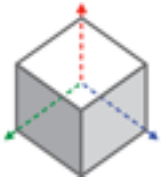
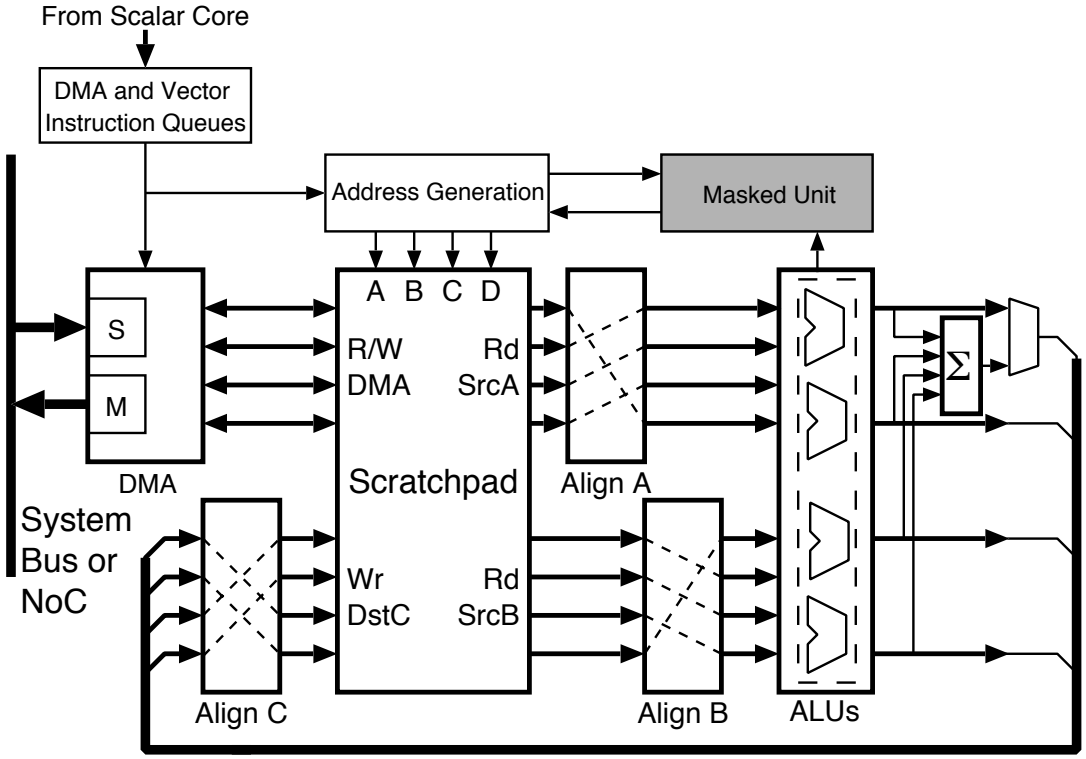
WAVEFRONT SKIPPING USING BRAMS FOR CONDITIONAL ALGORITHMS ON VECTOR PROCESSORS

Aaron Severance
Joe Edwards
Guy G.F. Lemieux



VectorBlox
embedded supercomputing

VectorBlox MXP
Soft Vector Processor (SVP)

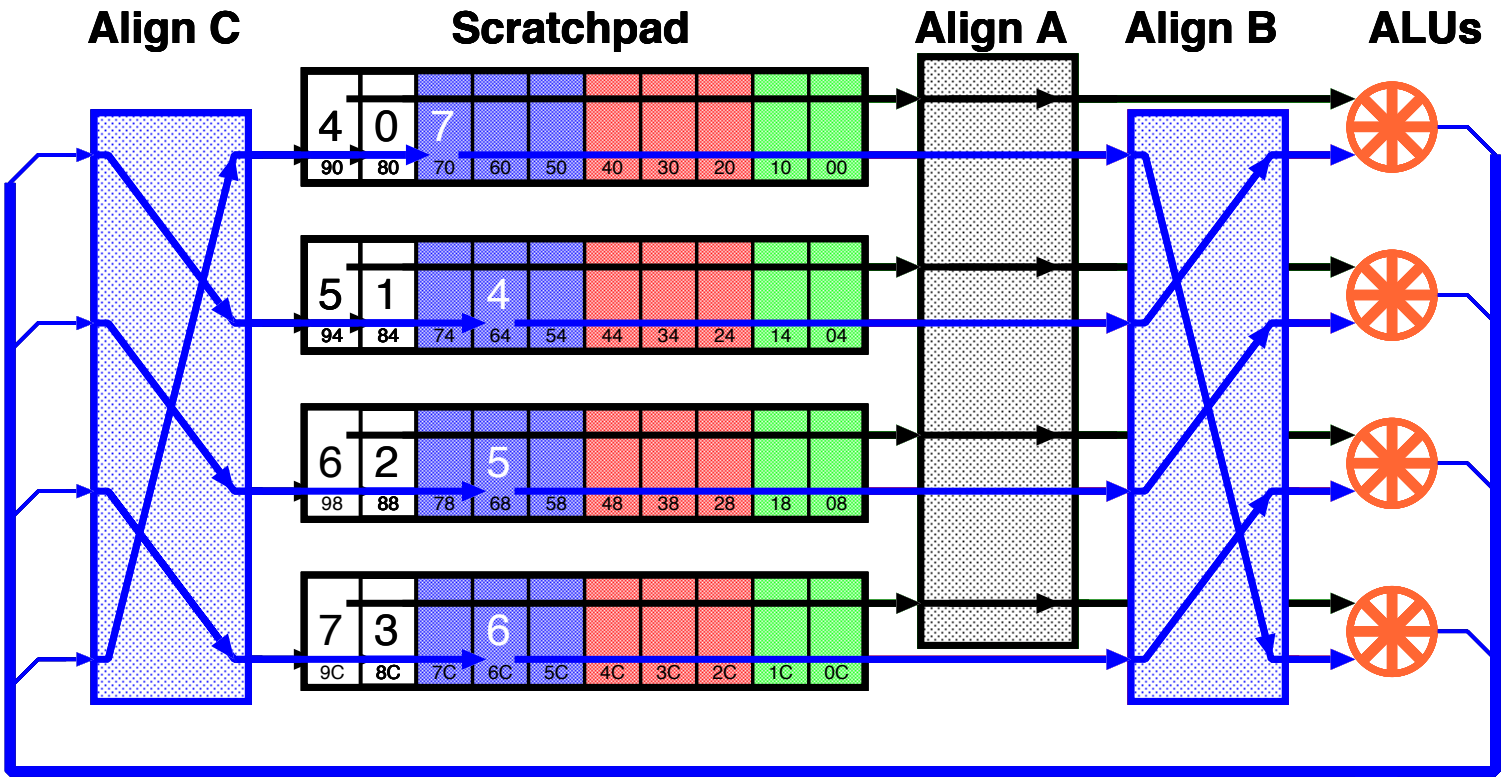


Software programmable

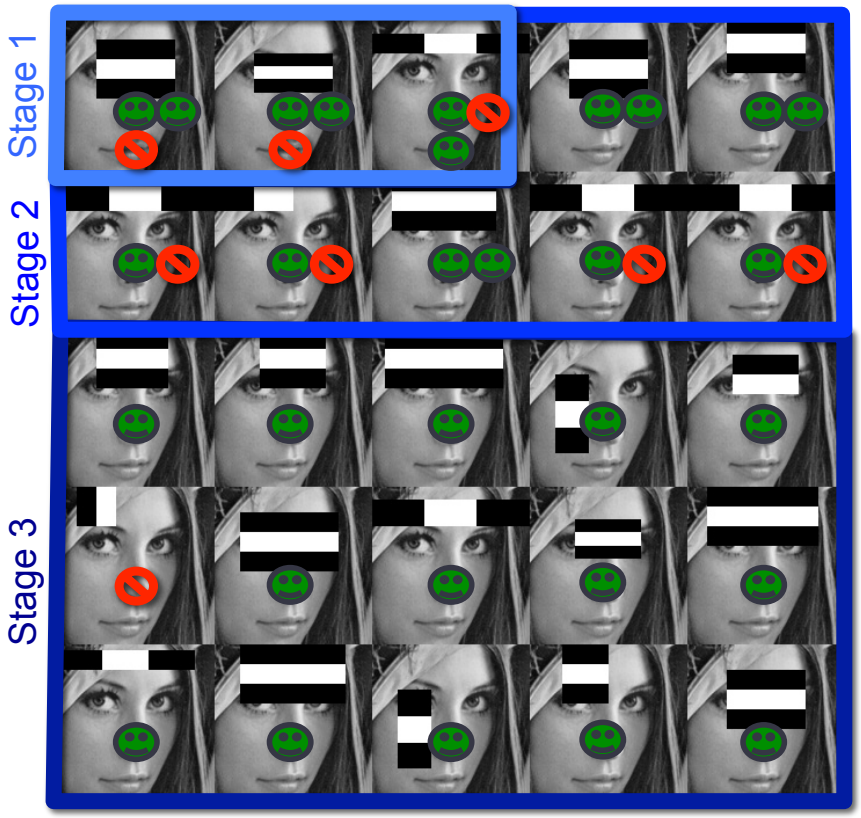
1 to 128 parallel ALUs
(4 shown)

Flat scratchpad memory
Vectors at any address,
have any length

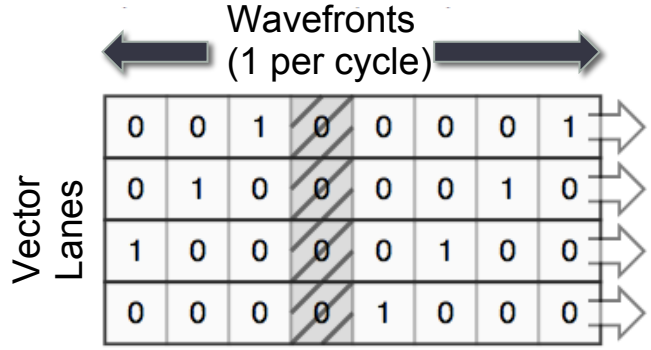
Vectors in Scratchpad Memory



Early Exit Algorithms



Divergent Control Flow on SVPs



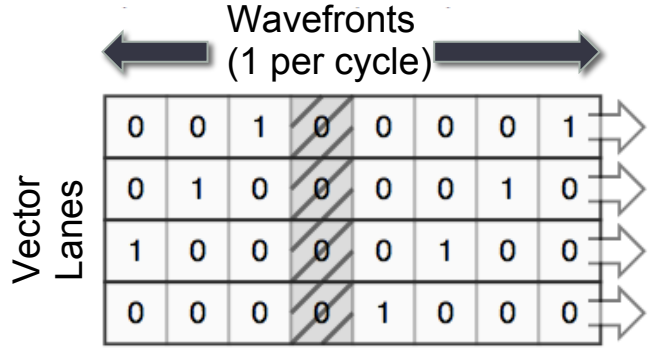
Mask Vector:
1's in corresponding vector are active
0's are disabled

- Predication/CMOV
  - Process entire vector
- Vector Compress
  - Load inputs, compress
  - Expand outputs, store

Vector Compress: 5x1 Sliding Window

- N*M compress operations (time)
- N*M temporary vectors (space)

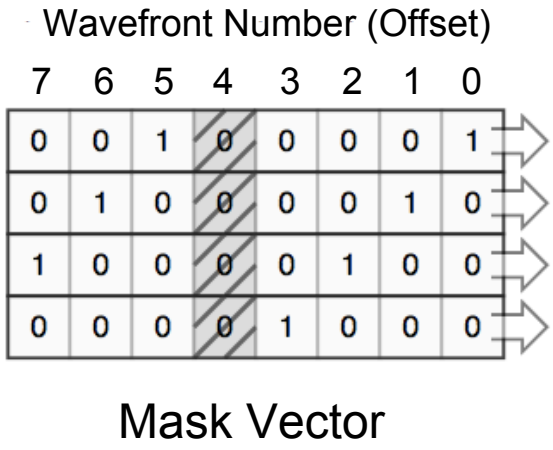
Divergent Control Flow on SVPs



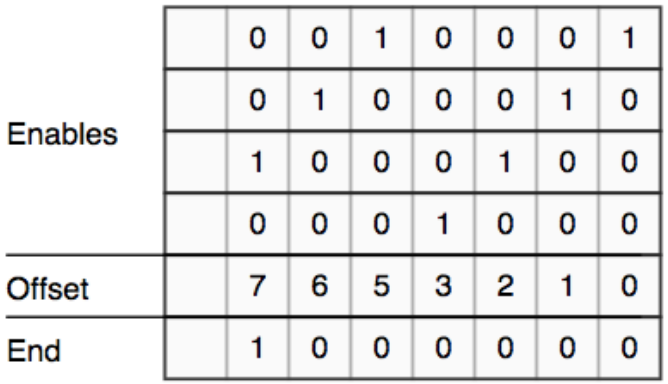
Mask Vector:
1's in corresponding vector are active
0's are disabled

- Predication/CMOV
  - Process entire vector
- Vector Compress
  - Load inputs, compress
  - Expand outputs, store
- Compress + Scatter/Gather
  - Compress addresses
  - Load/store indexed
- Density-time
  - Scan mask for next valid wavefront

Wavefront Skipping using BRAMs



Mask Vector



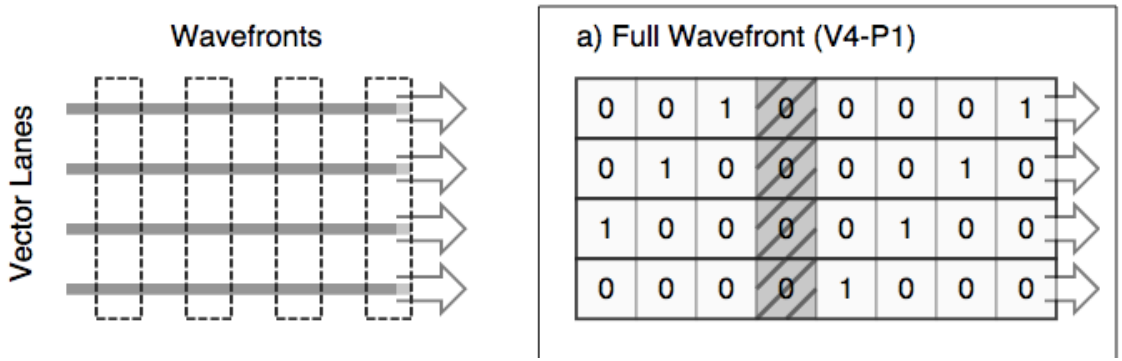
Mask BRAM

- mask_setup() comparison populates mask BRAM
  - Only wavefronts with active elements stored
- Masked execution adds offsets from mask BRAM to base

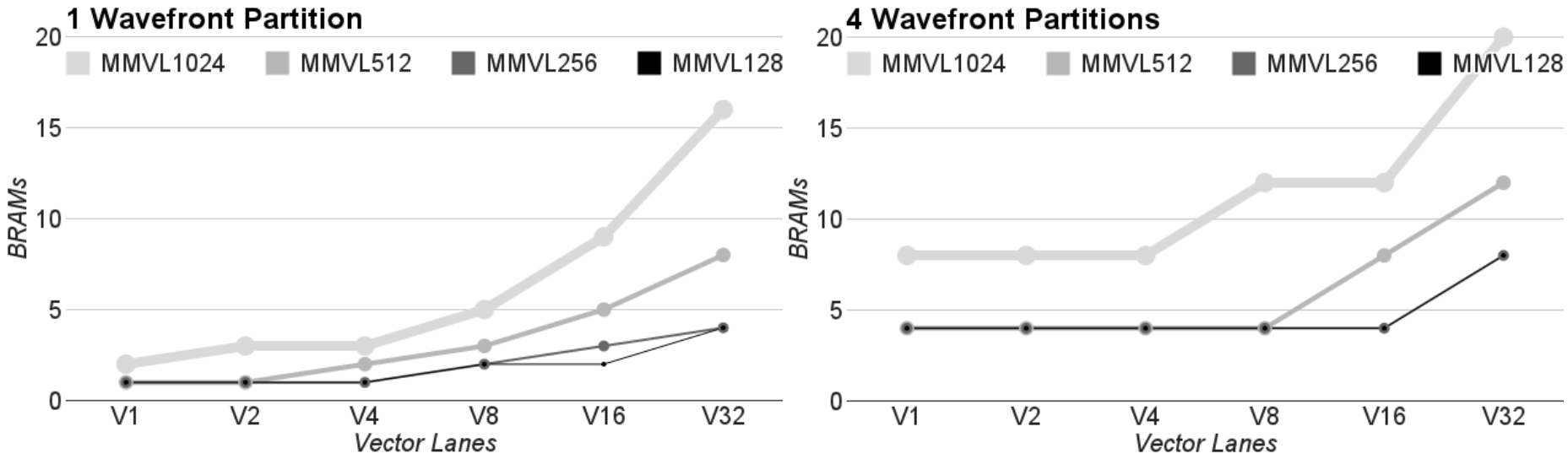
Code Example

```
#define STRIDE 5

//Toy functition to double every fifth element in the vector v_a
void double_every_fifth_element( int *v_a, int *v_temp int vector_length ){
  int i;

  //Initialize every fifth element of v_temp to zero
  for( i = 0; i < vector_length; i++ ) {
    v_temp[i] = i % STRIDE;
  }

  //Set the vector length which will be processed
  vbx_set_vl( vector_length );

  //Set mask for every element equal to zero
  vbx_setup_mask( CMOV_Z, v_temp );

  //Perform the wavefront skipping operation: multiply all non-masked off
  //elements of v_a by 2 and store the result back in v_a
  vbx_masked( SVW, MUL, v_a, 2, v_a );
}
```

Partial Wavefront Skipping



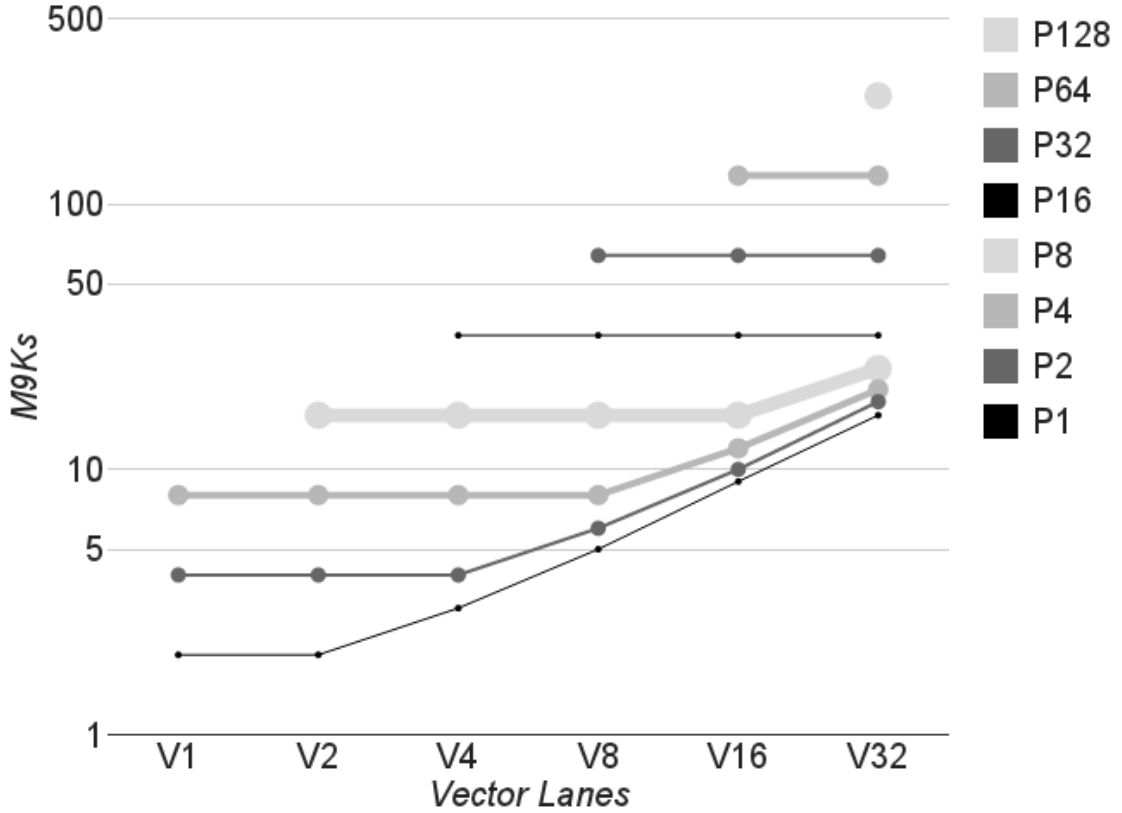
BRAM Usage for Varying MMVL



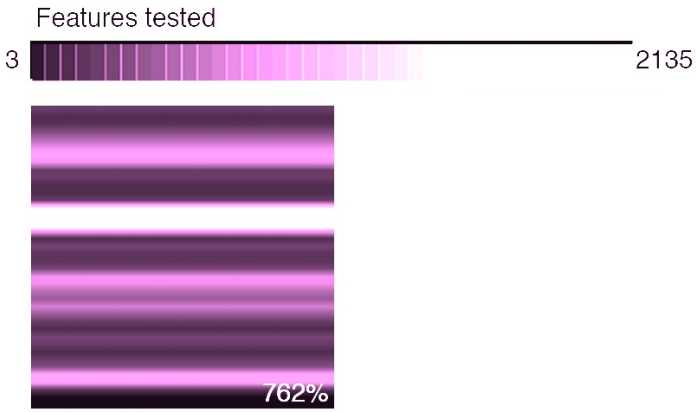
Increasing Maximum Masked Vector Length (MMVL):

- More wavefronts, more speedup with sparse masks
- Deeper BRAMs needed

BRAM Usage for Multiple Partitions

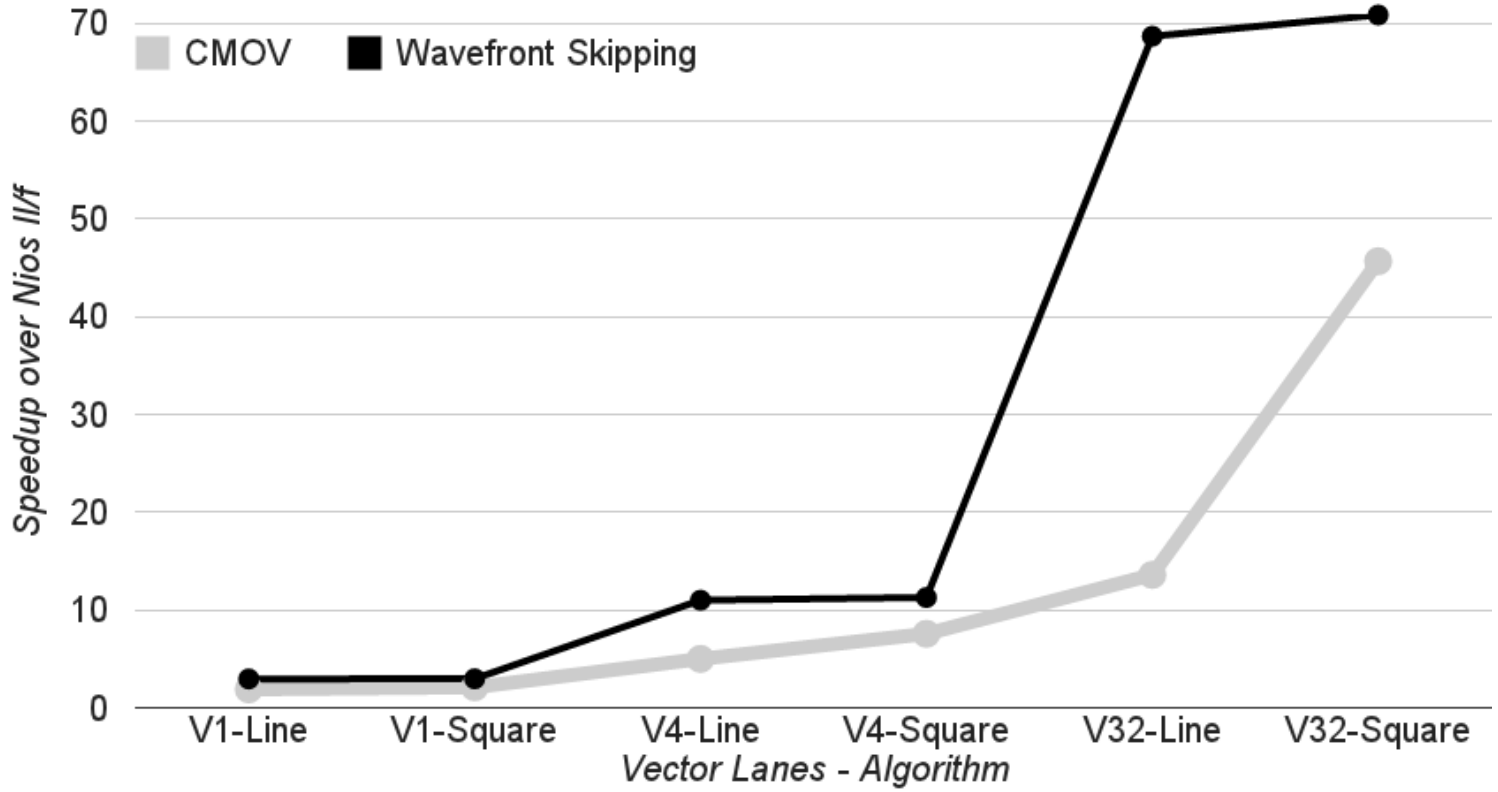


Work Done for Different Strategies

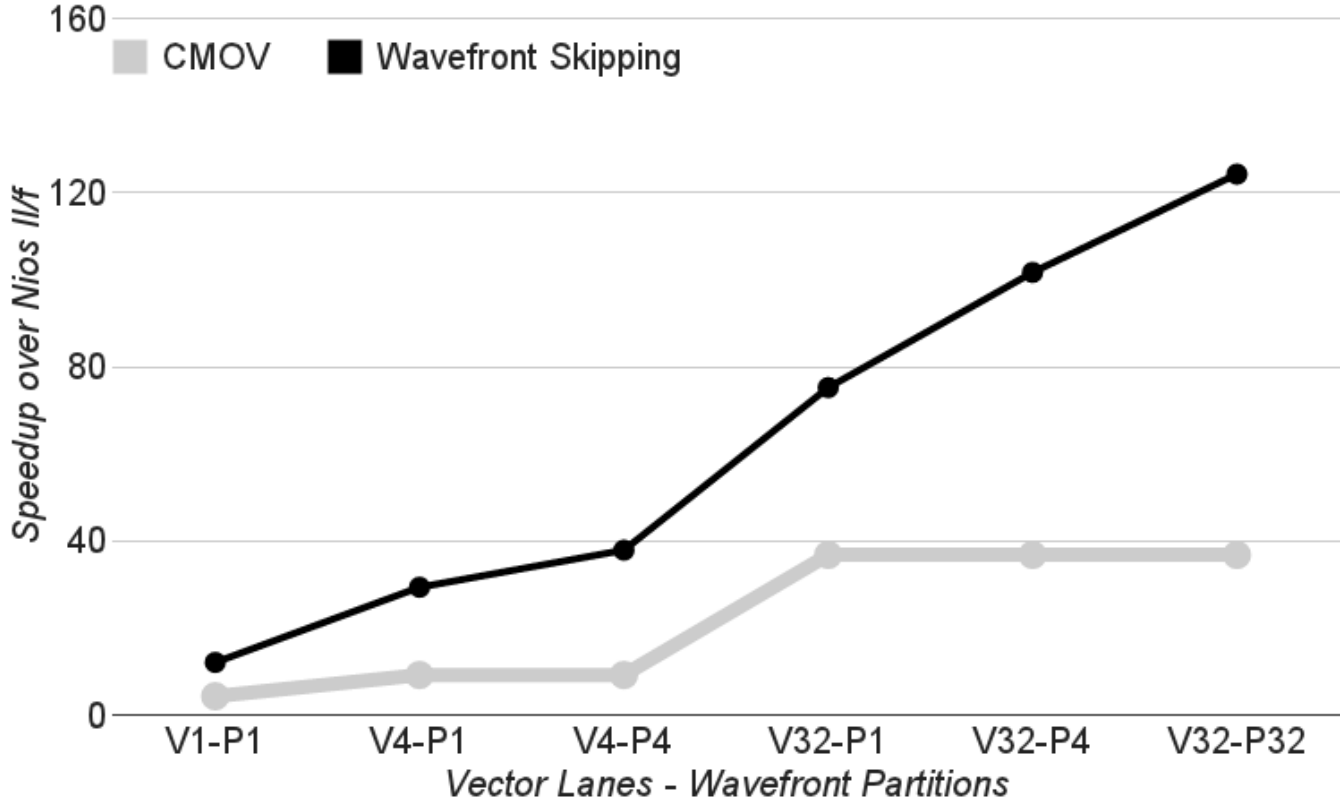


a) CMOV (Row)

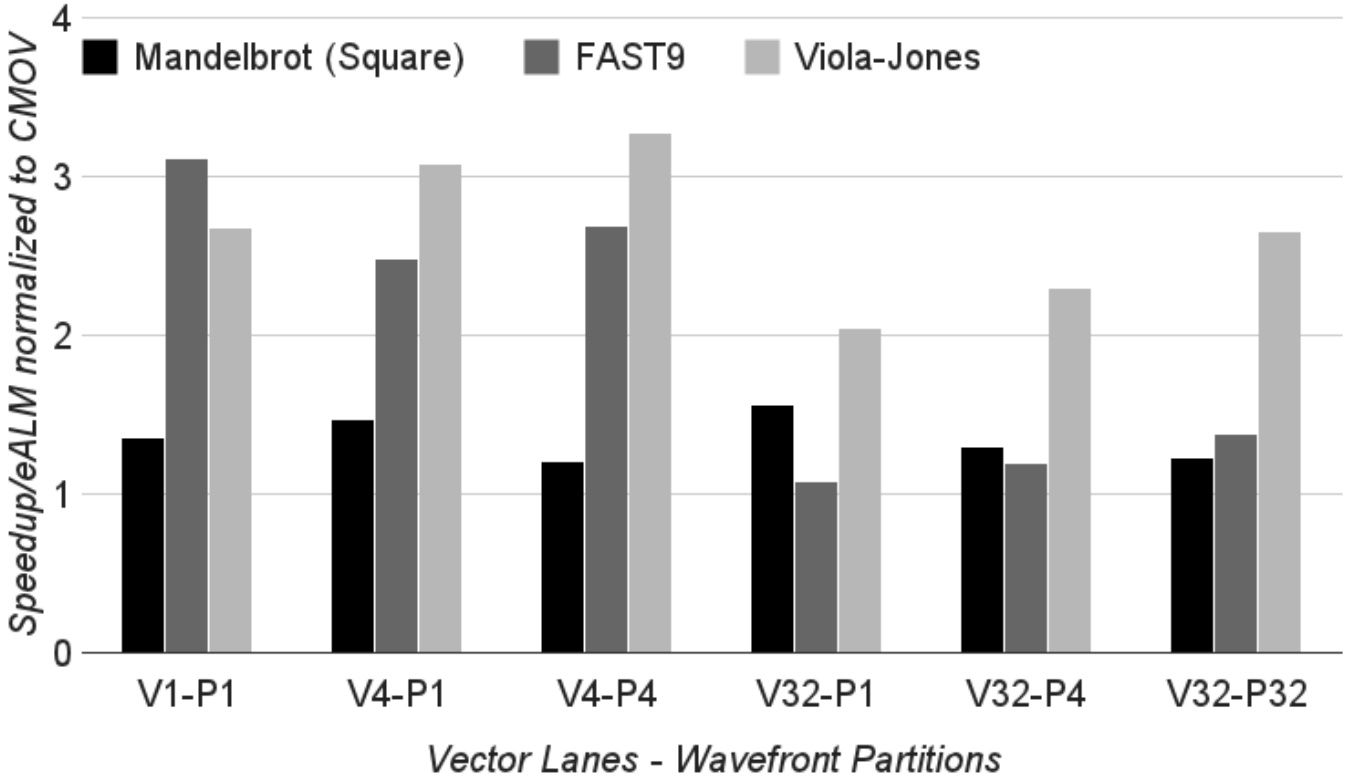
Different Strategies (Mandelbrot Set)



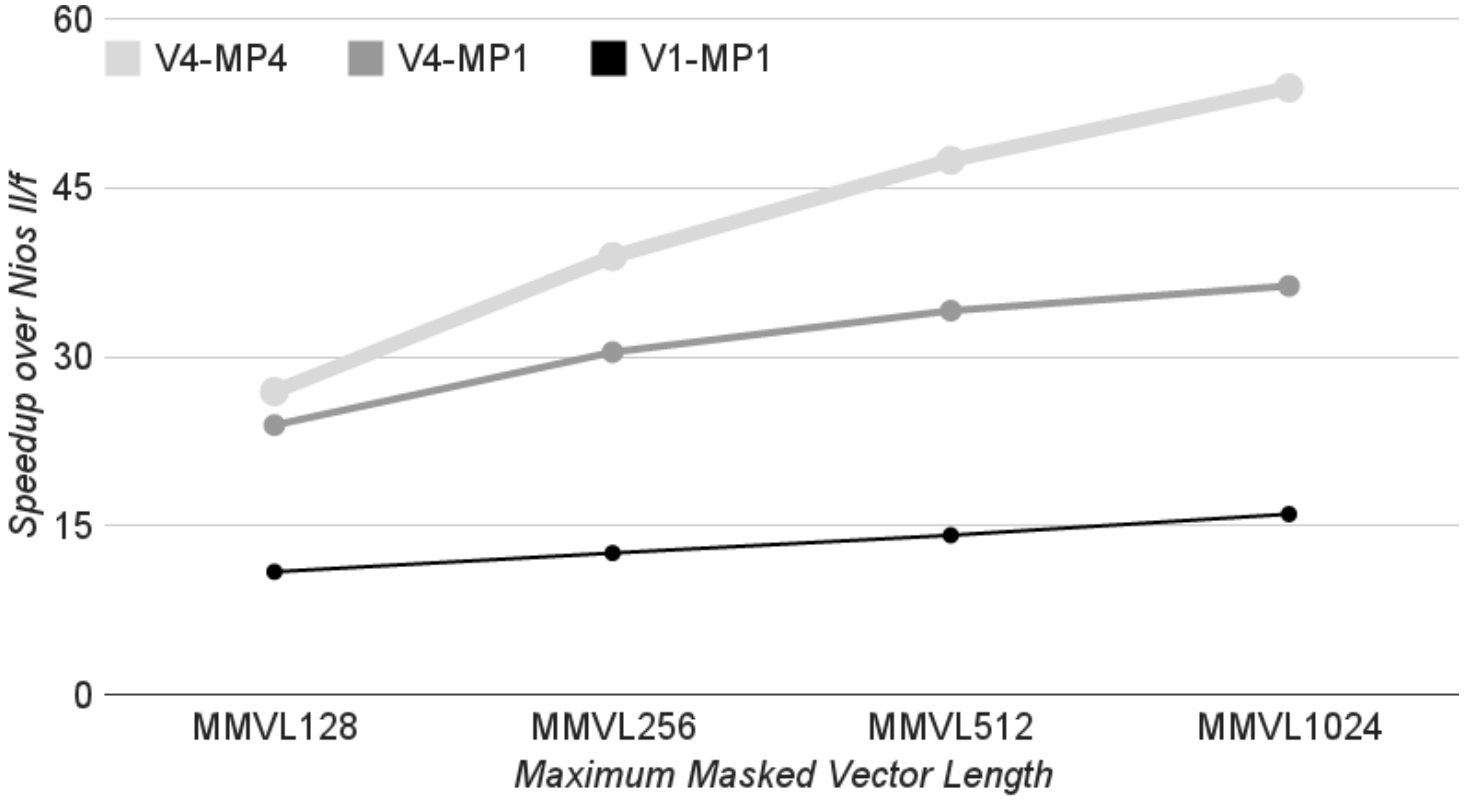
Face Detect Results



Speedup per Area (eALM)



Varying MMVL (Face Detect)



Limitations and Future Work

- Large area cost for multiple partitions
  - Area overhead on V4: 1 partition 4%, 4 partitions 26%
- Single mask
  - Requires extra instructions to store/restore

Conclusions

- BRAMs used to implement Wavefront Skipping
- Wide configurations space
  - Full Wavefront Skipping uses <5% more eALMs
  - Multiple partitions, longer MMVL can use 100’s of BRAMs
- 3x higher performance per area on early exit algorithms

Thank You!