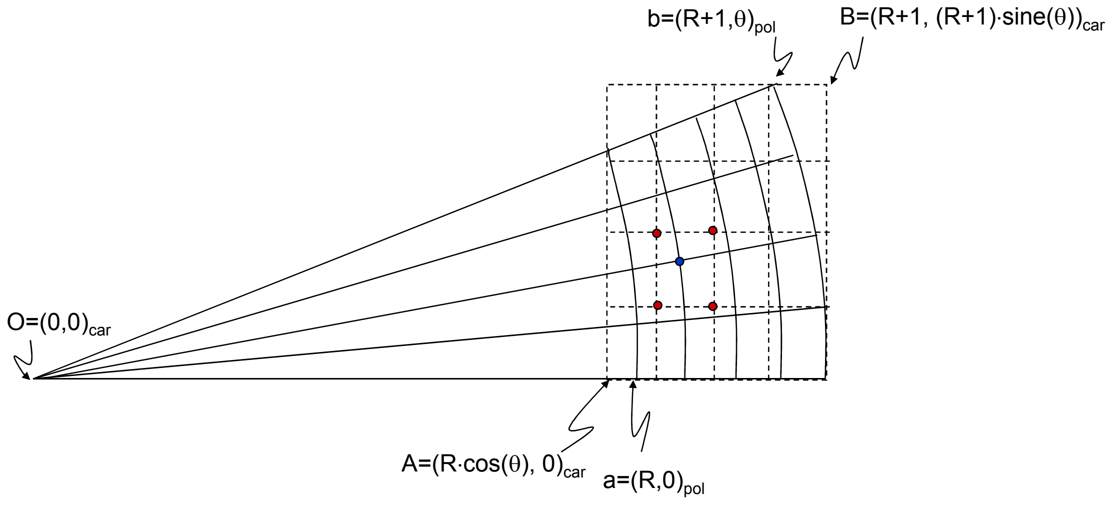
**Supported by:** CEDA bluespec XILINX

# MEMOCODE 2009 Design Contest: Cartesian-to-Polar Interpolation

**Revision March 6, 2009:** The lower bound on θ is changed to greater or equal to pi/256. Edits are highlighted below. θ>0 in the original specification.

The 2009 MEMOCODE Co-Design Contest is organized as a part of the 2009 MEMOCODE Conference (http://csg.csail.mit.edu/Memocode2009). The 2009 contest is organized by Forrest Brewer (UCSB) and James C. Hoe (CMU). The 2009 contest is generously supported by IEEE CEDA, Bluespec and Xilinx.

In the 2009 MEMOCODE Design Contest, you are tasked to implement a system to compute the values on an N×N grid in polar coordinates by interpolating the values from an enclosing N×N grid in the Cartesian coordinate (see Figure below).



$b=(R+1,\theta)_{pol}$
$B=(R+1, (R+1)\cdot \text{sine}(\theta))_{car}$
$O=(0,0)_{car}$
$A=(R\cdot\cos(\theta), 0)_{car}$
$a=(R,0)_{pol}$

The region C is defined by a bounding box, in Cartesian coordinates, with points $A=(R\cdot\cos(\theta),0)_{Cartesian}$ and $B=(R+1, (R+1)\cdot\text{sine}(\theta))_{Cartesian}$ on opposite corners, where $10\leq R\leq 100$, $(\pi/256)\leq\theta\leq(\pi/4)$. Region C is evenly spanned by an NxN grid where N is an integer between $10\leq N\leq 1000$. The values associated with the grid points are stored in a two-dimensional array CART where CART[0][0] is the value of the lower-left-most point; CART[0][N-1] is the value of the lower-right-most point; CART[N-1][N-1] is the value of upper-right most point.

The region P is defined by a bounding box, in polar coordinates, with points $a=(R,0)_{polar}$ and $b=(R+1, \theta)_{polar}$ on opposite corners. Region P is fully-enclosed by region C. Region P is also evenly spanned by an NxN grid. The values associated with the grid points are stored in a two-dimensional array POL where POL[0][0] is the value of the lower-left-most point; POL[0][N-1] is the value of the lower-right-most point; POL[N-1][N-1] is the value of the upper-right most point.

You are tasked to implement a system that computes the contents of the output array POL given the input array CART. The entry POL[t][r] corresponds to the point $(R+r/(N-1), \theta\cdot t/(N-1))_{Polar}$ or

$(x=(R+r/(N-1))\cdot\cos(\theta\cdot t/(N-1)),\ y=(R+r/(N-1))\cdot\text{sine}(\theta\cdot t/(N-1)))_{\text{Cartesian}}$. The value of POL[t][r] is the simple average of the four enclosing Cartesian grid points: $\text{CART}[\lfloor y/dy\rfloor+1][\lfloor(x-R\cdot\cos(\theta))/dx\rfloor]$, $\text{CART}[\lfloor y/dy\rfloor][\lfloor(x-R\cdot\cos(\theta))/dx\rfloor]$, $\text{CART}[\lfloor y/dy\rfloor+1][\lfloor(x-R\cdot\cos(\theta))/dx\rfloor+1]$ and $\text{CART}[\lfloor y/dy\rfloor][\lfloor(x-R\cdot\cos(\theta))/dx\rfloor+1]$), where $dx=((R+1)-R\cos(\theta)))/(N-1)$ is the separation between the Cartesian grid points in the x-dimension, and $dy=(R+1)\cdot\text{sine}(\theta)/(N-1)$ is the separation between the Cartesian grid points in the y-dimension. Your implementation must be able to handle all valid values of N, R and θ as parameters without recompilation or resynthesis. Specifically, the parameters N, R and θ and the contents of CART cannot be treated as compile time constants.

Your system must accept R and θ as arguments in the double-precision floating-point format. The values at the grid points are complex values with the real and imaginary components in a 16-bit two's-complement fixed-point format, 8 bits used for fraction. Your system must produce results that are accurate to $\pm$ 1/64 in each real and imaginary value relative to the results computed by the reference implementation (based on double precision calculations), except for those polar grid points whose $y/dy$ or $(x-R)/dx$ is within $\pm$ 1/32768 of an integral value and the peripheral points POL[0][ ], POL[N-1][ ], POL[ ][0], POL[ ][N-1].

In the reference implementation, each complex value is stored as a 4-byte quantity: the more significant 16 bits are used for the real component and the less significant 16 bits are used for the imaginary component. In your implementation, you are allowed to customize the layout of POL and CART in memory. You are also allowed to customize the layout of the real and imaginary components. To support standardized testing, you must provide the following *set* and *get* functions (or the equivalent of) if you use an alternate data layout. These *set* and *get* functions may not include calculations or conversions unrelated to "moving bits" from one place to another. When timing your executions, the test input and the resulting output must start and finish in an off-chip, external memory; preloading the input data into on-chip storage is not permitted. Any time spent in pre/post-processing and analysis of the input and output data must be included in the measured runtime.

```
SHORT16 getRPOL(void *base, int r, int t, int N);
SHORT16 getIPOL(void *base, int r, int t, int N);
SHORT16 getRCART(void *base, int x, int y, int N);
SHORT16 getICART(void *base, int x, int y, int N);

void setRPOL(void *base, int r, int t, int N, SHORT16 val);
void setIPOL(void *base, int r, int t, int N, SHORT16 val);
void setRCART(void *base, int x, int y, int N, SHORT16 val);
void setICART(void *base, int x, int y, int N, SHORT16 val);
```

---

Please email jhoe+MC09@ece.cmu.edu to receive instructions on downloading the reference software-only implementation of this challenge as a Xilinx EDK 9.2 project for the XUPV2 board. In the request, please specify: 1. a team name; 2. team member names and affiliations; and 3. the contacting email address.

There are no obligations associated with receiving or using the open-source-licensed reference implementation. The reference implementation includes a rudimentary testing and timing jig. We like to acknowledge Rachata Ausavarungnirun (CMU) for preparing this year's reference implementation. We will release additional test cases in the final week of the contest. Additional blind testcases will be applied after your submission to check your finished entry for correctness.

Please refer to the Official Contest Rules (http://www.ece.cmu.edu/~jhoe/distribution/mc09contest/contestrules09.pdf) for additional details. Please refer to the contest website (http://www.ece.cmu.edu/~jhoe/mc09.html) for more background, news and updates. It behooves you to examine the published 2007 and 2008 contest entries to get ideas and to harvest useful IPs.