Application-Specific Logic-in-Memory for Polar Format Synthetic Aperture Radar

Qiuling Zhu, Eric L. Turner, Christian R. Berger, Larry Pileggi, Franz Franchetti

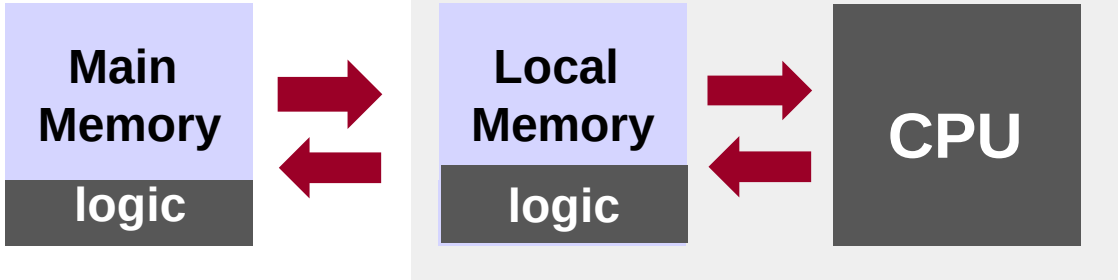
September 22, 2011

Application-Specific Logic-in-Memory

Can we *push* some memory-intensive computational *logic into or close to the memory* by constructing a smart and efficient *"Logic in memory"* block ?

Traditional:

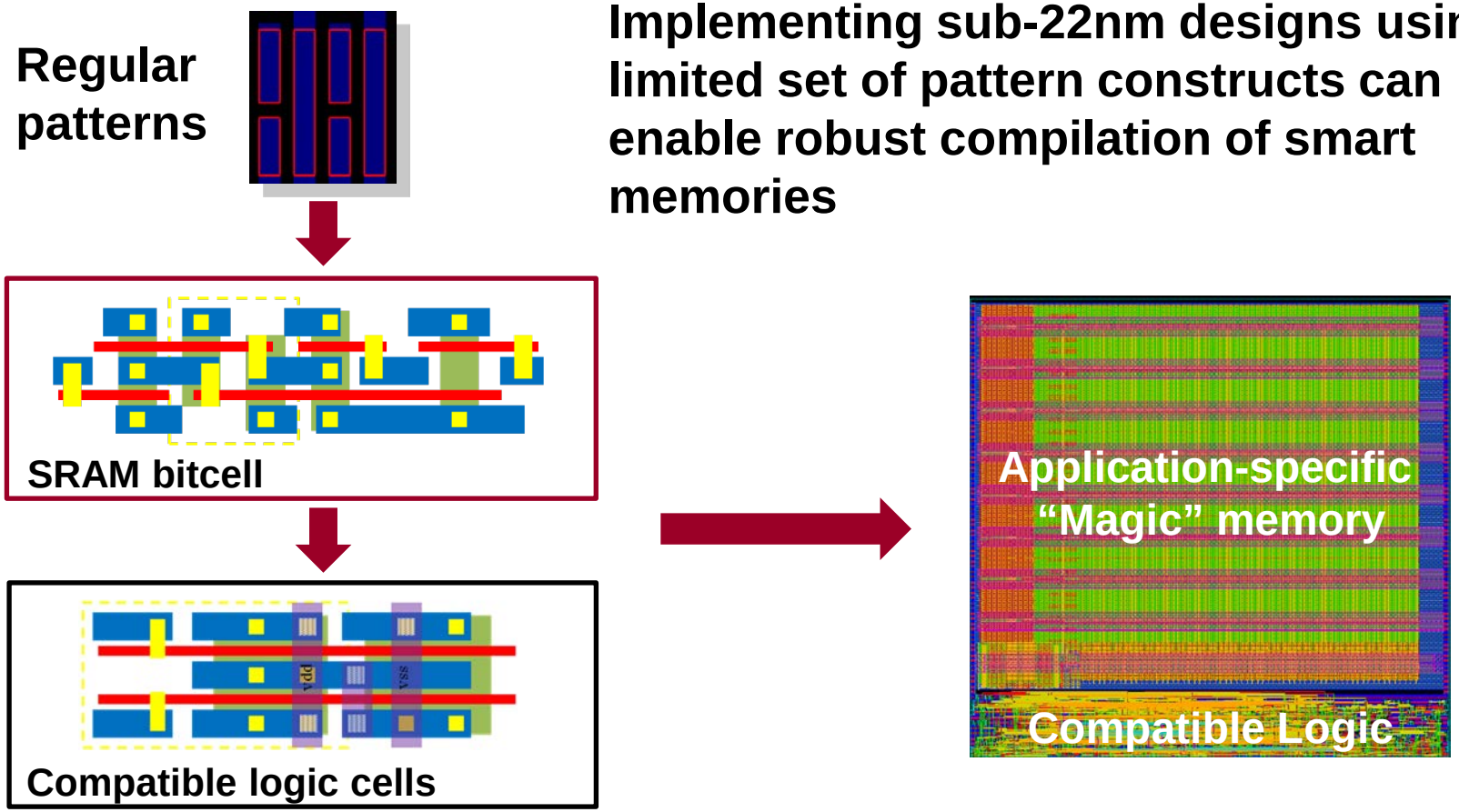
Logic-in-memory:



Enabling Technology: Regular Patterns

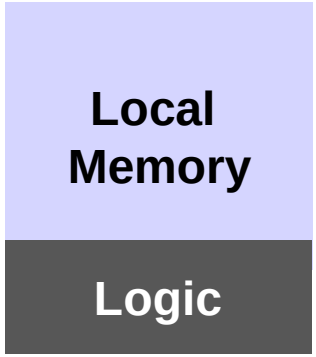
D. Morris, et. al, "Design of Embedded Memory and Logic Based On Pattern Constructs" , Symp.VLSI Technology, June 2011.

Implementing sub-22nm designs using a limited set of pattern constructs can enable robust compilation of smart memories



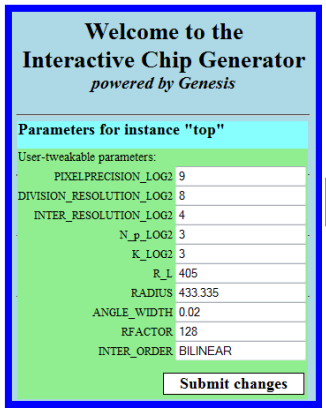
Tool Chain: Chip Generator and Memory Compiler

Logic in Memory

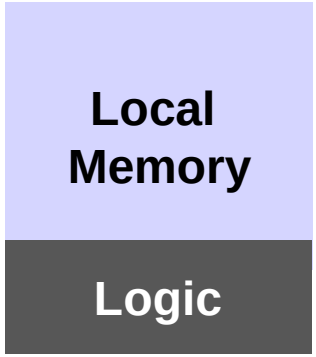


Tool Chain: Chip Generator and Memory Compiler

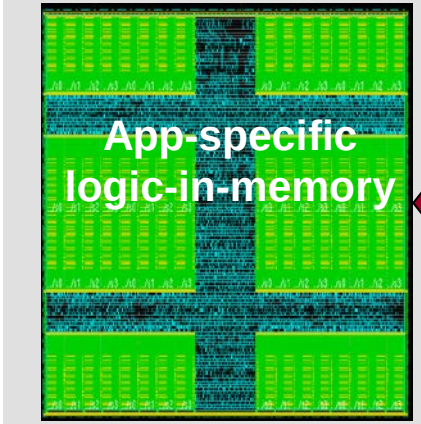
Chip Generator



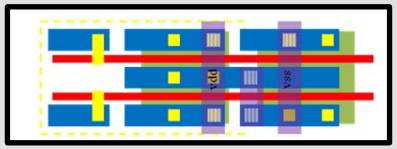
Logic in Memory



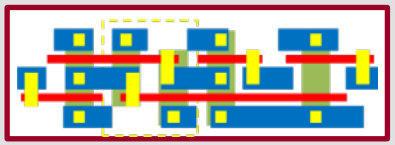
Smart Memory Compiler



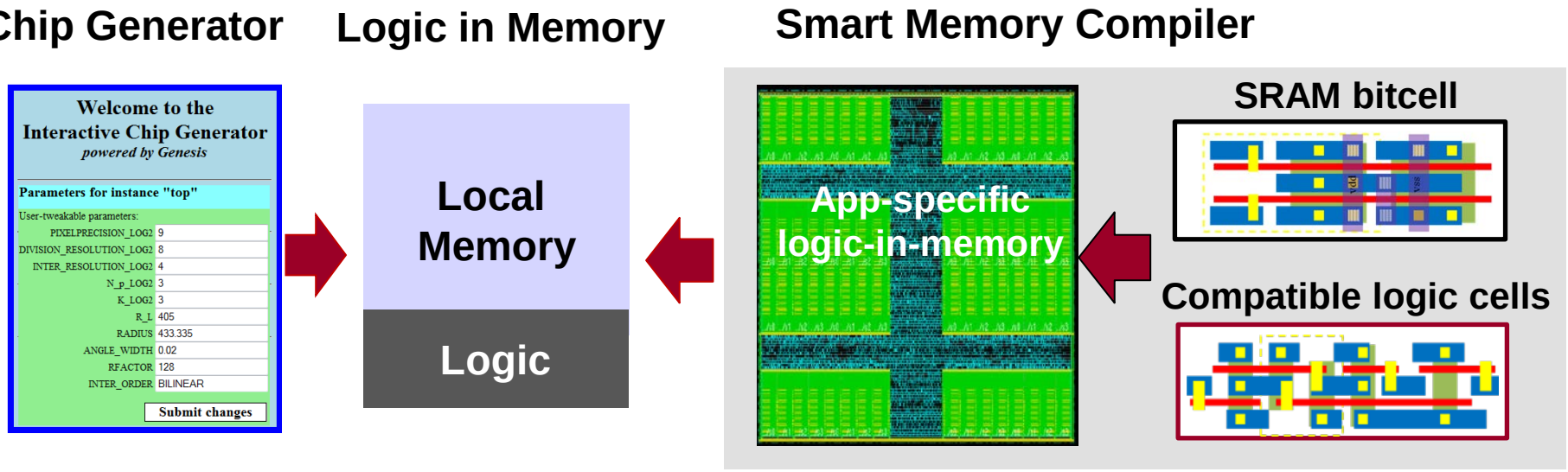
SRAM bitcell



Compatible logic cells



Tool Chain: Chip Generator and Memory Compiler



Chip Generator

- Generates designs from high-level parameterization and specification
- Utilizes Stanford's chip generator platform (Genesis 2)

Smart Memory Compiler

- Map memory and logic onto a set of pre-characterized pattern constructs
- Allow flexible synthesis of logic and memory functionalities in place of hard IP

Big Question: Impact on Algorithms

Logic-in-memory changes the relative cost of operations, requiring new types of algorithms.

Traditional

- **Data storage and processing are logically and physically split**
- **Algorithms are optimized w.r.t. cost measure as**

  Operation count, minimum number of memory accesses, reuse,…

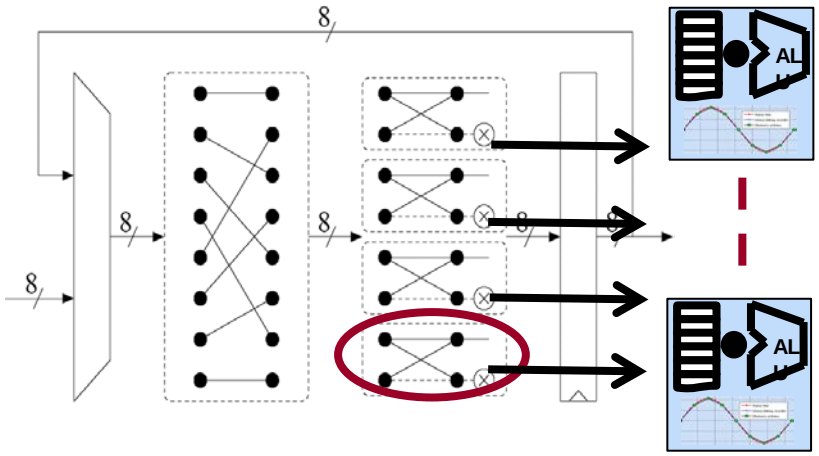
  eg. FFT: O(log n), Matrix Multiplication: O(n)

Logic-in-memory

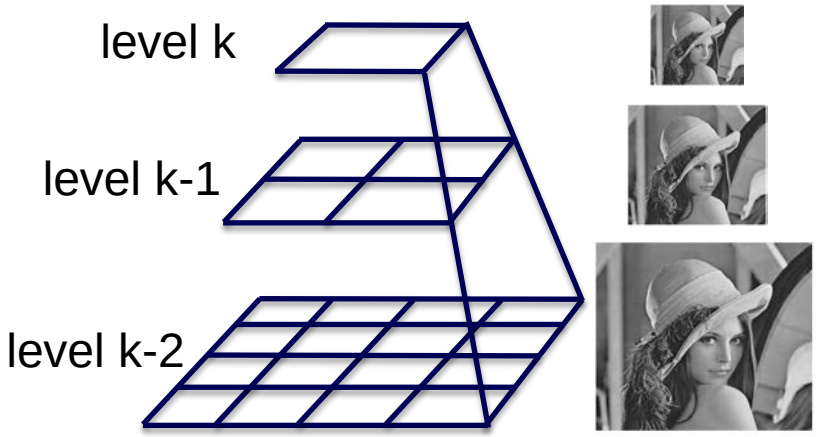
- **Local data dependency**
- **Regular memory access pattern**
- **Simple computational logic**
- **Cost measure changes**

Case Study: Interpolation Memory

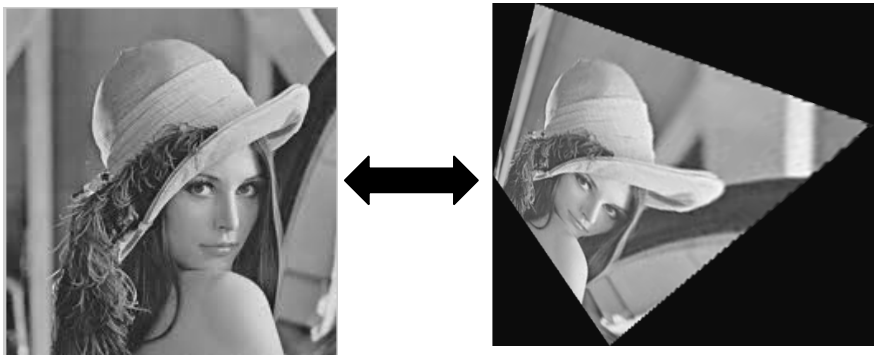
- Ex 1: FFT Twiddle Factor



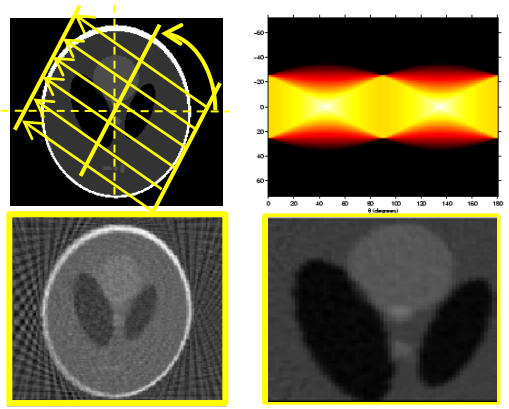
- Ex 2: Image Pyramid Memory



- Ex 3: Geometry Transformation



- Ex 4: Tomography Backprojection



Outline

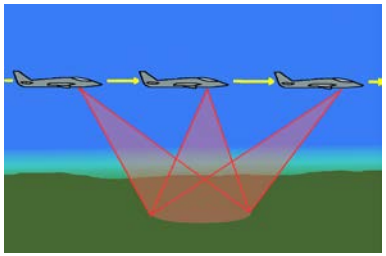
- **SAR Polar Format Algorithms for Logic-in-Memory**

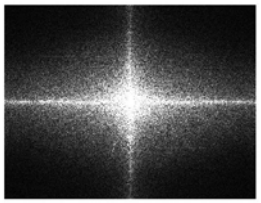
  Extension: Partial Reconstruction

- **Implementation and Design Automation**
- **Experimental Results**
- **Summary**

Synthetic Aperture Radar (SAR)

Data acquisition





Synthetic Aperture Radar (SAR)

Data acquisition

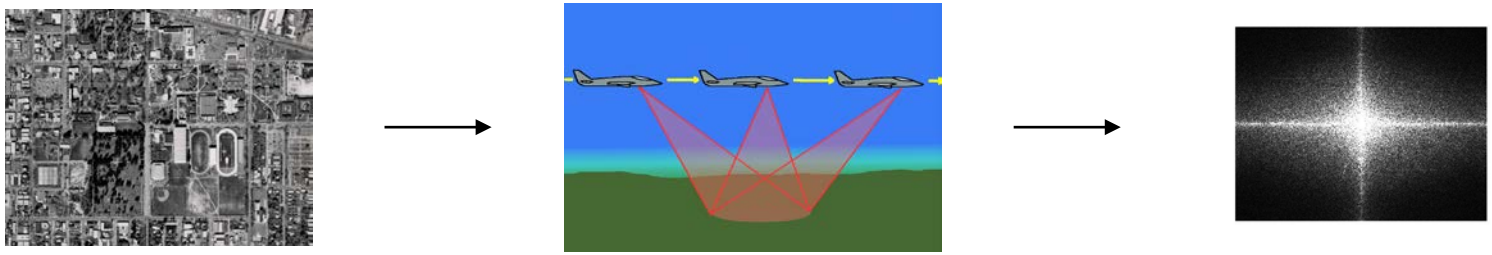
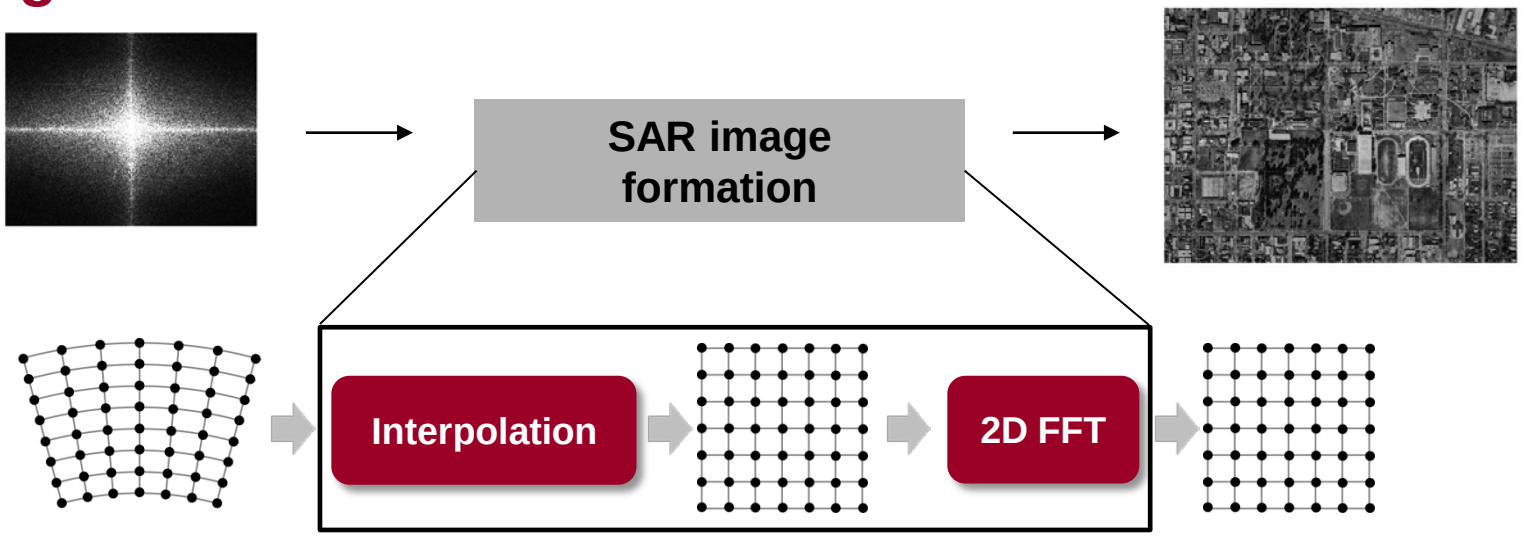
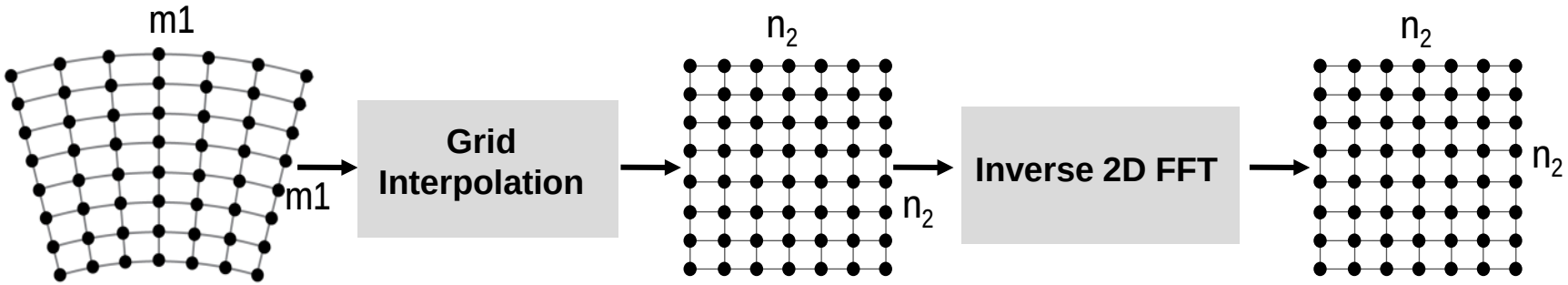


Image formation



FFT Upsampling Based Polar Reformatting



SAR image formation:

- **Range interpolation**
  - FFT upsampling based
- **Cross range interpolation**
- **2D inverse FFT**

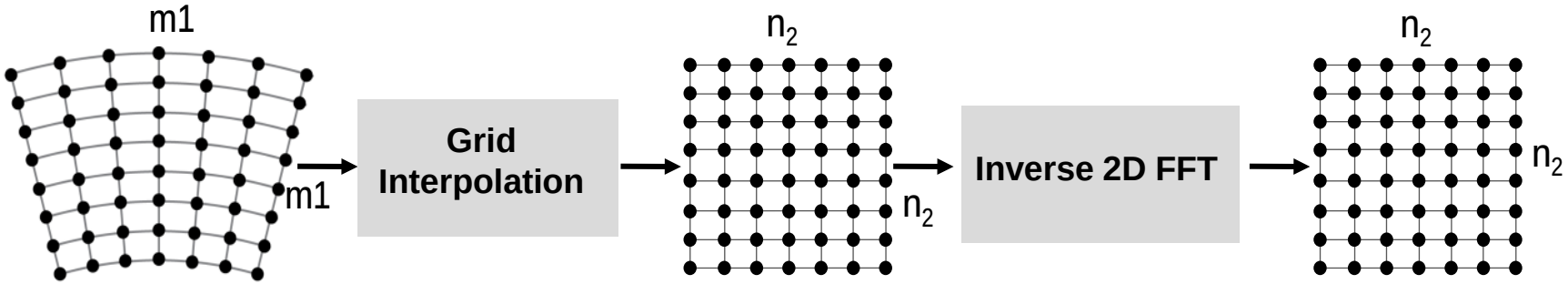
Computational cost:

Interpolation: $10lm_1 \cdot (m \cdot \log_2(m) + n \cdot \log_2(n))$

2D IFFT: $10 \cdot n_2^2 \cdot \log_2(n_2)$

I is the number of segments per range line, m is the input segment size and n is the size of the upsampled output segment.

FFT Upsampling Based Polar Reformatting



SAR image formation:

- **Range interpolation**
  - FFT upsampling based
- **Cross range interpolation**
- **2D inverse FFT**

Computational cost:

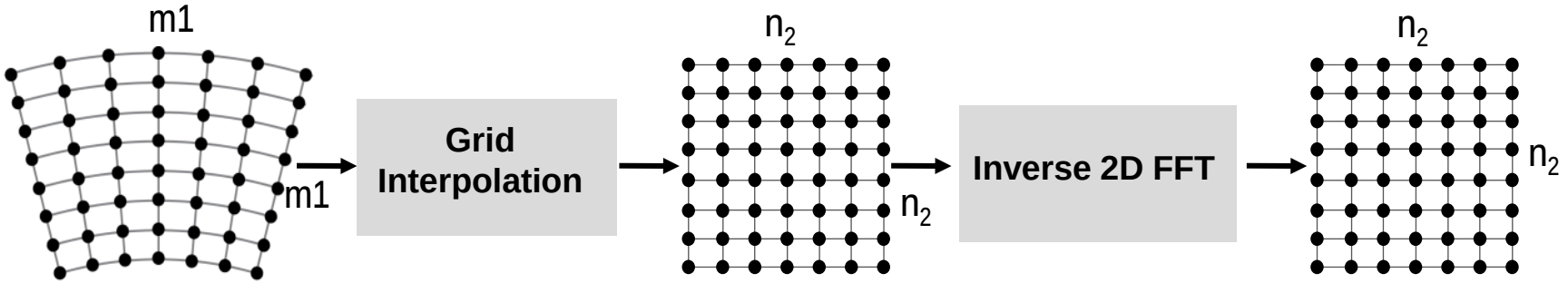
Interpolation: $10lm_1 \cdot (m \cdot \log_2(m) + n \cdot \log_2(n))$

2D IFFT: $10 \cdot n_2^2 \cdot \log_2(n_2)$

I is the number of segments per range line, m is the input segment size and n is the size of the upsampled output segment.

Data transferring cost:

FFT Upsampling Based Polar Reformatting



SAR image formation:

- **Range interpolation**
  - FFT upsampling based
- **Cross range interpolation**
- **2D inverse FFT**

Computational cost:

Interpolation: $10lm_1 \cdot (m \cdot \log_2(m) + n \cdot \log_2(n))$

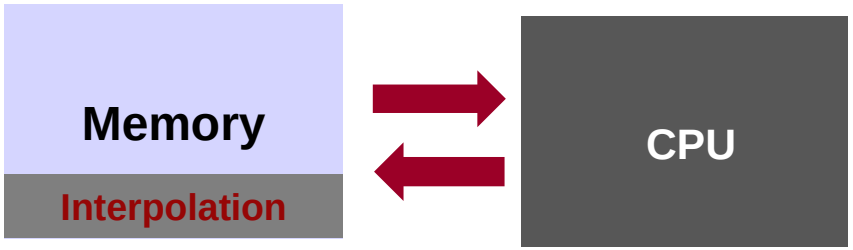
2D IFFT: $10 \cdot n_2^2 \cdot \log_2(n_2)$

I is the number of segments per range line, m is the input segment size and n is the size of the upsampled output segment.

Data transferring cost:

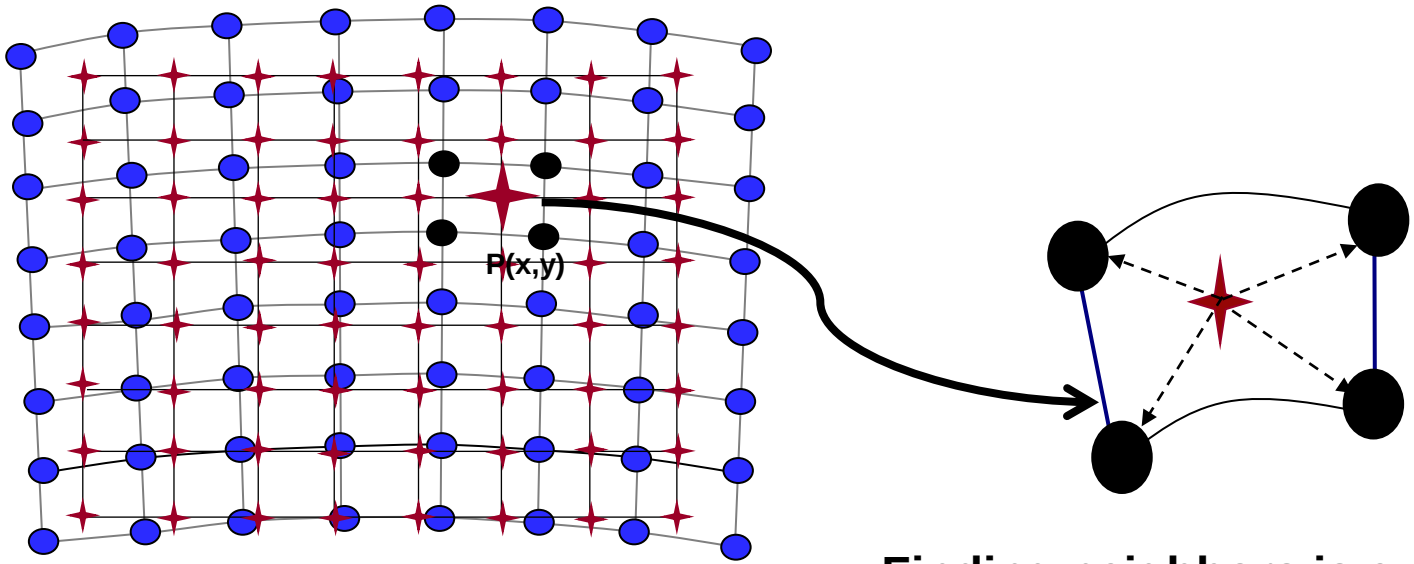
Logic-in-Memory Interpolation

- **Needs new algorithm**



Local Interpolation Based Polar Reformatting

Approach: direct local interpolation



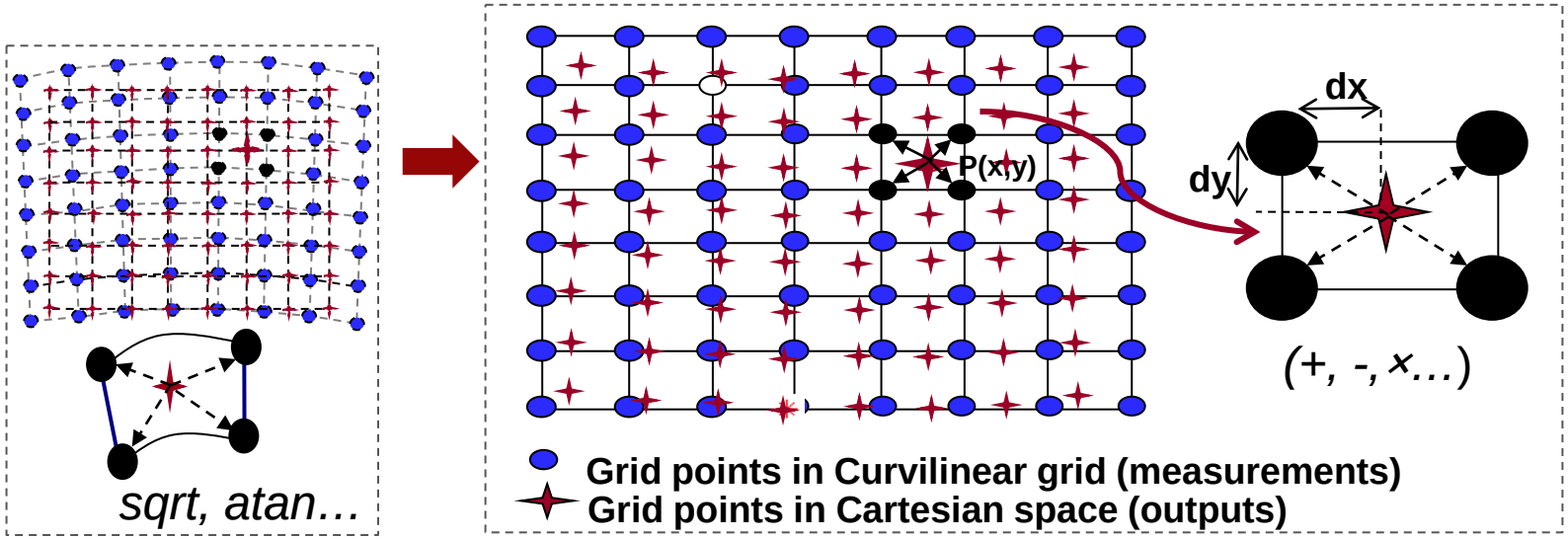
Finding neighbors is expensive

Grid points in Curvilinear grid (measurements)

Grid points in Cartesian space (outputs)

***sqrt, atan* operations are expensive in Logic-in-memory**

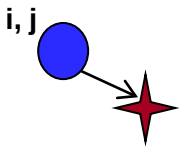
Local Interpolation Based Polar Reformatting



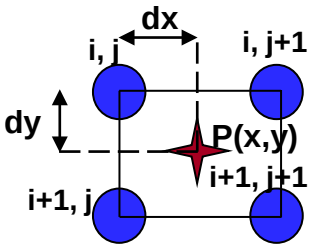
Steps:

- **Coordinate transformation**
  - Four-corner image perspective geometric transformation
  - Avoid ***sqrt*** and ***atan***
- **2D surface interpolation**
  - Simple logic computation
  - bilinear, bicubic,…

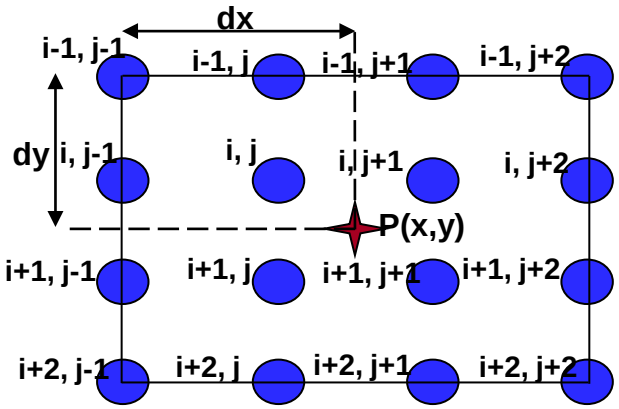
2D Interpolation



Nearest Neighbor



Bilinear Interpolation



Bicubic Interpolation

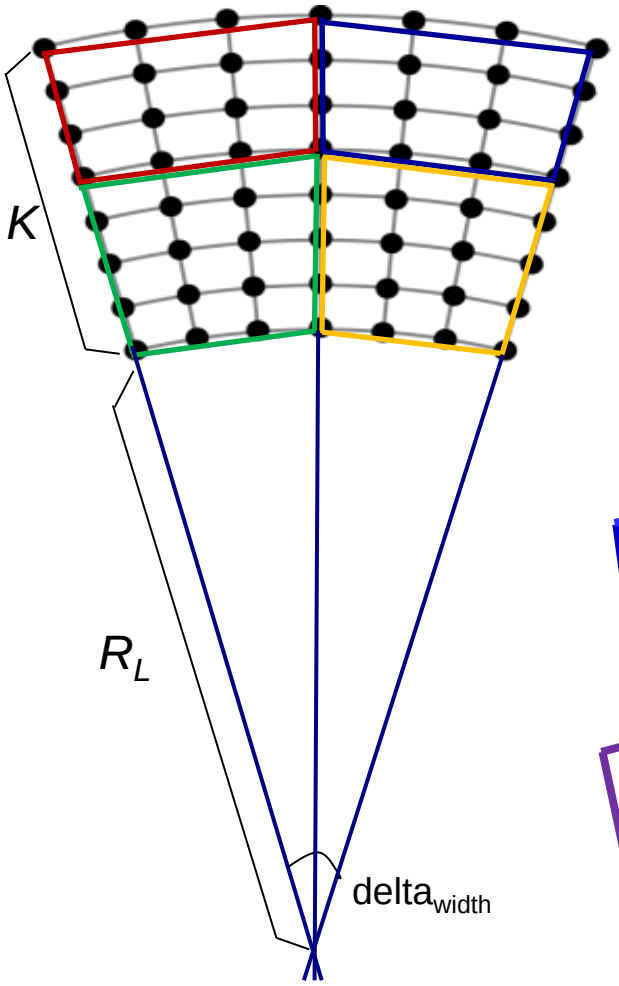
Dividable 2D interpolation

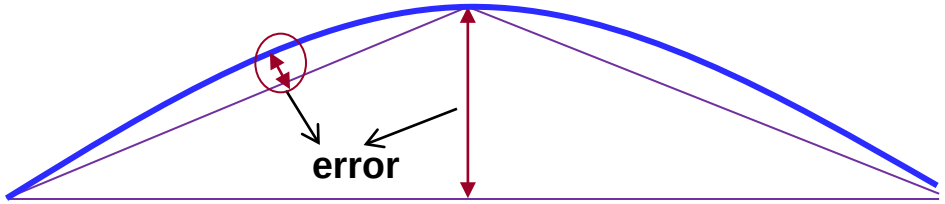
- **Bilinear:** (2 horizontal + 1 vertical) 1D interpolations
- **Bicubic:** (4 horizontal + 1 vertical) 1D interpolations
- **1D interpolation:** Newton divided difference form based polynomial interpolation

Suitable for Logic in Memory

- **Localized computation:** Outputs are only decided by their neighbors
- **Regular memory access:** Continuous or block data array access
- **Simple computational logic:** Adders, subs, boolean operations …

Tiling: Accurate Geometry Approximation

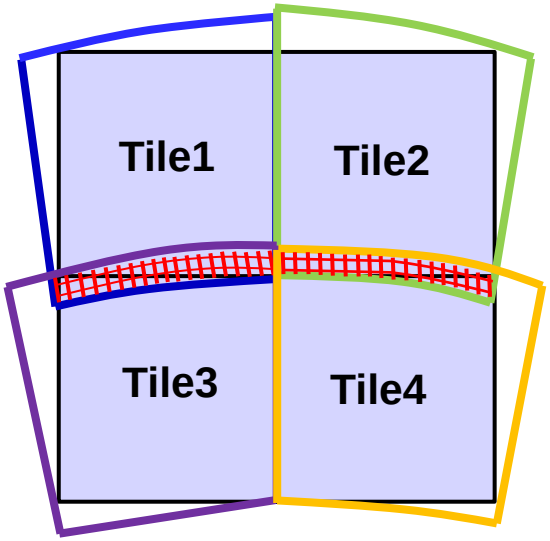




Geometry approximation conditions:

- $delta_{width}$ is small enough
- R_L is large enough

Solution: Image tiling



Tile in the Cartesian grid

- Output oriented tiling
- Easy to identify boundary and tile overlap

Outline

- SAR Polar Format Algorithms for Logic-in-Memory

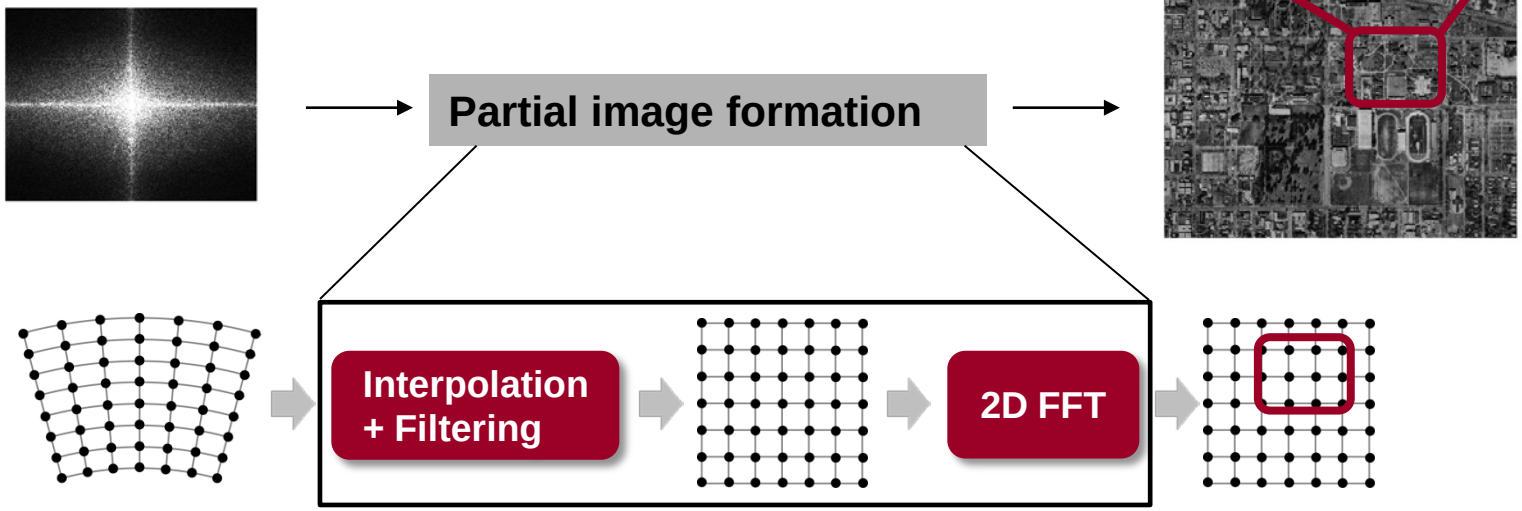
  Extension: Partial Reconstruction

- Implementation and Design Automation
- Experimental Results
- Summary

SAR Partial Reconstruction

- **Scenario:** Big image, small screen, pan-and-zoom (e.g. handheld device)
- **Bad approach:** reconstruct everything, display only region of interest
- **Better:** reconstruct only what will be displayed
  requires sophisticated filtering before reconstruction

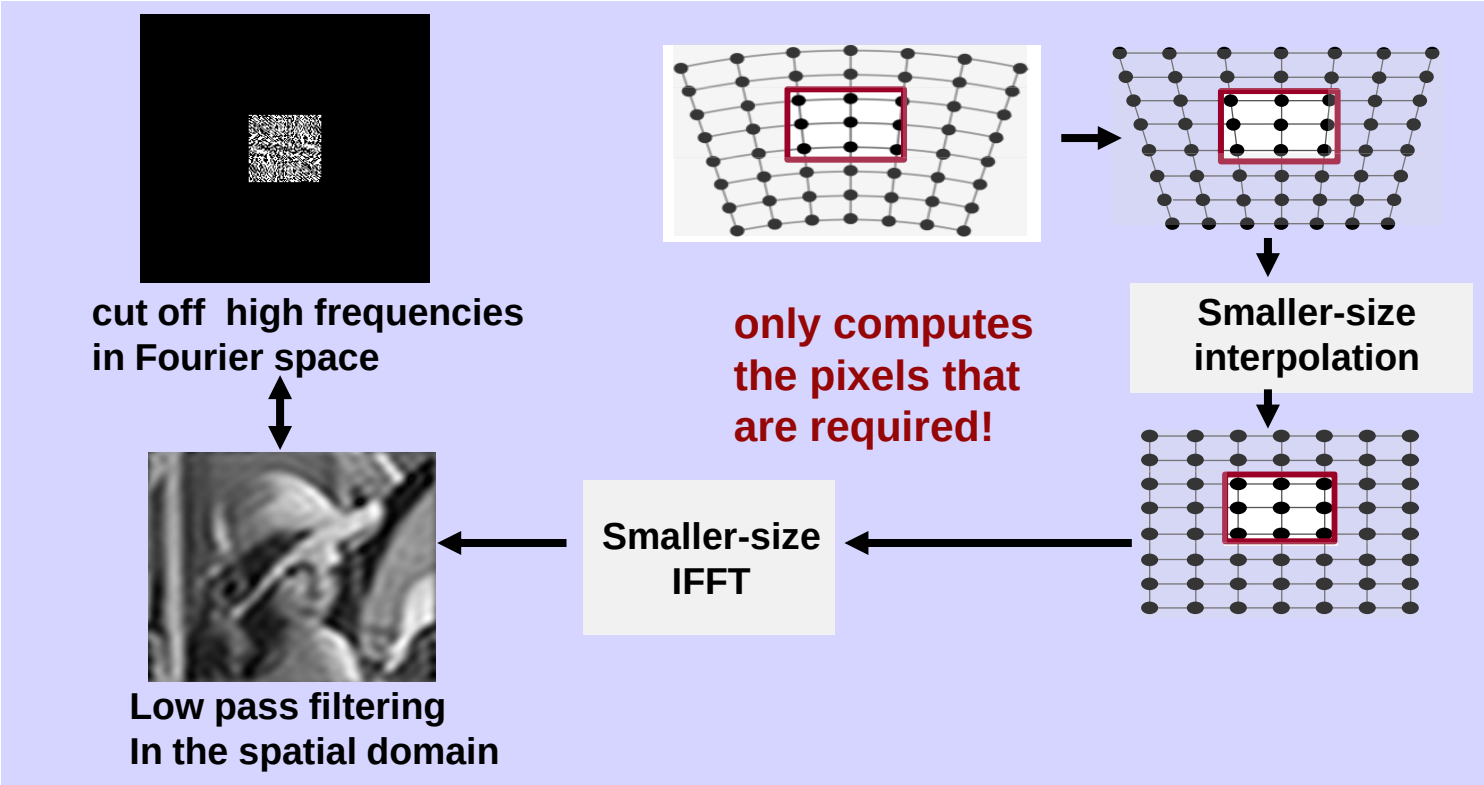
Partial Image formation



Partial Reconstruction I

Reconstructs and displays low-resolution full-size image

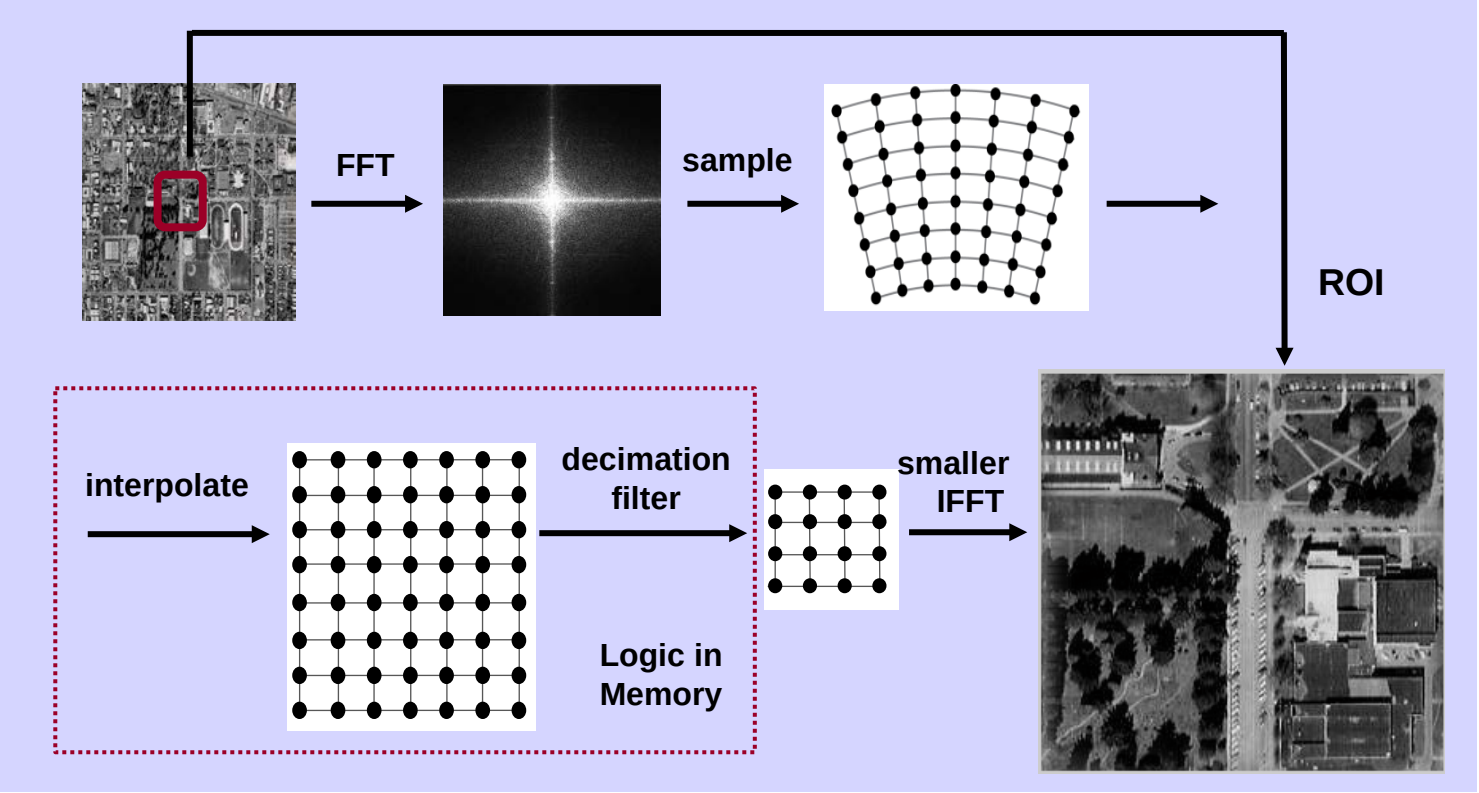
- **Traditional:** Interpolate all, full-size large IFFT then decimation
- **Alternative:** Partial interpolation then smaller-size IFFT
- **Theory behind:** Multiplication in the Frequency is identical to convolution in the spatial space.



Partial Reconstruction II

Reconstructs and displays a high-resolution image portion

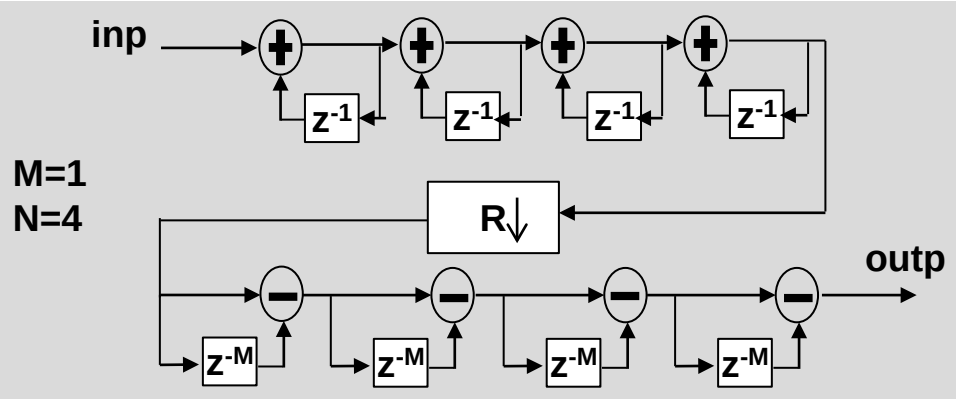
- **Traditional:** Full-size large IFFT, reconstruct all then cut off unnecessary region
- **Alternative:** Decimation filtering and then smaller-size IFFT
- **Theory behind:** Multiplication in the space is identical to convolution in the Fourier domain. Displacement in time is equivalent to phase shift



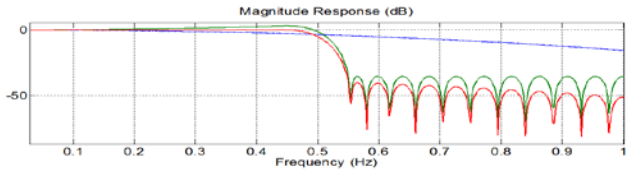
Decimation Filter Implementation

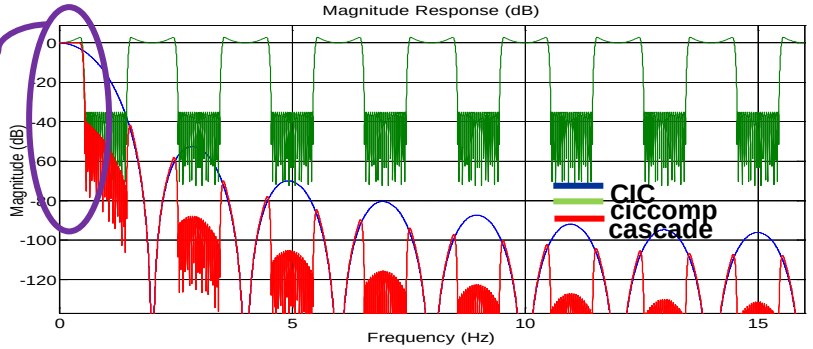
- **FIR Polyphase filter is expensive at high decimation factors**
- **Cascaded Integrated Comb(CIC) filter is more economical**
  - Large decimation factors
  - No multiplication
  - CIC compensation is required

CIC filter structure



Frequency Response:





CIC Spec: Decimation factor = 16; N = 4; M= 1
CIC Comp Spec:
Fp = 0.45; Fst = 0.55; Ap = 0.1dB, Ast = 35dB;
45 stages; downsample = 2 ; total decimation factor = 32 ;

Outline

- SAR Polar Format Algorithms for Logic-in-Memory

  Extension: Partial Reconstruction

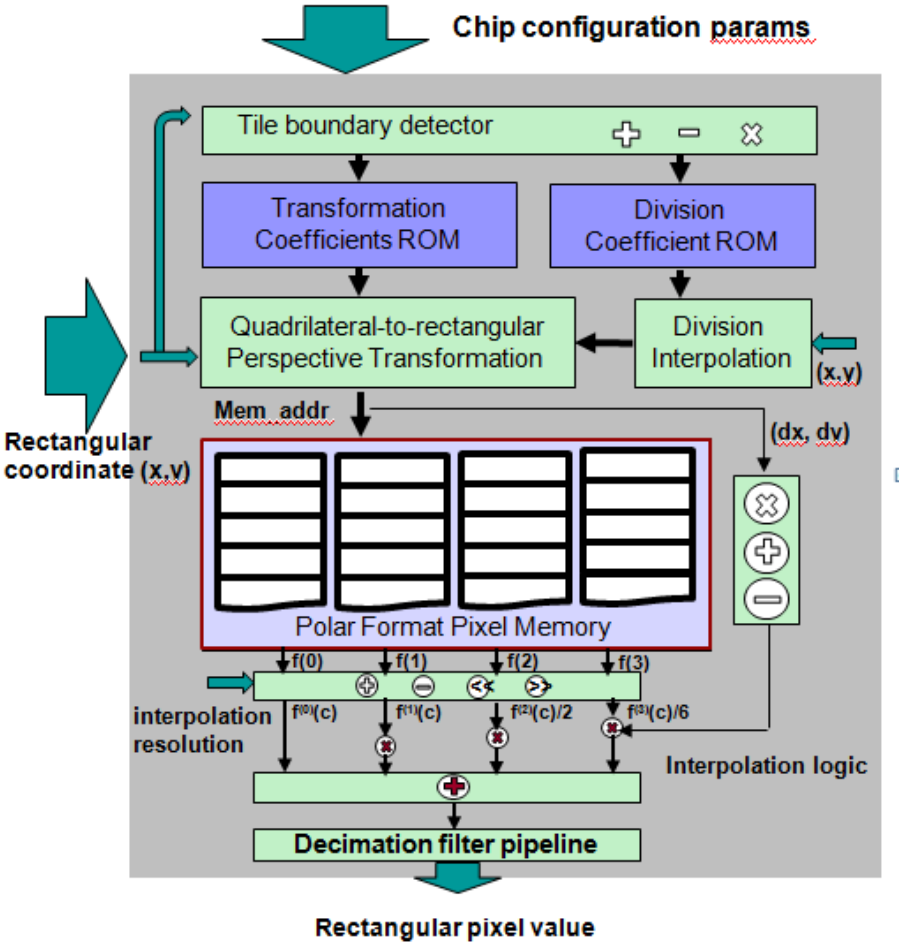
- **Implementation and Design Automation**

- Experimental Results

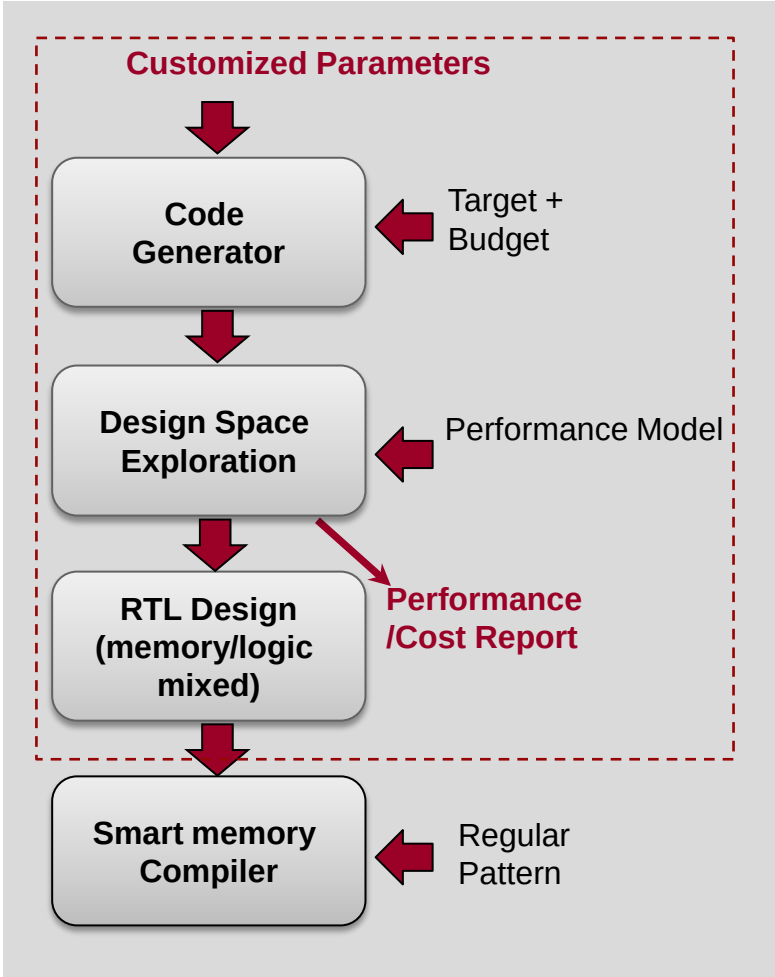
- Summary

Design Automation and Optimization

Hardware Structure

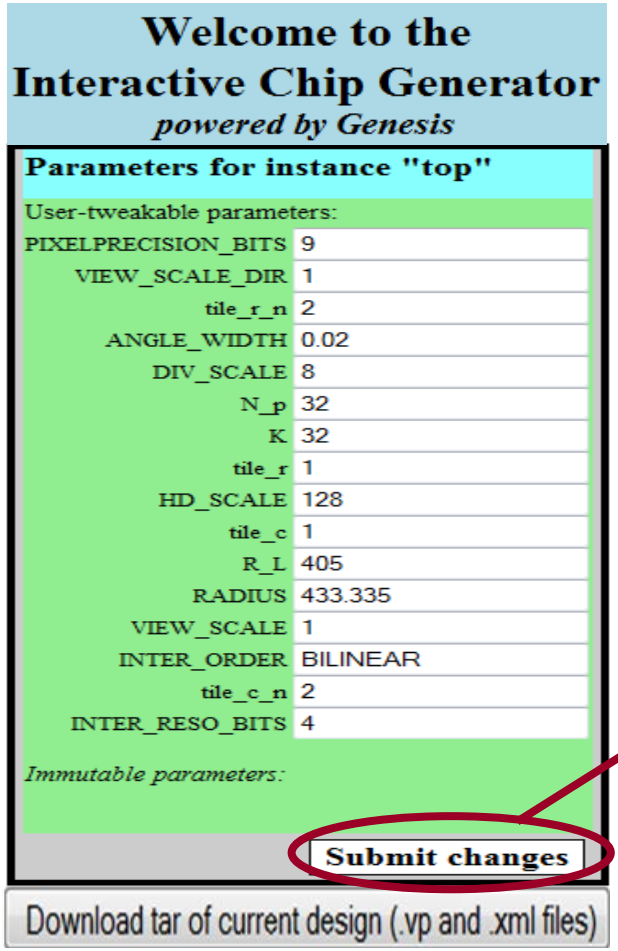


Design Automation Flow:



Chip Generator

http://genesis.web.ece.cmu.edu/gui/scratch/mydesign-10545.php



Reference: *O. Shacham, O. Azizi, M. Wachs, et. al, "Rethinking Digital Design: Why Design Must Change", Micro, IEEE, Dec 2010.*

```
//-----------------------------------------------------------------------------
//  THIS FILE WAS AUTOMATICALLY GENERATED BY THE STANFORD GENESIS2 ENGINE
//  FOR MORE INFORMATION, CONTACT OFER SHACHAM FROM THE STANFORD VLSI GROUP
//  THIS VERSION OF GENESIS2 IS NOT TO BE USED FOR ANY COMMERCIAL USE
// ---------------------- Begin Unique Status Report ----------------------
// Parameter -->N_p_LOG2<-- = -->8<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// Parameter -->INTER_RESOLUTION_LOG2<-- = -->4<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// Parameter -->K_LOG2<-- = -->8<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// Parameter -->ANGLE_WIDTH<-- = -->0.02<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// Parameter -->DIVISION_RESOLUTION_LOG2<-- = -->8<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// Parameter -->RFACTOR<-- = -->128<-- (Priority = _GENESIS2_INHERITANCE_PRIORITY_=3)
// ------------------------ End Unique Status Report ------------------------
module transform_unq1 (clk, reset, start, addr_row, addr_column, Cartesian_grid_out  ) ;
input clk, reset, start ;
input  [7 : 0] addr_row ;
input  [7 : 0] addr_column ;
output Cartesian_grid_out ;
//create rectangular matrices
wire signed [12 : 0] rect_x_mat ;
wire signed [17 : 0] rect_y_mat ;
assign rect_x_mat = -1460 + 11 * (addr_column) ;
assign rect_y_mat = 51815 + 13 * (256 - addr_row - 1) ;
//Interpolation
linear_1D_inter_unq1  linear_div_inter1(.offset({1'b0, addr_column}), .out1(w00_integer),
        .out2(w01_integer), .inter_value(pm1_div));
linear_1D_inter_unq1  linear_div_inter2(.offset({1'b0, addr_column}), .out1(w10_integer),
        .out2(w11_integer), .inter_value(pm2_div));
linear_1D_inter_unq1  linear_div_inter3(.offset({1'b0, addr_row}), .out1(pm1_div),
        .out2(pm2_div), .inter_value(wp_inter));
assign rect_xx = jj * wp_inter  ;
assign rect_yy = ii * wp_inter  ;
assign rect_x = rect_xx >> (28) ;
assign rect_y = rect_yy >> (28);
```

Outline

- SAR Polar Format Algorithms for Logic-in-Memory

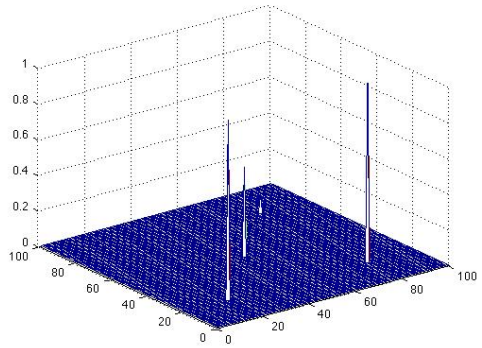
  Extension: Partial Reconstruction

- Implementation and Design Automation
- **Experimental Results**
- Summary

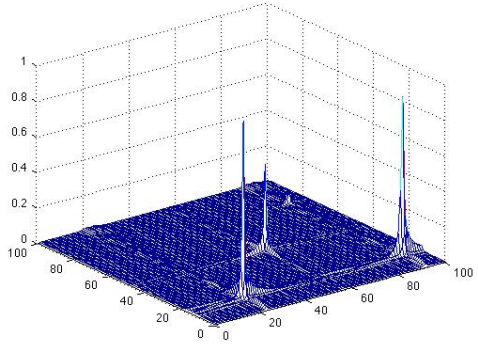
Reconstruction Quality vs. FFT SAR

Perfect reconstruction of point targets

original

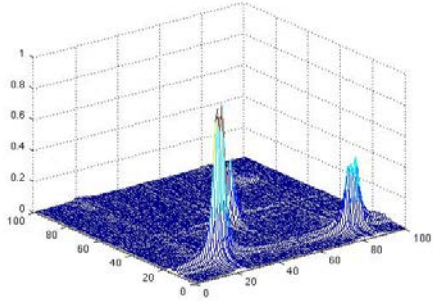


hermitian image

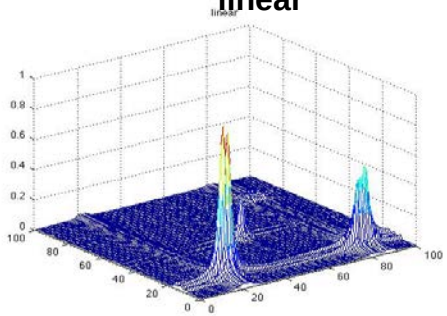


Actual reconstruction algorithms

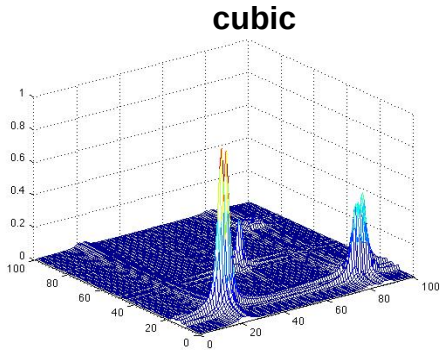
FFT-based



linear

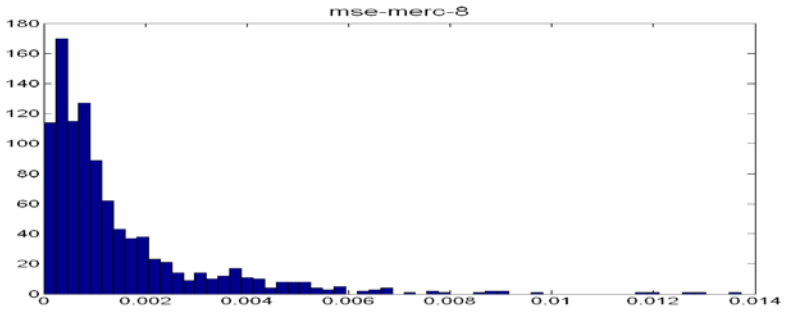


cubic

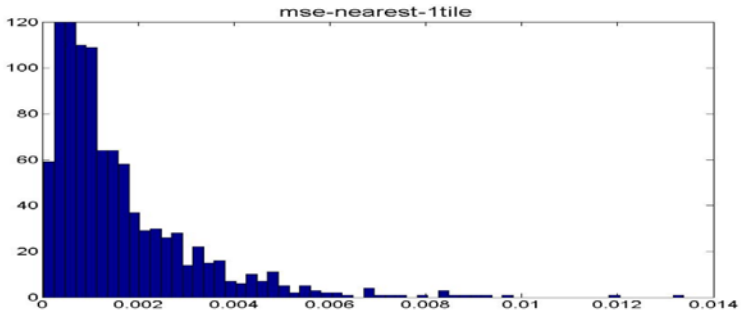


Is FFT-based SAR better than interpolation-based SAR?

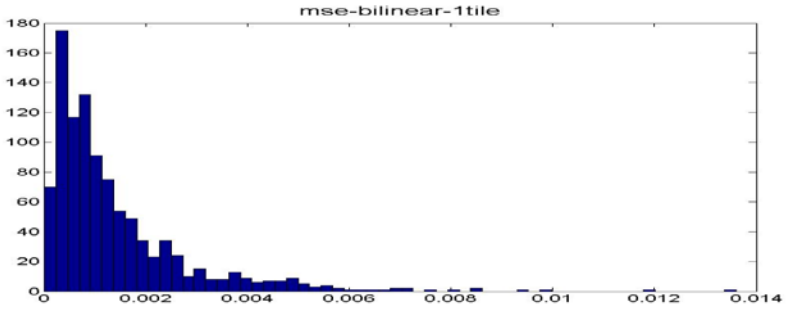
Can FFT and Interpolation Be Distinguished?



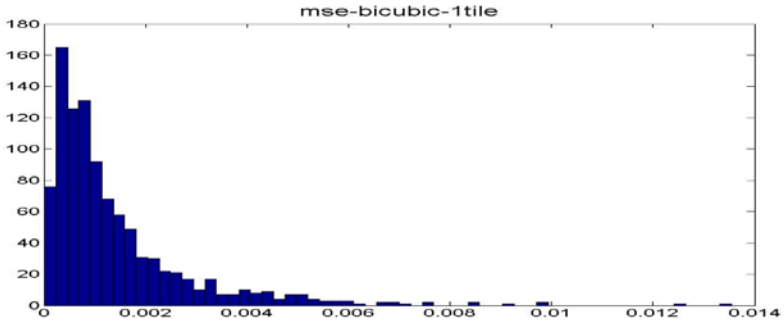
FFT interpolation



nearest neighbor interpolation



bilinear interpolation



bicubic interpolation

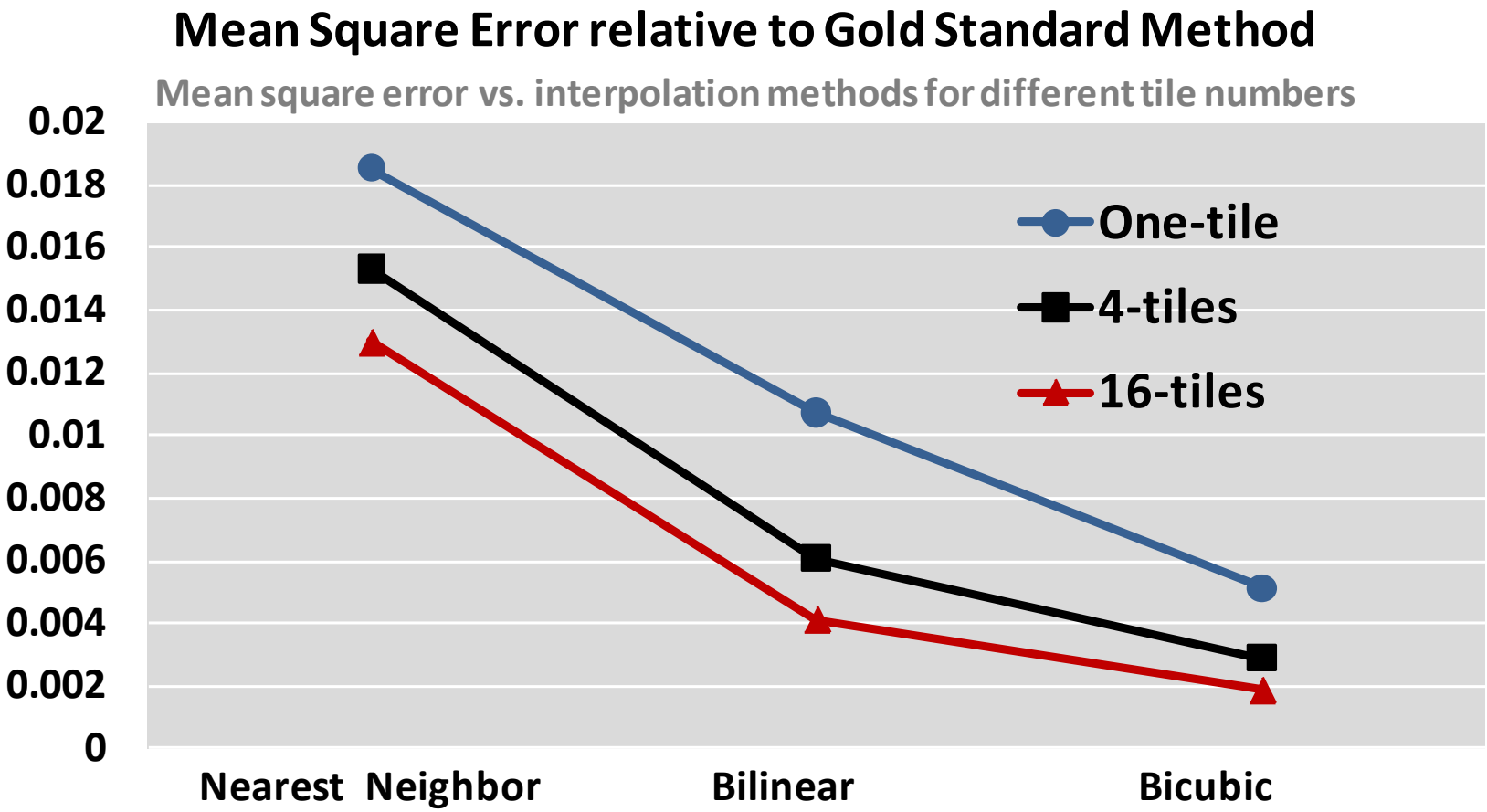
Answer: Hypothesis Testing

Hypothesis testing for linear and FFT: P(Error) = 0.495

Random guessing: P(Error) = 0.5

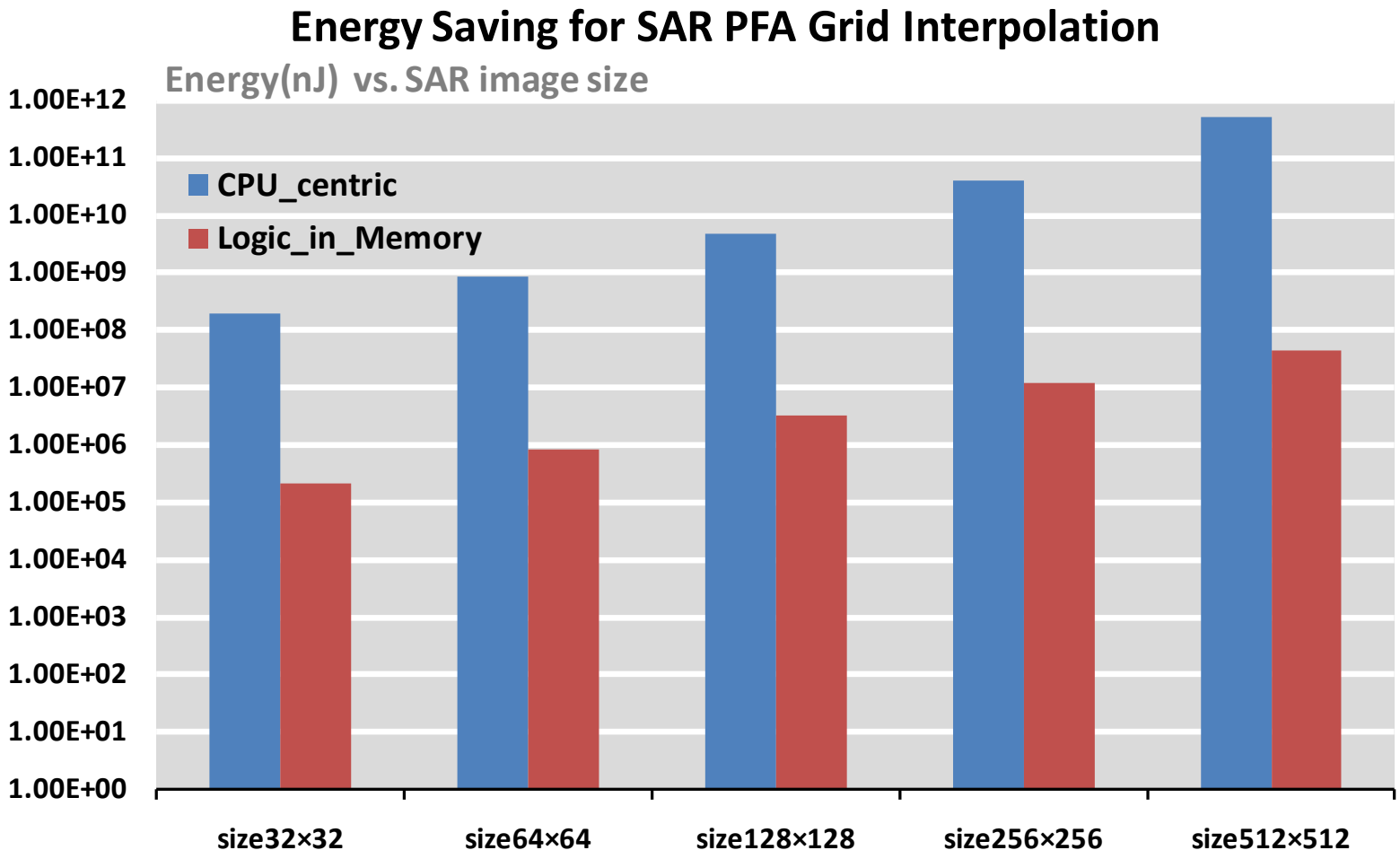
Results are statistically indistinguishable. Interpolation is as good as FFT

Accuracy Improvement Through Tiling



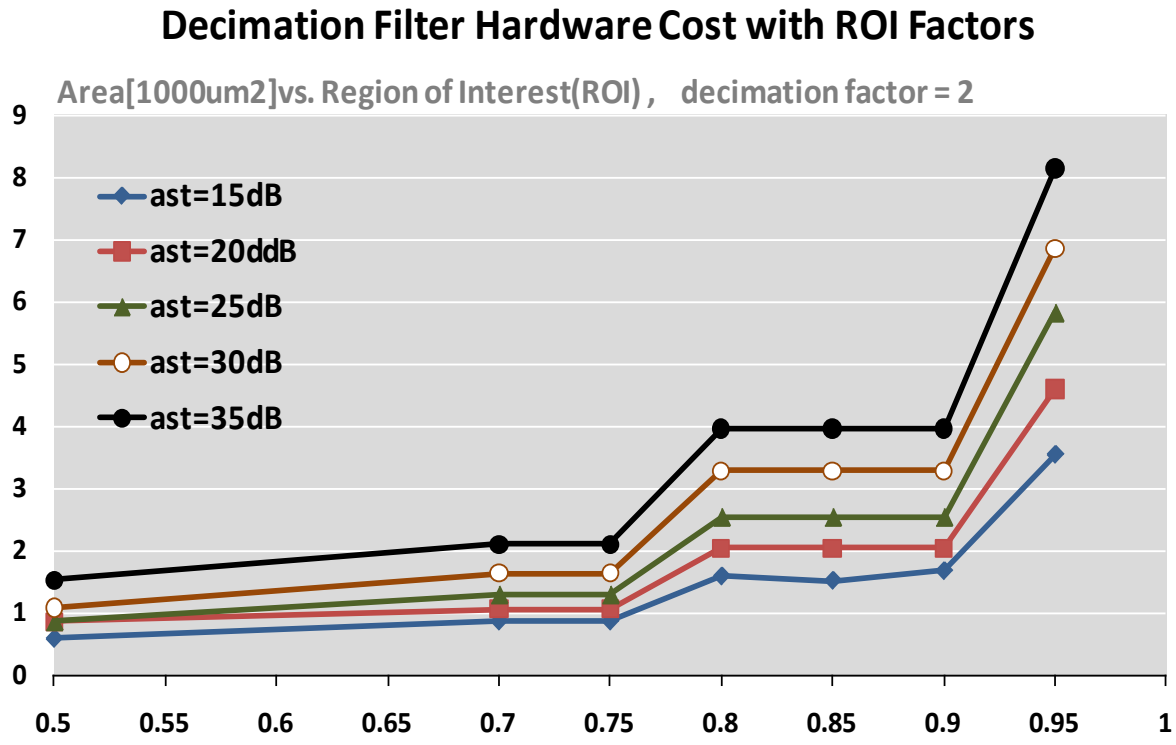
MSE decreases with more tiling and higher interpolation order

Energy Saving for Logic-in-Memory

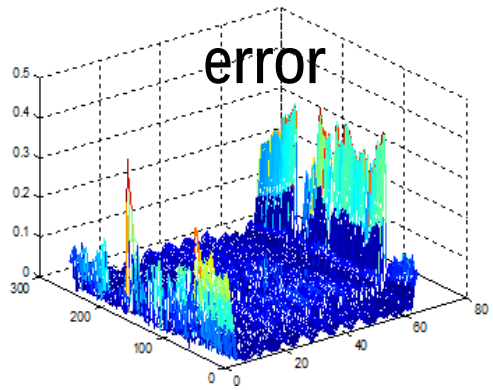


Energy saving increases with the increasing of problem size

Accurate Region-of-Interest by Sacrificing Border

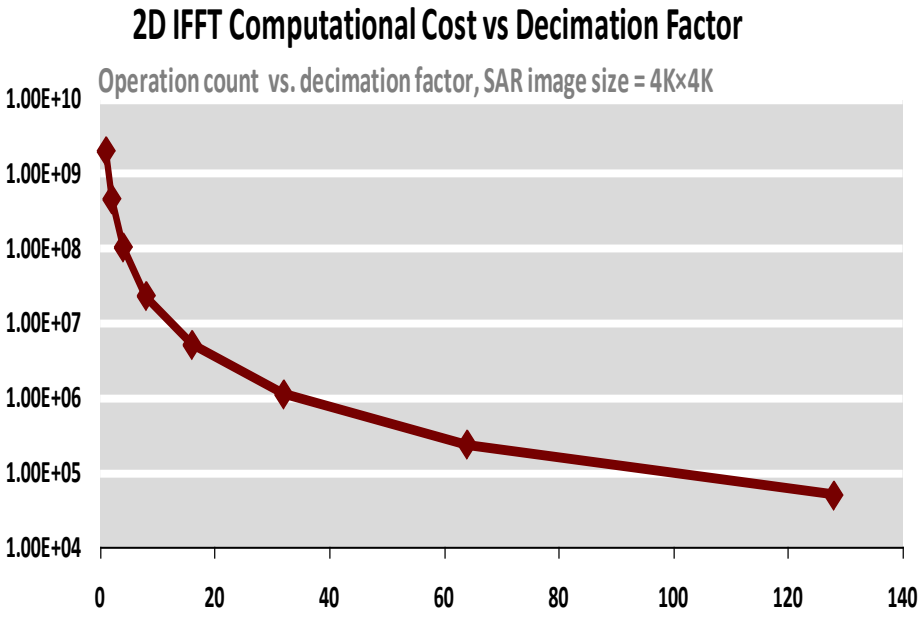


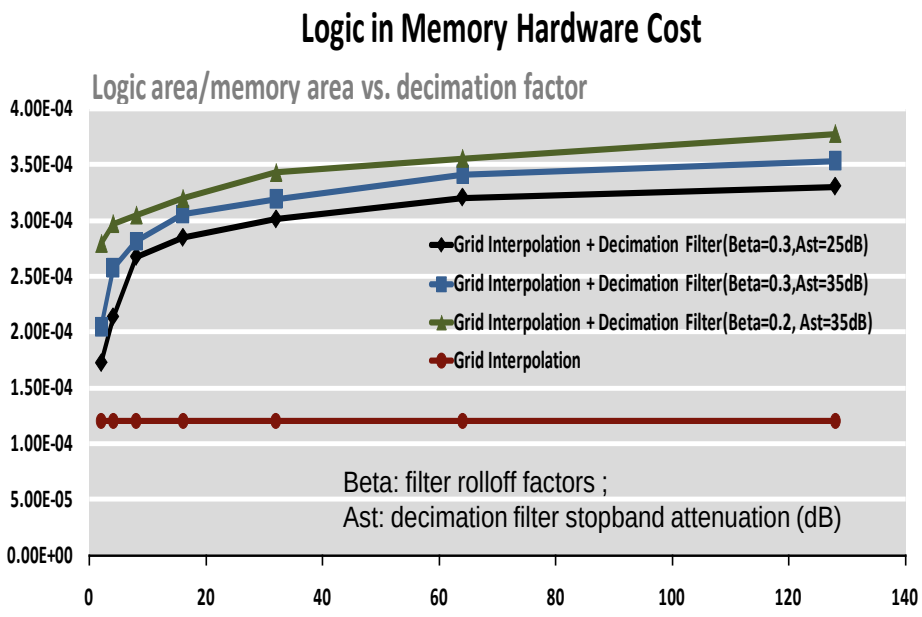
ast: decimation filter stopband attenuation (dB)



Imperfect image edge is resulting from non-steep filter transition region

Partial Reconstruction: Operation saving vs. Cost





- IFFT operation counts decreases exponential with increasing decimation factors
- Logic hardware cost is negligible compared with memory cost
- Decimation filter cost slightly increases when increasing decimation factors

Outline

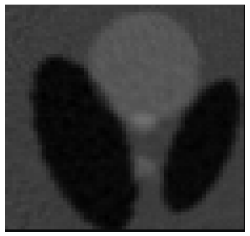
- SAR Polar Format Algorithms For Logic-in-Memory

  Extension: Partial Reconstruction

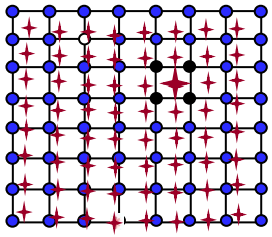
- Implementation and Design Automation
- Experimental Results
- **Summary**

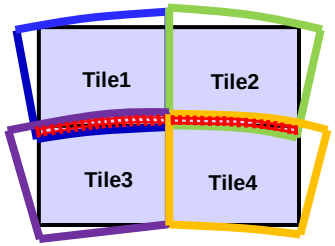
Summary

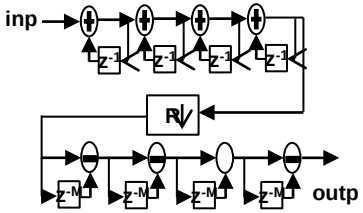
Logic in Memory and its applications for interpolation

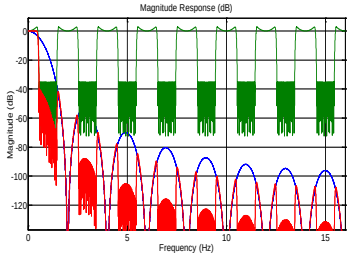


Logic in Memory for SAR FPA and partial reconstruction









Evaluation and integration with Genesis2