From Concept to Silicon (Algorithm to Hardware Design Technology)

Vason P. Srini

April 2007

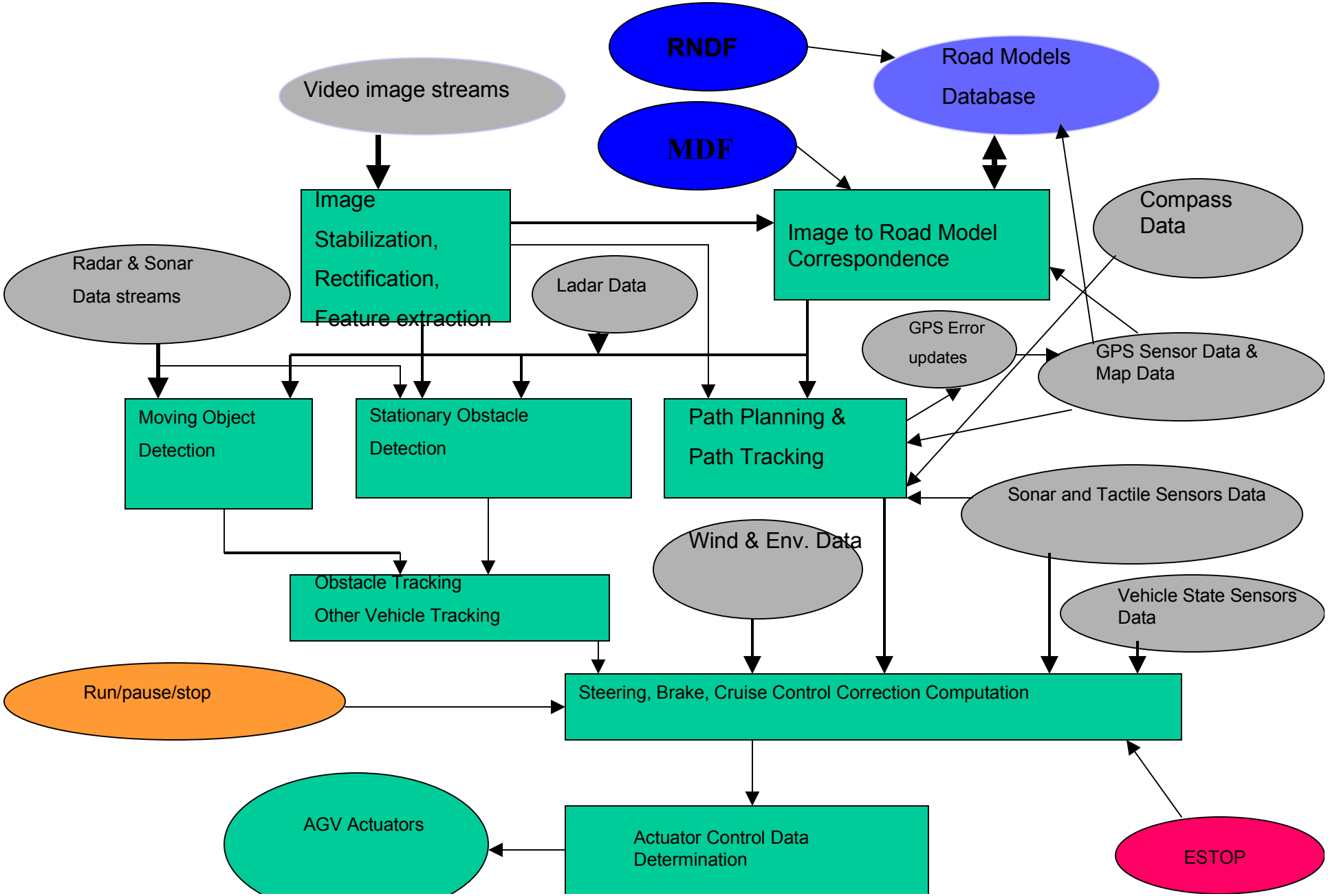
srini@eecs.berkeley.edu

UCB CHESS Seminar

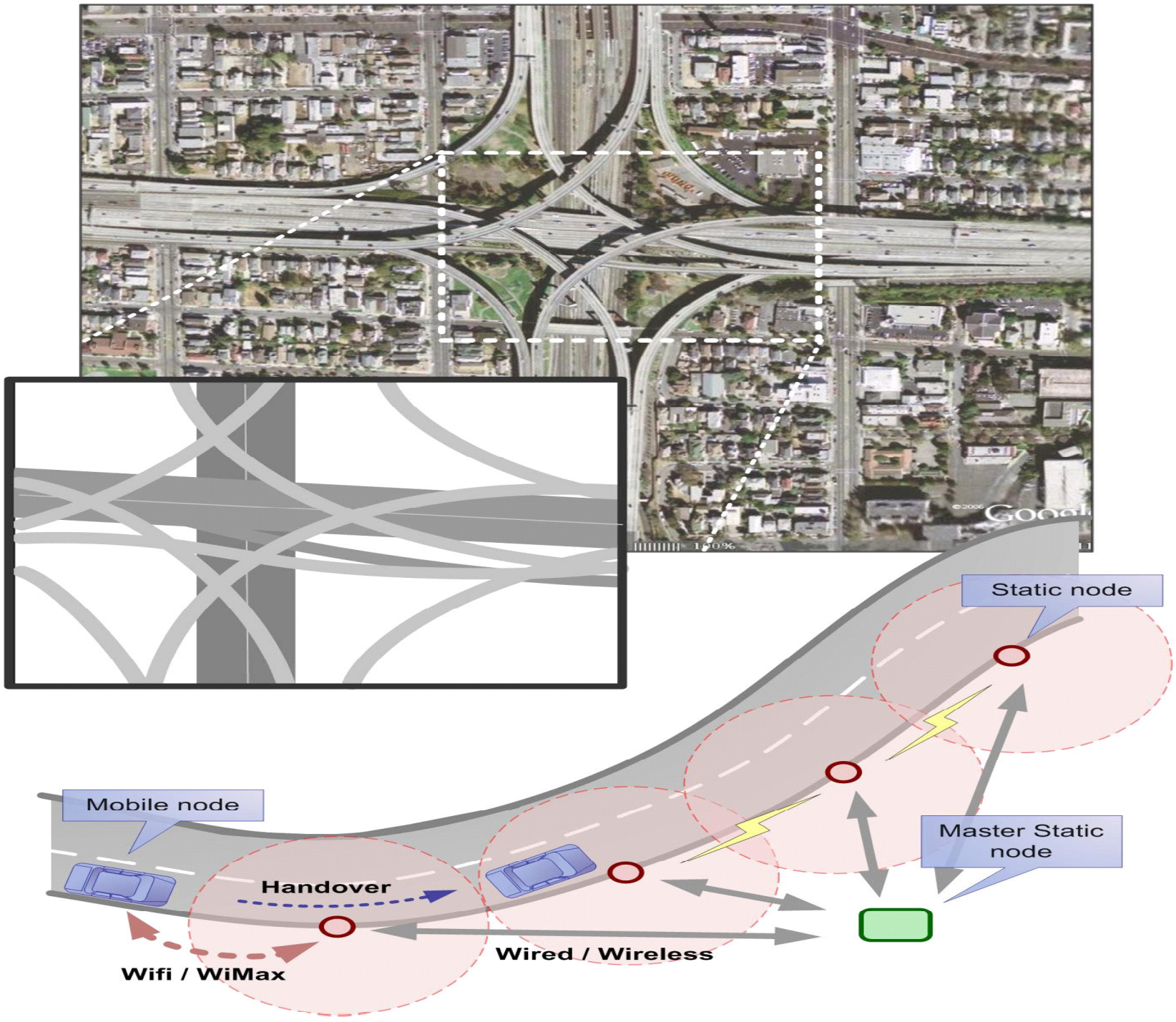
Motivation

- Computing support for the Berkeley Sydney Driving Team participating in DGC3
- Auto assist for future cars and transportaion systems
- Support collision avoidance and adaptive cruise control (ACC) in cars, buses, and trucks.
- Support the development of advanced sensors such as 3D Flash Ladar

Real-time Integrated Sensors and GN&C Processing



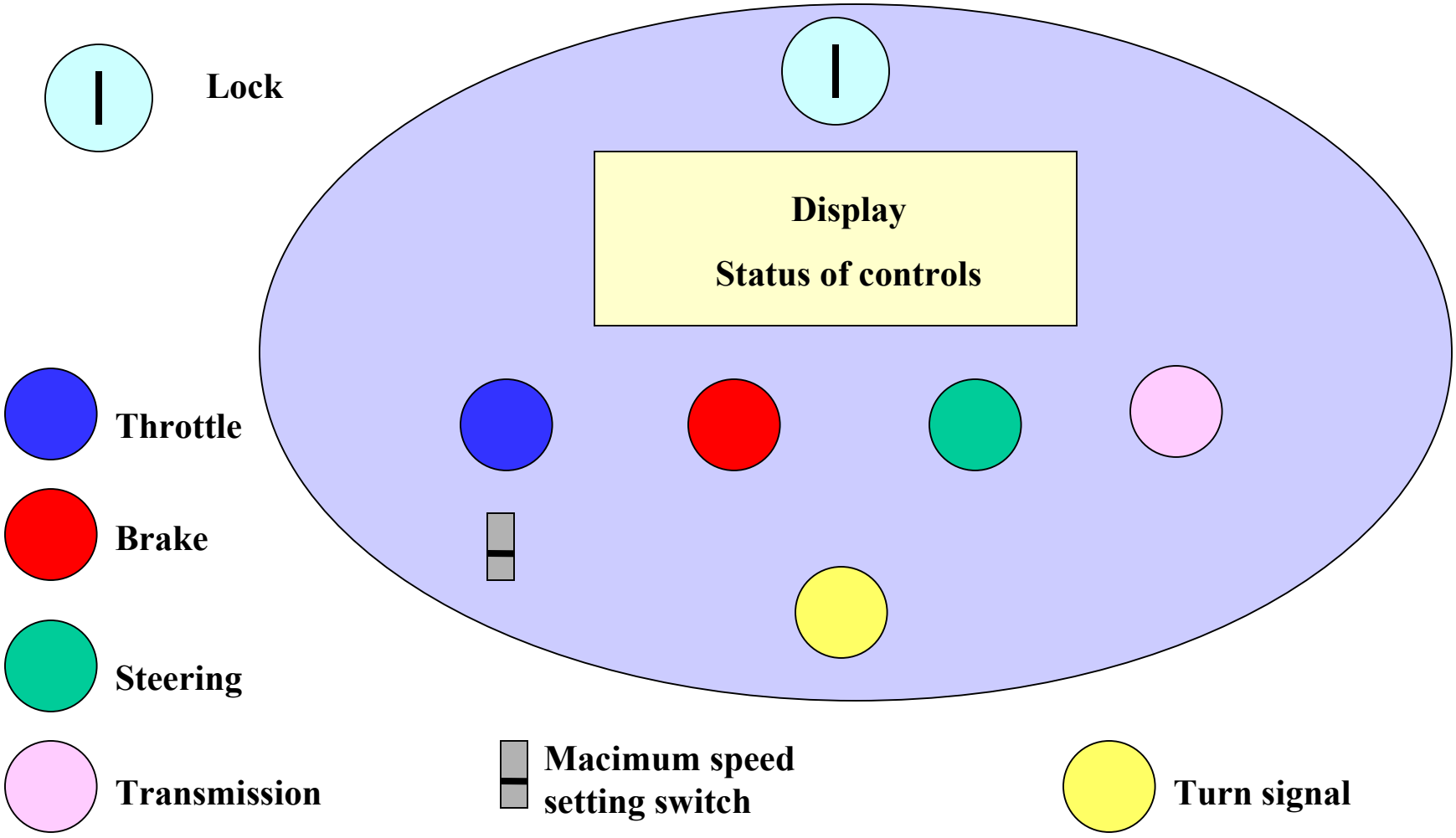

270
495
267
29
Snow
blockage
295
Start
From
Maryland
Destination
1
201
Congestion
washington dc
66
110
50
395
Traffic accident
95
Image © 2006 TerraMetrics
Image © 2006 Sanborn


Static node
Mobile node
Handover
Master Static node
Wired / Wireless
Wifi / WiMax

2007 Lexus LS 460L

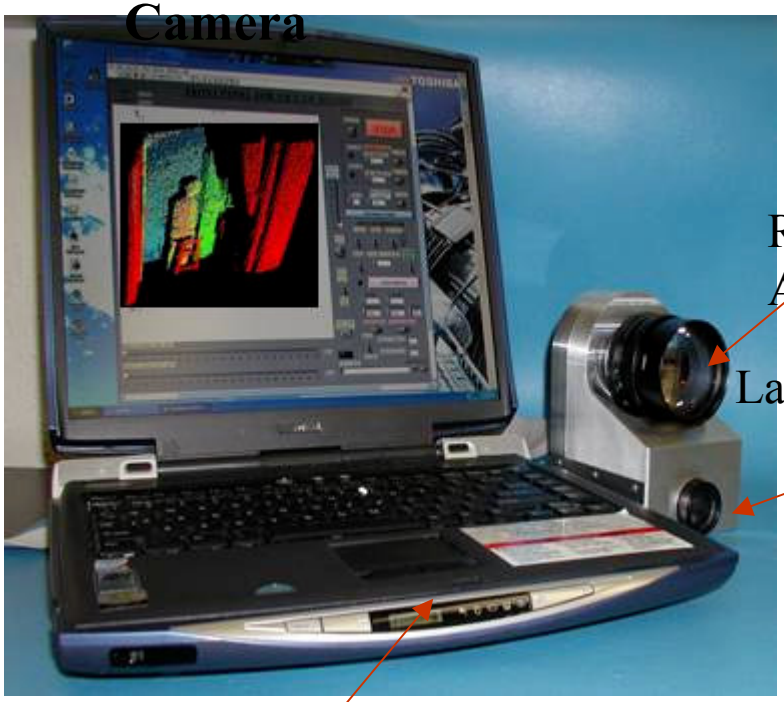

Map View
Work
66
12

Auto Control Panel



Advantages of 3-D Flash Ladar: Size, Weight and Use is Consistent with Ordinary 2-D Camera

Hand Held .5 Hz 3-D Camera



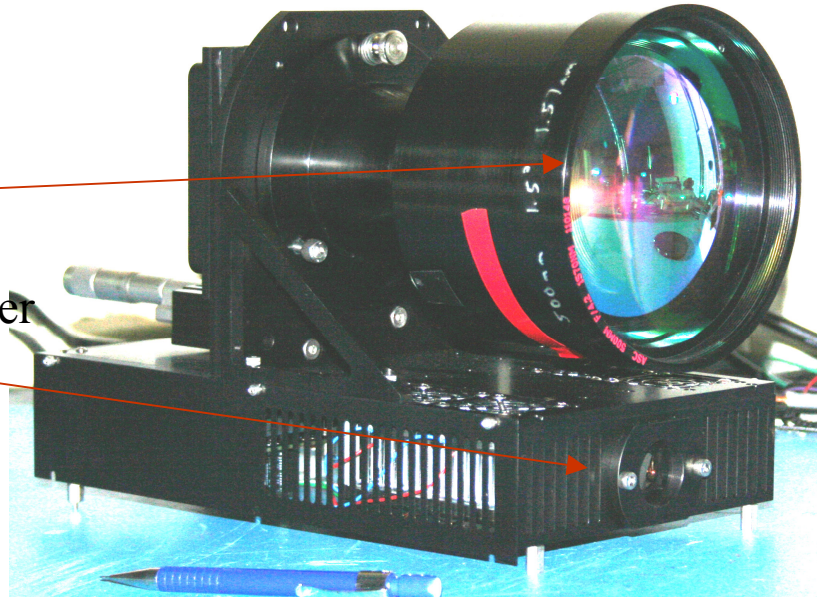
Receiver Aperture

Laser Transmitter Aperture

Laptop Processes and Displays 3-D Images

Controls Camera Functions

Hand Held 30 Hz 3-D "Video" Camera

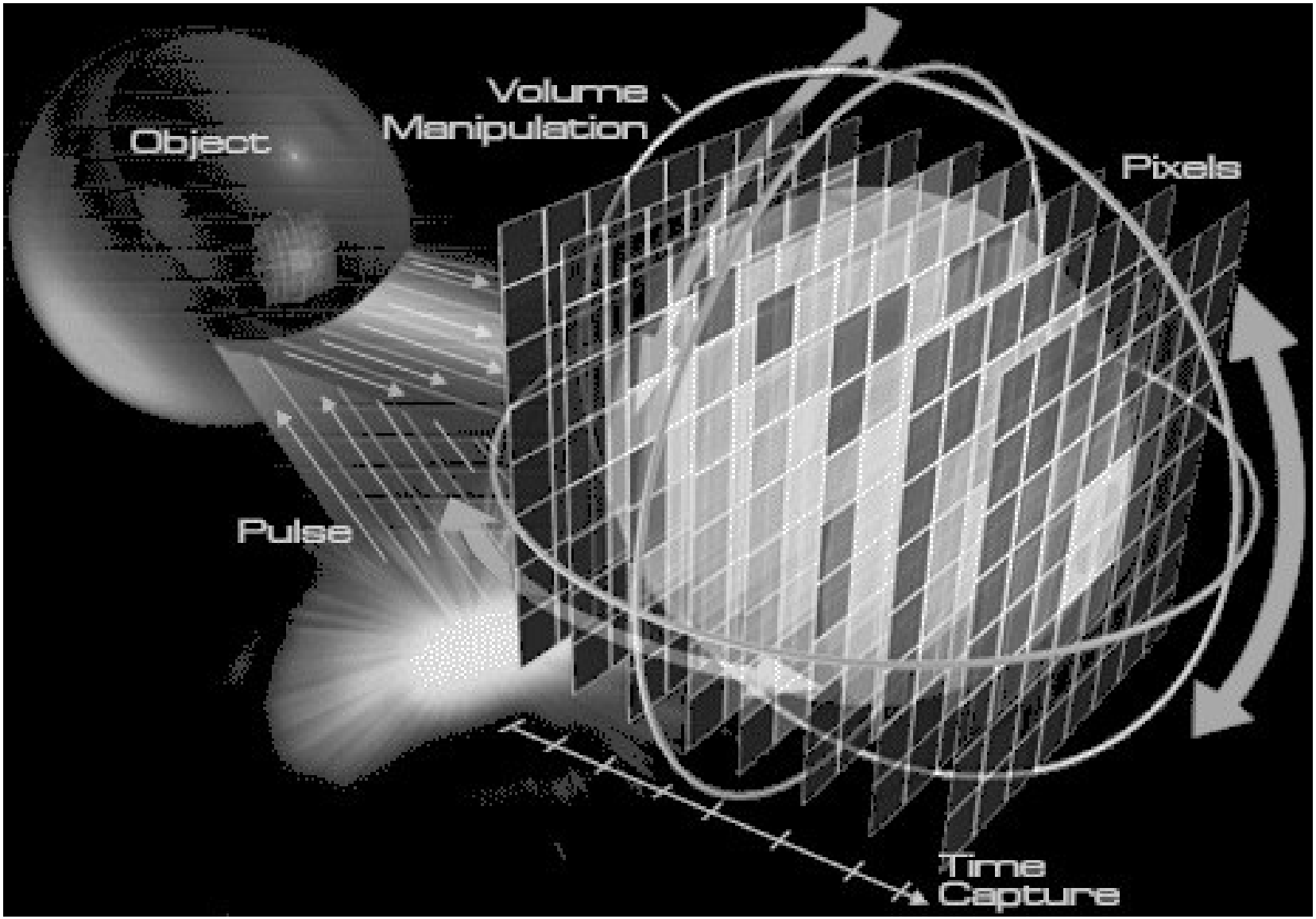


Modular Lens Substitution

Kilometer Range

Conduction Cooled

3D Flash Ladar



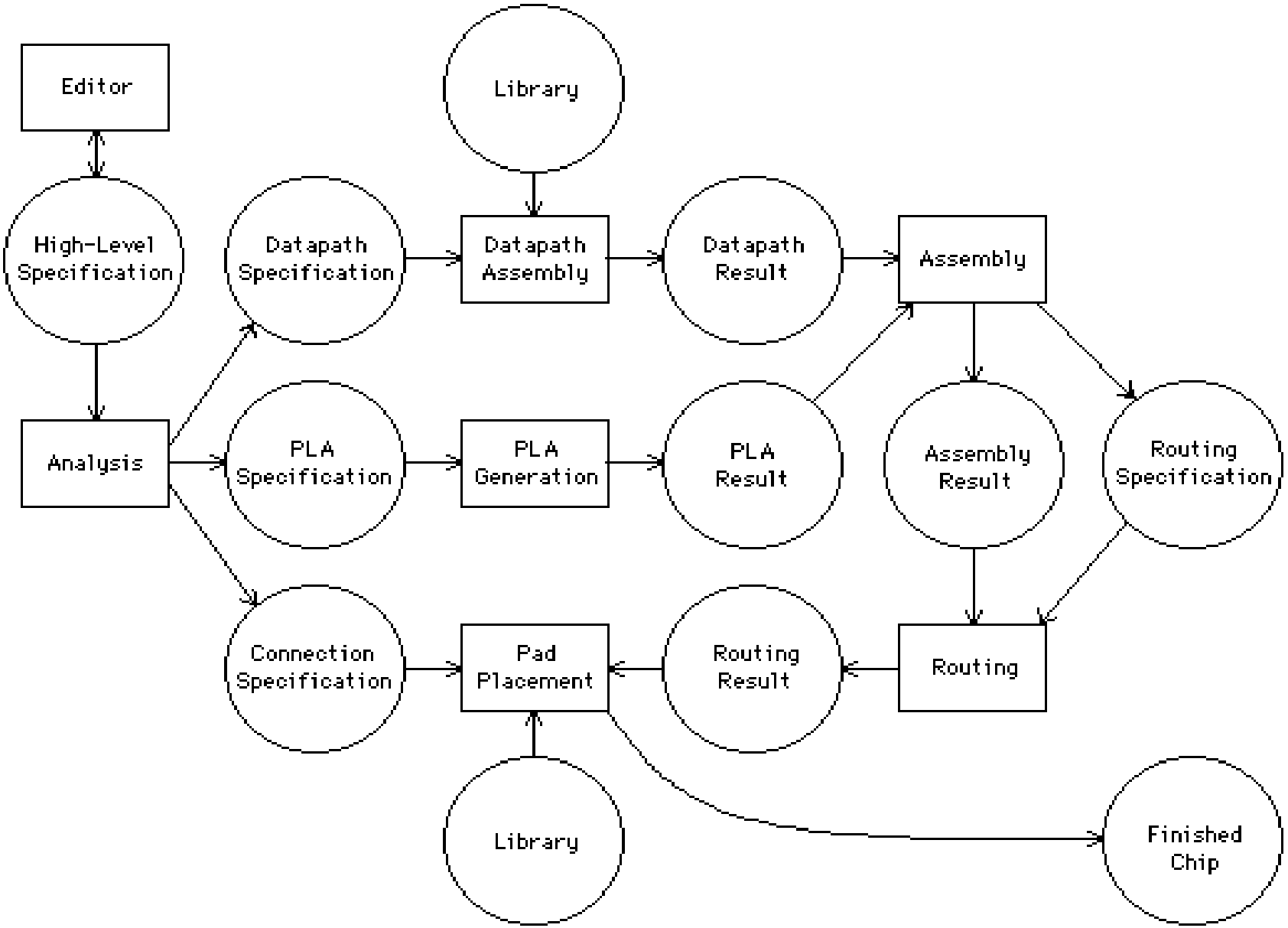
Outline

- History
- Electronic system design (ESD) process
- Register transfer (RT) based approach
- Component based approach
- System design using libraries and Simulink
- Custom (proprietary) algorithms and rapid hardware mapping using library of components
- Multiple vehicle tracking in dynamic environments

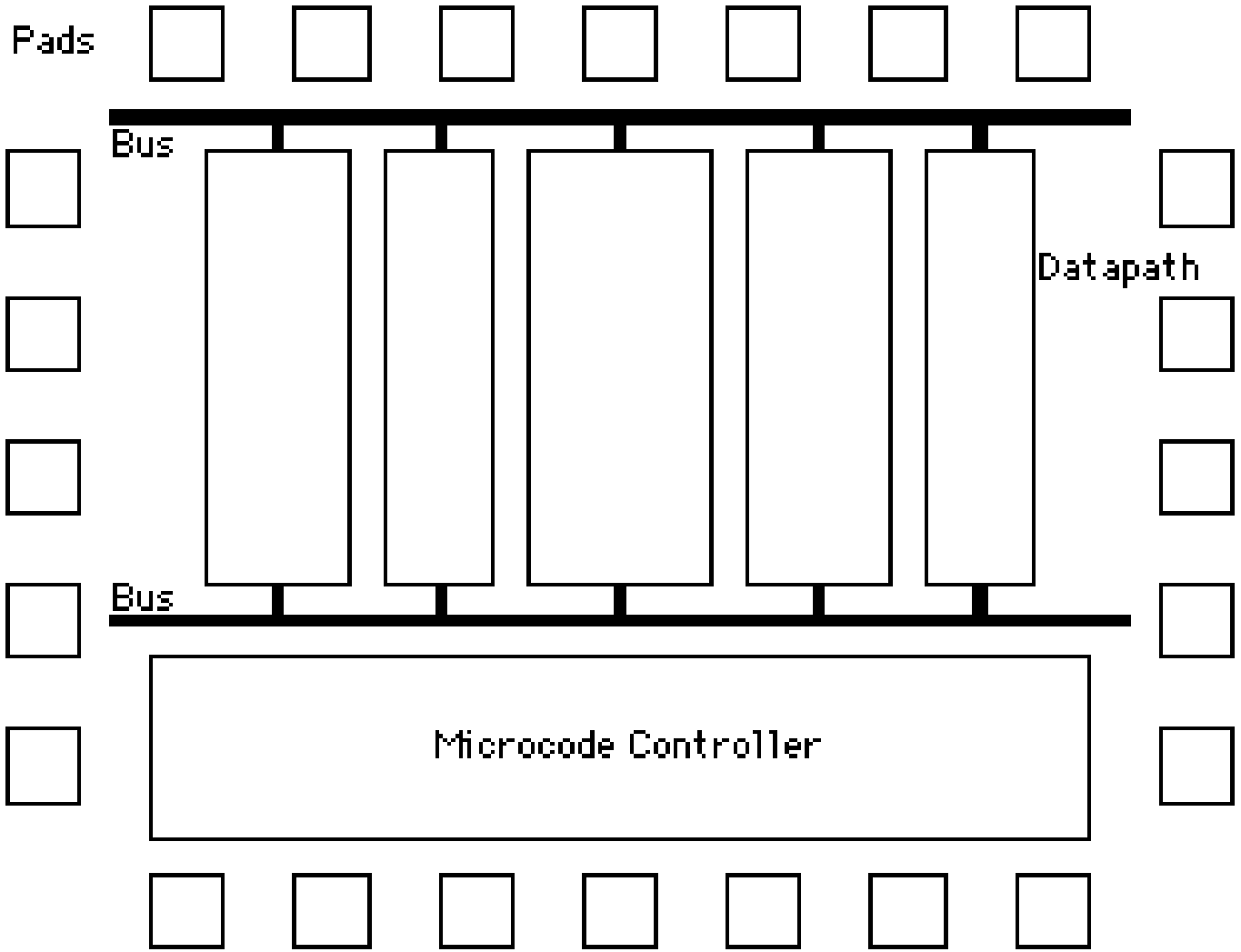
History

- MacPitts/Silc - Lisp based (simple DSP alg.) - MIT
- Bristle blocks - ISP description - Caltech
- Lager - Lisp based structural level description of DSP alg. - UCB
- Many attempts at silicon compilers (e.g. ASP - Advanced silicon compiler in Prolog)
- C/CatapultC/SpecC/ to hardware
- Ptolemy to C and to DSP hardware
- Matlab to hardware
- Simulink to hardware

Silicon Compiler/Assembler



Bristle Blocks Architecture

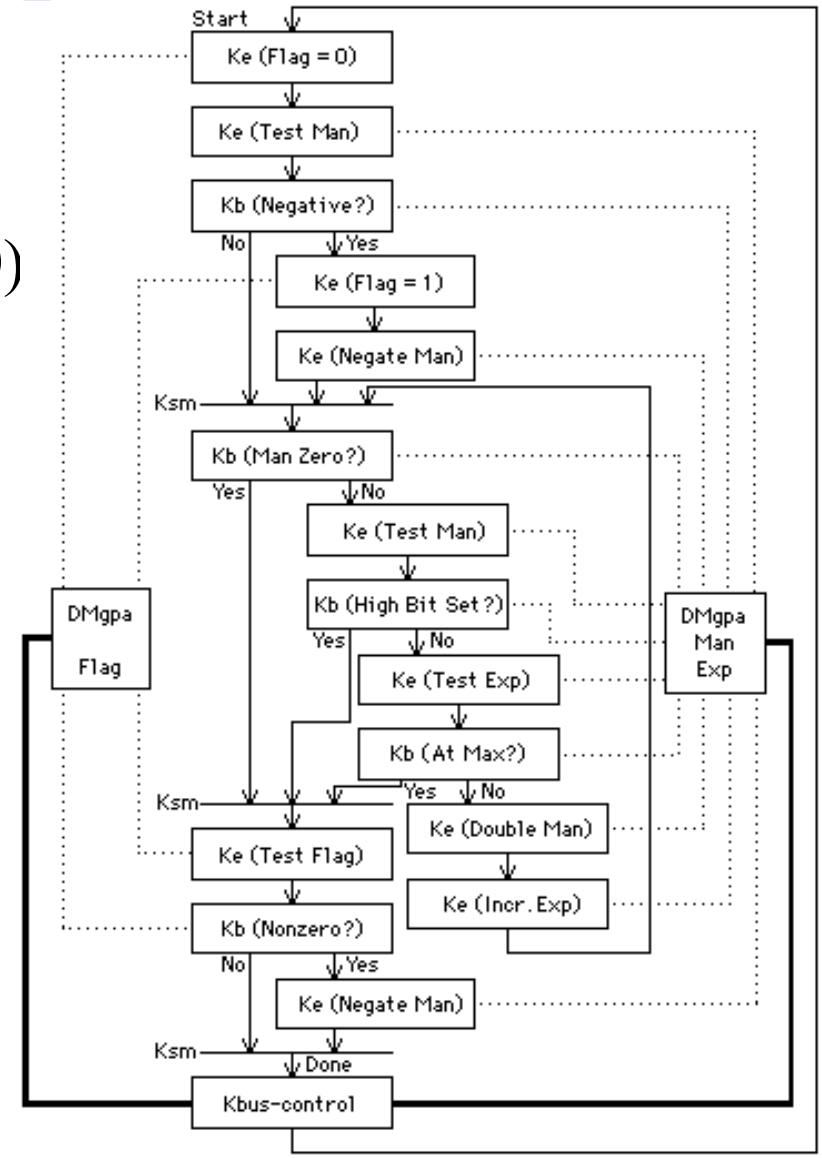


ISP Description

OP := M[PC]<0:2>

(OP = 1) (M[OPERAND] AC; AC 0)

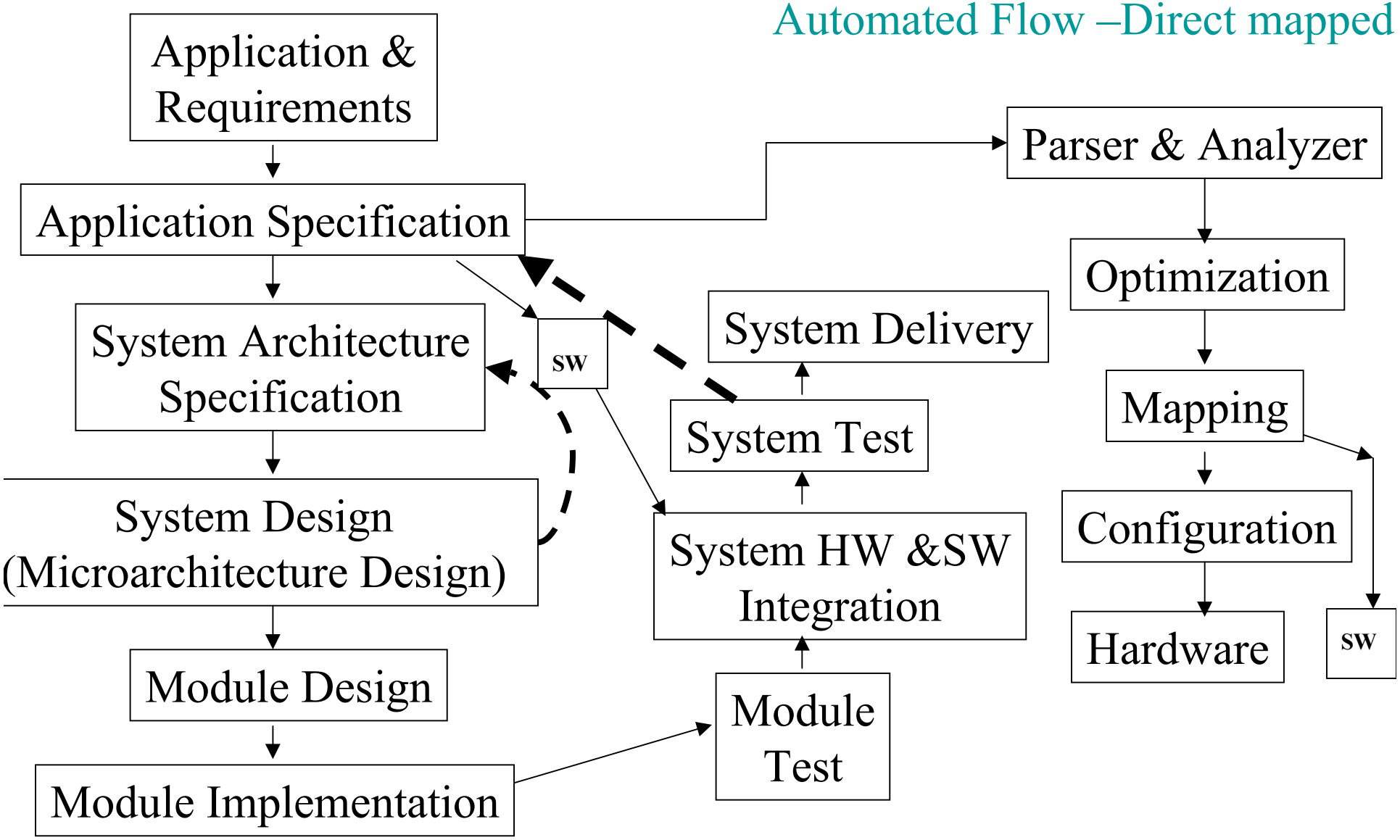
(AC AC + T; next M[D] AC)



Register-transfer layout to normalize a floating-point number.

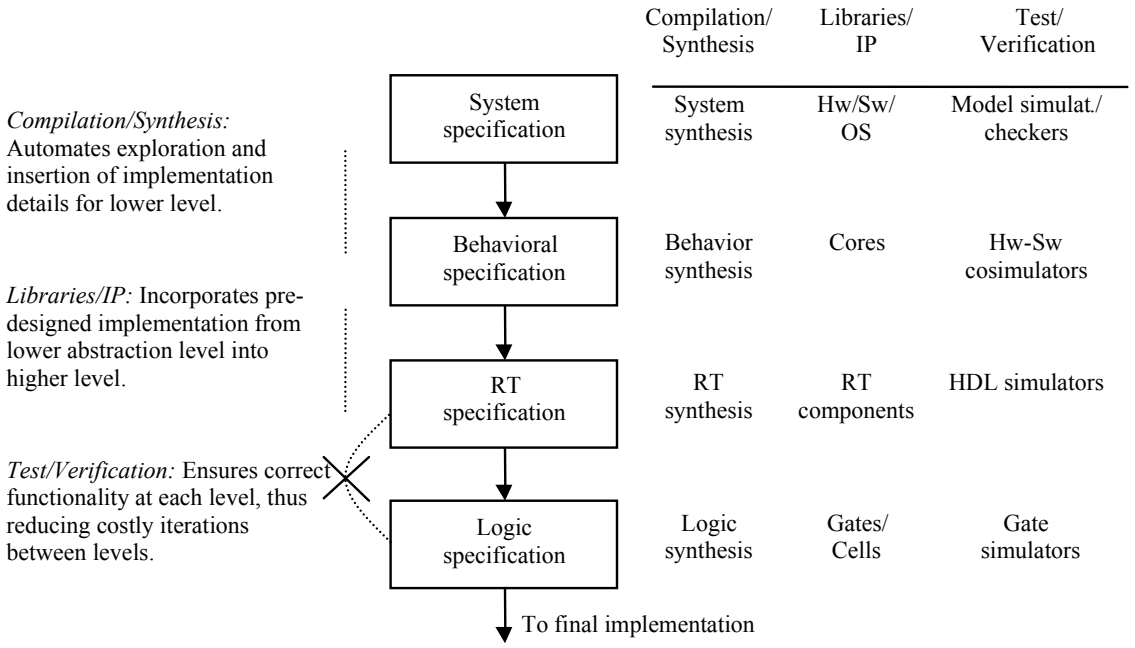
ESD Process

Conventional Flow

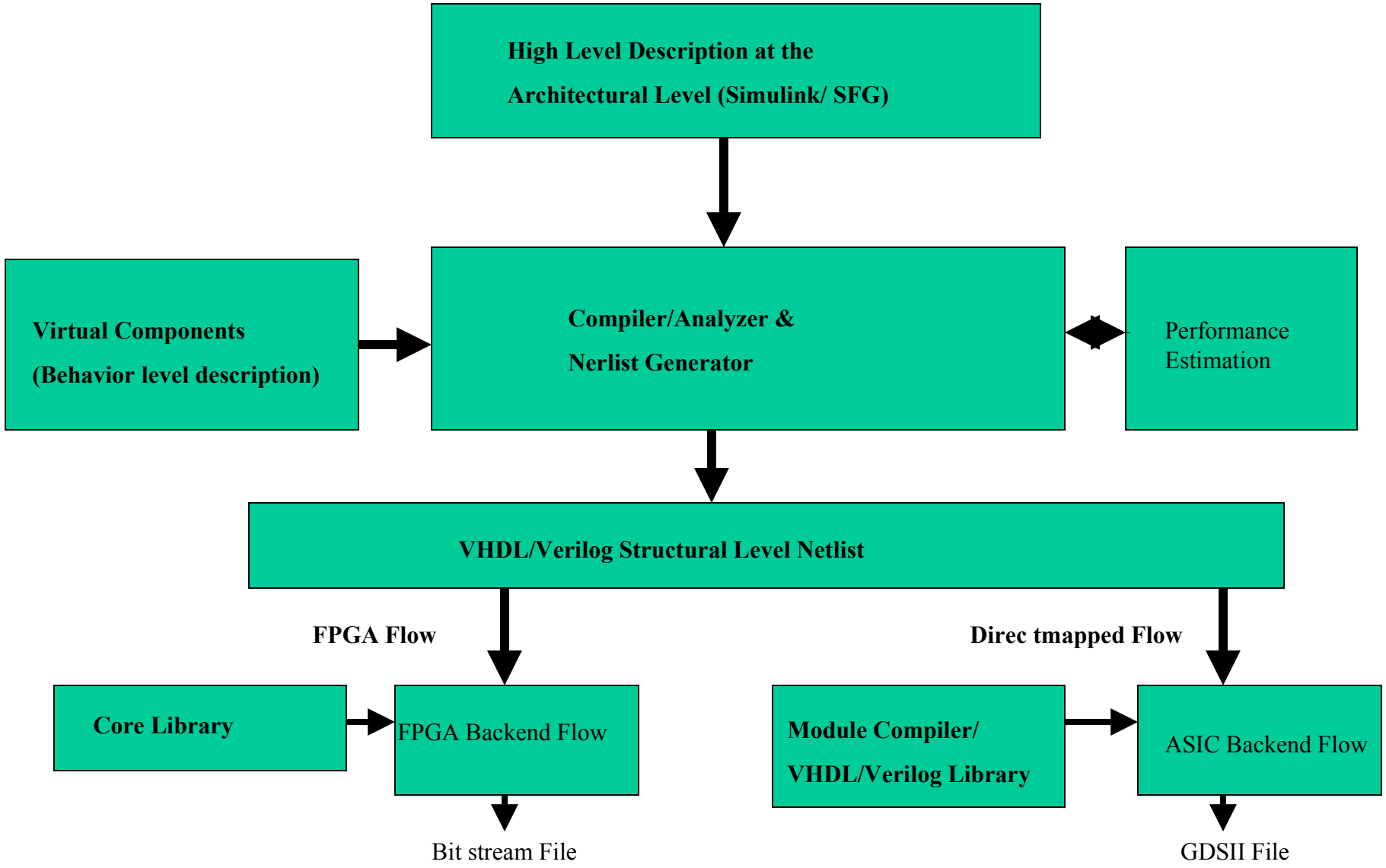


HW Design Technology - RT based Approach

The manner in which we convert our concept of desired system functionality into an implementation

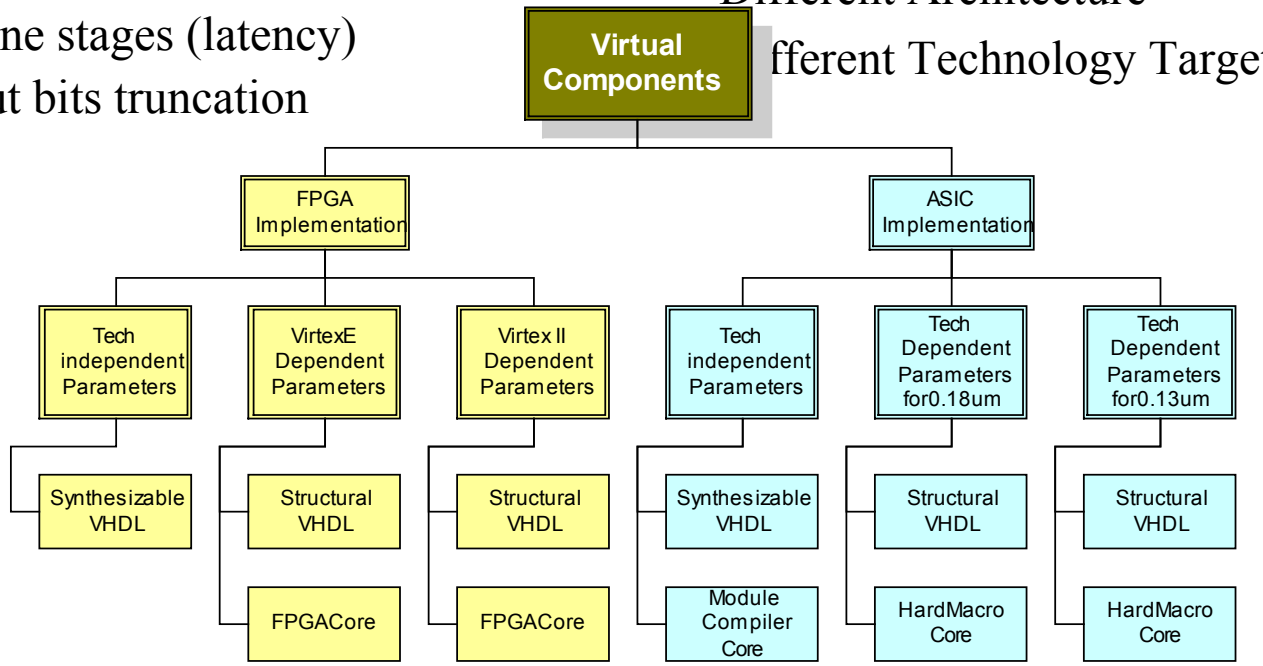


HW Design Technology - Component based Approach

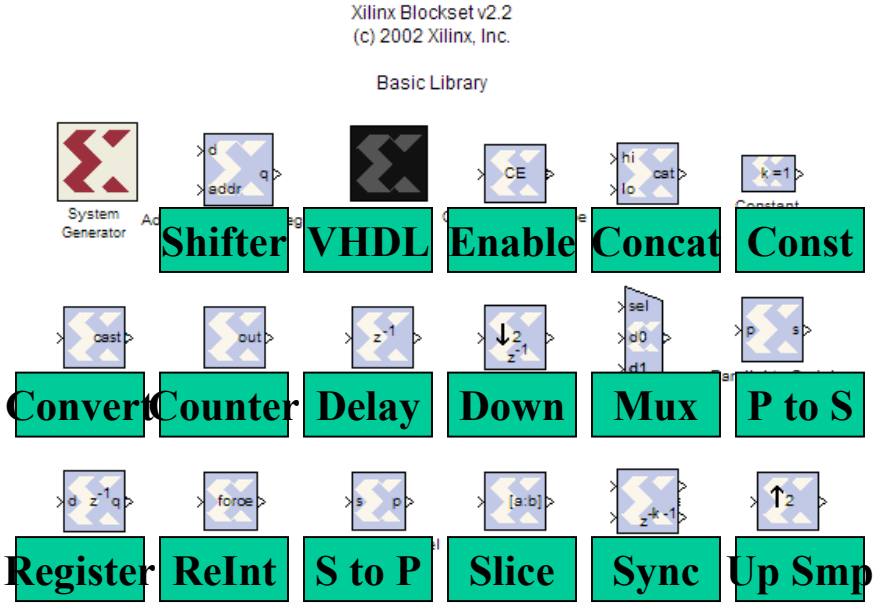


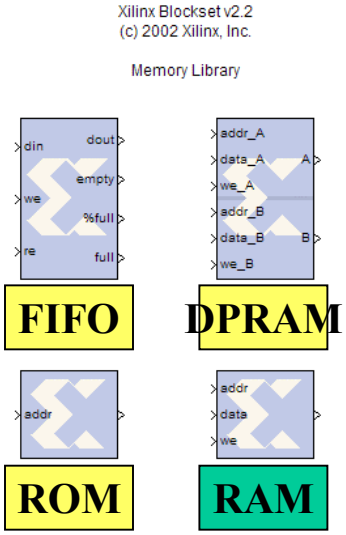
Virtual Component Library

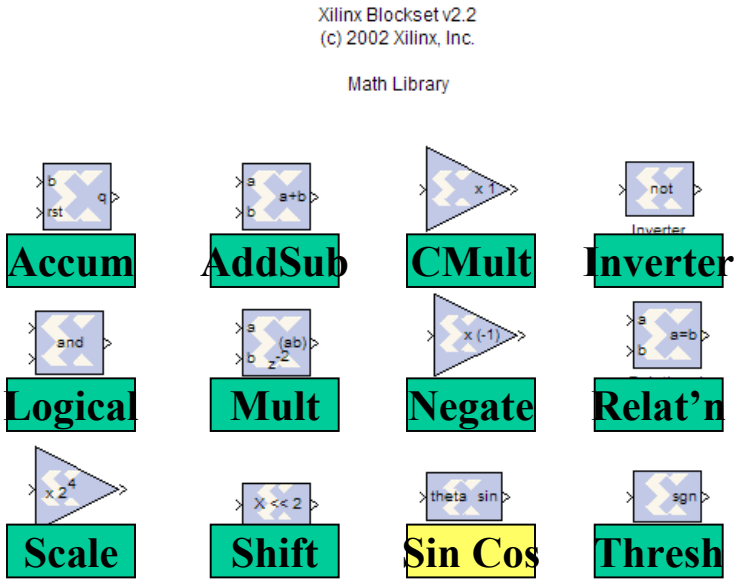
- Parameterized system level blocks
  - Bit-width
  - Pipeline stages (latency)
  - Output bits truncation
- Customizable block set library
  - Different Architecture
  - fferent Technology Target



Basic Blocks







FPGA+ASIC Support

FPGA Support Only

Goals

- Develop high level design technologies to facilitate the mapping of application programs to high performance and low-power embeddable hardware and software.
- Need domain specific component libraries.
- Require an unified approach to HW and SW design with late binding to HW and SW.
- Need a program analyzer to detect parallelism and data types present in programs.

Specification

- Constructive specification of software using Matlab/Java / C++.
- Matlab/Simulink or PtolemyII models for system level and architectural level and simulation.
- System C, Superlog (verilog & C), C++ specification for systems.
- Rosetta, UML, and Statecharts

Susan Edge Detector - Matlab

```
name1 = 'input_image_';
format = '.JPG';
for i = 1: num_images
input = imread(strcat(name1, int2str(i), format));
% check to see if the image is a color image...
d = length(size(input));
if d==3
    image=double(rgb2gray(input));
elseif d==2
    image=double(input);
end
result = susan_edge_detector(image, threshold);
imshow(result);
end
```

```
function image_out = susan_edge_detector(image,threshold)
close all
clc
% mask for selecting the pixels within the circular region (37 pixels, as
% used in the SUSAN algorithm
mask = ([ 0 0 1 1 1 0 0 ;0 1 1 1 1 1 0;1 1 1 1 1 1 1;1 1 1 1 1 1 1;
          1 1 1 1 1 1 1;0 1 1 1 1 1 0;0 0 1 1 1 0 0]);
% the output image indicating found edges
R=zeros(size(image));
% define the USAN area
nmax = 3*37/4;
% padding the image
[a b]=size(image);
new=zeros(a+7,b+7);
[c d]=size(new);
new(4:c-4,4:d-4)=image;
```

```
for i=4:c-4

    for j=4:d-4
        current_image = new(i-3:i+3,j-3:j+3);
        current_masked_image = mask.*current_image;
        current_thresholded =
susan_threshold(current_masked_image,threshold);
        g=sum(current_thresholded(:));
 if nmax<g
            R(i,j) = g-nmax;
        else
            R(i,j) = 0;
        end
    end
end
image_out=R(4:c-4,4:d-4);
```

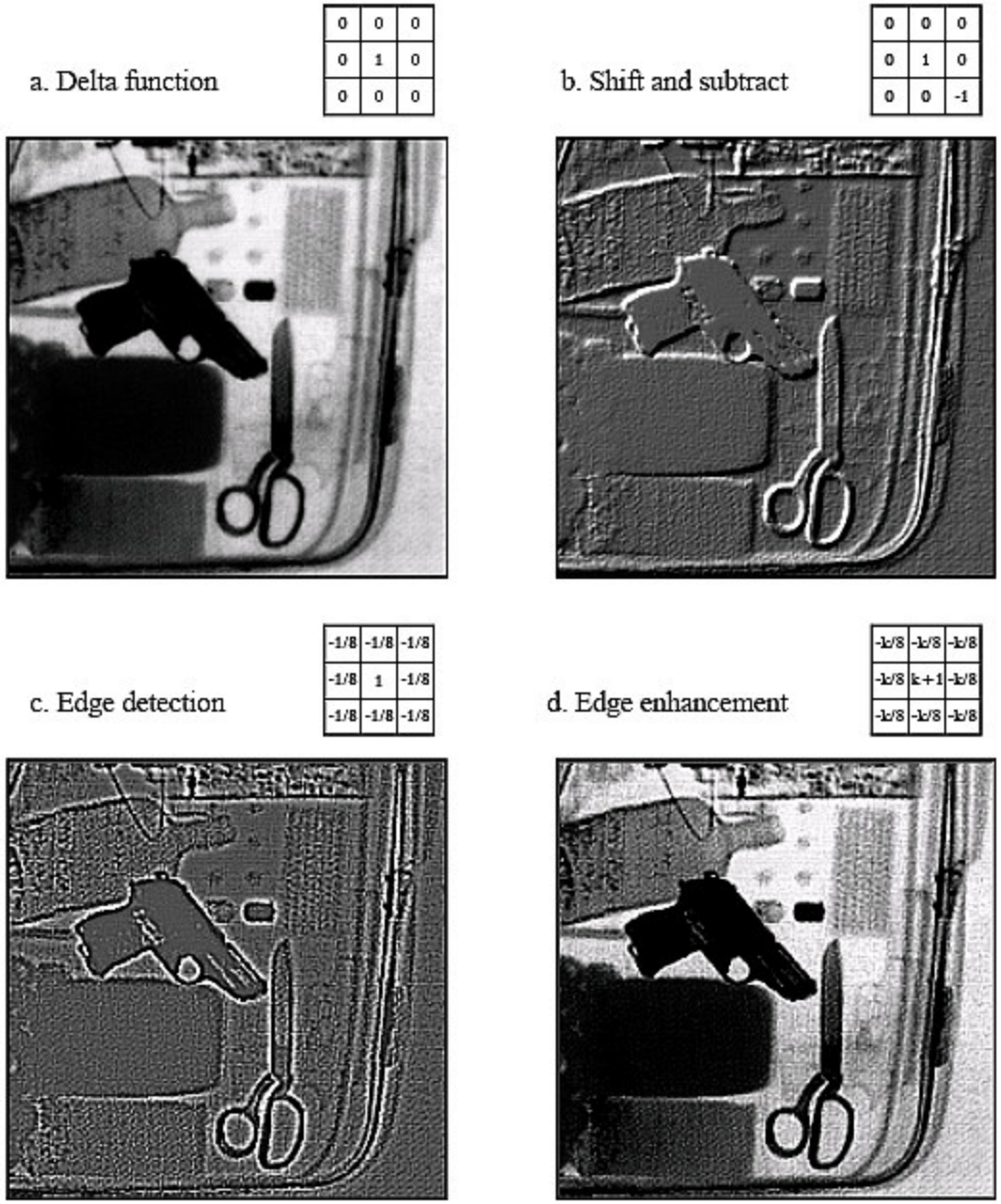


FIGURE 24-4
3×3 edge modification. The original image, (a), was acquired on an airport x-ray baggage scanner. The shift and subtract operation, shown in (b), results in a pseudo three-dimensional effect. The edge detection operator in (c) removes all contrast, leaving only the edge information. The edge enhancement filter, (d), adds various ratios of images (a) and (c),

Autonomous System Libraries

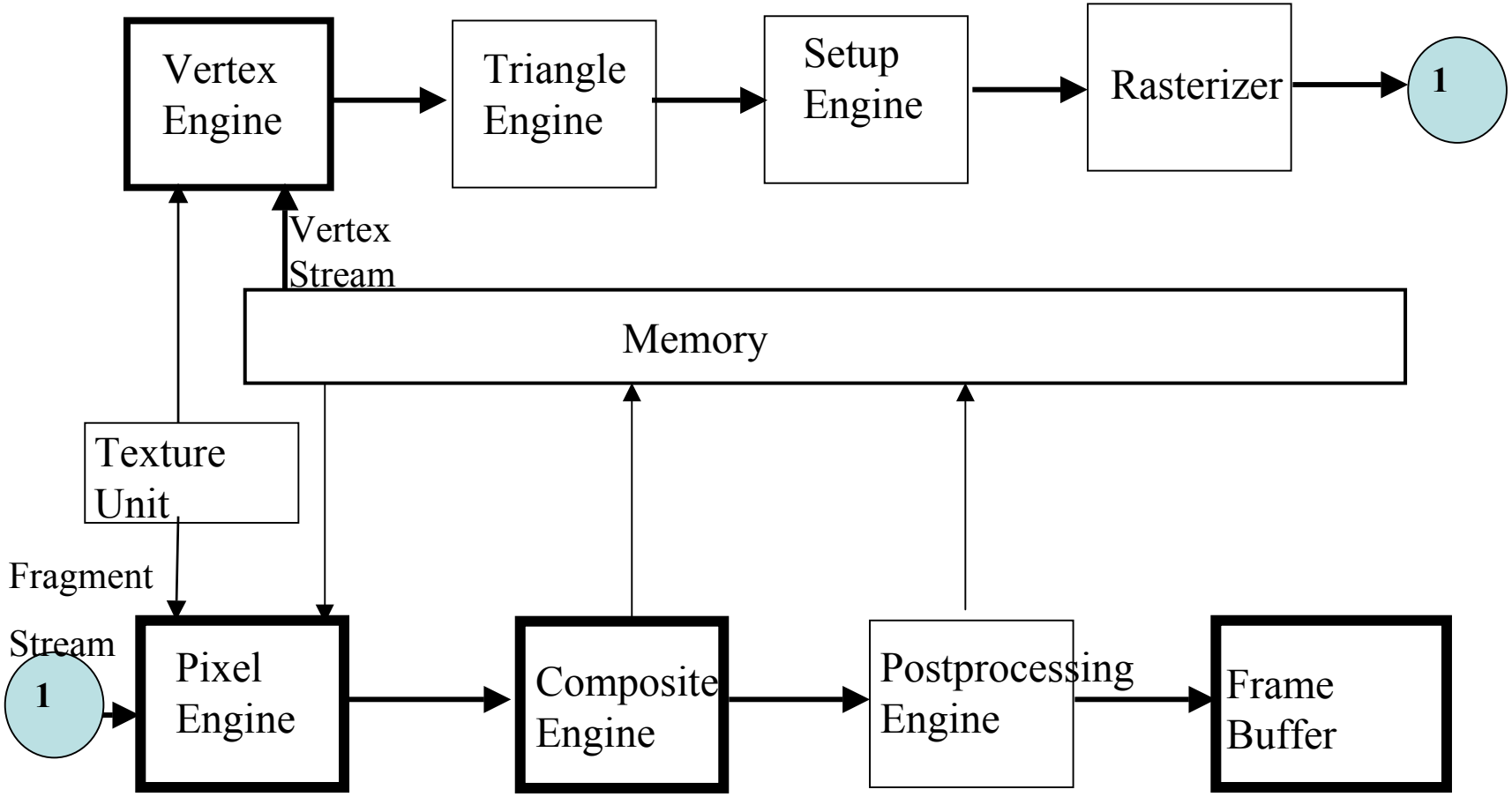
- Rebel - Recursive Bayesian library
- Kalman - Kalman filter library including extended and unscented Kalman filters
- Radar library
- Image processing library
- 3D visualization library
- Path planning library
- Model predictive control (MPC) library

3D- Graphics and Imaging Library

- Geometry engine - Floating point datapath, multiple units operating in a SIMD manner
- Geometry sequencer and distributed microinstruction memory
- Command engine for distributing data to geometry engines
- Raster engine - 25 to 40 pipeline stages with carefully tuned arithmetic units in each to do Z buffering, depth-cueing, alpha- blending, raster-ops, dithering, stenciling, clipping region checks, and special effects

- Raster manager to supply data and control the raster engine pipeline stages
- Image planes (24 bitplanes), depth planes (24 depth planes), stencil planes (1to 40 , overlay/underlay planes (4 bits), window clipping planes (4-bit planes)
- Video timing controller and display generator
- Multimode graphics processor
- Warp engine -3D image warping and image based rendering
- Volume rendereing – DVR using ray casting and 3D texture mapping

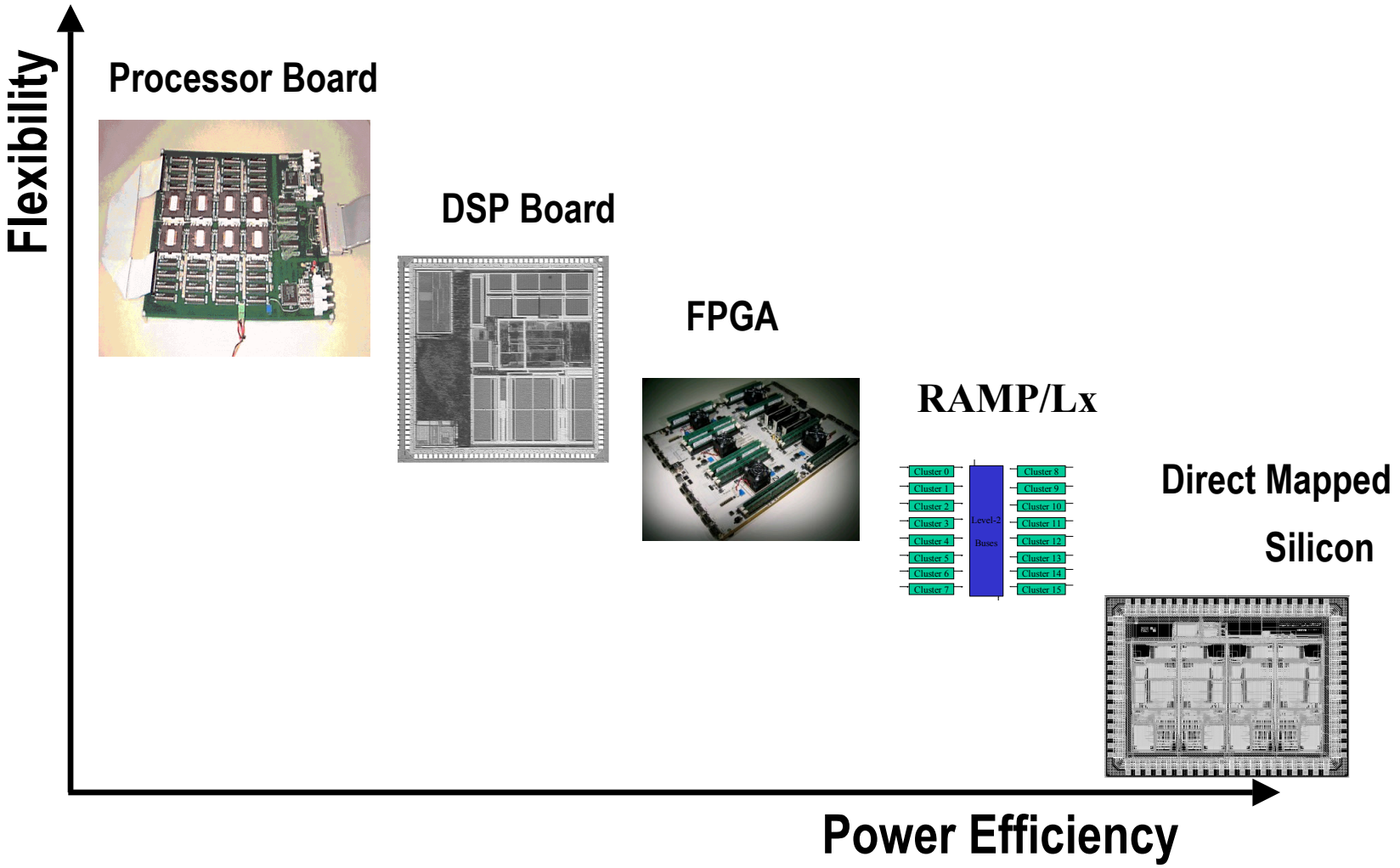
Stream Based VGVI Pipeline - Graphics



Mapping to System

- Boards containing microprocessors, memory, controller, FPGAs, ASICs, RTOS, compilers, debuggers, and IDE.
- DSP boards and supported OS and software.
- FPGA boards - BEE2, Xilinx ML310, Calinx - and related software.
- Reconfigurable clusters of processors and memory (RAMP) with related software.
- Direct mapping to silicon and custom software.

HW Direct Mapping Minimizes Power



10x to 100x Difference in Power

Direct Mapping for Wireless Algorithms

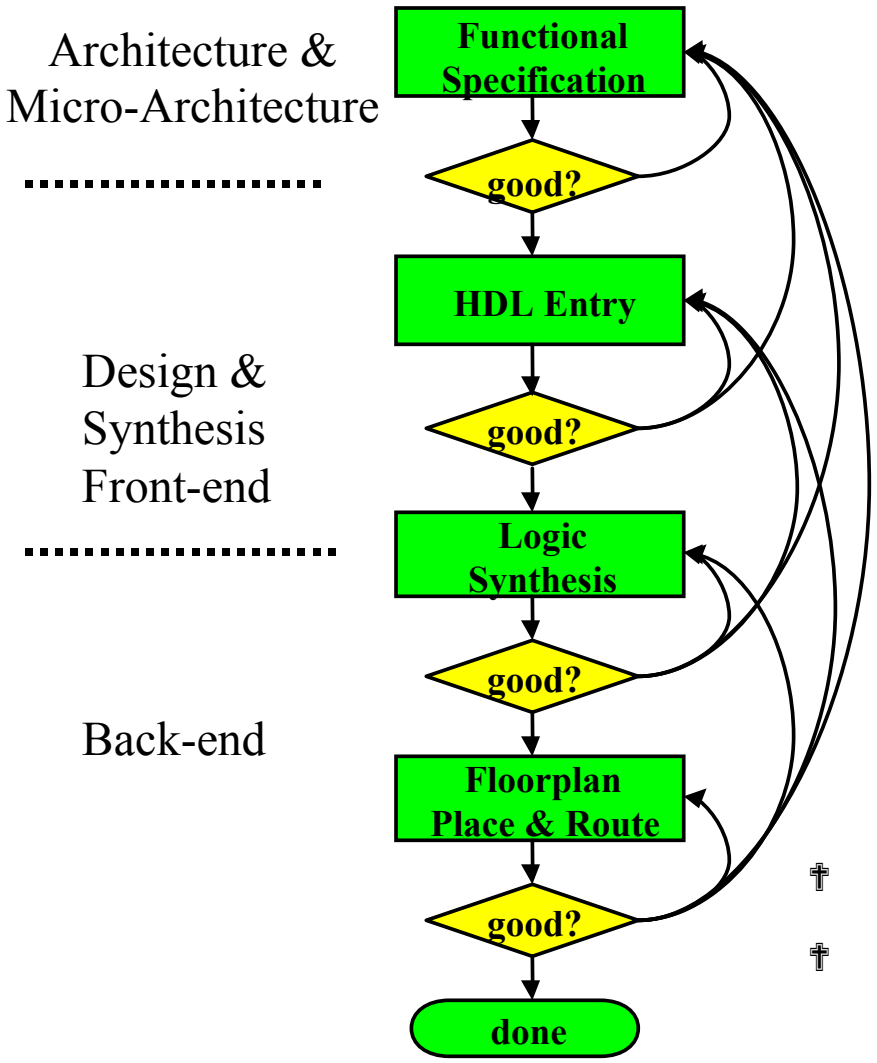
$$P \propto f \times C \times V_{DD}^2$$

Use low V_{DD} / parallelism

- Low processing rates (wireless baseband ~25 Msps)
- High Complexity
- Low Power

Example:	CDMA / adaptive MMSE	FDMA / multi-antenna
BW efficiency (b/s/Hz)	1.8	3.6
No. of Parallel DSP's	23	87
power (mW)	303	1149
Direct-mapped area (mm^2)	3	10
power (mW)	6.8	19

Typical ASIC Design Flow



Difficulties:

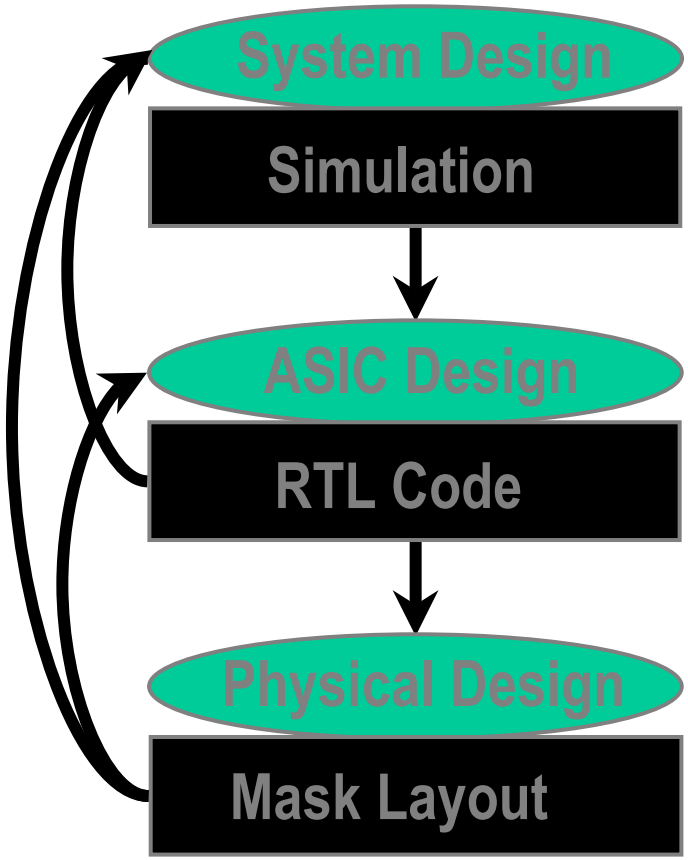
- Logic Verification
- Timing Closure
- Routing Congestion

Problem:

Indeterminate Design Time

- Design Decisions made at Every Step
- Unsolvable Problems Arise

Standard DSP-ASIC Design Flow

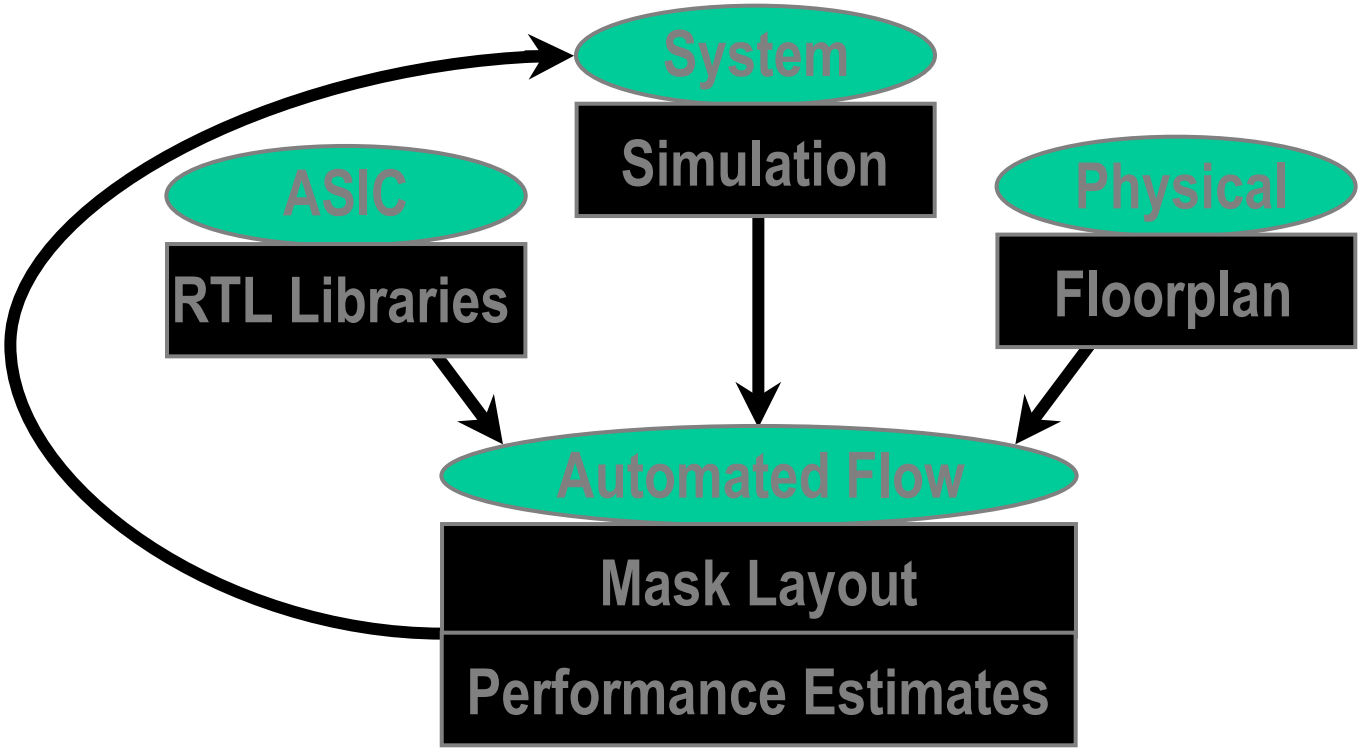


Problems:

- Separation of engineering teams makes exploration hard
- Uncontrolled looping when pipeline stalls
- Feedback to system designer is an aberration, but should be encouraged

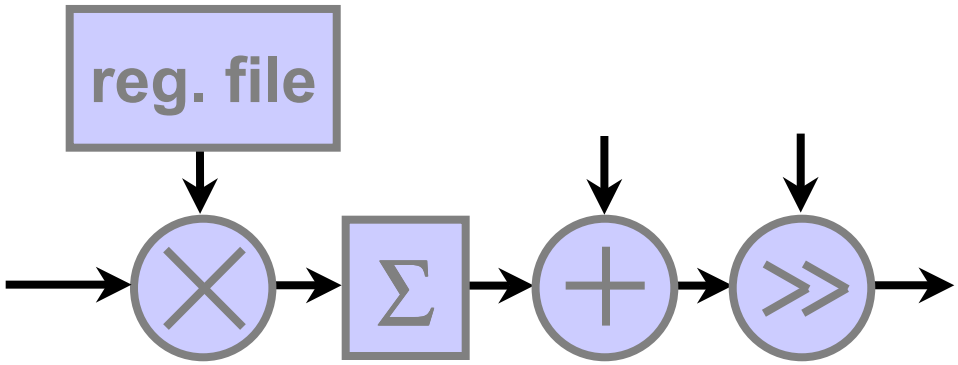
Prohibitively Long Design Time
for Direct Mapped Architectures

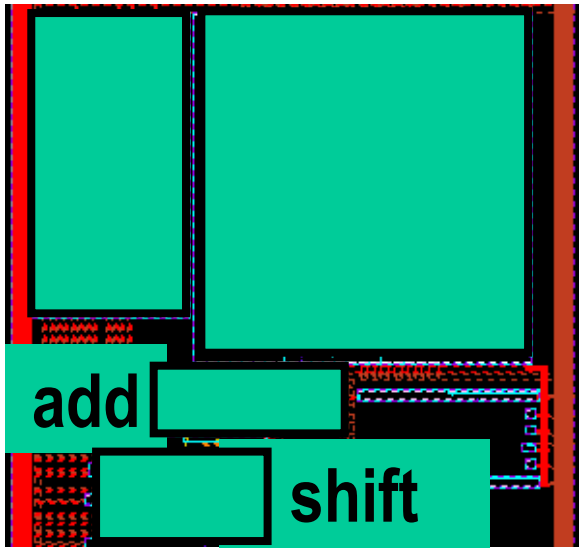
HW Direct Mapped Design Flow



- Encourages iterations of layout
- Controls looping
- Reduces the flow to a single phase
- Depends on fast automation

Capturing Design Decisions



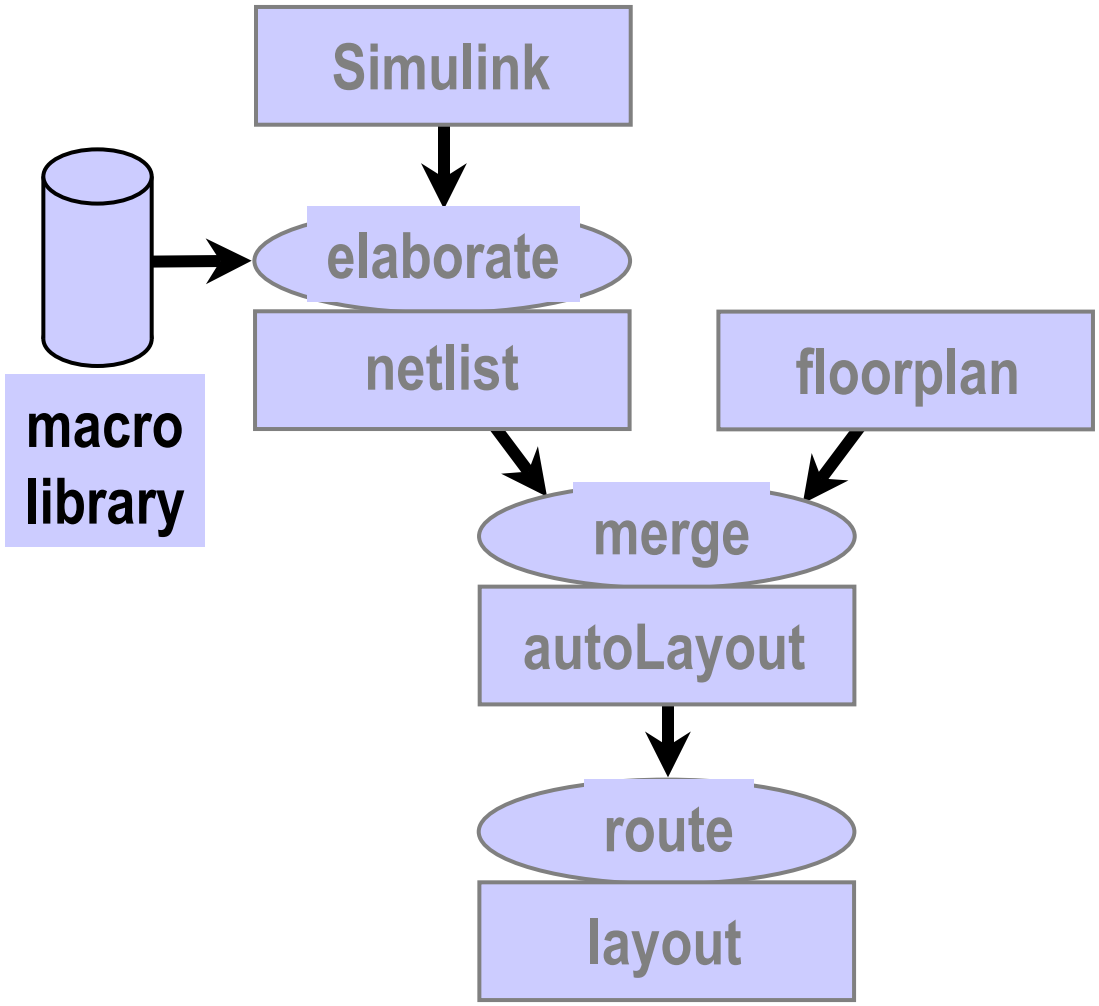


Categories:

- Function - basic input-output behavior
- Signal - physical signals and types
- Circuit - transistors
- Floorplan - physical positions

How to get layout and performance estimates quickly?

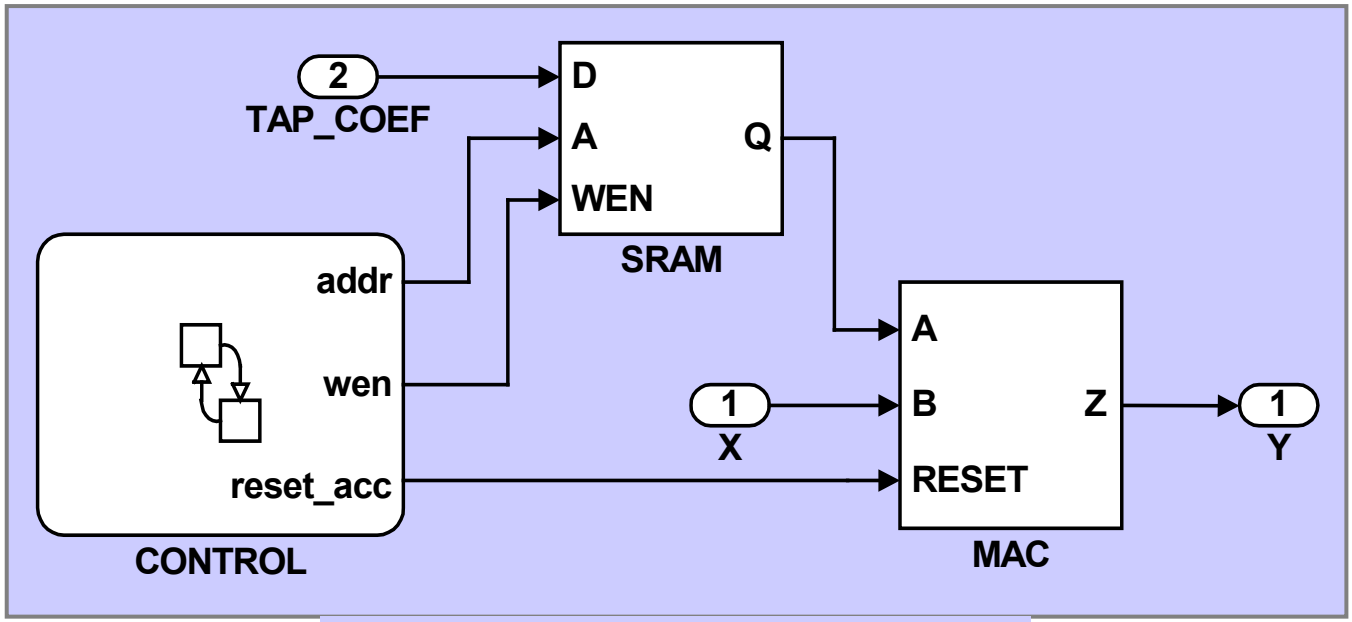
Automated Design Flow



New Software:

- Generation of netlists from Simulink
- Merging of floorplan from last iteration
- Automatic routing and performance analysis
- Automation of flow as a dependency graph (UNIX MAKE program)

Why Simulink?

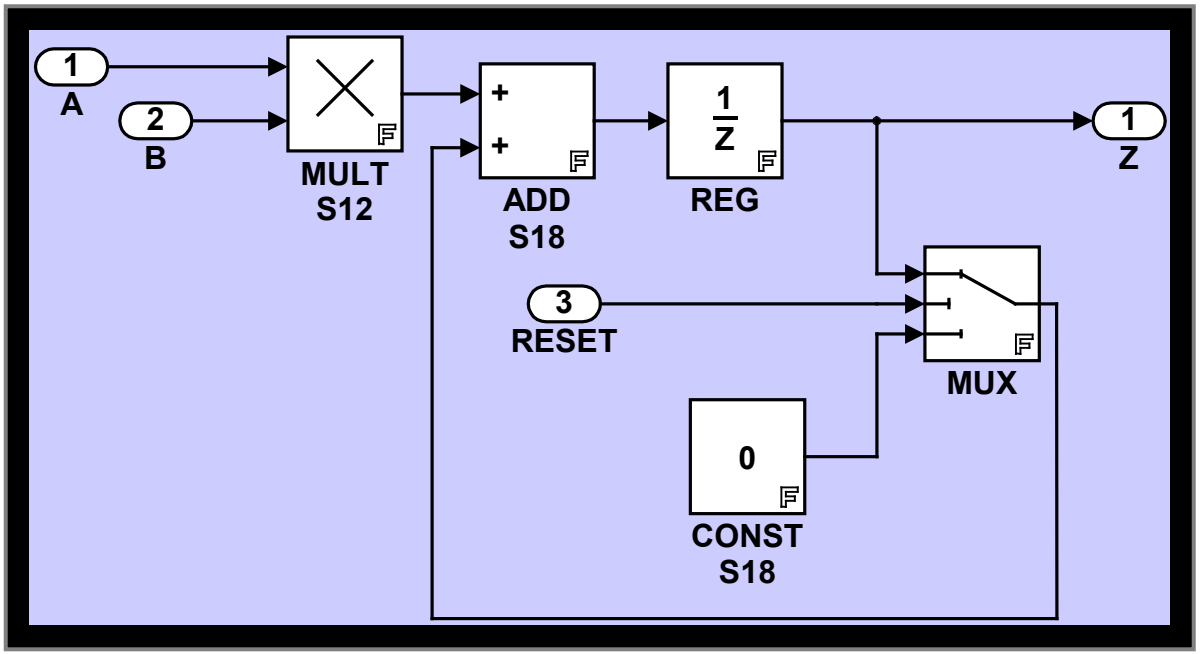


Time-Multiplexed FIR Filter

- Simulink is an easy sell to algorithm for developers
- Closely integrated with popular system design tool Matlab
- Successfully models digital and analog circuits

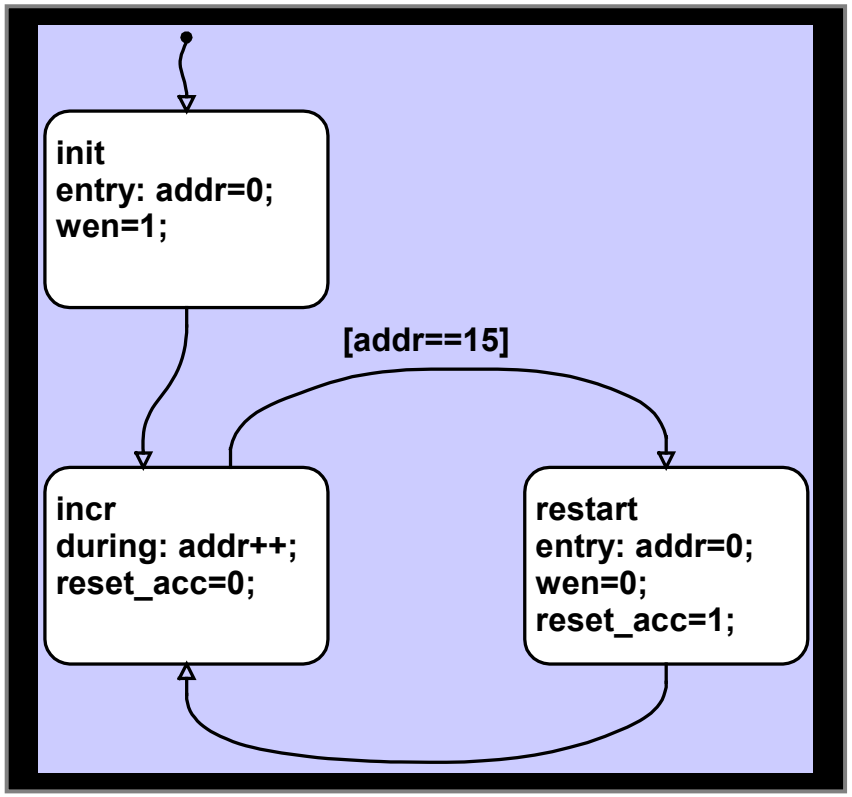
Simulink Models Datapath Logic

- Dataflow primitives (parallelism)
- Fixed-Point Types
- Completely specify function and signal decisions
- No need for RTL



Multiply / Accumulate

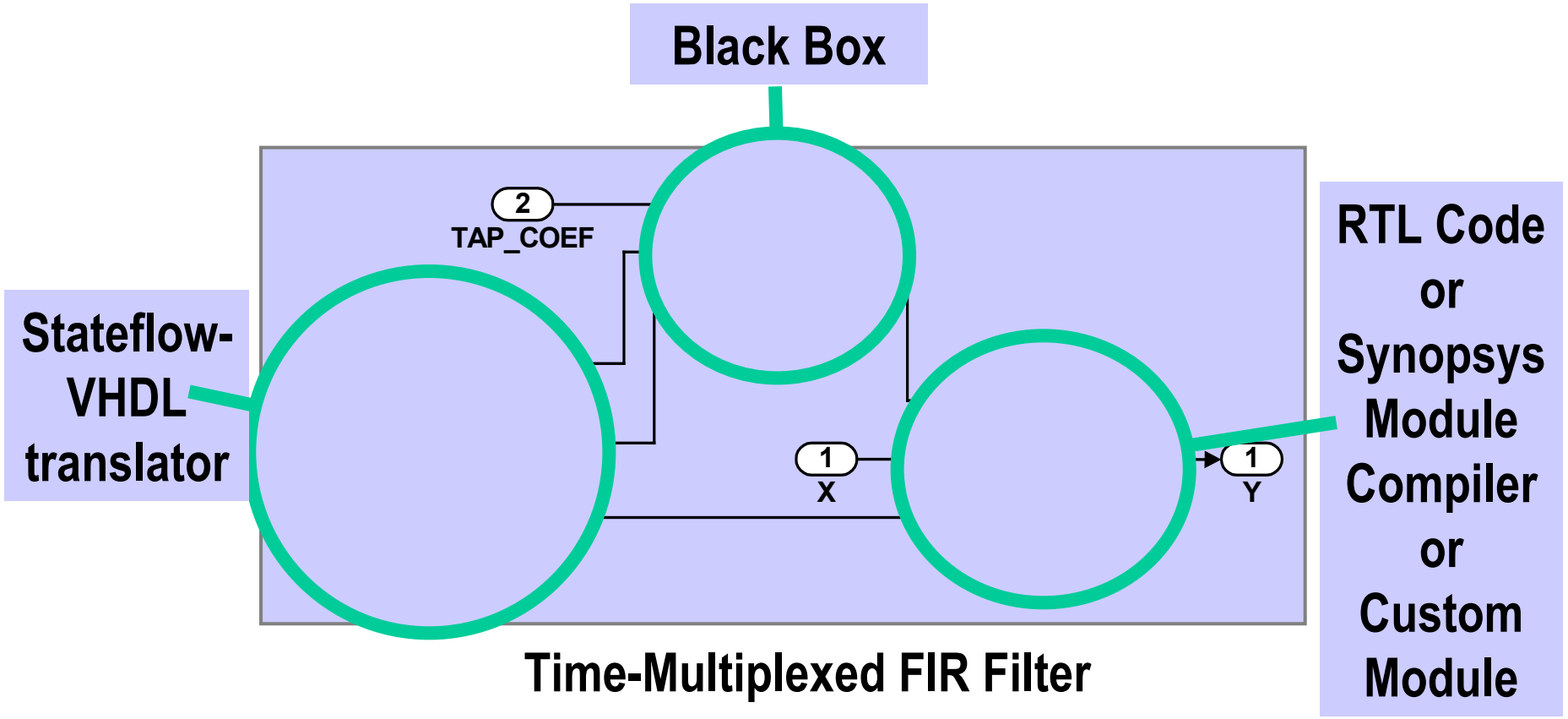
Stateflow Models Control Logic



Address Generator / MAC Reset

- Extended finite state-machine editor
- Co-simulation with Simulink
- New Software: Stateflow-VHDL translator
- More complete capture of function decisions

Specifying Circuit Decisions



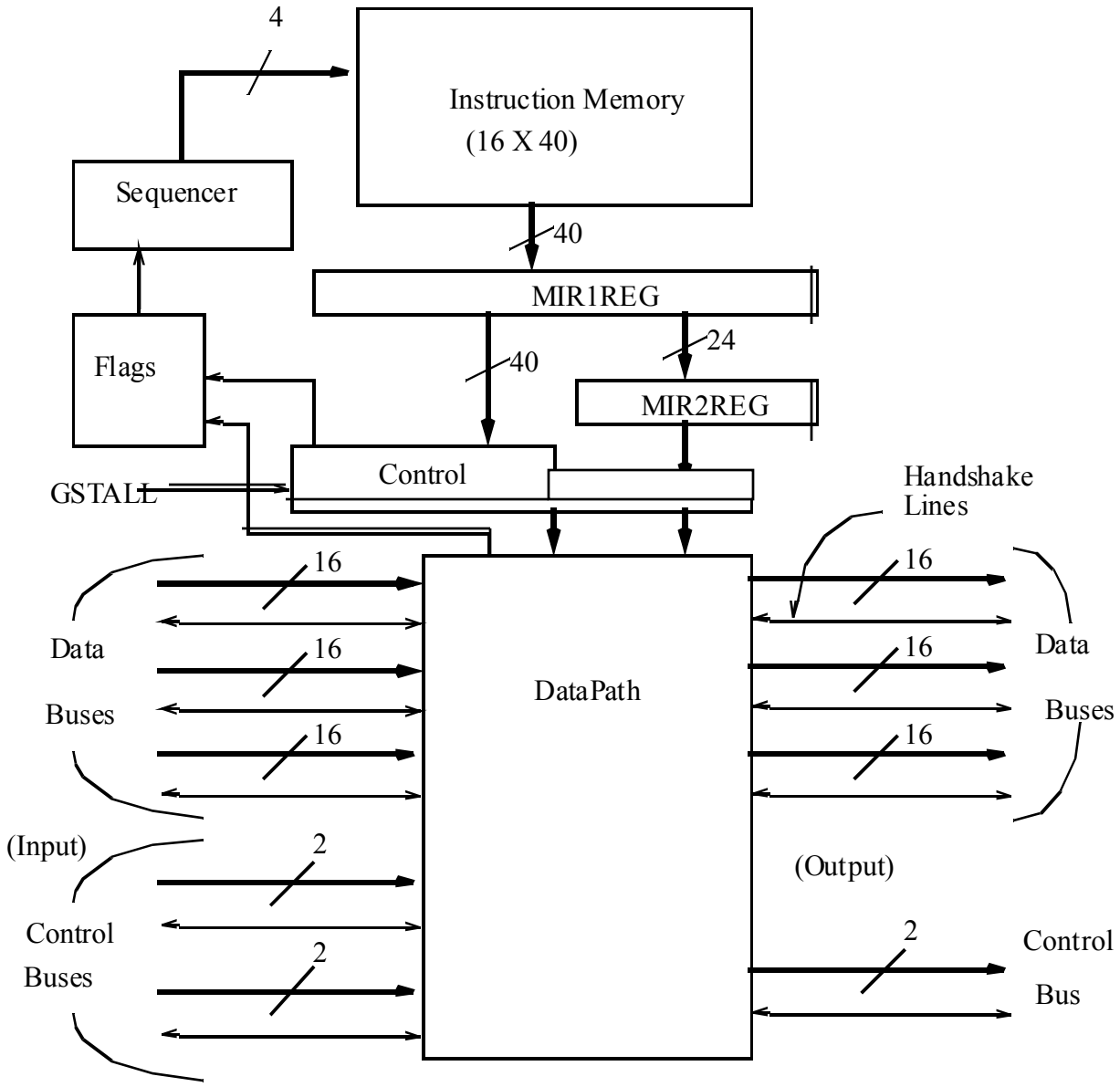
- Macro choices embedded in Simulink
- Cross-check simulations required

Application Characteristics

- Continuous stream of data coming from physical interfaces and network.
- Lots of integer and fixed point DSP calculations.
- Results sent as streams to displays, speakers, recorders, and compute servers.
- Real-time response with selectable quality of service for audio and video.
- High I/O and network bandwidth for video and audio applications.
- Short kernel loops provide instruction locality.

Embedded Processor's Architecture

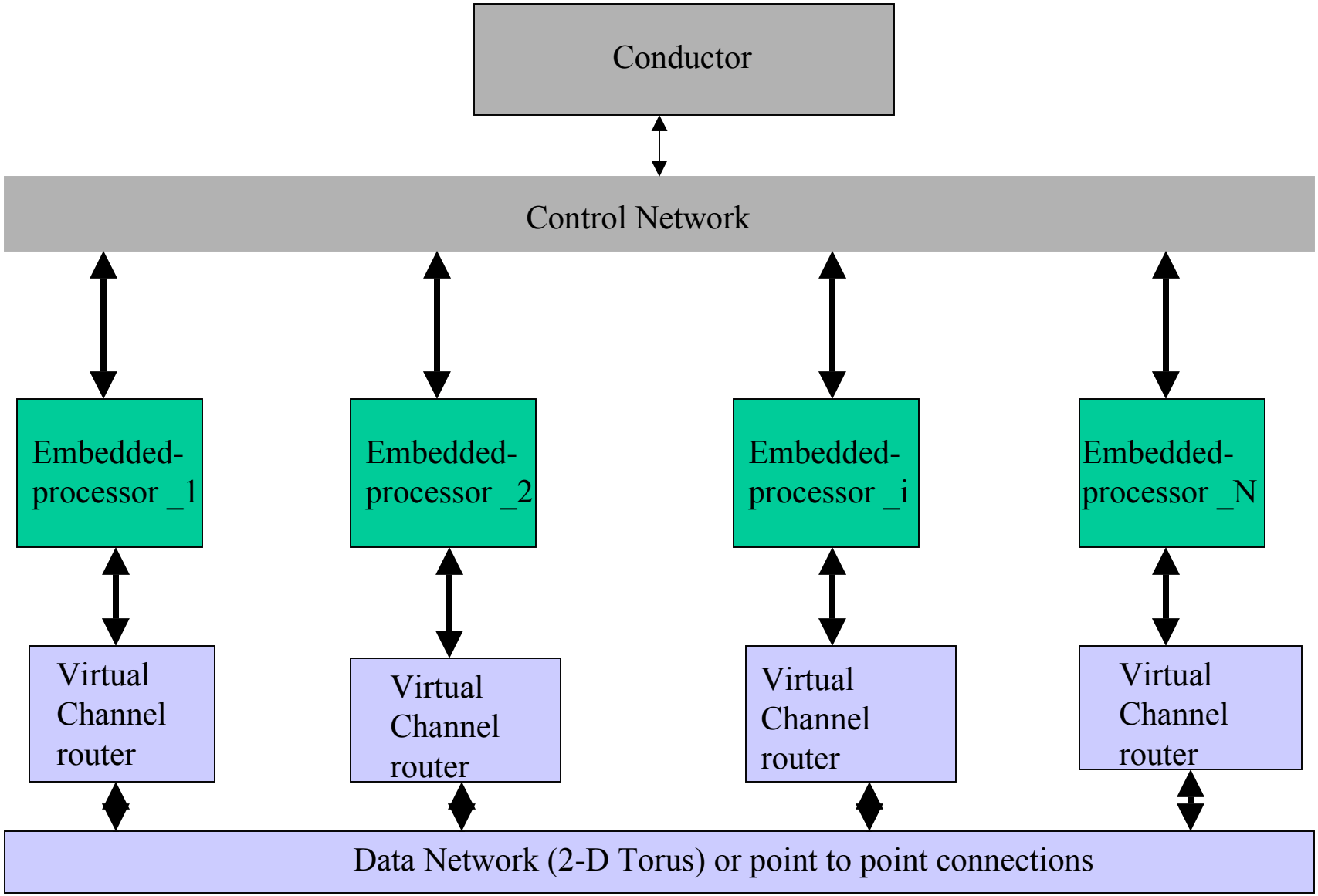
- Load/Store architecture
- Fixed length instructions
- Three address machine (source1, source2, destination) using a shared register file
- Single thread execution and no context switching
- No function calls, interrupts, exceptions
- Separate instruction and data memory units
- I/O is done using memory mapping
- Five stage pipeline (I-fetch, decode/operand fetch, execute, reg. write, store)
- Branch instructions are delayed by one cycle with delay slots filled by useful instructions
- Simple sequencer


4
Instruction Memory
(16 X 40)
Sequencer
40
MIR1REG
Flags
40
24
MIR2REG
Control
GSTALL
Handshake
Lines
16
16
Data
16
16
DataPath
Data
Buses
16
16
Buses
(Input)
2
(Output)
Control
2
Control
Buses
2
Bus

Multiple Embedded-processors

- Multiple embedded-processors are interconnected for computation, and control token and datagram flow.
- Multiple processors are executing concurrently.
- Each processor executes one or more functions of an embedded application.
- A conductor commands each embedded-processor to execute functions once or repeat at a certain rate, receives status, and makes global decisions.
- Distributed control with globally asynchronous communication between embedded-processors, and locally synchronous communication.
- Data communication between embedded-processors using virtual channels, routers, and buffers.

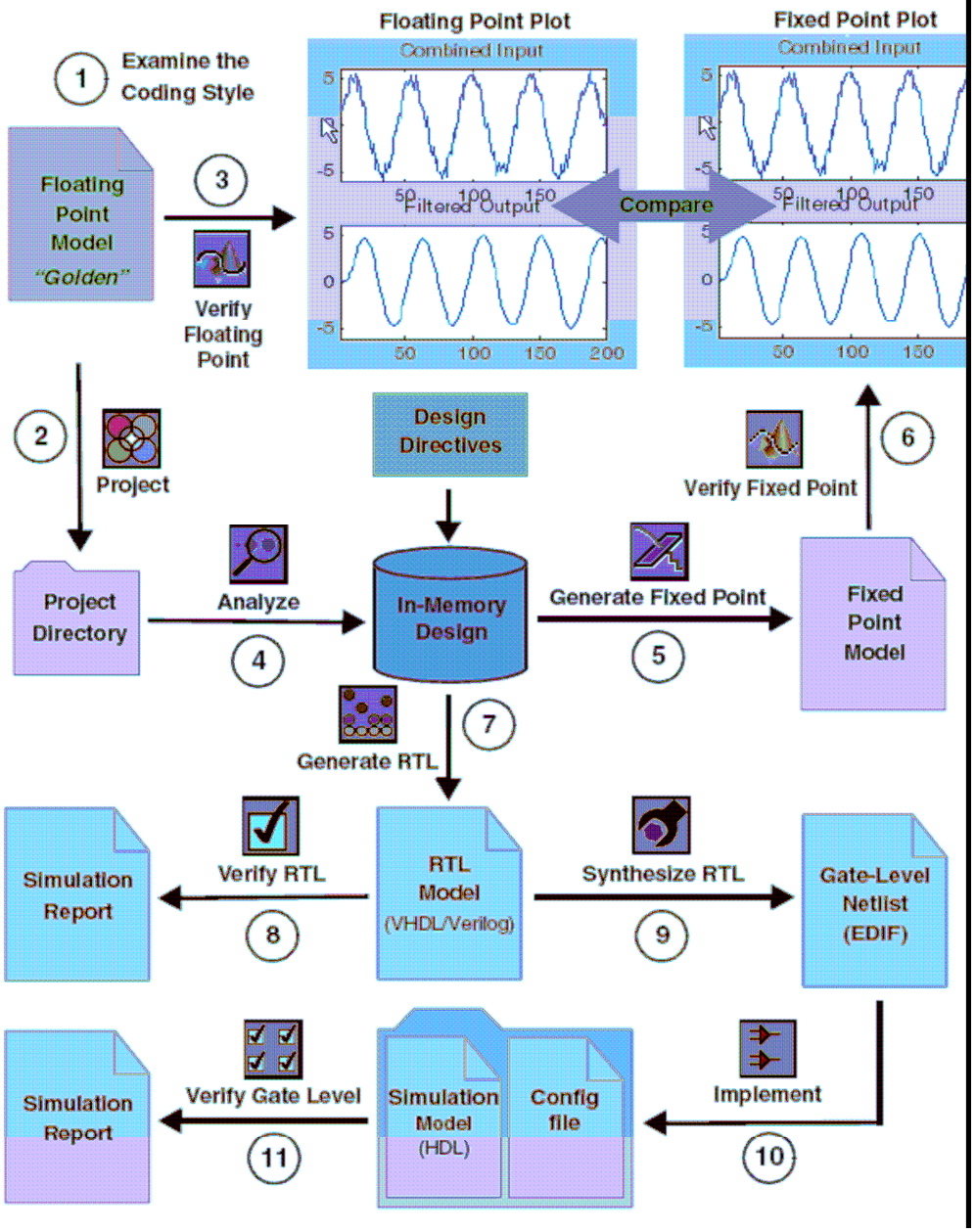
Block diagram of Embedded System



Library Development

- AccelDSP and library element development
- AccelWare
- System generator block preparation

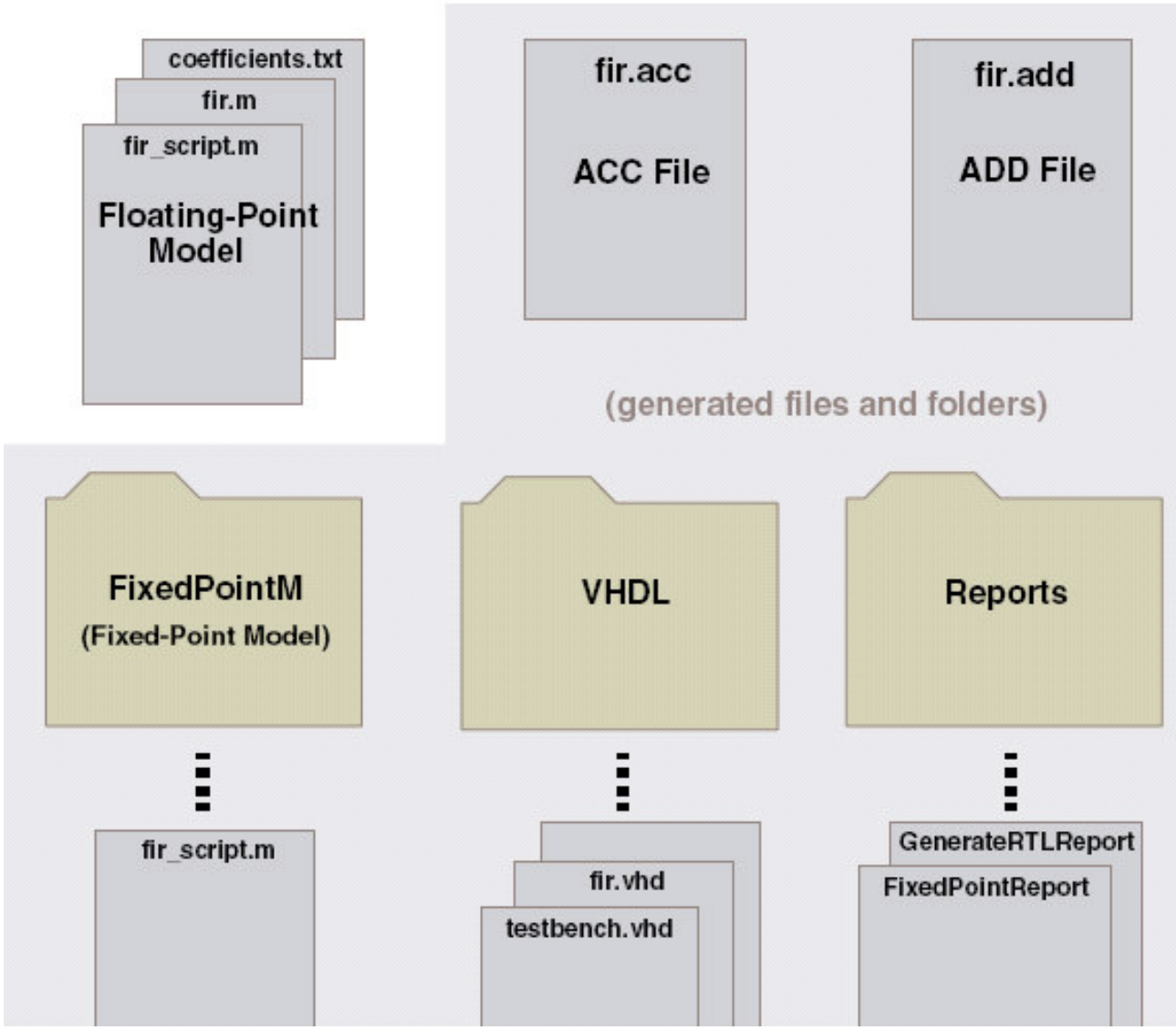
The AccelDSP ISE Synthesis Flow



1. **Examine the Coding Style of the Floating-Point Model**. You should first verify that the MATLAB design conforms to minimum AccelDSP style guidelines that are explained in the manual titled MATLAB for Synthesis Style Guide.
2. **Create an AccelDSP Project**. You invoke AccelDSP Synthesis, then click **Project** and specify the name of a new Project file. The Project file is placed in the Project Directory where all future AccelDSP-generated files are saved.
3. **Verify the Floating-Point Model**. You should have verification constructs in your MATLAB script file to apply stimulus and plot results. This output plot is the "golden" reference for comparing future results. If you have verified the floating-point model outside of AccelDSP Synthesis, you may skip this step.
4. **Analyze the Floating-Point Model**. This step creates an in-memory model of your design. In a later pass through this flow, you can add design directives that guide AccelDSP toward finding the best hardware architecture for your design.
5. Generate a Fixed-Point Model. This step generates a fixed-point model of the design, then places the design files in a newly generated project sub-folder named MATLAB.
6. **Verify the Fixed-Point Model**. When you click **Verify Fixed Point**, AccelDSP automatically runs a MATLAB fixed-point simulation. You then visually compare the Fixed-Point Plot with the Floating-Point Plot to verify a match.

7. **Generate an RTL Model**. This step generates an RTL Model from the in-memory design data base. The RTL model can be generated in VHDL or Verilog format.
8. **Verify the RTL Model with your HDL Simulator**.You click Verify RTL to run your HDL simulator on the generated Testbench. PASSED or FAILED will be indicated in a simulation report.
9. **Synthesize the RTL Design into a Gate-Level Netlist**. This step invokes a pre-specified RTL synthesis tool on your design. The generated gate-level netlist is ready for place and route using the ISE implementation tools.
10. **Implement the Gate-Level Netlist**. You click on **Implement** to invoke the ISE implementation tools to place and route the design. The generated files of interest are a gate-level HDL simulation file and a configuration file containing the bitstream for configuring the FPGA hardware.
11. **Verify the Gate Level Design**.This step uses the AccelDSP Testbench to run a bit-true simulation check on the gate-level HDL simulation model. A PASSED indication means that the implemented design is bit-true with the original fixed-point MATLAB design.

Project Directory

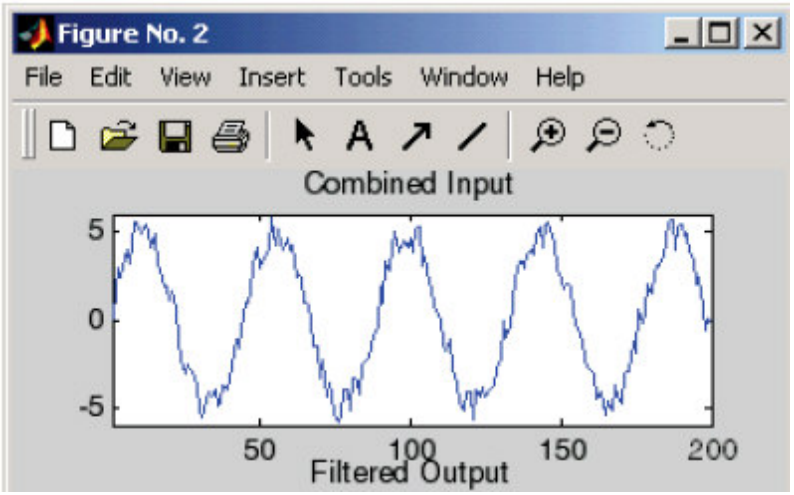


Verifying the Floating-Point Mode

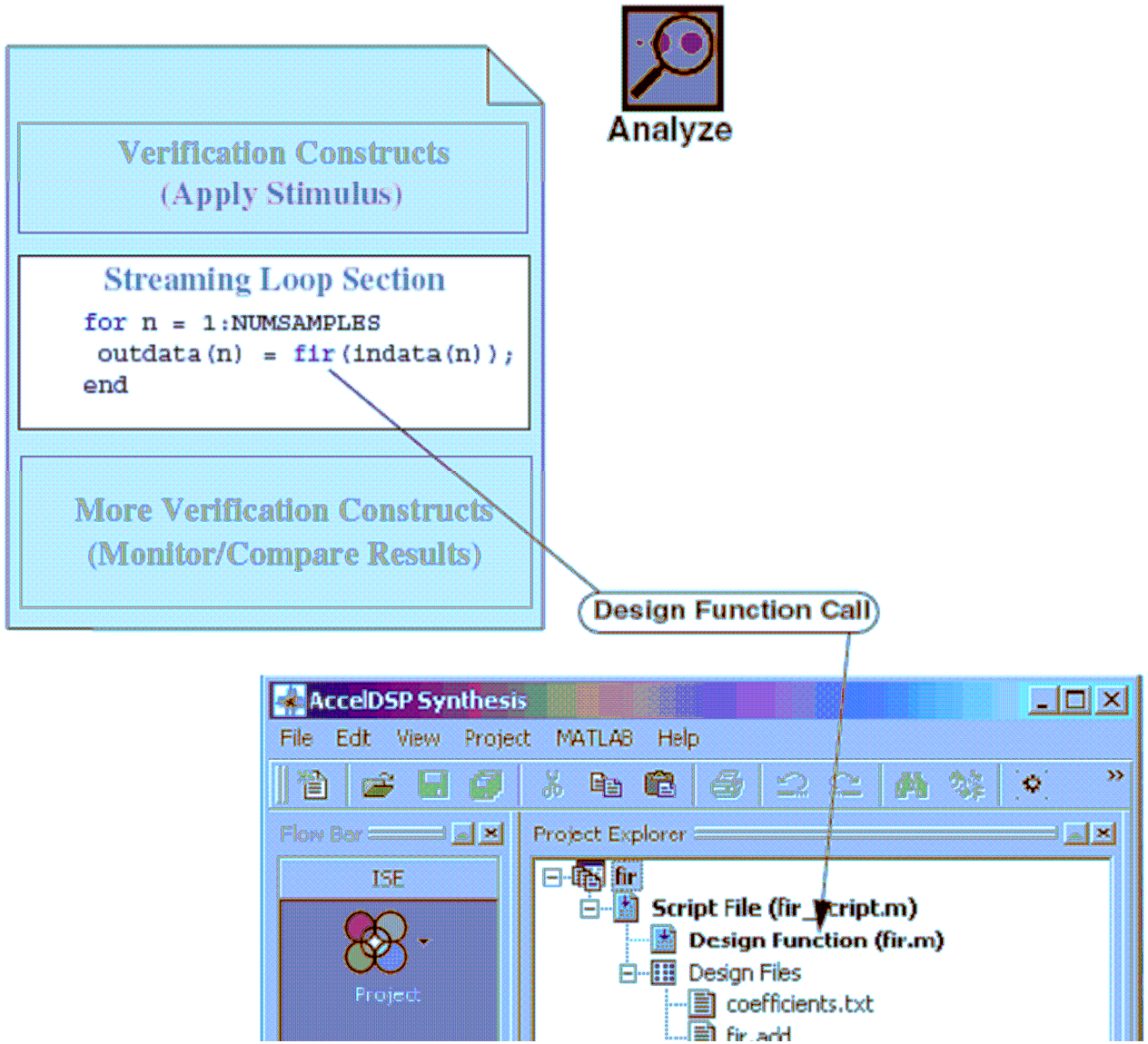
Verify Floating Point

Floating Point Model

(automatically invoked)

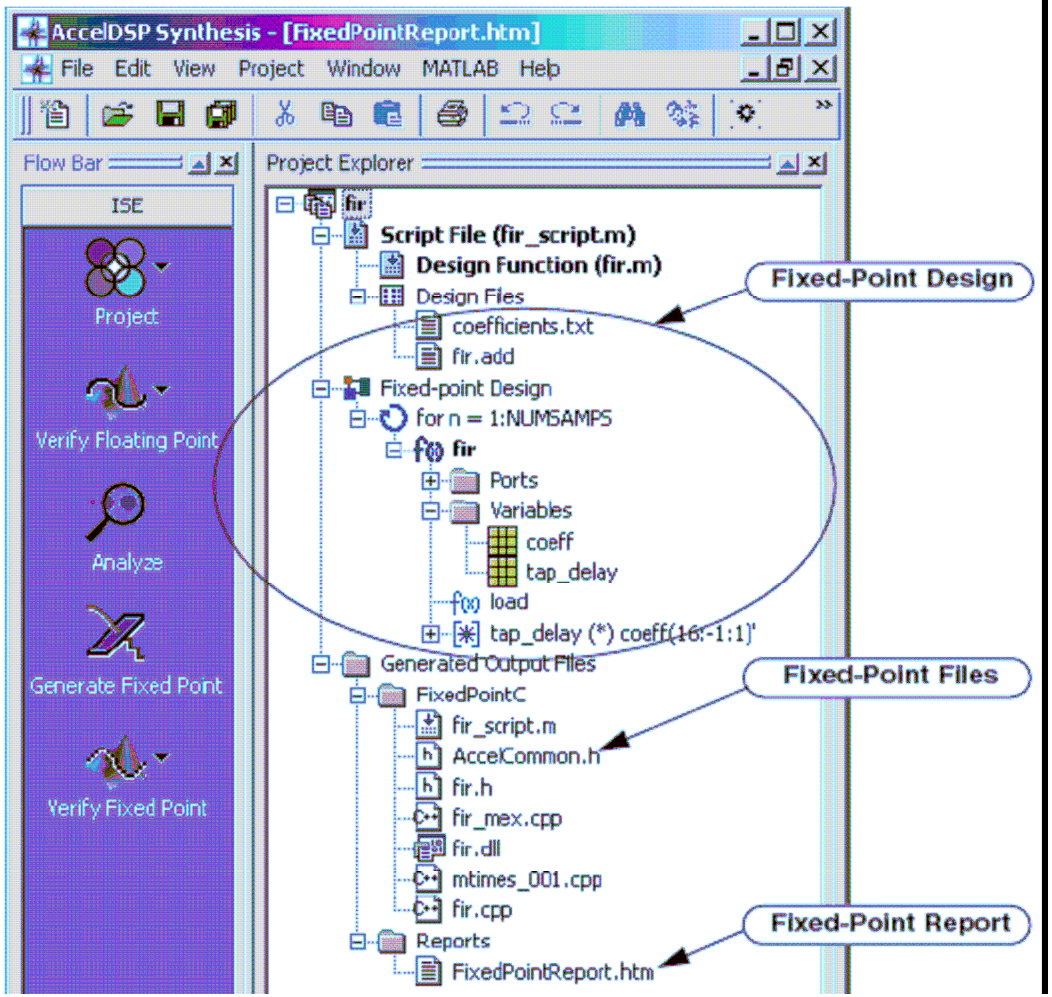


Analyzing the Floating-Point Model

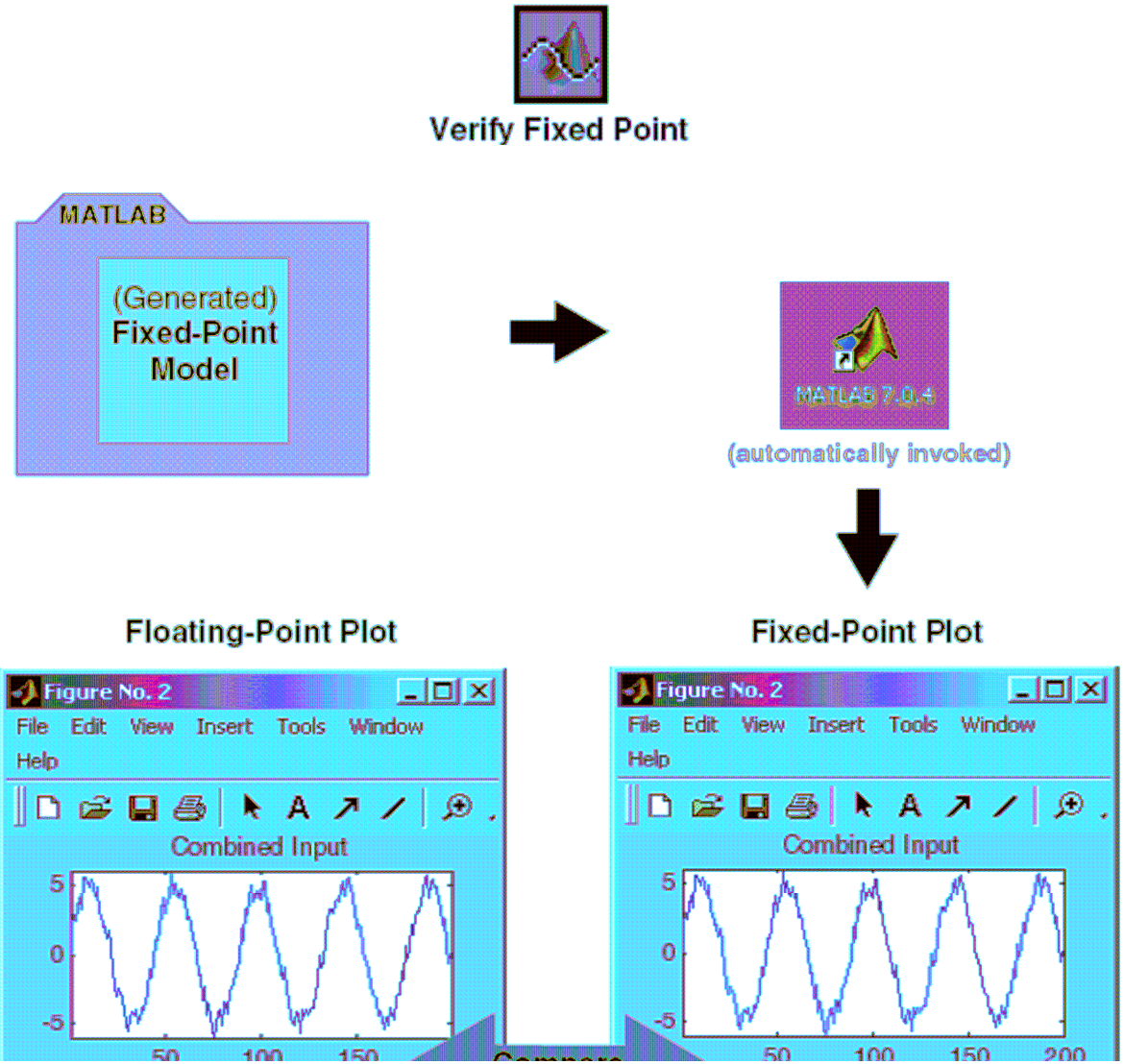


Generating the Fixed-Point Mode

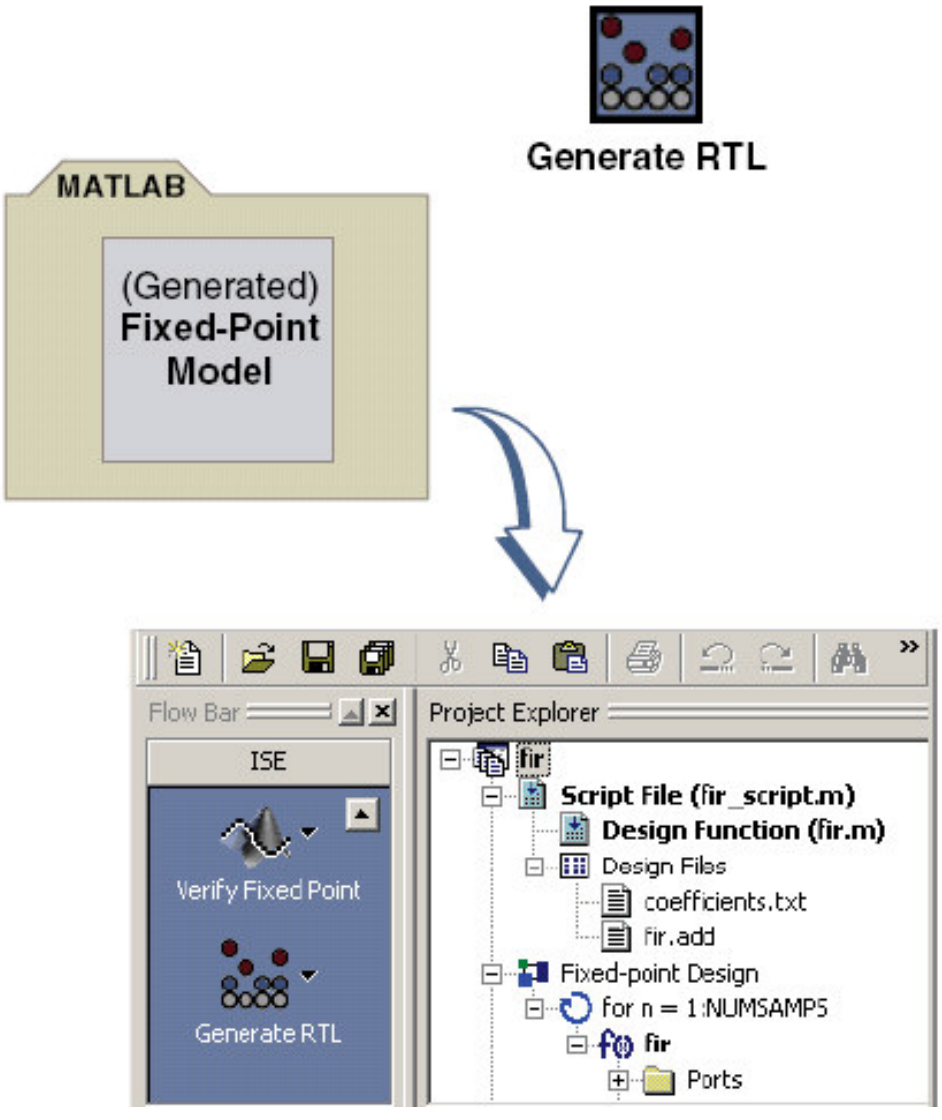
Generate Fixed Point

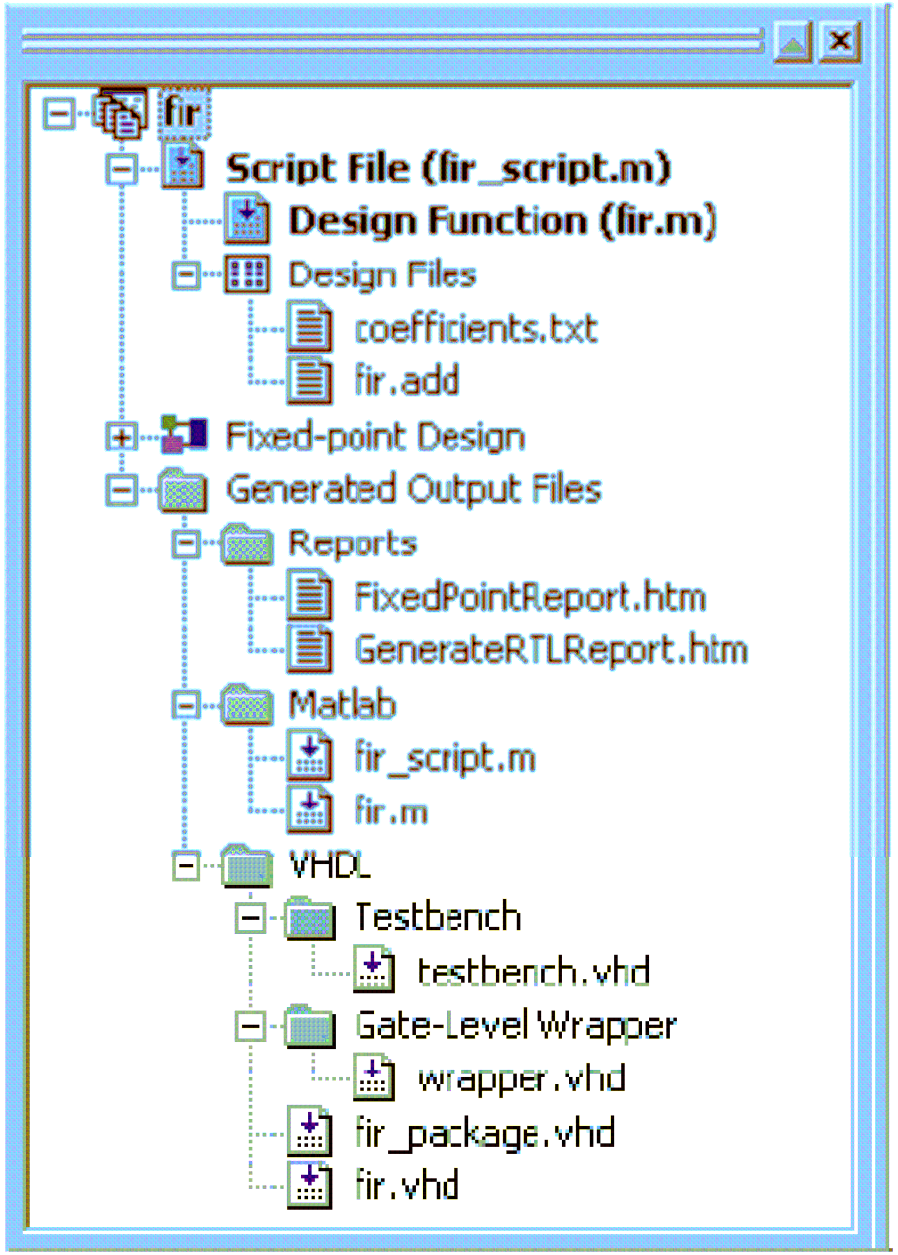


Verifying the Fixed-Point Model

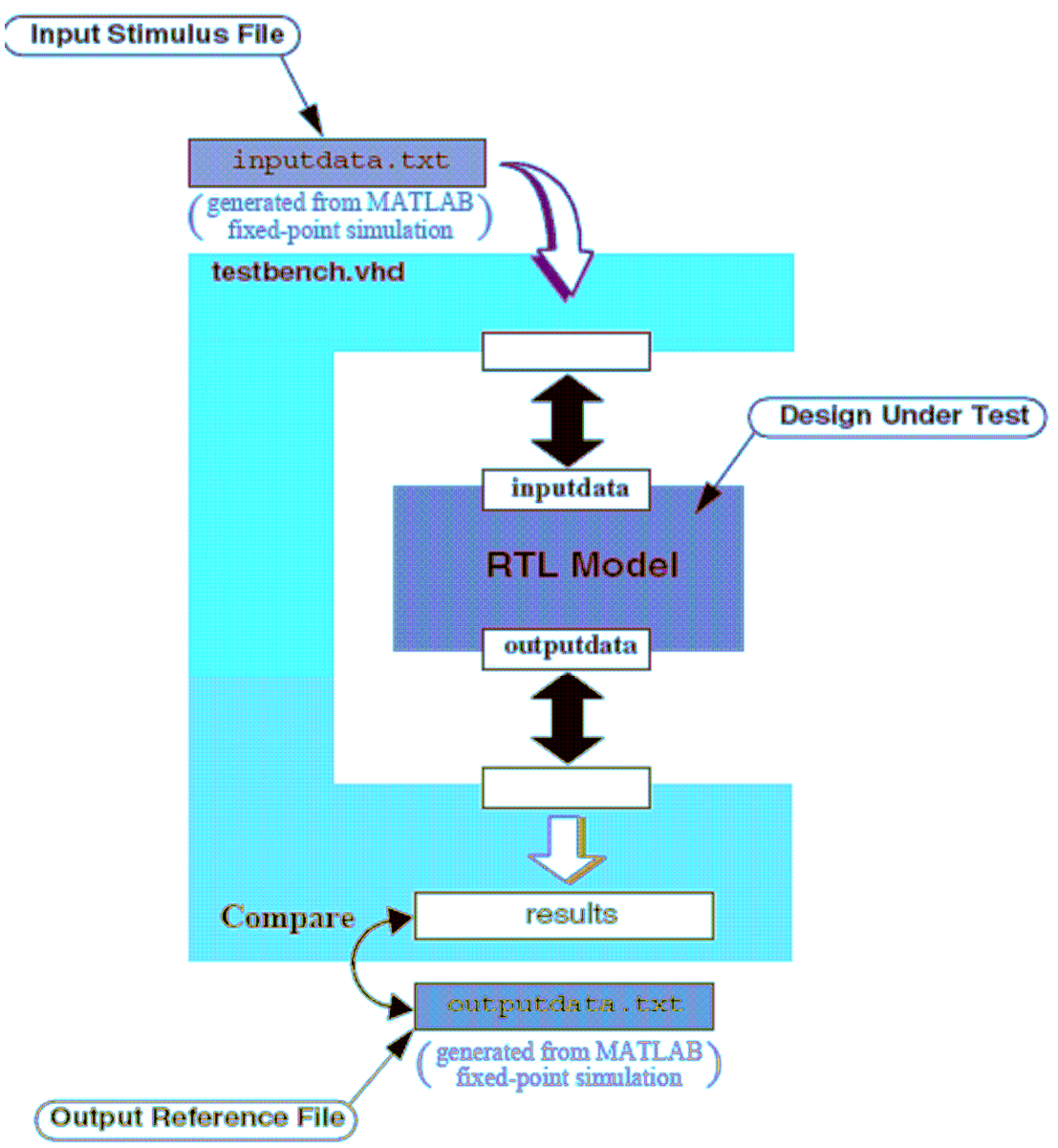


Generating the RTL Model

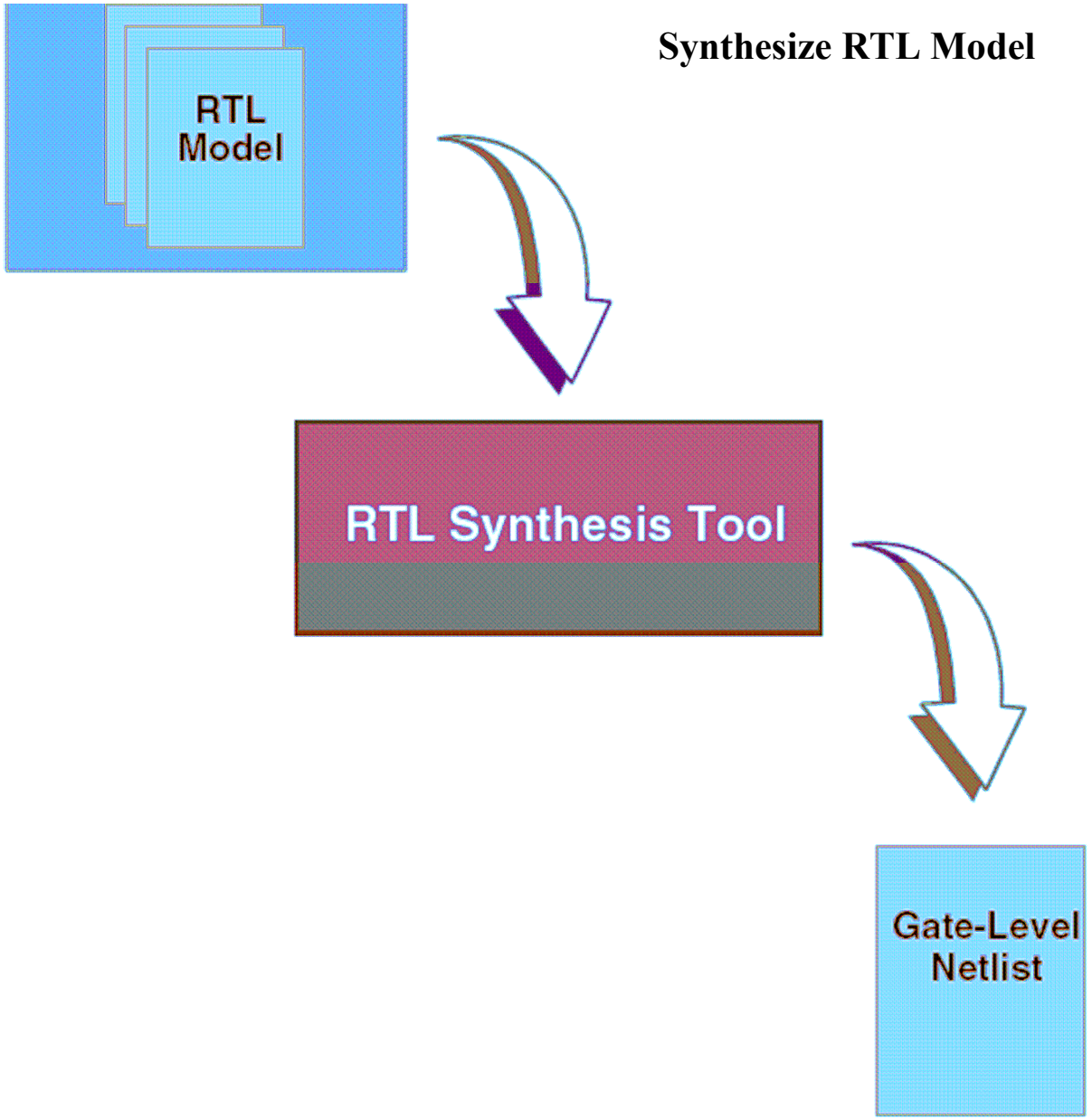



fir
Script File (fir_script.m)
Design Function (fir.m)
Design Files
coefficients.txt
fir.add
Fixed-point Design
Generated Output Files
Reports
FixedPointReport.htm
GenerateRTLReport.htm
Matlab
fir_script.m
fir.m
VHDL
Testbench
testbench.vhd
Gate-Level Wrapper
wrapper.vhd
fir_package.vhd
fir.vhd

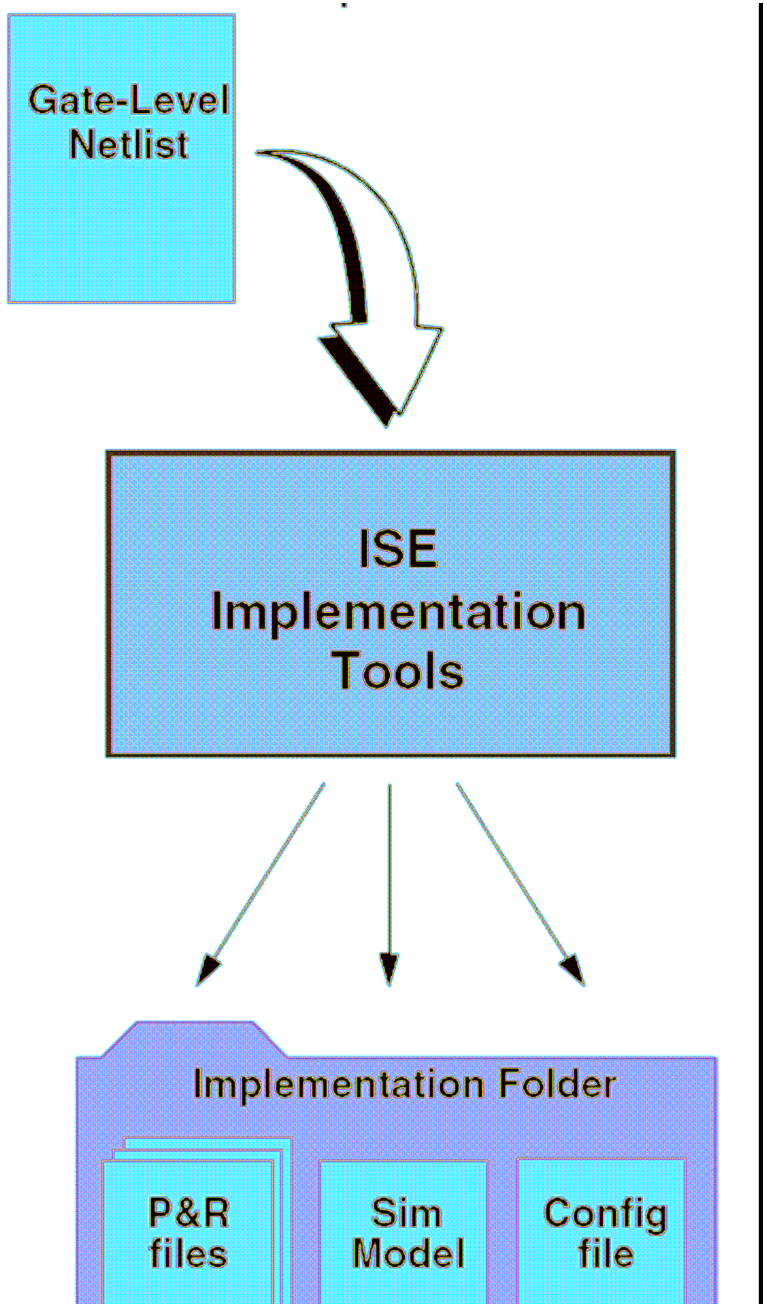
Verify RTL Model



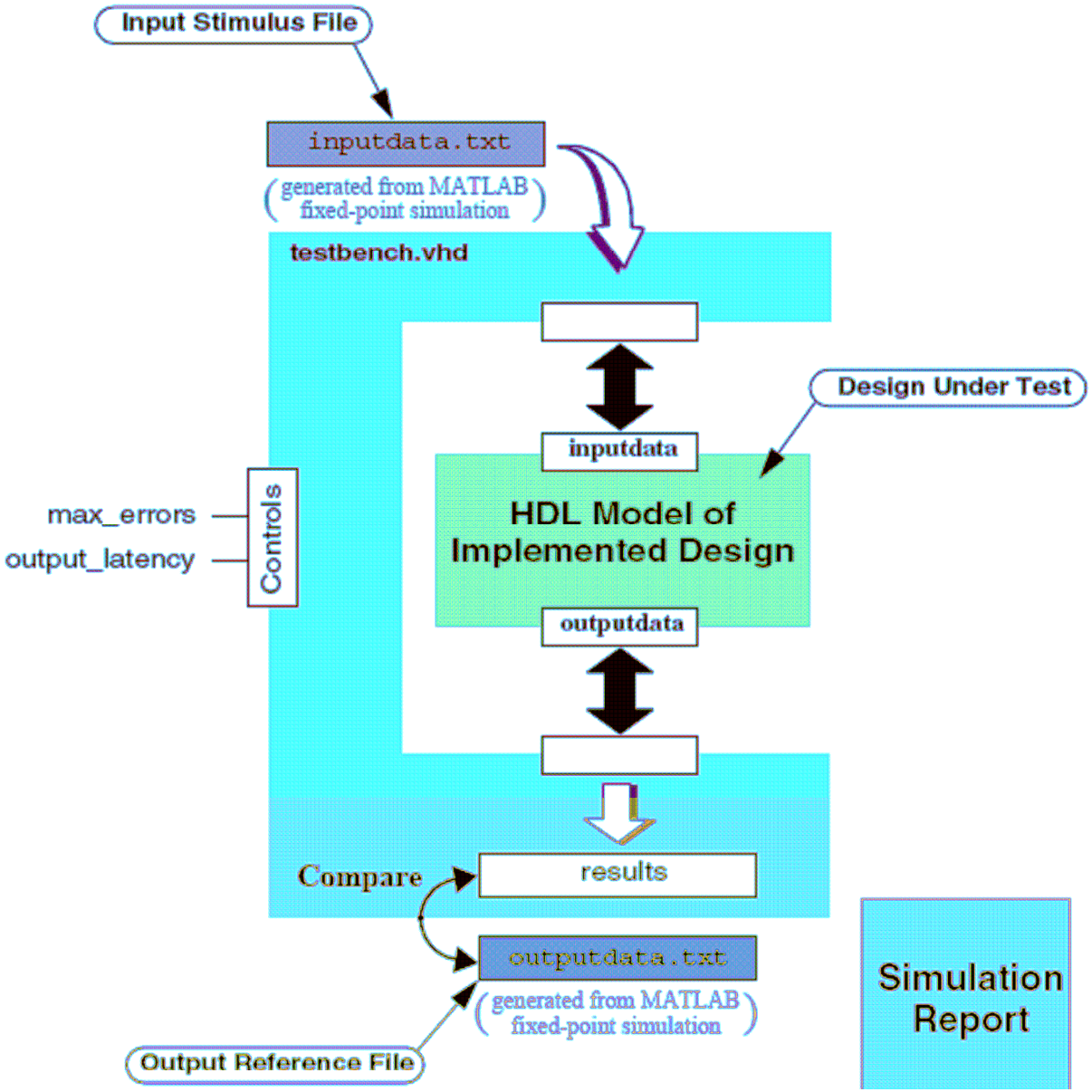
Synthesize RTL Model



Implement


Gate-Level Netlist
ISE Implementation Tools
Implementation Folder
P&R files
Sim Model
Config file

Verify Gate level Design



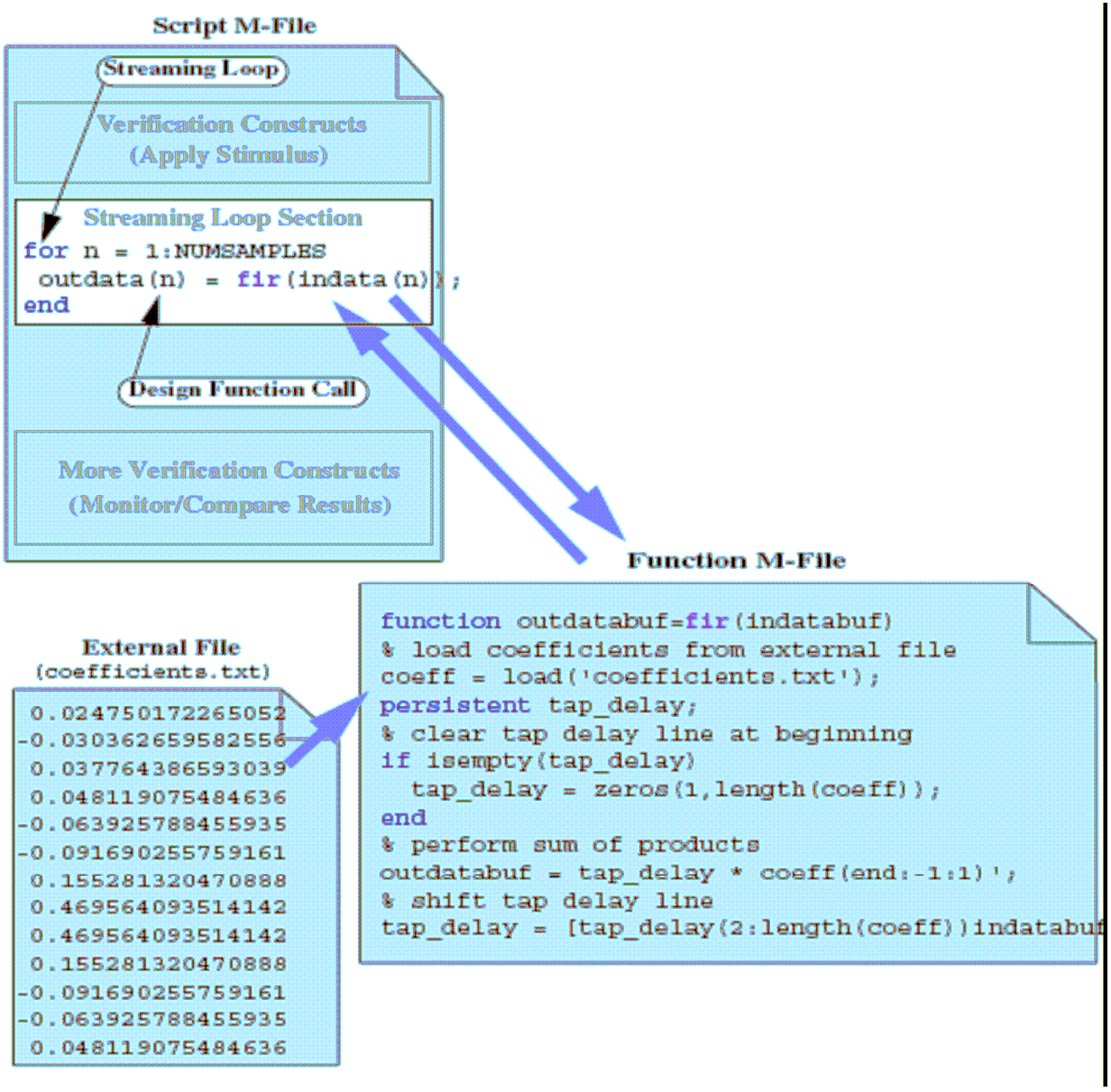
Design file Structure

- Script file in Matlab
  - Containing a streaming loop (for or While)
  - function call (top level)
  - other constructs for design verification (ignored by synthesis)
- Top level function file in Matlab
  - Contains the hardware to be synthesized
  - Inputs and outputs must be a variable that represents a scalar, row vector, or column vector
- The above two files can reference other function files in Matlab

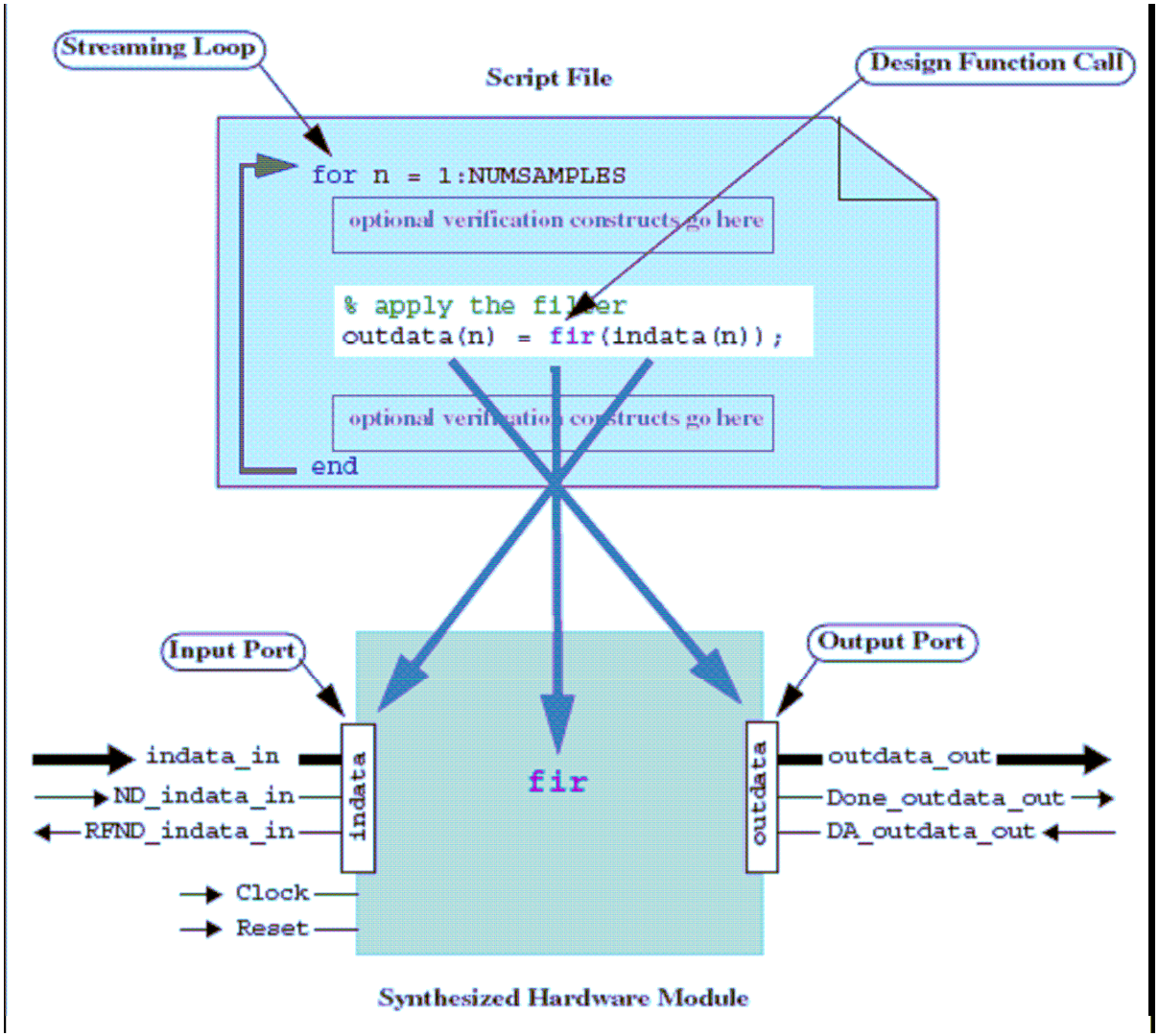
Limitations

- Matrices cannot be used as arguments to top level function and results cannot be matrices
- Function arguments and subexpression arguments are not allowed
- Array elements are not allowed as arguments

Basic files of a Design



Mapping Design to Hardware



Handshake Interface

- Global signals - clock and reset (reset has to be active high for one cycle)
- Input synchronizing signals
  - acInputAvail (input)
  - acInputReq (output)
- Output synchronizing signals
  - acOutputAvail (output)
  - acOutputAck (input)

Input Synchronization Signals

- acInputAvail (input) - Indicates that data on input port is valid. This signal is controlled by external design. Receiving device can capture data on the input ports during the rise edge of next clock cycle
- acInputReq (output) - It is set by the hardware module. If the signal is high the module is ready to receive data from input port. If the signal is low the external design should stop sending data.

Output Synchronization Signals

- acOutputAvail (output) - This signal is controlled by the hardware module. It is set high when data is valid on the output port. It will stay high until the receiving module sends a high acOutputAck signal.
- acOutputAck (input) - This signal is controlled by the external design. It is set to high when data is captured by the external design.

Next Steps for DGC3

- Design components for stereo processing and generate hardware using AccelDSP.
- Design components for image segmentation, classification, identification, and generate hardware using AccelDSP.
- Design components for multiple moving target tracking and generate hardware using AccelDSP.
- Subsystem integration from components and mapping to BEE2

References

Computer Aids for VLSI Design, 1994

http://www.rulabinsky.com/cavd/text/chap01-1.html

IEEE Computer, Dec. 2006