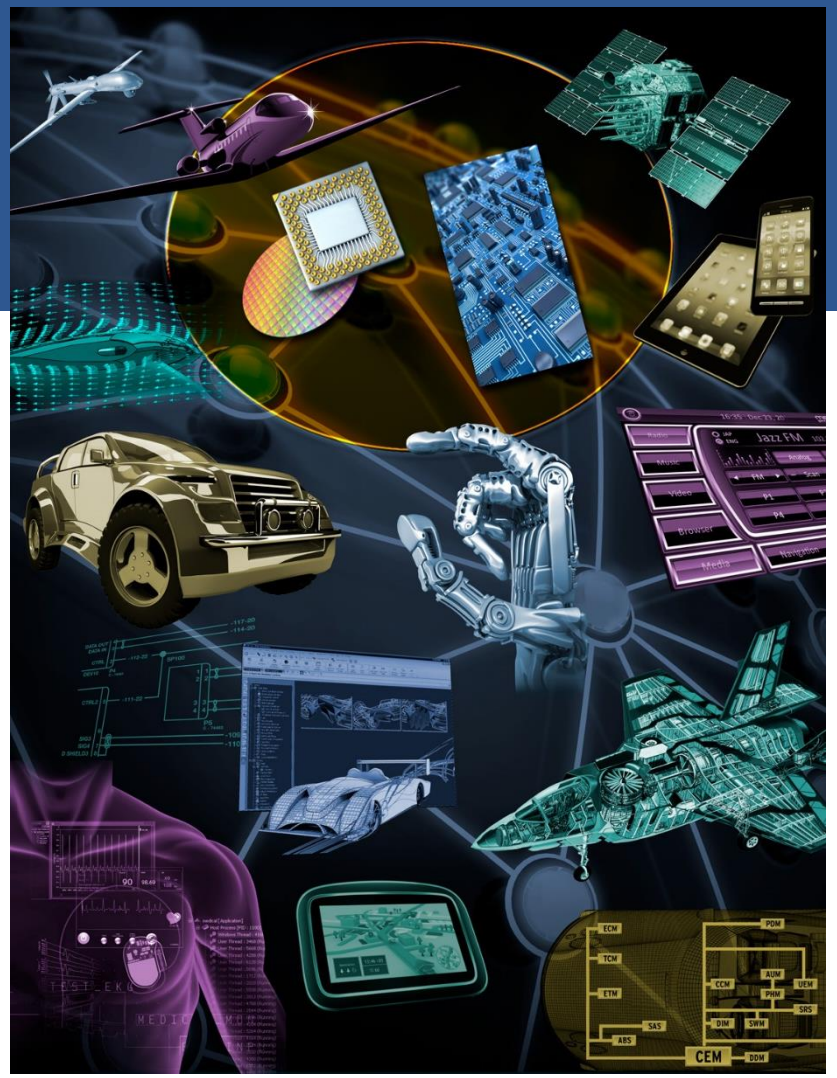

Preparing for Synthesis

Mentor®
A Siemens Business

Writing Synthesizable Code

- Some input language constructs not supported
  - Dynamic memory allocation
    - Physical hardware resources have finite amount of storage
  - Unions
    - Could be synthesized but could easily create bad logic
  - "float" and "double" native types
    - Often not desired or efficient HW but can be modeled using ac_float, user-defined implementation, etc.
    - Otherwise convert to ac_fixed or ac_int
  - Recursion with unfixed terminal depth
    - Hardware resources are finite
    - Template recursion is OK
  - Pointers to arrays mapped to memories on interfaces
  - Global variables shared between design blocks
- Globals for static const/ROMs are Ok

Converting Float and Double to Bit-Accurate Data-types

Double/float not synthesizable

```
double dx_sq, dy_sq, sum;
for (int i = 0; i < 128; i++) {
    for (int j = 0; j < 64; j++) {
        dx_sq = *(dx + i * 64 + j) * *(dx + i * 64 + j);
        dy_sq = *(dy + i * 64 + j) * *(dy + i * 64 + j);
        sum = dx_sq + dy_sq;
```

Use bit-accurate integer, fixed-point and ac_float

```
uint18 dx_sq, dy_sq;
ac_fixed<19,19,false> sum;//fixed point integer for sqrt
ac_fixed<8,3>at;
ac_fixed<16,9,false> sq_rt;

MROW:for (int i = 0; i < 128; i++) {
    MCOL:for (int j = 0; j < 64; j++) {
        dx_sq = dx[i][j] * dx[i][j];
        dy_sq = dy[i][j] * dy[i][j];
        sum = dx_sq + dy_sq;
```

Bound Array Declarations and Formals

- Catapult needs to know array bounds for internal arrays and design block interface variables
  - Internal pointers are ok to move data around

typedef ac_int<8,false> uint8;

```
void HogAlg(unsigned char *dat_in,
        double *magn,
        double *angle){
    double *dx, *dy;

    dy = (double*)malloc(128*64*sizeof(double));
    dx = (double*)malloc(128*64*sizeof(double));

    VerticalGradient(dy, dat_in);
    HorizintalGradient(dx, dat_in);
    MagnitudeAngle(dx, dy, magn, angle);

}
```

Pointer to array on HW interface not synthesizable

dynamic memory allocation not synthesizable

Bound arrays on HW interface

```
void CCS_BLOCK(HogSynthesizable)(uint8 dat_in[128][64],
        uint9 magn[128][64],
        ac_fixed<8,3>  angle[128][64]){

    int9 dx[128][64], dy[128][64];

    VerticalGradient(dy, dat_in);
    HorizontalGradient(dx, dat_in);
    MagnitudeAngle(dx, dy, magn, angle);

}
```

Bound internal array declarations

Using HLS Math Libraries

- math.h functions are not synthesizable
- Catapult math library implements most commonly used math functions for fixed point an integer types
  - Div, sqrt, atan2, sin/cos, etc
  - Implemented using piecewise linear, CORDIC and recursive algorithms
  - See User docs and toolkits
- Include path
  - <Catapult Install Tree>/Mgc_home/shared/include
- #include <math/mgc_ac_math.h>
- #include <ac_math.h>

```
for (int j = 0; j < 64; j++) {
    dx_sq = *(dx + i * 64 + j) * *(dx + i * 64 + j);
    dy_sq = *(dy + i * 64 + j) * *(dy + i * 64 + j);
    sum = dx_sq + dy_sq;
    *(magn + i * 64 + j) = sqrt(sum);
    *(angle + i * 64 + j) = atan2(dy[i * 64 + j], dx[i * 64 + j]);
}
```

Math.h not directly supported

```
MCOL:for (int j = 0; j < 64; j++) {
    dx_sq = dx[i][j] * dx[i][j];
    dy_sq = dy[i][j] * dy[i][j];
    sum = dx_sq + dy_sq;
    //Catapult's math library implementation of sqrt and atan2
    sqrt(sum, sq_rt);
    atan2((ac_fixed<9,9>) (dy), (ac_fixed<9,9>) (dx), at);
    magn[i][j] = sq_rt.to_uint();
    angle[i][j] = at;
}
```

Use Catapult's mgc_ac_math.h and ac_math.h implementations

Checking the Code for Bugs

- C based languages have a certain amount of ambiguity and errors can easily create bad RTL
  - Make sure that the code is “clean”
- Initialize variables before their read
  - Uninitialized Memory Reads (UMR) can be very unpredictable. Simulation behavior varies between compilers and platforms
  - Synthesis tool may optimize away variables that are UMR
- Make sure that there are no out-of-bounds array reads/writes (ABW, ABR)
  - Simulation may pass/fail intermittently
  - This is bad logic in hardware
- Linting and Property Checking Tools
  - Valgrind
  - Purify
  - Catapult Design Checks
    - can find most user-errors like these at the push of a button

Disabling code

- Debug code and status messages need to be eliminated from Synthesis Code
  - High level behavioral description is often used for architecture exploration which usually have this kind of information included
  - Can make the code non-synthesizable
  - May result in unnecessary hardware
- Unsynthesizable code can be excluded from synthesis using a compiler pre-processor definition (Catapult solution)

```
#ifndef __SYNTHESIS__
    printf (...)
    cout << ...
#endif
```

A Siemens Business

www.mentor.com