SYNOPSYS®
Silicon to Software™

It Takes Two - Balancing Data Movement and Compute in a Radar Application for Maximum Performance on a Vector DSP

Pieter van der Wolf, Principal Product Architect

Synopsys ARC® Processor Summit 2022

Agenda

- Introduction
  - Sensor fusion applications and processing requirements
- Data movement and data processing on a vector DSP
  - Problem definition illustrated with a Radar case
- Techniques to balance data movement and compute
  - To achieve maximum performance on a vector DSP
- Conclusion

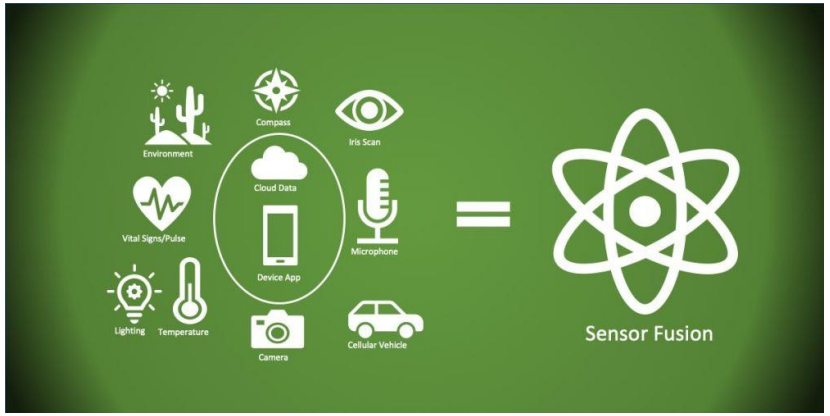
Sensor Fusion

Key Trends

- Sensor fusion combines data from multiple sensors to obtain more complete and accurate results
  - Use of multiple sensing devices enables context awareness
  - Enabling new applications and new user experiences

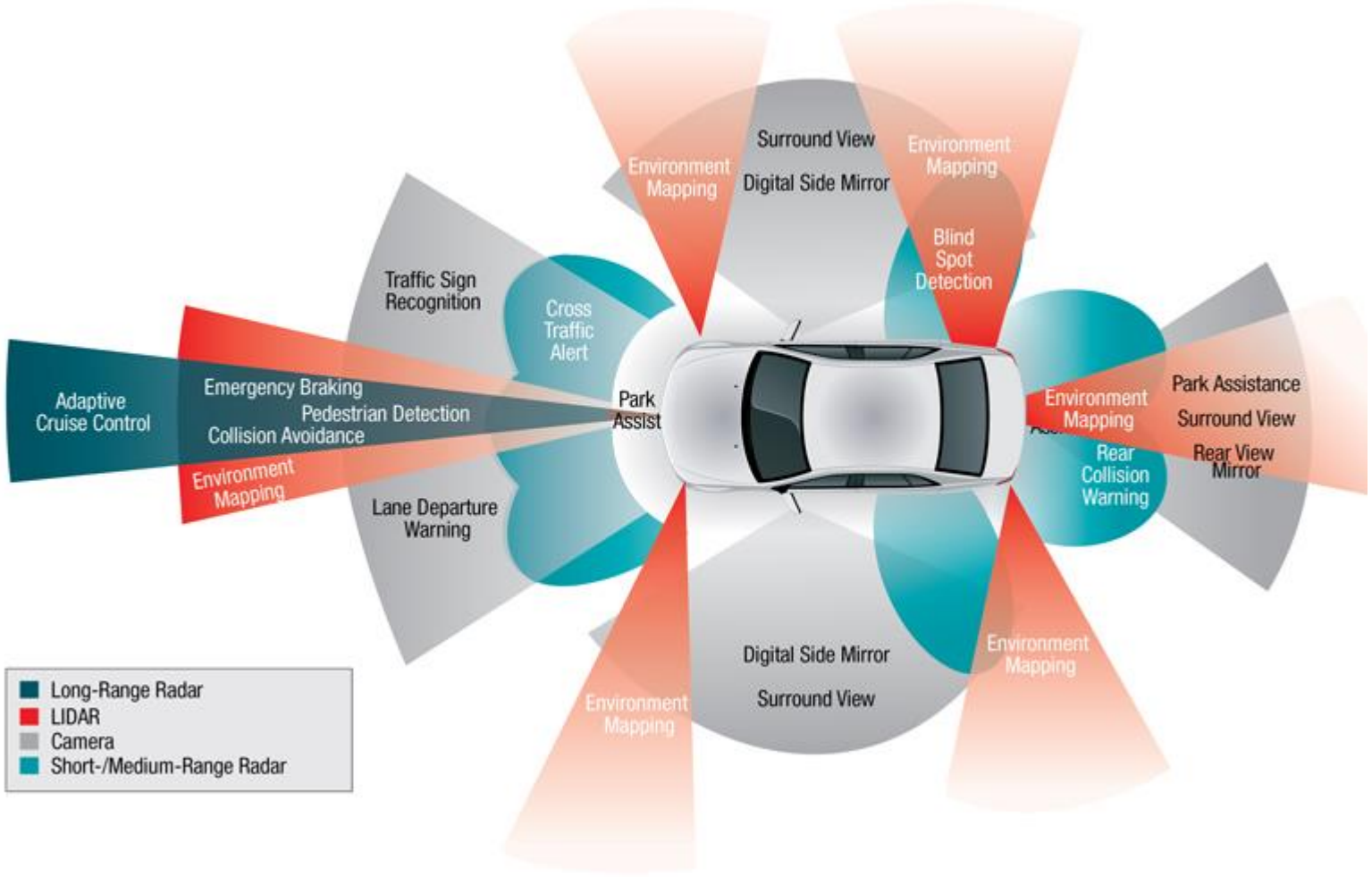
- Use of sensors fueled by miniaturization (MEMS) and lower cost of sensors
  - Camera, microphone, accelerometer, gyroscope, magnetometer
  - Sensors for RADAR and LiDAR

- Many new sensor fusion applications in different markets
  - Smart mobile devices
  - Automotive systems: autonomous driving, driver monitoring, ..
  - Smart home devices: smart appliances, home control & security, ..
  - Health: wearables, fitness devices, ..
  - Industrial control and robotics



(Image: SAR Insight and Consulting)

Advanced Driver Assistance System (ADAS)



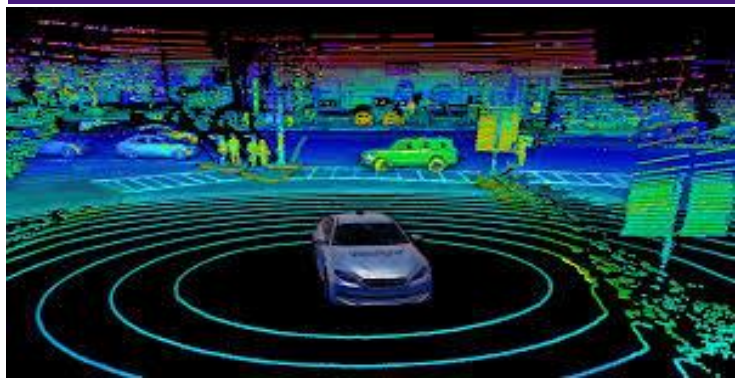
- Large number of subsystems
- Using different technologies
- Performing different functions
- Performance up to 100s TOPs
- Low component cost is key
- Low power consumption is key

Automotive ADAS Heavily Reliant on Sensor Fusion

RADAR

- Detect objects based on reflected microwaves
- Robust operation in different light and weather conditions
- Key component of Level 3+ and autonomous vehicles

LiDAR



- Measures time-of-flight of reflected (laser) light
- Produces 3D point cloud
- Wide field of view with good angular resolution
- Some sensitivity to weather conditions (e.g. heavy rain)

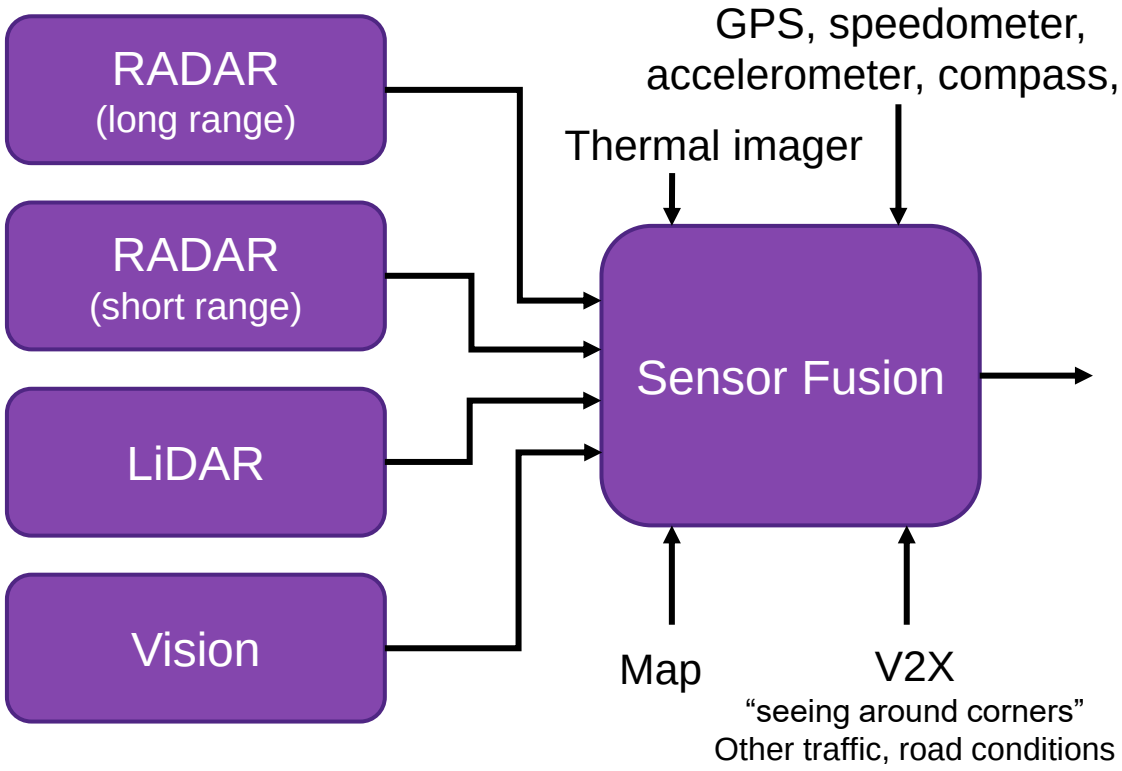
Vision

- Analysis of camera data
- Fast and accurate object classification using AI/ML
- Good angular resolution
- Sensitive to light and weather conditions

Different technologies have complementary strengths

RADAR, LiDAR and Vision Have Complementary Strengths

- No single type of sensor is perfect
- Sensor fusion combines data from multiple sensors to obtain more complete and accurate results

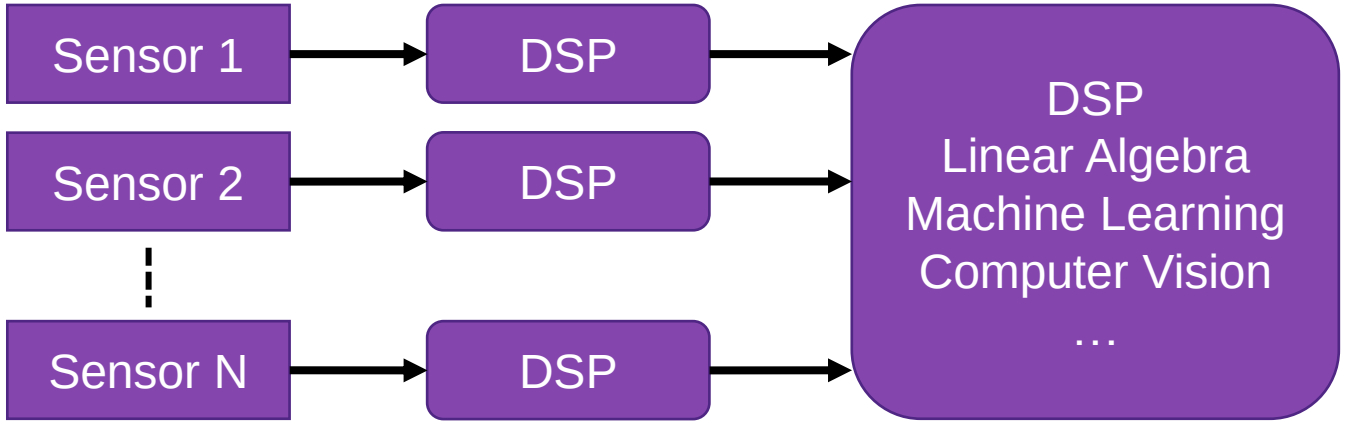


Sensor fusion tasks

- Localization
  - “Where am I?”
- Mapping
  - “What’s around me?”
  - Model of environment with objects and their properties
- Path planning
  - “What to do next?”

Sensor Fusion Requires Heterogenous Processing

Wide Range of Workloads, Depending on Sensor, and Role in the Processing Chain



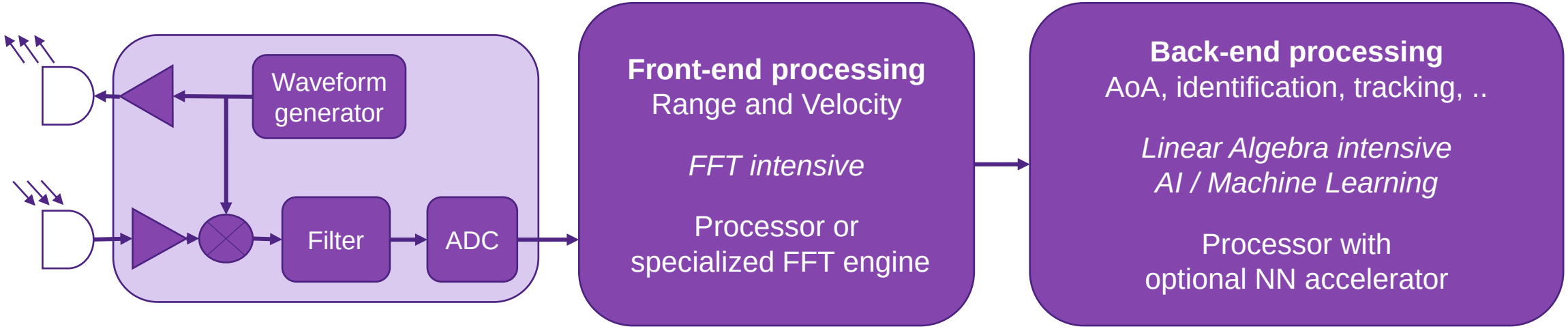
- Front-end: DSP for processing raw sensor data
  - Voice: Extract spectral features (filtering, transforms)
  - Camera: Image signal processing, ..
  - Radar: Range & velocity FFTs, CFAR, ..
- Back-end: processing requirements vary based on application
  - Tasks include object detection, recognition, tracking, prediction, ..
  - Algorithms and data types largely depend on application

Vector DSP requirements

- Versatile
  - Execute mix of workloads
  - Offer rich ISA and feature set
- Scalable
  - Range of compute requirements
  - Family of vector DSPs
  - With good software portability
- High performance
  - Demanding workloads
  - Lots of data to process
  - Lots of data to move

Generic Architecture for FMCW Radar

Frequency-Modulated Continuous-Wave



Example RADAR case

- 4 channels, 50 frames/s, 256 chirps/frame, 512 samples/chirp
- Range FFTs: 4 x 50 x 256 = 51,200 complex FFTs of length 512 per second
- Velocity FFTs: 4 x 50 x 512 = 102,400 complex FFTs of length 256 per second

} >52M FFT points per second

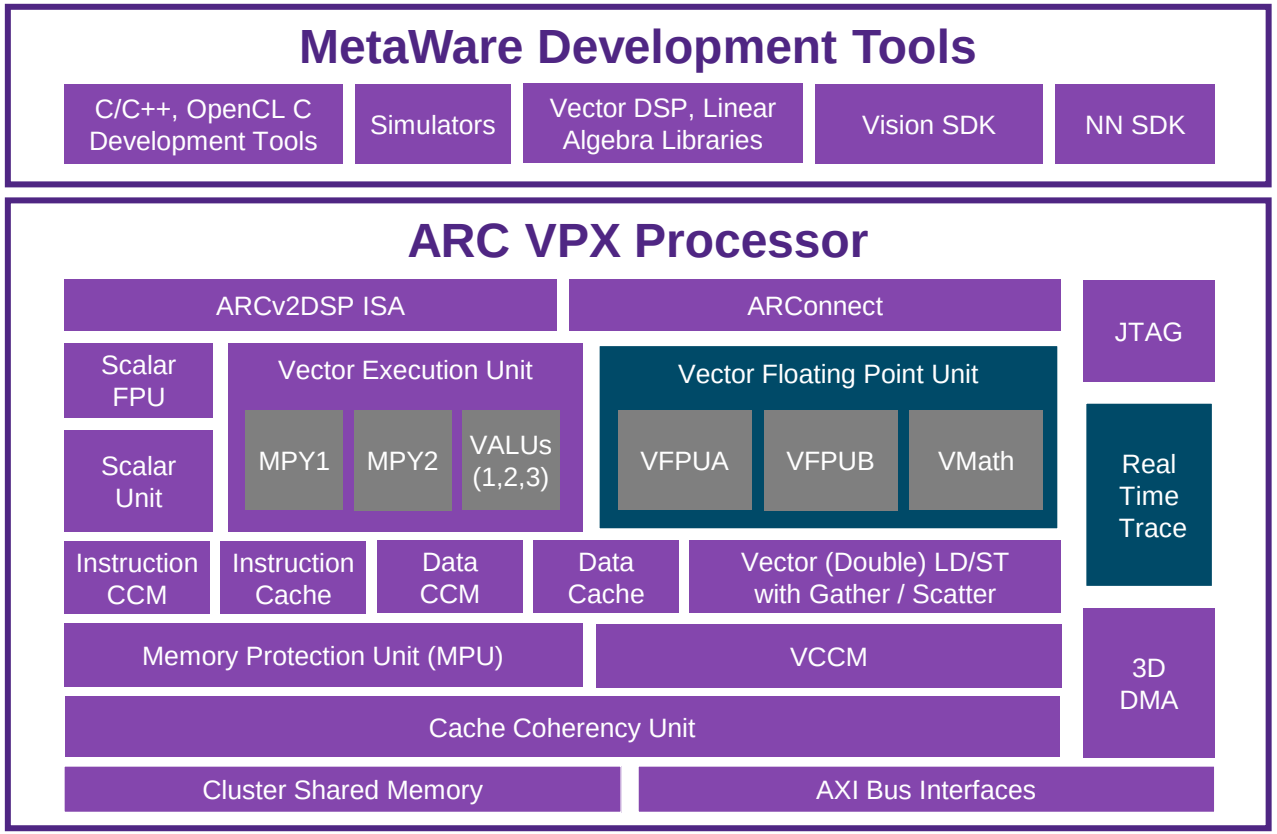
- Size of radar cube with 32b+32b complex data is 4.2MB
- Bandwidth requirement for a single data stream is 50 x 4.2 = 210MB/s

Agenda

- Introduction
  - Sensor fusion applications and processing requirements
- Data movement and data processing on a vector DSP
  - Problem definition illustrated with a Radar case
- Techniques to balance data movement and compute
  - To achieve maximum performance on a vector DSP
- Conclusion

Synopsys ARC VPX DSP IP Family

Next-Generation DSP Architecture for a Data Centric World



- Advanced **SIMD / VLIW DSP IP** addresses broad range of DSP and AI/ML workloads including RADAR/LiDAR, vision and sensor fusion
  - Vector lengths of **128-bit, 256-bit and 512-bit** enable users to select optimum PPA for required workload
  - **Scalable** and **configurable** to tune performance, area and power for specific SoC requirements
- **Vector-length agnostic** programming model allows for smooth migration of S/W among members of VPX family
  - **Fully C-programmable** with optimizing compiler
- Special features for precision results on latest algorithms
  - **Ultra high-performance floating-point** processing
  - **Hardware acceleration** for Math operations, FFT, ..
  - Architecture, tools and libraries optimized for **efficient ML/AI,** with option to combine with NPX NN accelerator
- ISO 26262 **ASIL B and D** compliant versions for safety

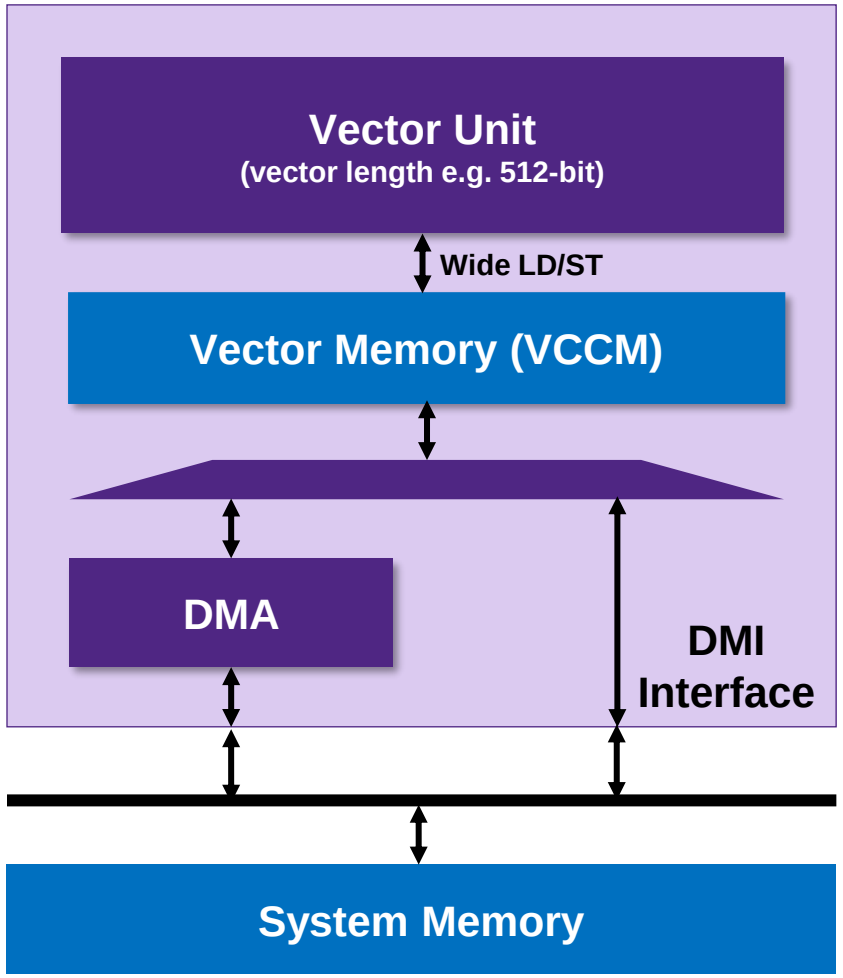
Licensable Option

VPX5	512-bit vector SIMD/VLIW DSP
VPX5x2	Dual Core 512-bit vector SIMD/VLIW DSP
VPX5x4	Quad Core 512-bit vector SIMD/VLIW DSP

VPX3	256-bit vector SIMD/VLIW DSP
VPX3x2	Dual Core 256-bit vector SIMD/VLIW DSP
VPX2	128-bit vector SIMD/VLIW DSP
VPX2x2	Dual Core 128-bit vector SIMD/VLIW DSP

Moving Data In/Out a Vector DSP

Via DMA or DMI Interface

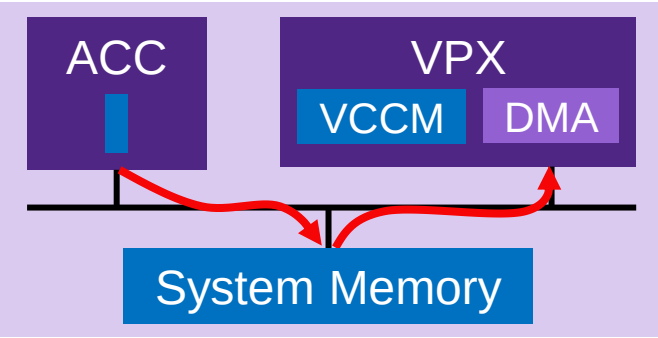


- Vector DSPs have local memory that can be accessed efficiently
  - To support efficient processing of data by wide vector unit
  - Vector unit performs wide LD/ST operations on VCCM to access data
  - VCCM is limited in size so data must be moved from L2/L3 memory
- External data moved to/from vector memory using DMA or DMI
  - DMA has initiator interface(s) on system NoC
    - For example, to move data from/to System Memory
  - DMI offers target interface(s)
    - For example, to allow external accelerators to move data to/from VCCM
- Benchmark data often assumes data is present in VCCM
  - And the overhead of moving data in/out VCCM is not accounted for
- Challenge is to minimize overheads of DMA/DMI data transfers

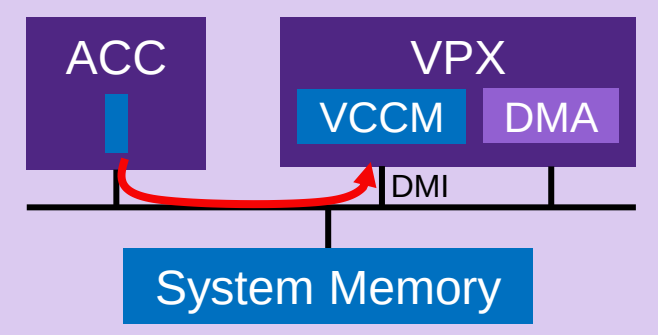
Interfacing of Hardware Accelerators to VPX

Options for flexible and efficient coupling of hardware accelerators

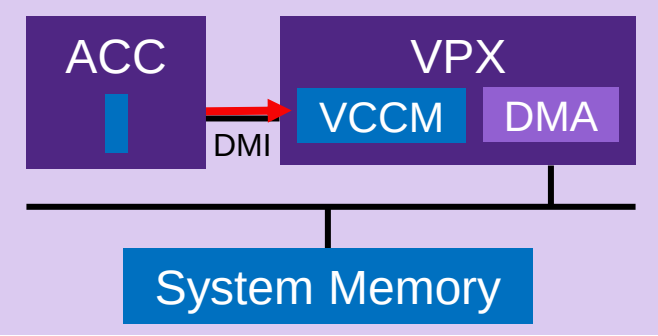
1. Transferring data via system memory
   - Flexible buffer sizing to handle different use cases
   - Relaxed task scheduling constraints
   - Moving data over system interconnect twice, VPX DMA is initiator



2. Transferring data over system interconnect directly to/from VCCM
   - Buffer sizes constrained by VCCM size
   - Buffer handling to be aligned with processing on VPX
   - Moving data over system interconnect once, using target interface of VPX

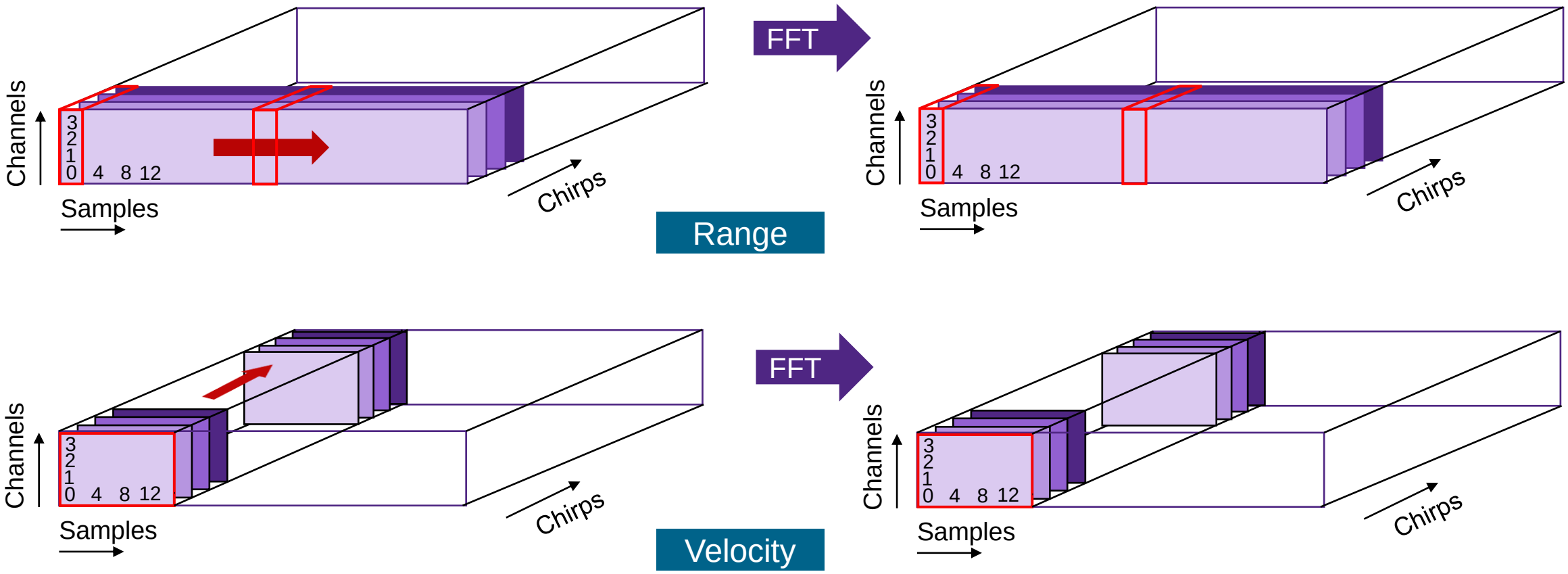


3. Direct transfer of data to/from VCCM
   - Buffer sizes constrained by VCCM size
   - Buffer handling to be aligned with processing on VPX
   - No data transfer over system interconnect, using target interface of VPX



FFT Processing Example for Radar

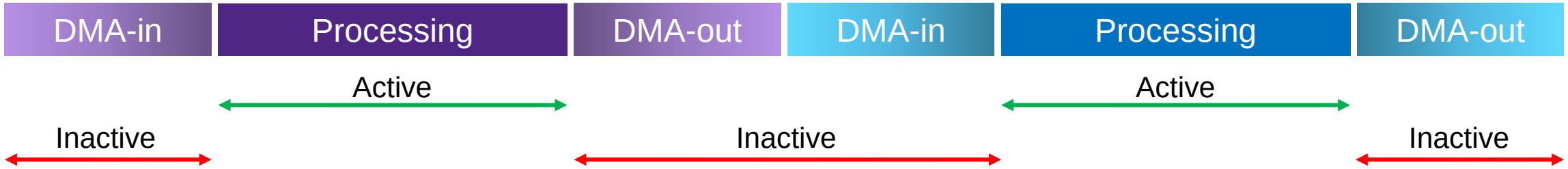
Organization of input & output data in vector memory for batch of 16 FFTs



Demands advanced DMA capabilities and a vector unit with flexible LD/ST access patterns

Balancing Data Movement and Compute

Problem definition



- Performing DMA and processing steps sequentially gives low utilization of vector DSP
  - Vector DSP is idle during DMA transfers
  - MHz budget of vector DSP cannot be fully utilized
  - Only two buffers required, one input buffer and one output buffer

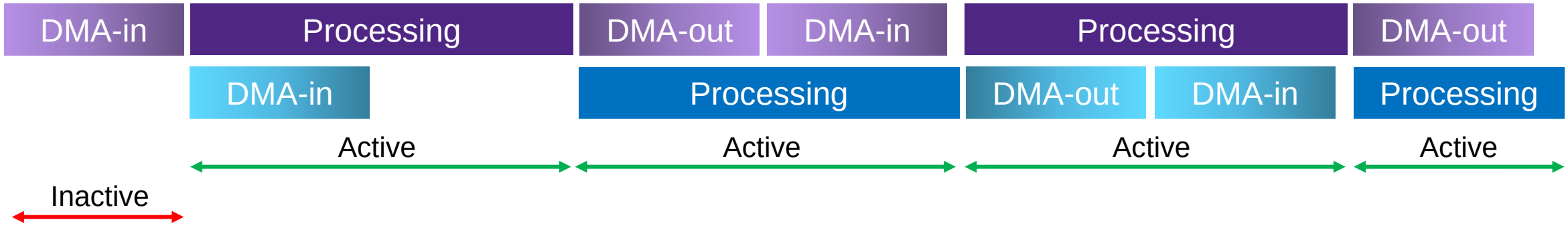
- Objectives
  - Perform DMA transfers in parallel to processing in order to hide DMA latency
  - Properly balance data movement and compute so that full MHz budget of vector DSP can be utilized

Agenda

- Introduction
  - Sensor fusion applications and processing requirements
- Data movement and data processing on a vector DSP
  - Problem definition illustrated with a Radar case
- Techniques to balance data movement and compute
  - To achieve maximum performance on a vector DSP
- Conclusion

Piecemeal Processing with DMA Operating in the Background

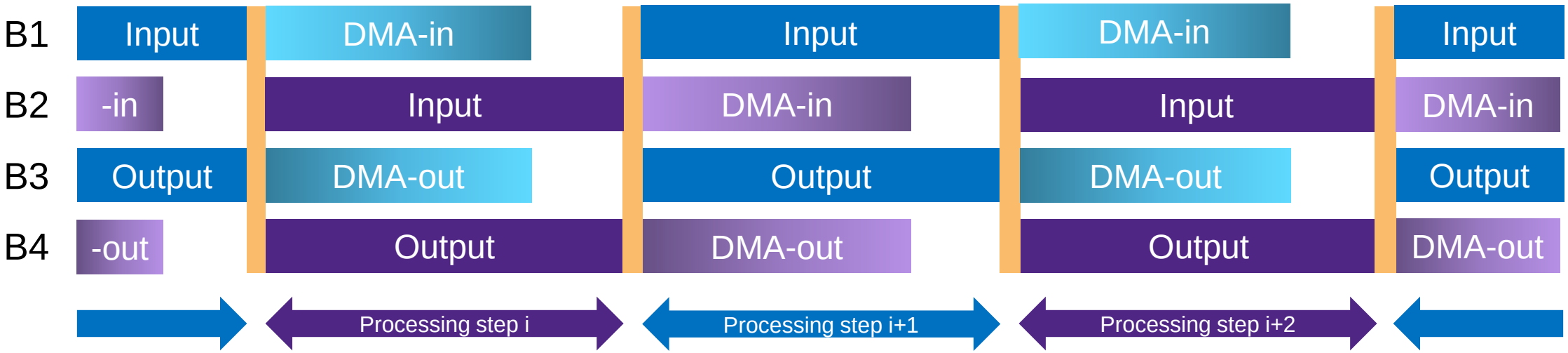
Hide DMA latency



- Data is processed in piecemeal fashion
  - Fits well with ‘streaming’ applications like Radar
- DMAs are performed in parallel to data processing in order to hide DMA latency
  - DMA to store output data of previous processing step
  - DMA to load input data of next processing step
- Enables high utilization of MHz budget of vector DSP
  - But latency of first input DMA cannot be hidden
- Parallel DMAs demand some form of double buffering

Example DMA Scheme for Hiding DMA Latency

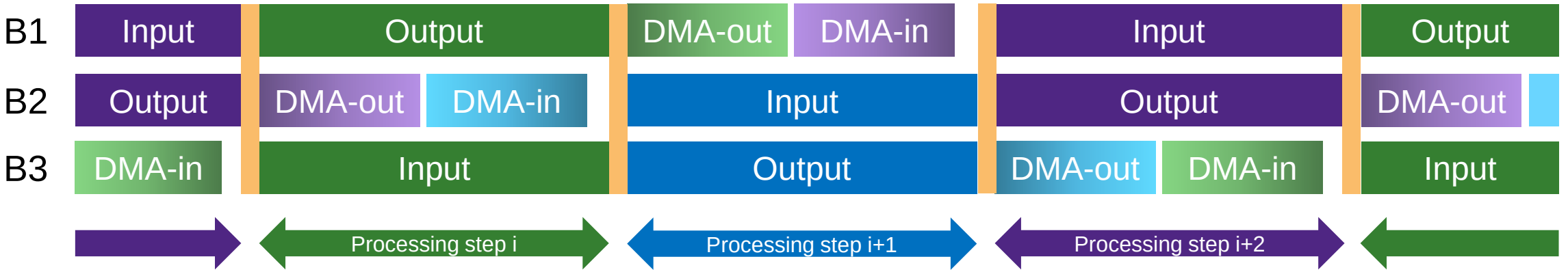
Double-buffering with four buffers in VCCM



- Double buffer scheme with four buffers in VCCM (B1, B2, B3, B4)
  - Two input buffers (B1, B2) and two output buffers (B3, B4)
- Upon each processing step
  - Two buffers are used for processing, one for input and one for output
  - DMA transfers data from/to other two buffers
- Software “rotates” buffer pointers when synchronizing with DMA and setting up transfers (orange bars)

Example DMA Scheme for Hiding DMA Latency

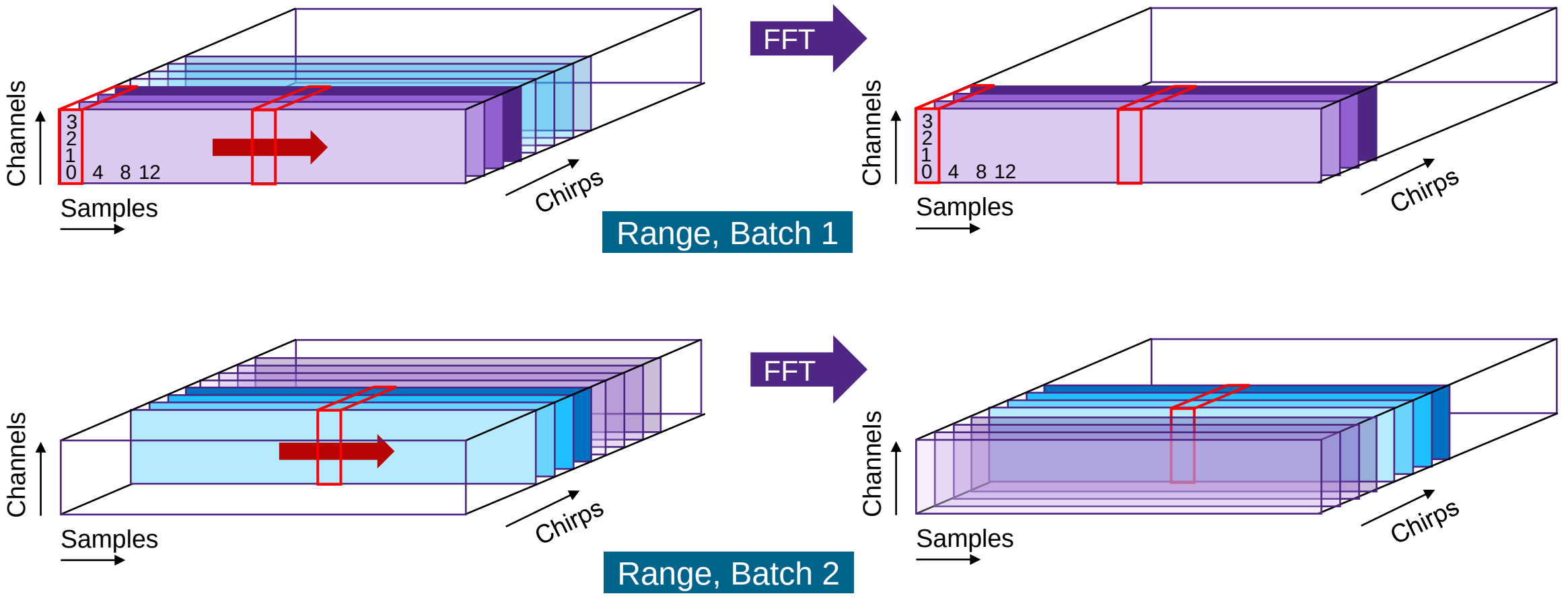
Double-buffering with three buffers in VCCM



- Double buffer scheme with three buffers in VCCM (B1, B2, B3)
- Upon each processing step
  - Two buffers are used for processing, one for input and one for output
  - DMA transfers data from/to third buffer
  - Input buffer of previous step is used as output buffer for current step
- Software “rotates” buffer pointers when synchronizing with DMA and setting up transfers (orange bars)
- Benefit of a 3-buffer scheme is that less VCCM buffer space is required than with a 4-buffer scheme

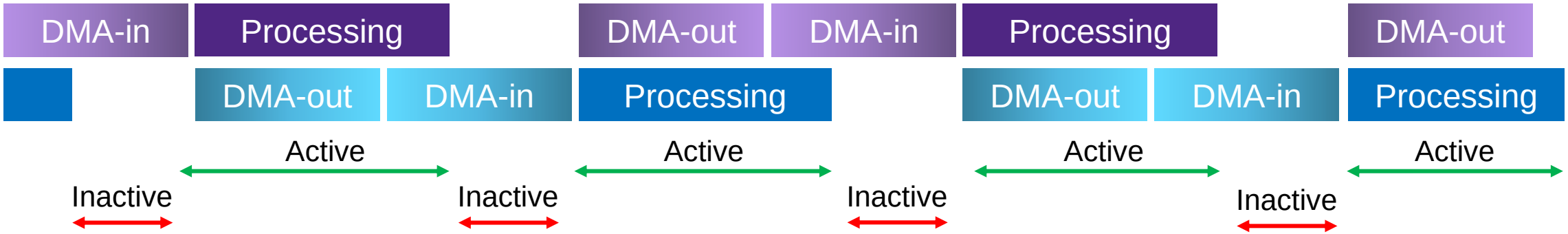
FFT Processing Example for Radar

Parallel processing and DMA transfers for batch of 16 FFTs



Balancing Data Movement and Compute

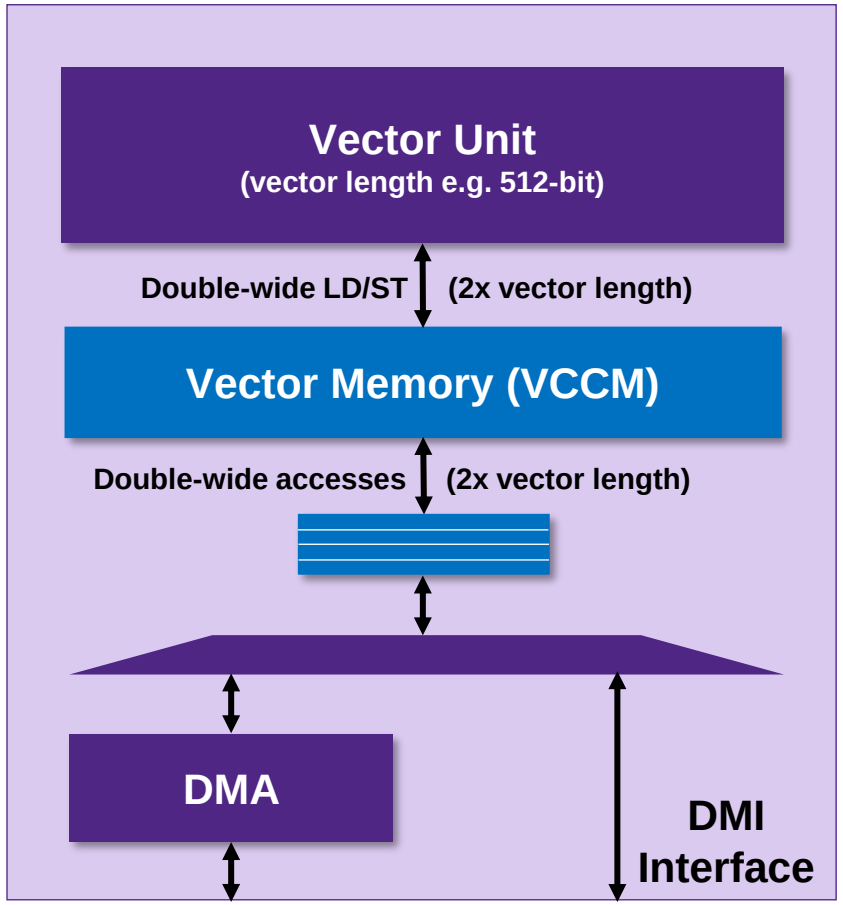
Avoiding DMA bottlenecks



- DMA may become bottleneck if data movement and compute are not properly balanced
  - MHz budget of vector DSP cannot be fully utilized
  - If processing step takes N cycles and DMA needs to move M bytes, then throughput > M/N bytes/cycle
- Avoid too fine-grain processing tasks
  - Moving data in/out for small processing tasks can give large overheads and increase power consumption
  - Fuse processing into bigger tasks where appropriate
- Allocate sufficient DMA bandwidth
  - Use wider memory interface(s) for DMA, allocate multiple DMA channels

Smart VCCM Arbitration to Avoid Processor Stalls

With high-bandwidth access to VCCM



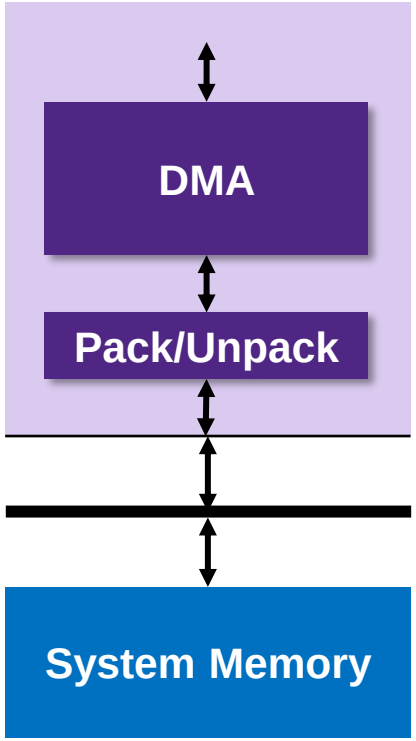
- Processor core and DMA/DMI compete for access to VCCM
  - Must minimize stalls for LD/ST of processor core
  - DMA/DMI needs a throughput guarantee for access to VCCM
- Unified VCCM allows flexible buffer allocation
  - Buffers can be anywhere in VCCM, no artificial boundaries
- Unified VCCM is arbitrated on a per cycle basis
  - Per cycle either the core or DMA/DMI gets access to full VCCM
  - Core takes priority as long as DMA/DMI is not under-served
- Allows full memory bandwidth to be utilized
  - Double-wide memory access every cycle
  - Access by either core or DMA/DMI, with programmable ratio
- Smart arbiter allows easy programming of bandwidth guarantees
  - Employing simple accounting scheme

High cycle efficiency for processing on core with predictable system performance

Packing / Unpacking DMA Data

To reduce memory footprint and memory bandwidth

- Many applications work on large data sets that must be stored in system memory
  - And must then be moved by the DMA to be processed on the vector DSP
  - For example, images, radar cubes, AI/ML data sets
- Data sets can often be stored with smaller bit-widths than required for processing
  - For example, 12 bit per pixel element for image data that is to be processed in 16 bit
  - Particularly relevant for SoCs without DDR, using on-chip SRAM
- DMA packing / unpacking allows for on-the-fly (de-)compression of data
  - Unpack data when moving from system memory into vector DSP
  - Pack data when moving from vector DSP to system memory
- Benefits
  - Reduces memory footprint in system memory for large data sets
  - Reduces memory bandwidth for moving data from/to system memory



ARC VPX: A Quantitative Radar Case

VPX provides ample performance and DMA bandwidth for Radar processing

- Example RADAR case
  - 4 channels, 50 frames/s, 256 chirps/frame, 512 samples/chirp

- FFTs
  - Range FFTs: 4 x 50 x 256 complex FFTs of length 512 per second
  - Velocity FFTs: 4 x 50 x 512 complex FFTs of length 256 per second
  - On VPX5 in single-precision floating-point (fp32) these FFTs take less than 40MHz
  - **VPX can efficiently handle the FFT processing requirements of the RADAR case**

- DMA
  - Size of radar cube in fp32 is 4.2MB
  - Bandwidth requirement for a single data stream is 210MB/s
  - VPX5 DMA can provide bandwidth of up to 256B/cycle, so available bandwidth at e.g. 800MHz is 205GB/s
  - **VPX has ample DMA bandwidth for the RADAR case**

Agenda

- Introduction
  - Sensor fusion applications and processing requirements
- Data movement and data processing on a vector DSP
  - Problem definition illustrated with a Radar case
- Techniques to balance data movement and compute
  - To achieve maximum performance on a vector DSP
- Conclusion

Conclusion

- Advanced sensor fusion applications demand versatile and scalable vector DSPs
  - The ARC VPX family of SIMD / VLIW IP addresses a broad range of workloads
- It takes two - data movement and compute need to be properly balanced
  - To achieve best performance on a vector DSP
- Various techniques must be employed to balance data movement and compute
  - Perform DMAs in parallel to data processing in order to hide DMA latency
  - Compute tasks must be sufficiently coarse-grain to hide DMA latency and reduce data movements
  - Allocate sufficient DMA bandwidth
  - Employ smart arbitration for vector memory to avoid stalls of the processing on the core
  - Use packing / unpacking to reduce memory footprint and memory bandwidth
- Employing these techniques, ARC VPX can efficiently perform Radar processing

SYNOPSYS®
Silicon to Software™

Thank You