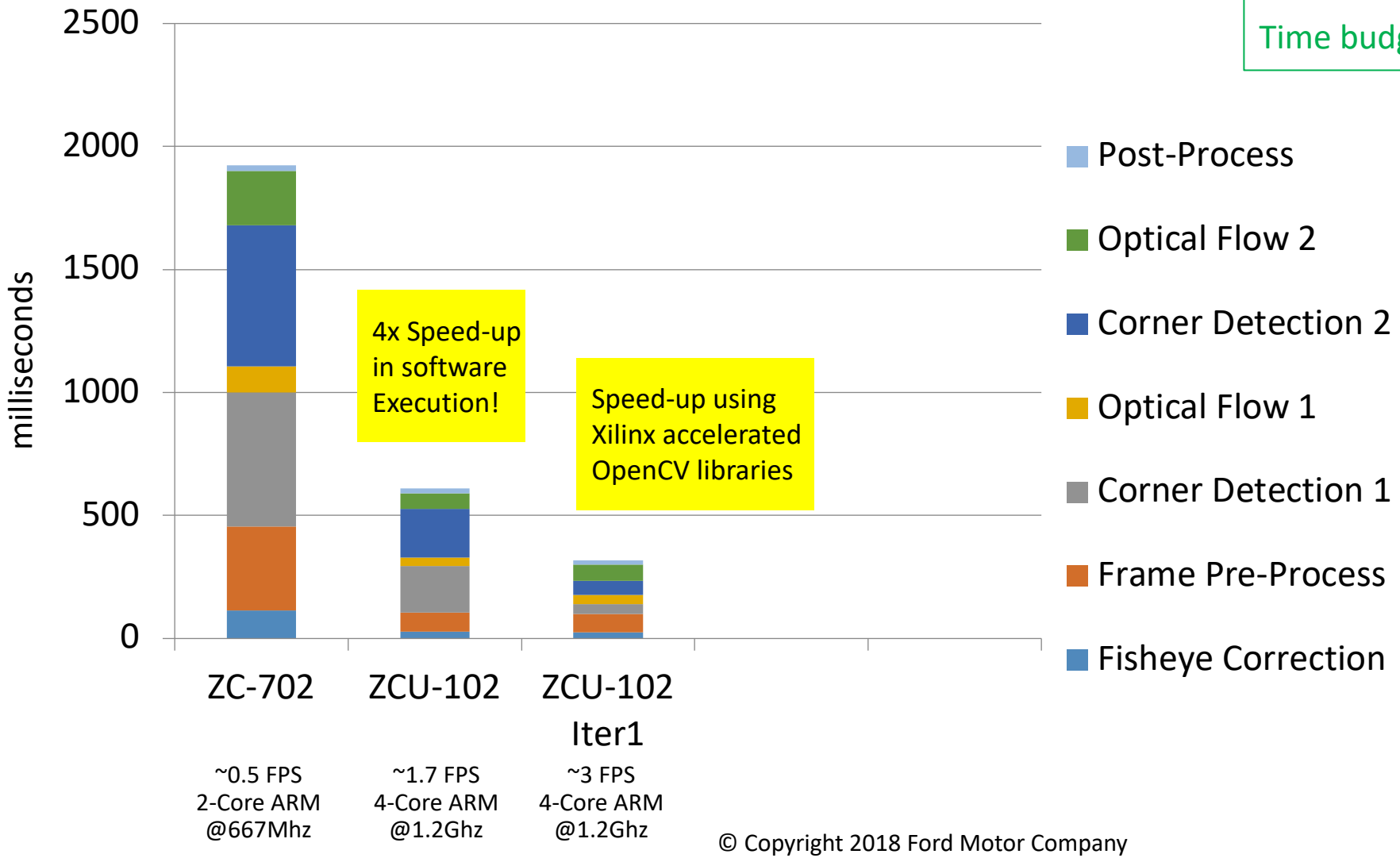
- Moved to Zynq Ultrascale+ device family
- Achieved ~4x performance improvement running in software mode

Ford
Go Further

Results: Time to Process a Frame

Time budget per frame for 15 FPS = 66ms

Time budget per frame for 10FPS = 100ms



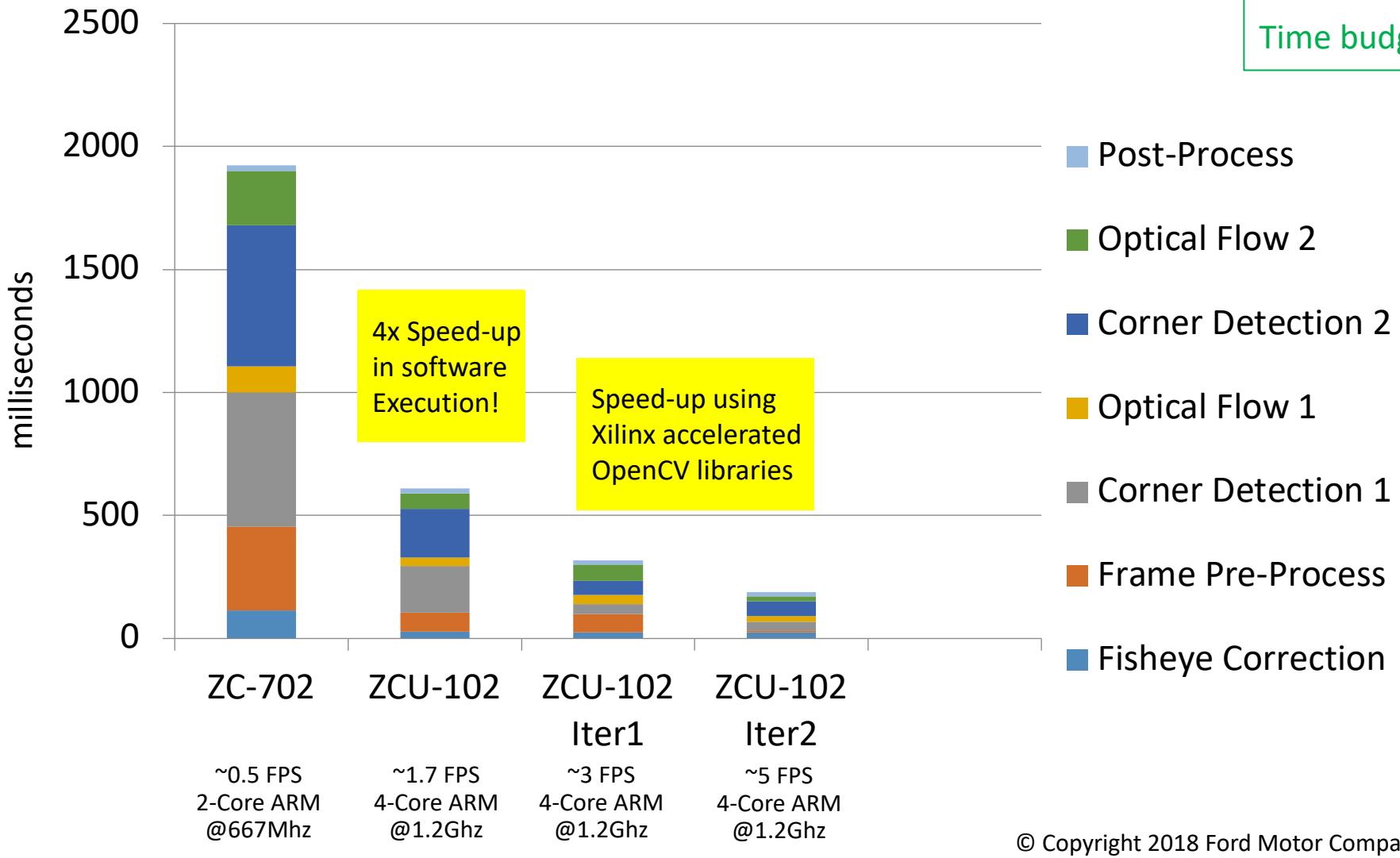
- Moved Corner Detection function to PL using "Fast Corner" block from the Xilinx reVISION library

Ford Go Further

Results: Time to Process a Frame

Time budget per frame for 15 FPS = 66ms

Time budget per frame for 10FPS = 100ms



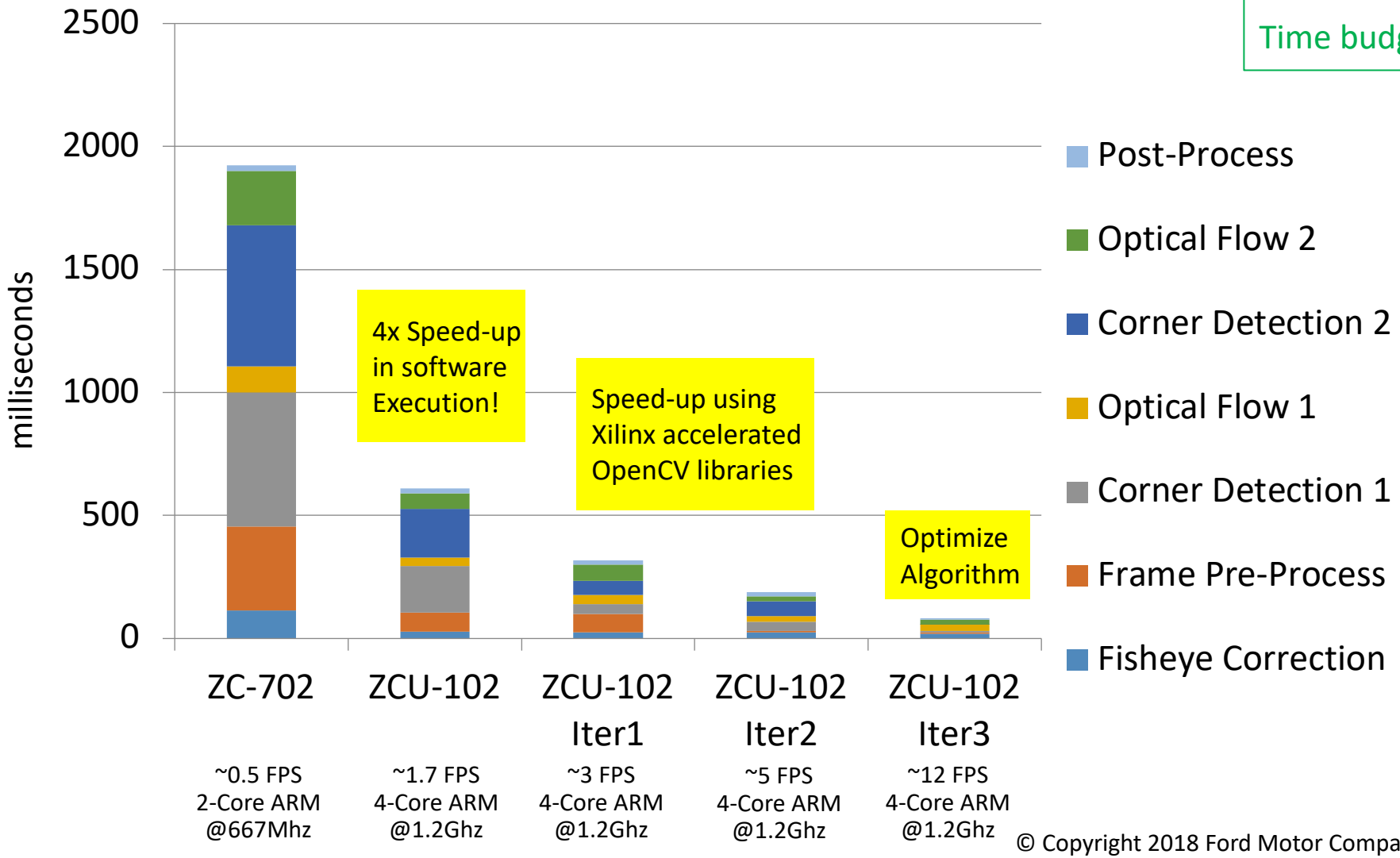
- Moved Optical Flow function to PL using the Xilinx reVISION library
- Optimized frame pre-process in software

Ford Go Further

Results: Time to Process a Frame

Time budget per frame for 15 FPS = 66ms

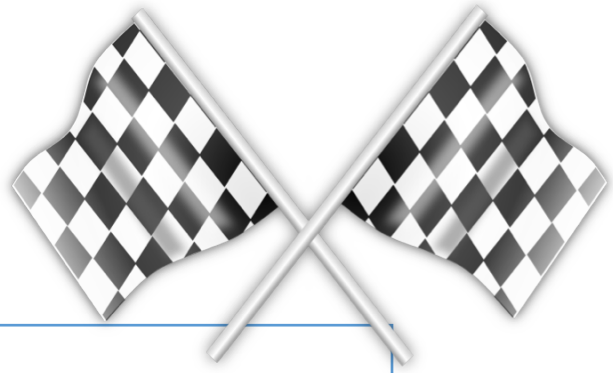
Time budget per frame for 10FPS = 100ms



- Optimized Algorithm
  - Process smaller ROIs
- Achieved target of 12 frames/sec!

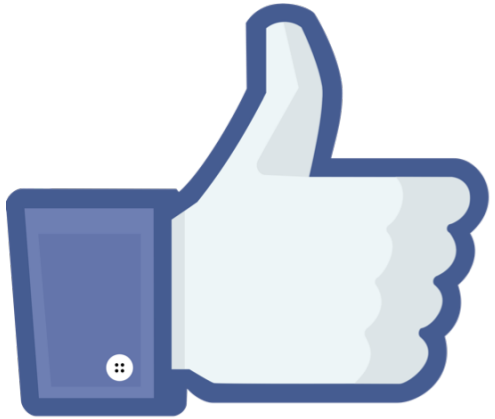
Ford Go Further

Project Results



- Completed initial setup and profiling within 1 month
- Succeeded in handing off project to Ford engineering team
  - 2 Algorithm/Research eng., 1 FPGA implementation and 2 embedded SW dev's
- Achieved target frame rate of 12 fps
- Overall project using Xilinx FPGA completed in 10 months, on schedule
  - Algorithm implementation portion on the FPGA was completed in 4 months
- Passed 1st gate of multi-year production release process

What We Liked



- SDSoC tool & environment was easy to use
- Familiar C/C++ development environment
- Short learning curve for an embedded software engineer
- Relevant examples from Xilinx to help learn the environment
- Xilinx support for early access OpenCV libraries was instrumental to a successful project implementation
- Excellent field support from Xilinx

What Could Be Improved

- Expand support for more OpenCV library functions accelerated in hardware
- Make it easier to import C/C++ algorithms developed on a PC to Xilinx SDSoC environment

Acknowledgments

The FORD Project Team

Vidya Murali

Bindu Sairamesh

Nikhil Rao

Bruno Costa

Yi Zhang

Dongran Liu

The Xilinx Team

Alvin Clark

Quang Nguyen

Kamran Khan

Varun Santhaseelan

David Lam

Summary

- SDSoC-based top-down flow was used successfully by embedded SW developers with limited FPGA design experience
- Key to success was setting up the platform and training developers on design methodology
- Early access to Accelerated OpenCV libraries and Xilinx support were essential


XDF
XILINX
DEVELOPER
FORUM