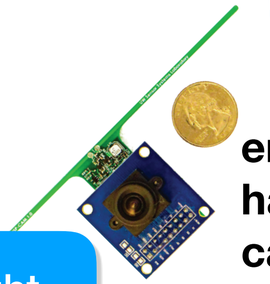
Exploring Computation-Communication Tradeoffs in Camera Systems

Amrita Mazumdar
Armin Alaghi
Thierry Moreau
Luis Ceze
Sung Kim
Mark Oskin
Meghan Cowan
Visvesh Sathe

IISWC 2017

Camera applications are a prominent workload with tight constraints

low-power



energy harvesting camera

light weight

light weight

real-time processing

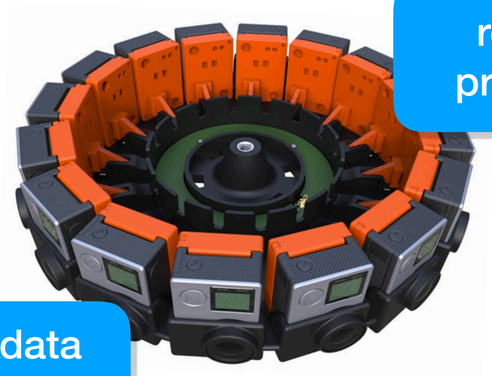
augmented reality glasses

low-power

real-time processing

video surveillance cameras

large data size



large data size

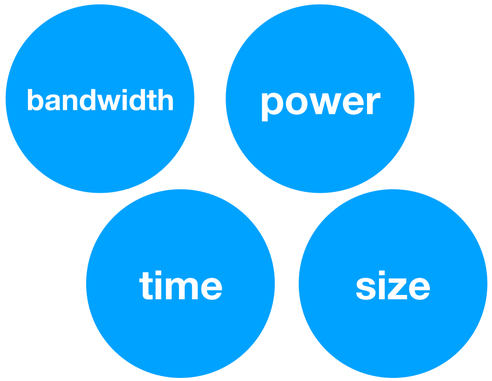
3D-360 virtual reality camera rig

Hardware implementations compound the camera system design space

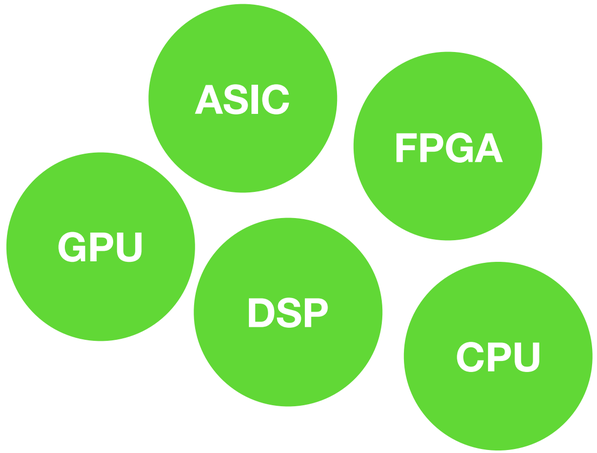
camera system

DogChat™

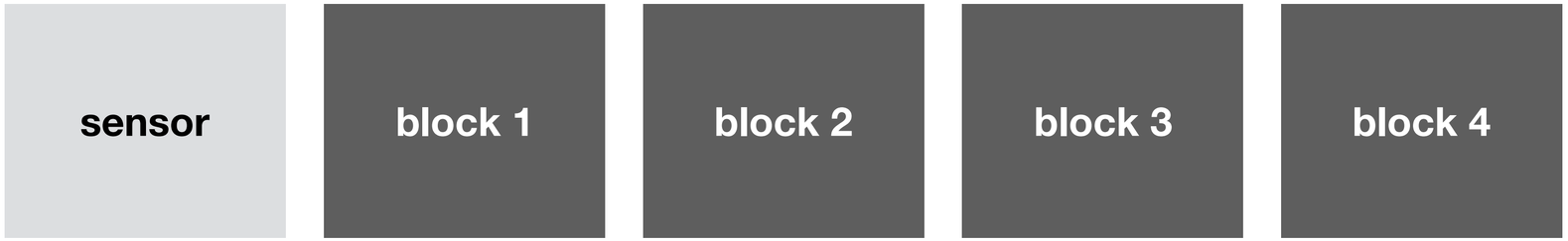
constraint



implementation



We can represent camera applications as camera processing pipelines to clarify design space exploration



functions in the application

We can represent camera applications as camera processing pipelines to clarify design space exploration

DogChat™

Developers can trade off between computation and communication costs

DogChat™

sensor

image processing

face detection

feature tracking

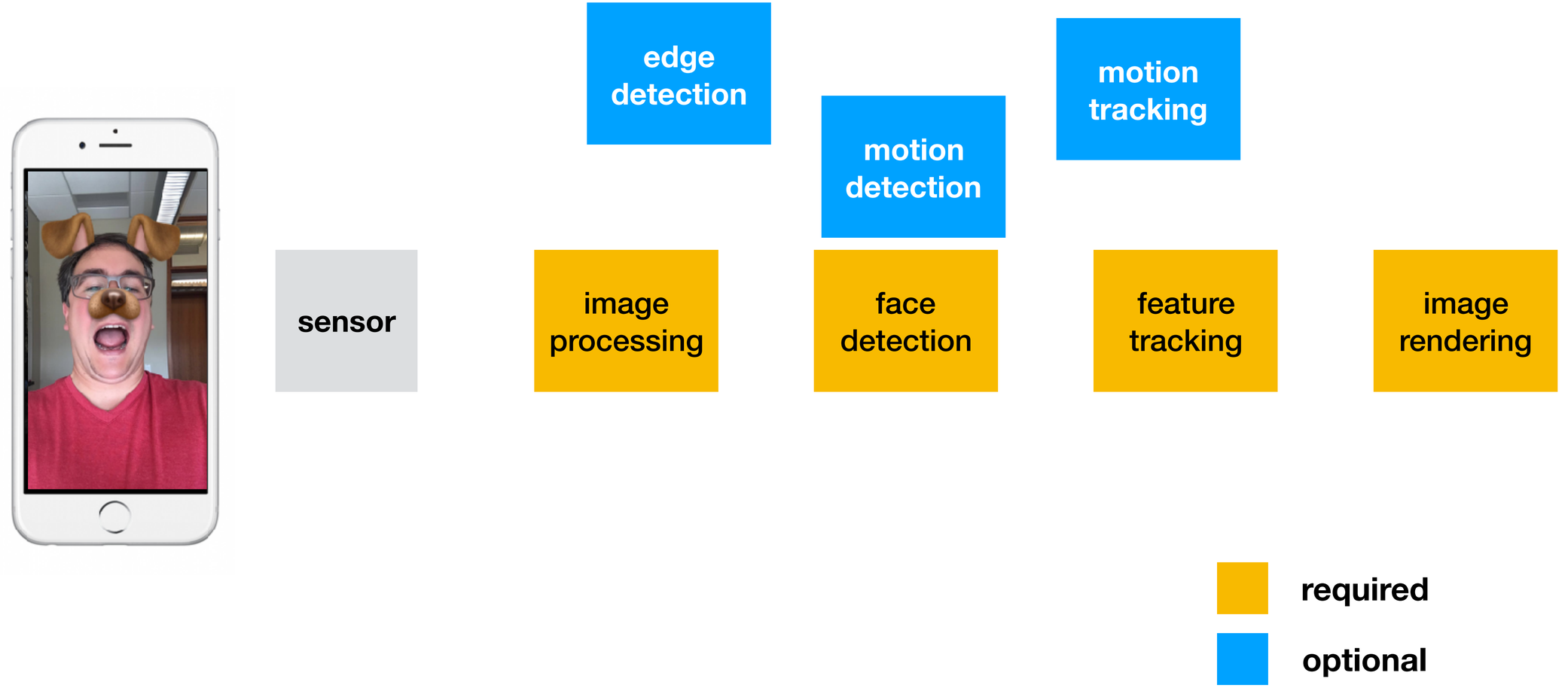
image rendering

offloaded to cloud

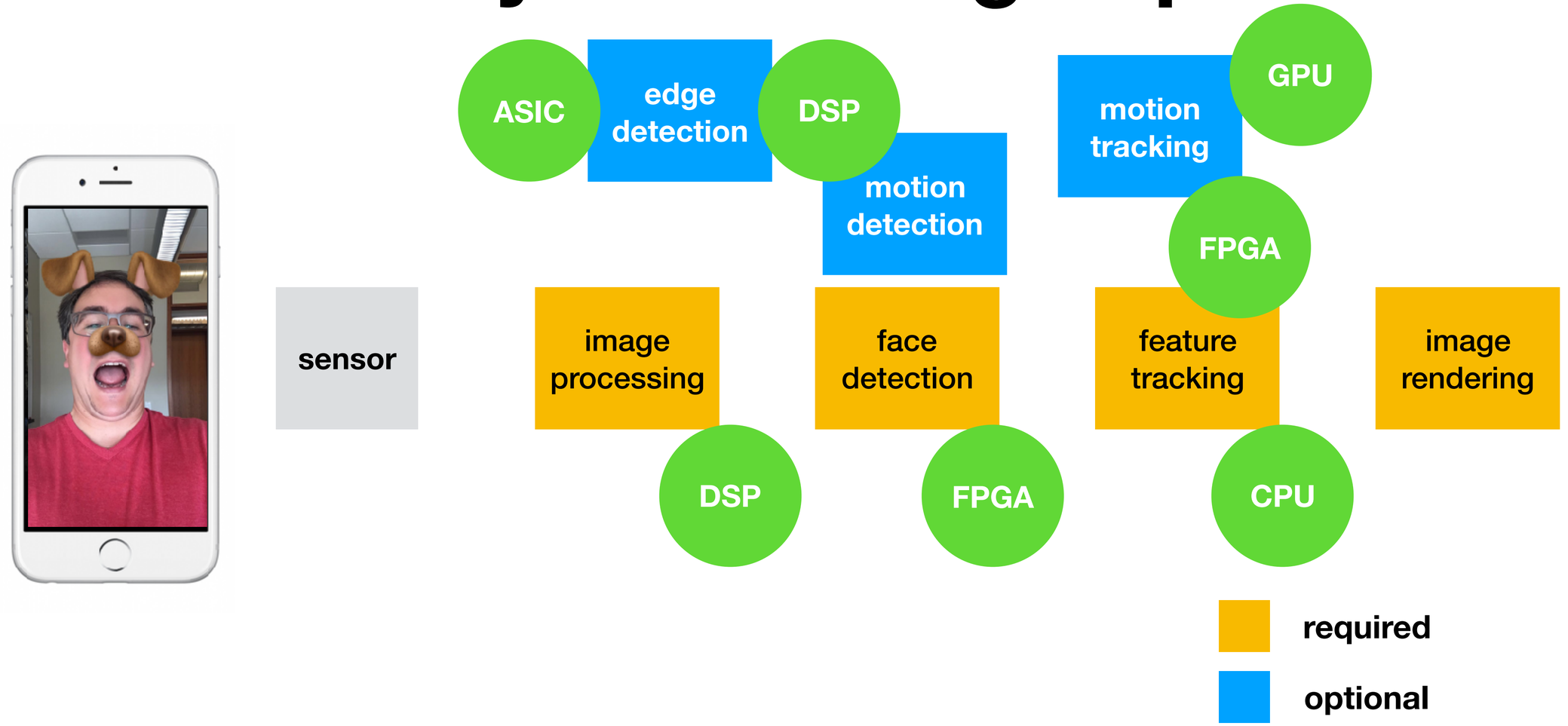
Developers can trade off between computation and communication costs

DogChat™

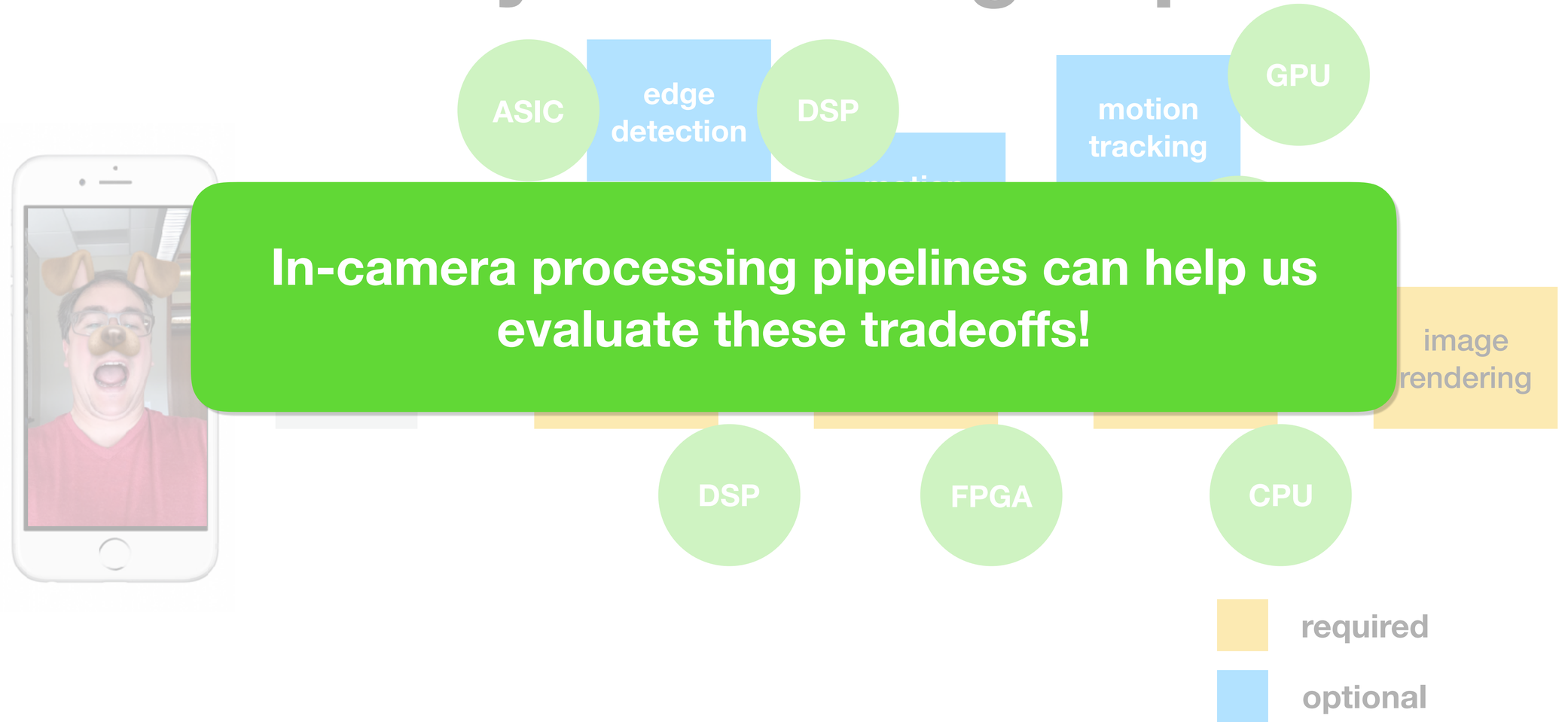
Optional and required blocks in camera pipelines introduce more tradeoffs



Custom hardware platforms explode the camera system design space



Custom hardware platforms explode the camera system design space

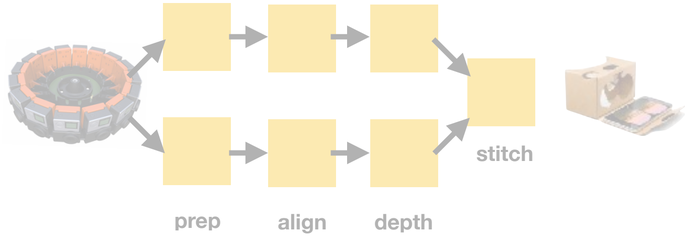


In-camera processing pipelines can help us evaluate these tradeoffs!

Challenges for modern camera systems

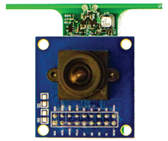
Low-power: face authentication for energy-harvesting cameras with ASIC design

Low latency: real-time virtual reality for multi-camera rigs with FPGA acceleration



Challenges for modern camera systems

Low-power: face authentication for energy-harvesting cameras with ASIC design

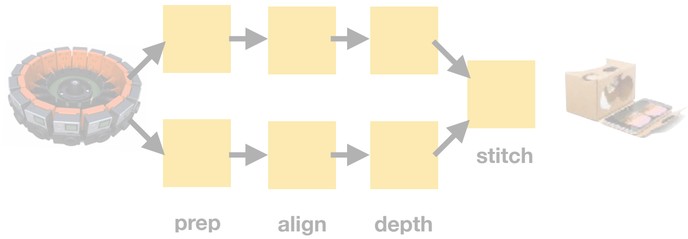


motion detection

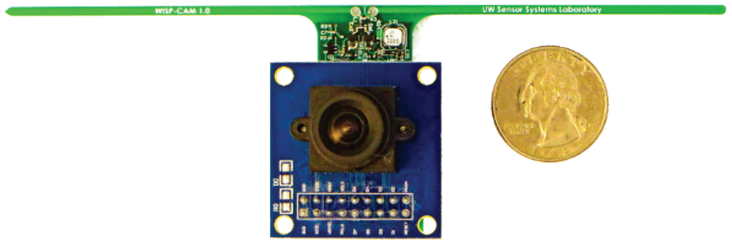
face detection

neural network

Low latency: real-time virtual reality for multi-camera rigs with FPGA acceleration



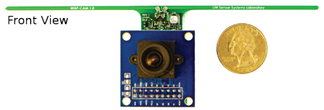
Face authentication with energy harvesting cameras



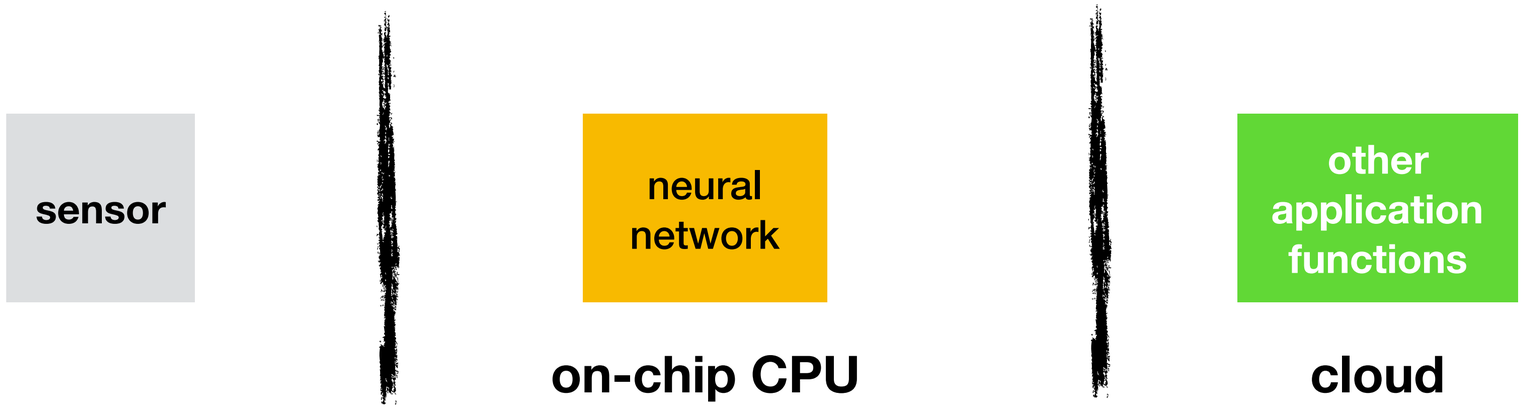
WISP Cam
energy-harvesting camera
powered by RF
1 frame / second
~1 mW processing / frame

Face authentication with energy harvesting cameras

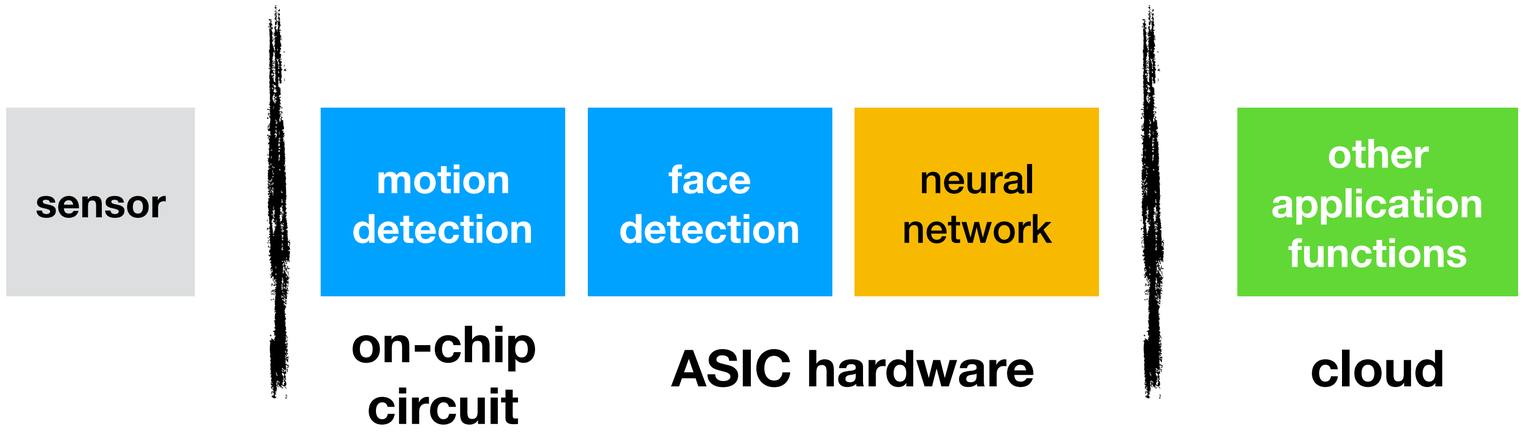
Is this Armin? ✅



CPU-based face authentication neural networks can exceed WISPcam power budgets



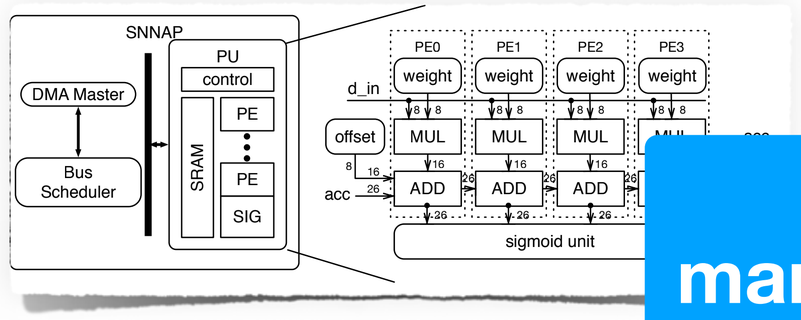
CPU-based face authentication neural networks can exceed WISPcam power budgets



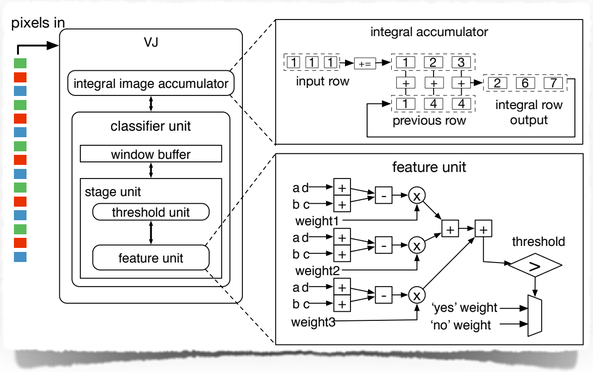
adding optional blocks can reduce power consumption for a neural network

Exploring design tradeoffs in ASIC accelerators

neural network



face detection



many more details in paper!

Evaluated NN topology and har… impact on energy and accuracy

…eaming face detection …celerator

Selected a 400-8-1 network topology and used 8-bit datapaths for optimal energy/accuracy point

Explored classifier and other algorithm parameters to optimize energy optimality

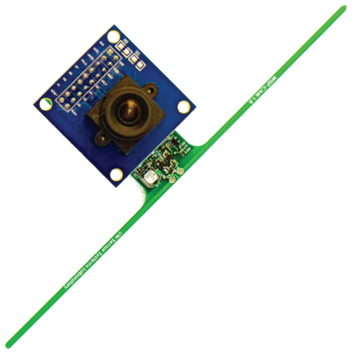
Evaluation

Which pipeline achieves the lowest overall power?

Synthesized ASIC accelerators in Synopsys

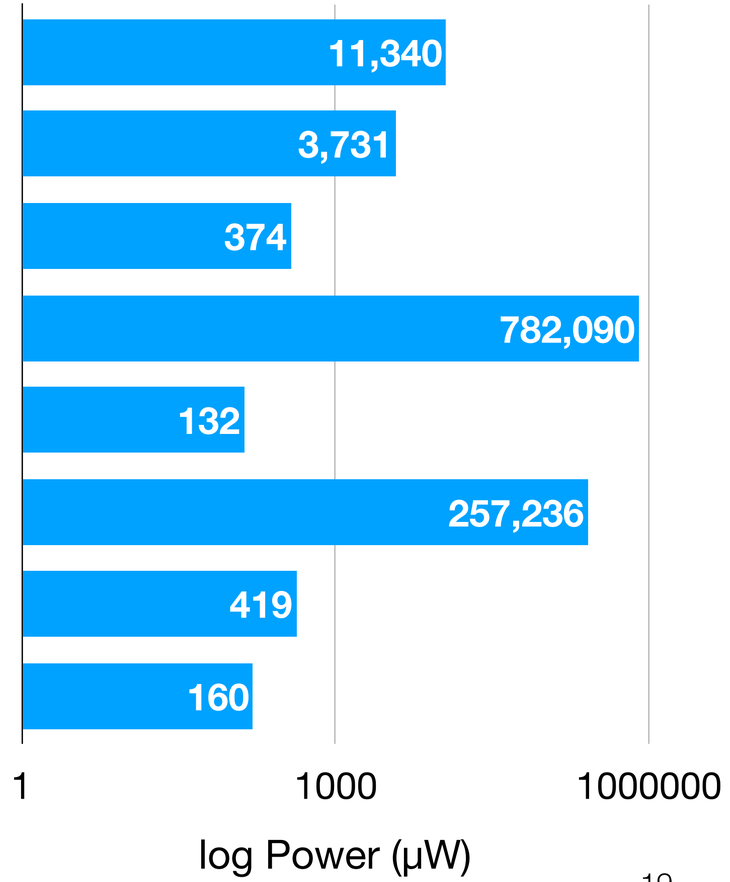
Constructed simulator to evaluate power consumption on real-world video input

Computed power for computation and transfer of resulting data for each pipeline configuration



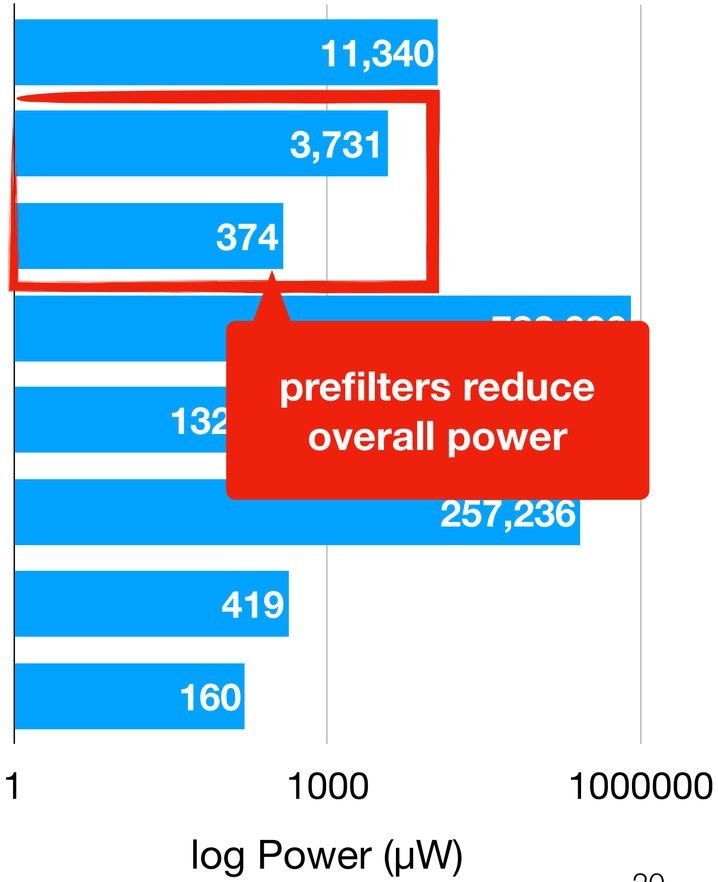
Which pipeline achieves the lowest power consumption?

platform configuration				(ratios) compute	transfer
sensor					
sensor	motion				
sensor		face detect			
sensor			NN		
sensor	motion	face detect			
sensor	motion		NN		
sensor		face detect	NN		
sensor	motion	face detect	NN		



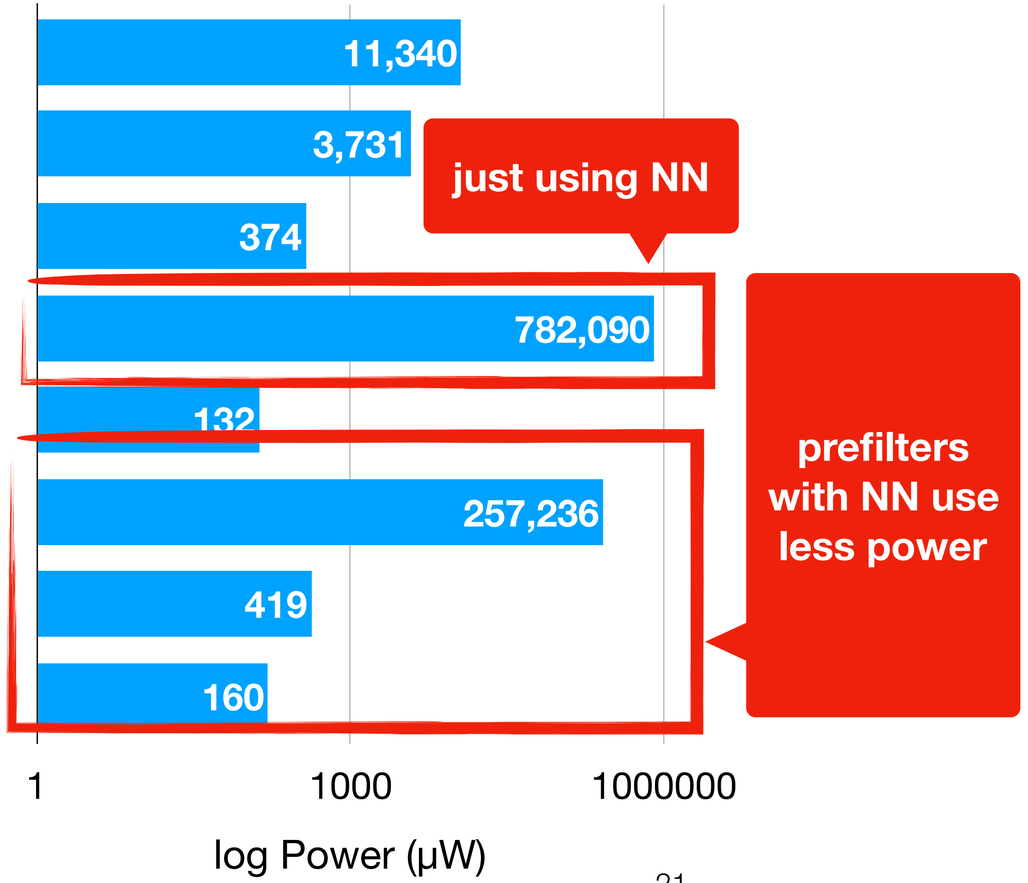
Which pipeline achieves the lowest power consumption?

platform configuration				(ratios) compute	transfer
sensor				<1%	>99%
sensor	motion			<1%	>99%
sensor		face detect		10%	90%
sensor			NN	16%	84%
sensor	motion	face detect		>99%	<1%
sensor	motion		NN	>99%	<1%
sensor		face detect	NN	>99%	<1%
sensor	motion	face detect	NN	>99%	<1%



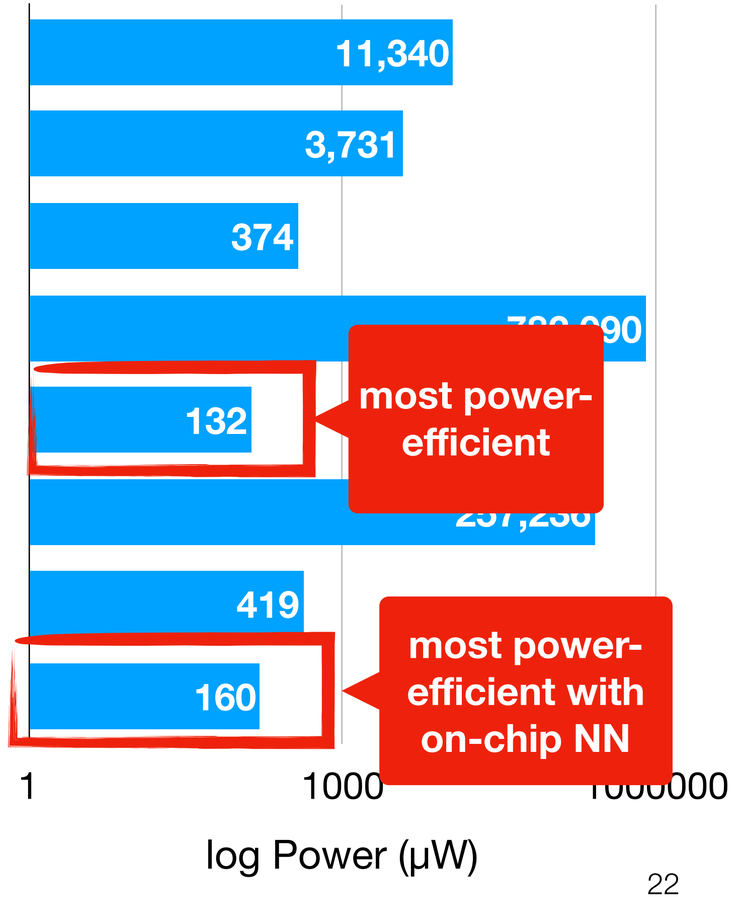
Which pipeline achieves the lowest power consumption?

platform configuration				(ratios) compute	transfer
sensor				<1%	>99%
sensor	motion			<1%	>99%
sensor		face detect		10%	90%
sensor			NN	16%	84%
sensor	motion	face detect		>99%	<1%
sensor	motion		NN	>99%	<1%
sensor		face detect	NN	>99%	<1%
sensor	motion	face detect	NN	>99%	<1%

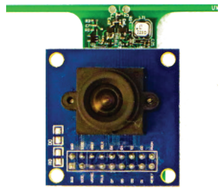


Which pipeline achieves the lowest power consumption?

platform configuration				(ratios) compute	transfer
sensor				<1%	>99%
sensor	motion			<1%	>99%
sensor		face detect		10%	90%
sensor			NN	16%	84%
sensor	motion	face detect		>99%	<1%
sensor	motion		NN	>99%	<1%
sensor		face detect	NN	>99%	<1%
sensor	motion	face detect	NN	>99%	<1%



In-camera processing for face authentication



motion detection

face detection

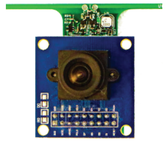
neural network

In isolation, even well-designed hardware can show sub-optimal performance

Optional blocks can improve the overall cost, if they balance compute and communication better than the original design

Challenges for modern camera systems

Low-power: face authentication for energy-harvesting cameras with ASIC design

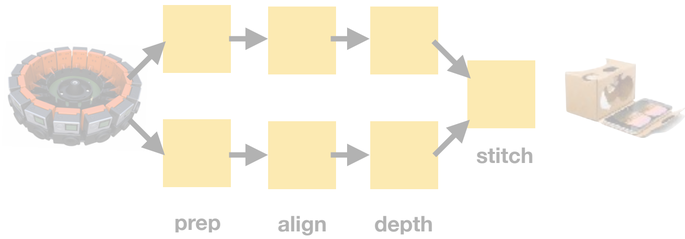


motion detection

face detection

neural network

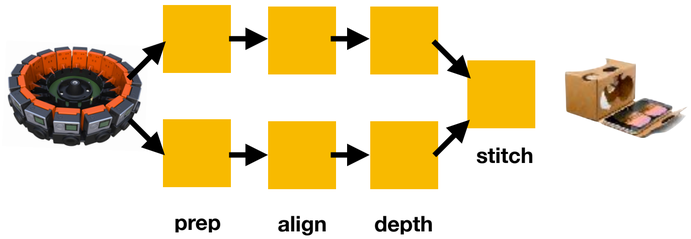
Low latency: real-time virtual reality for multi-camera rigs with FPGA acceleration



Challenges for modern camera systems

Low-power: face authentication for energy-harvesting cameras with ASIC design

Low latency: real-time virtual reality for multi-camera rigs with FPGA acceleration



Producing real-time VR video from a camera rig

16 GoPro cameras
4K-30 fps
3.6 GB/s raw video

Goal:
30 fps
3D-360 stereo video
1.8 GB/s output

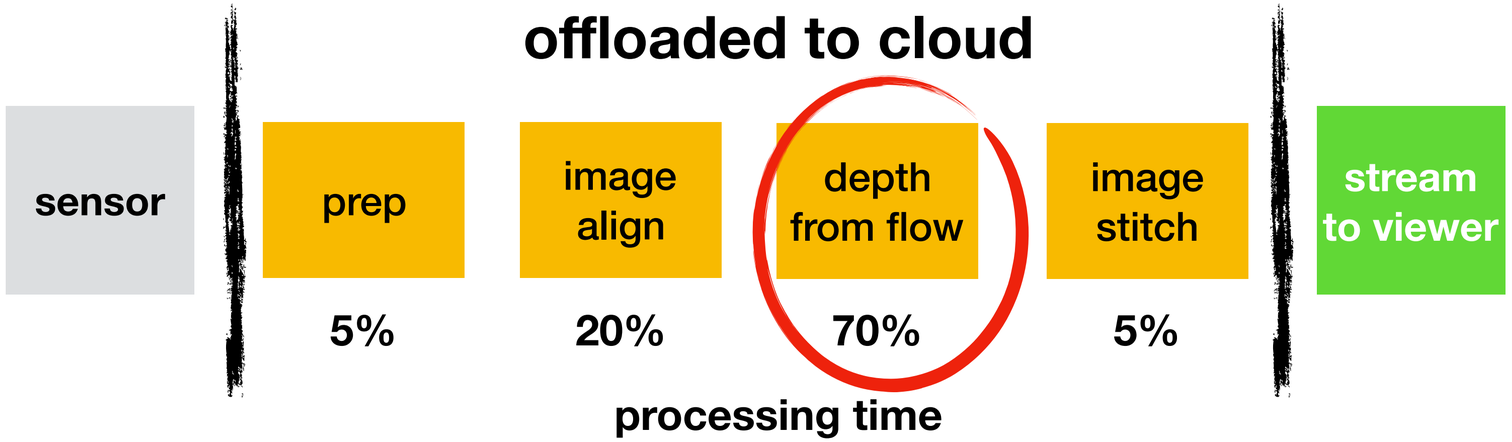
Producing real-time VR video from a camera rig

16 GoPro cameras
4K-30 fps
3.6 GB/s raw video

cloud processing prevents real-time video

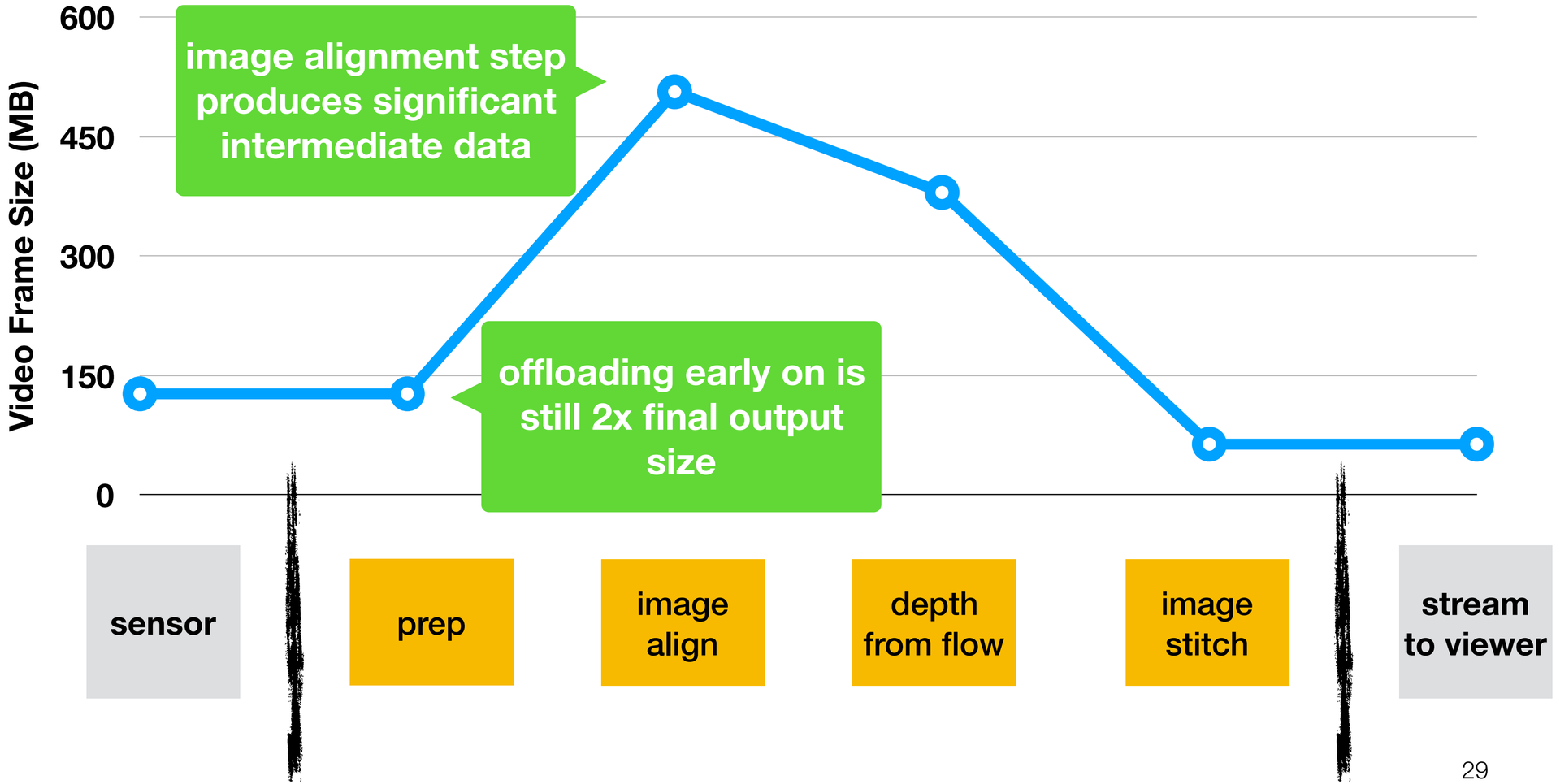
Goal:
30 fps
3D-360 stereo video
1.8 GB/s output

VR pipeline is usually offloaded to perform heavy computation



need to accelerate “depth from flow” to achieve high performance

Offloading before the costly step doesn't avoid compute-communication tradeoffs



Evaluation

Which pipeline achieves the highest frame rate?

Designed a simple parallel accelerator for Xilinx Zynq SoC, simulated for Virtex UltraScale+

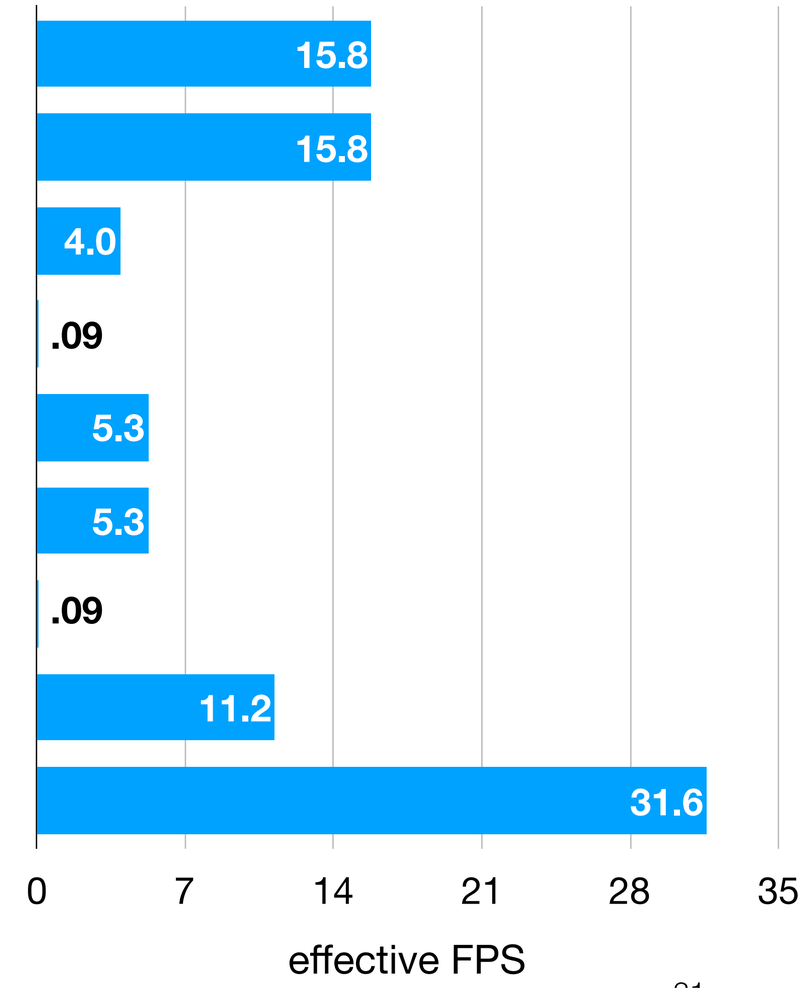
Evaluated against CPU and GPU implementations in Halide

Assumed 2GB/s network link for communication

implementation details in paper

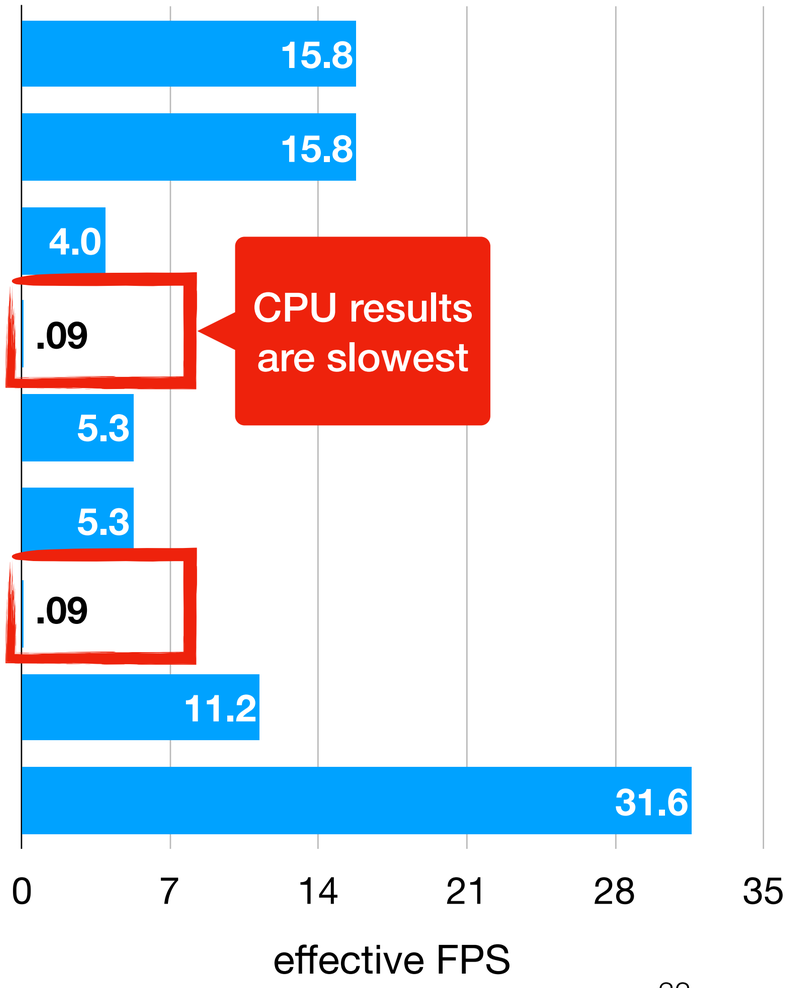
Which pipeline achieves the highest frame rate?

pipeline configuration					(FPS) compute	transfer
sensor						
sensor	prep					
sensor	prep	align				
sensor	prep	align	depth (CPU)			
sensor	prep	align	depth (GPU)			
sensor	prep	align	depth (FPGA)			
sensor	prep	align	depth (CPU)	stitch		
sensor	prep	align	depth (GPU)	stitch		
sensor	prep	align	depth (FPGA)	stitch		



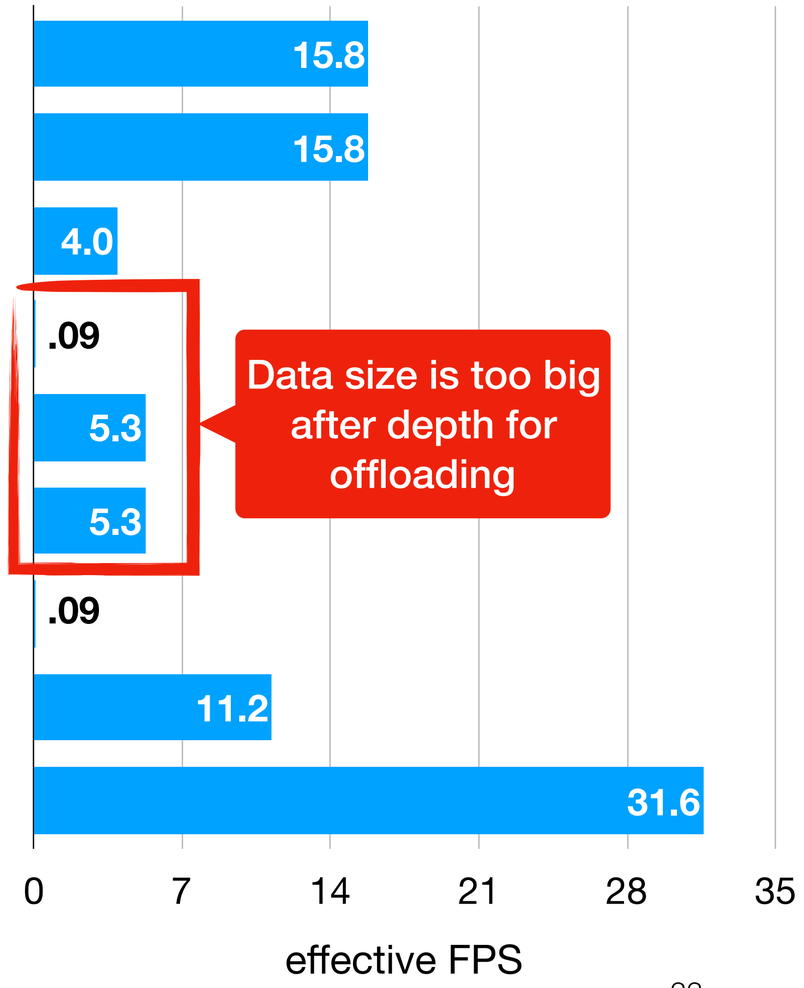
Which pipeline achieves the highest frame rate?

pipeline configuration					(FPS) compute	transfer
sensor					100	15.8
sensor	prep				100	15.8
sensor	prep	align			100	3.95
sensor	prep	align	depth (CPU)		0.09	5.27
sensor	prep	align	depth (GPU)		11.2	5.27
sensor	prep	align	depth (FPGA)		174	5.27
sensor	prep	align	depth (CPU)	stitch	0.09	31.6
sensor	prep	align	depth (GPU)	stitch	11.2	31.6
sensor	prep	align	depth (FPGA)	stitch	174	31.6



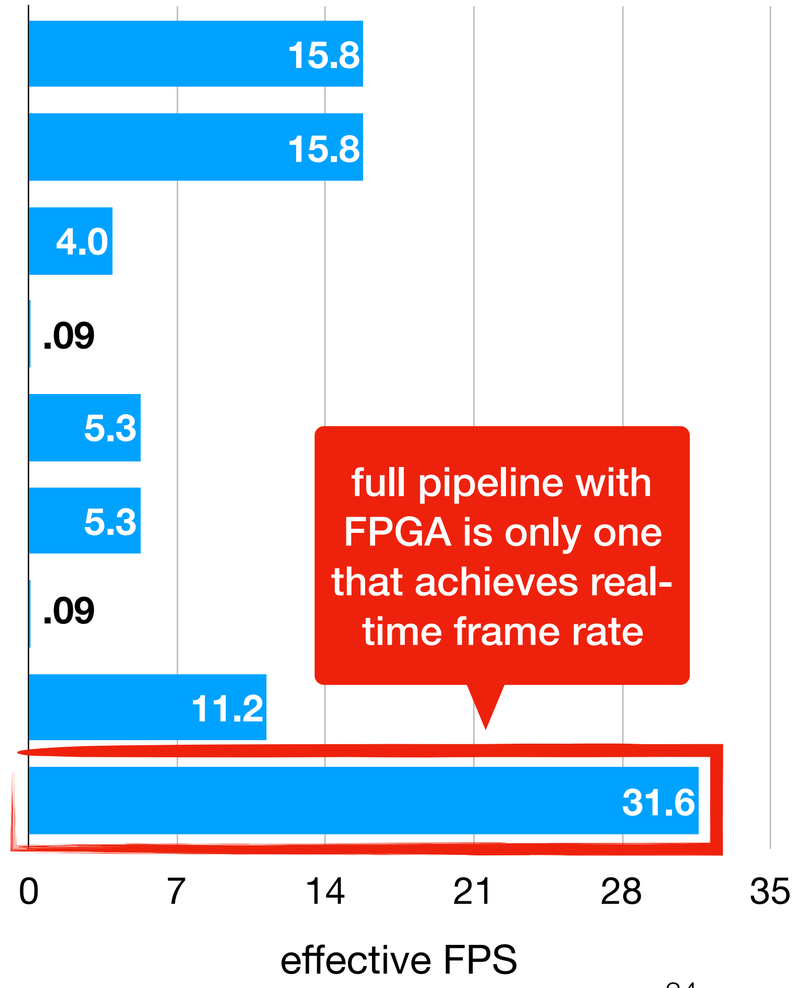
Which pipeline achieves the highest frame rate?

pipeline configuration					(FPS) compute	transfer
sensor					100	15.8
sensor	prep				100	15.8
sensor	prep	align			100	3.95
sensor	prep	align	depth (CPU)		0.09	5.27
sensor	prep	align	depth (GPU)		11.2	5.27
sensor	prep	align	depth (FPGA)		174	5.27
sensor	prep	align	depth (CPU)	stitch	0.09	31.6
sensor	prep	align	depth (GPU)	stitch	11.2	31.6
sensor	prep	align	depth (FPGA)	stitch	174	31.6

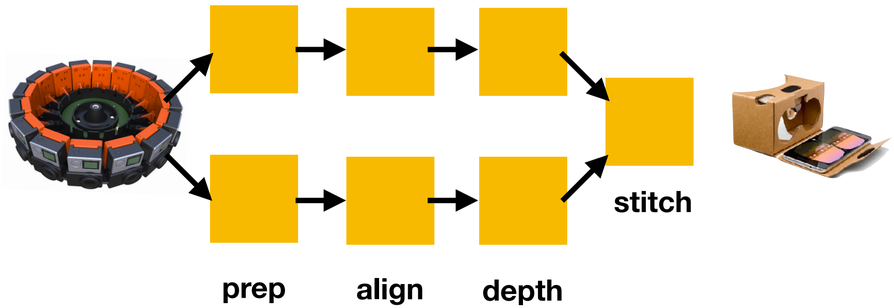


Which pipeline achieves the highest frame rate?

pipeline configuration					(FPS) compute	transfer
sensor					100	15.8
sensor	prep				100	15.8
sensor	prep	align			100	3.95
sensor	prep	align	depth (CPU)		0.09	5.27
sensor	prep	align	depth (GPU)		11.2	5.27
sensor	prep	align	depth (FPGA)		174	5.27
sensor	prep	align	depth (CPU)	stitch	0.09	31.6
sensor	prep	align	depth (GPU)	stitch	11.2	31.6
sensor	prep	align	depth (FPGA)	stitch	174	31.6



In-camera processing for real-time VR video



Computation and communication together highlight benefits not seen when considered separately

For VR video, in-camera processing pipelines enable applications that could not even be achieved via cloud offload

In-camera processing pipelines help characterize camera systems

In-camera pipelines evaluate **computation-communication trade-offs**

Use **hardware-software co-design** to balance constraints and optimize designs

Achieve optimal performance by **considering bottlenecks in context of full system**

Thank you!