

CVPR 2006 Tutorial

Embedded Computer Vision and  
Real-Time Algorithms for  
Smart Cameras

Branislav Kisacanin & Mathias Kolsch

# Introduction - Overview

1:30-3:15 part I

3:15-3:35 coffee break

3:35-5:30 part II

# Tutorial Overview

- Motivation
- Media Processors
- FPGAs
- ASICs
- Porting to Embedded Platforms
- Algorithmic Considerations
- Resources, Conclusions, Future...

# Introduction - Presenters

- **Branislav Kisačanin, Delphi Corporation**
  - Projects: Vision-Based Active Driver Safety and Driver Assistance
    - Driver State Monitor (drowsiness, distraction)
    - SAVE-IT
  - Expertise: Embedded CV Algorithms & SW Implementation
    - Synergy between algorithms and SW
    - Synergy between optics and algorithms
- **Mathias Kölsch, Naval Postgraduate School**
  - Projects:
    - postural comfort zones
    - hand detection and tracking, posture recognition (HandVu)
    - multimodal, wearable AR system
    - current: UAV video processing (tracking, 3d viz, ...)
  - Expertise:
    - computer vision, computer graphics
    - augmented reality, HF/HCI, systems & networking

# Introduction - Style

- Combination of academic and industrial experiences and views
- Questions & comments encouraged
- We want to
  - Share what we know
  - and
  - Learn from you!

# For those not sure ...

- Embedded Computer Vision
  - Implemented on low-cost, low-power, minimal HW
  - Other definitions
    - » “Development requires an oscilloscope”
    - » Murray Gell-Mann’s (of the quarks fame) definition
  - Sacrifice computational power, gain from interaction with optical design
- Real-Time Algorithms
- Smart Cameras

# Terminology

- Embedded Computer Vision
- Real-Time Algorithms
  - Not a perfect definition, any algorithm can be real-time, given enough Crays
  - “Fast” or “Fast enough,” to help minimize HW
  - Example: DFT and FFT
  - Warning: # mults and adds,  $O$ -notation
  - Helps with a big goal: First to market
- Smart Cameras

# Terminology

- Embedded Computer Vision
- Real-Time Algorithms
- Smart Cameras
  - Camera + embedded vision platform
  - Running real-time algorithms
  - Cost advantage:
    - » Many cameras, video transfer, CPU
    - » Smart cameras, low-bandwidth data transfer, CPU

# Who are you?

- computer scientists
- engineers: electrical, controls, computer
- image processing vs. computer vision
- academia
- industry
- other?

# Embedded or CPU?

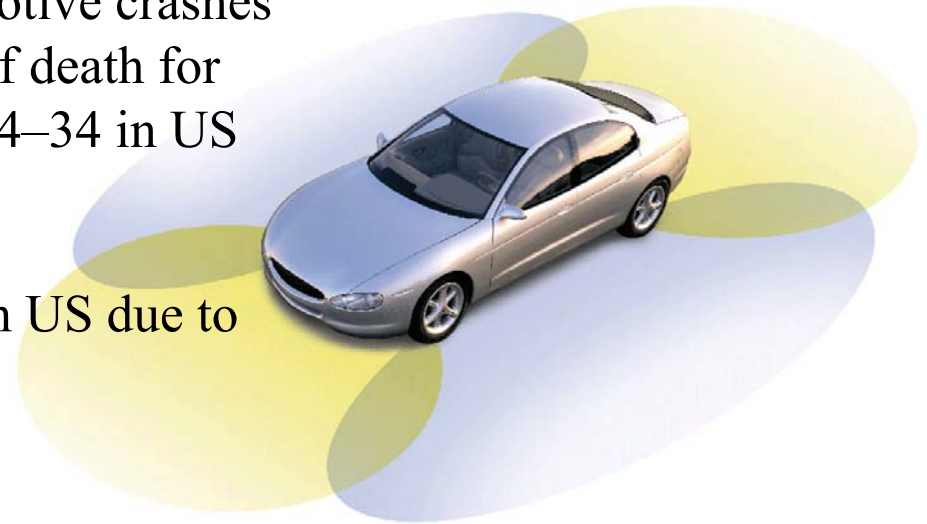
- Advantages of Embedded vs. CPU
  - Cost
  - Packaging
  - Power consumption
  - Temp ranges
- In the future ...
  - Smart cameras
  - Networks

# Embedded or CPU?

- Limitations of Embedded vs. CPU
  - Processing power
    - Pentiums have MMX, SSE, SSE2
    - PowerPCs have AltiVec
  - Development tools
  - Fixed-Point Arithmetic

# Example: Automotive Safety

- Vision for Active Safety
  - Recent NHTSA\* statistics (2001)
    - In 2000 and 2001 automotive crashes were the leading cause of death for children and people age 4–34 in US
  - NHTSA estimates
    - 1500+ deaths annually in US due to fatigue and drowsiness
    - Conservative estimate



\*National Highway Transportation Safety Administration

# Example: Automotive Safety

- Processor cost
- Size
- Power
- Temp range

# Questions? Comments?

# Embedded Processor Selection

- General rules we learned
  - Desirable features
    - Fast memory
    - Wide data bus
    - Parallelism
      - VLIW, SIMD
    - Other: power, heat, tools, cost, ...



# Embedded Processors

- Media Processors and DSPs
  - Computer Vision >> Image Processing >> DSP
  - Media Processors = extreme DSPs + video ports
    - Fast memory
    - Wide data bus and DMA
    - Parallelism
      - VLIW, SIMD
  - For products with medium volume (10,000-100,000/yr)
  - Texas Instruments, Philips, Analog, Equator
- FPGAs
- ASICs

# Embedded Processors

- Media Processors and DSPs
- FPGAs
  - higher NRE, lower volume cost than DSP (?)
  - customizable, thus high performance
  - medium power
  - design much more involved than DSP
  - projects with small to medium volume ( < 10,000/yr)
- ASICs

# Embedded Processors

- Media Processors and DSPs
- FPGAs
- ASICs (= Application Specific IC)
  - Development path from FPGAs
  - Examples: your own or Acadia, EyeQ, ...
  - For products with high volumes ( $> 100,000/\text{yr}$ )

# Other Hardware

- Mobile Phones
- Framegrabbers
- Smart Cameras
- GPU
- ...

# Image Sensor Digital Interfaces

- relay through memory
- analog frame grabber (ADC)
- media processors: video port → DMA to memory
- coming into video port from image sensor

# Tutorial Overview

- Motivation
- Media Processors
- FPGAs
- ASICs
- Porting to Embedded Platforms
- Algorithmic Considerations
- Resources, Conclusions, Future...

# Embedded Processors - Media

- Media Processors
  - DSP + video port
  - VLIW
  - SIMD
  - Compile-time optimization
  - Typically fixed-point HW
  - Example: TI DM64x family

# Embedded Processors - Media

- Entire embedded system on a single chip (SoC)
  - video ports
  - even analog components
  - Ethernet MAC
  - PCMCIA
  - USB controllers
  - Bluetooth
  - RS232 UART
  - IrDA
  - IEEE 1394 (FireWire) controller
  - display interface
  - flash memory interfaces
  - ADCs and DACs
- Core speed 100s MHz
- Cost \$10-\$100
- Power 50mW-5W

# Embedded Processors - Media

- Target market: set-top boxes, mobile phones, automotive industry, portable media devices, cable modems...
- Have “hardware support for video processing”
  - camera interface
  - video codec (MPEG2, MPEG4, H.263)
  - very little image processing, no computer vision
- Intel XScale architecture = ARM10 + stuff
- TI OMAP2 and DaVinci
  - OMAP = ARM11 + C55xx
  - DaVinci = ARM 11 + DM64x(x)

# Tools for Embedded Vision Design

- MATLAB
  - Algorithm development
  - Embedded targets (not mature yet)
- IDE
- Vision board
- Digital video camera (with Analog In)
- DVD burner/player

# Embedded Vision Libraries

(this is going to be short)

- TI
  - Fast Run-Time Support Library
  - Float ops on fixed-point architectures
  - DSPLIB: one-dimensional signal processing
  - IMGLIB two-dimensional data streams
- BLAS/LAPACK is available for some DSPs
- IPL port to DSP (Rinner et al.)
  - not possible entirely transparently

So you got that cool chip...

# So you got that cool chip...

- DSP Starter Kits (DSK)
- evaluation boards
- have JTAG and PCI or USB interface to host PC
- common peripherals: memory, I/O controllers, I/O connectors, camera, display

# Questions?

# Embedded Processors - FPGAs

*why?*

- performance example

*what?*

- hardware

Slides mostly courtesy Asst. Prof. Ryan Kastner:

[ExPRESS](http://express.ece.ucsb.edu/) - Extensible, Programmable,  
Reconfigurable Embedded SystemS

*how?*

<http://express.ece.ucsb.edu/>

- HDL, tools, ...

# Field-Programmable Gate Arrays

- *field*-programmable
- off-the-shelf, inexpensive (?)
- logic program *is* the microcontroller
- nowadays often include memory, and hard or soft cores:
  - GPP (ARM, MIPS, PowerPC)
  - DSP
  - Xilinx MicroBlaze & PicoBlaze
  - Altera Nios I & II

# FPGA Technologies

- floating gate (Flash, EPROM, EEPROM)
- **SRAM** (Static RAM)
  - SRAM is volatile, does not need refresh
  - as opposed to **DRAM** (Dynamic RAM)
  - do not confuse with Synchronous DRAM (**SDRAM**, state change sync'ed to clock)
- antifuse (open until blown)

# FPGAs

- see Ryan Kastner's slides on "Reconfigurable Computing Architectures", at <http://www.ece.ucsb.edu/~kastner/ece154/slides/ece154-17.pdf>

# FPGA Security

Tim Sherwood, UCSB:

- Security is considered a secondary consideration - if at all
  - The current work in FPGA security is primarily focused on preventing intellectual property theft
  - Basic security primitives, such as separation, authentication methods for fair resource sharing, and memory protection are all up to applications writers to implement (which means they don't get implemented)
- The embedded design community recognizes this as one of the most serious impediments to progress in the field

# FPGA Security: RCsec

- joint UCSB and NPS project
- aims to address the problems of logic/memory separation on FPGAs
  - **practical:** designs are augmented with security information that can be synthesized directly into logic
    - example: memory protection with “reference monitor”
  - **formal:** build systems that support:
    - multiple levels of security or trustworthiness, all interacting on a single chip

contact: Tim Sherwood, UCSB

# Embedded Processors - FPGAs

*why?*

- performance example

*what?*

- hardware

*how?*

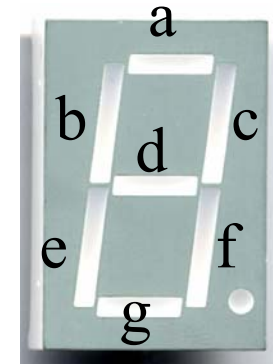
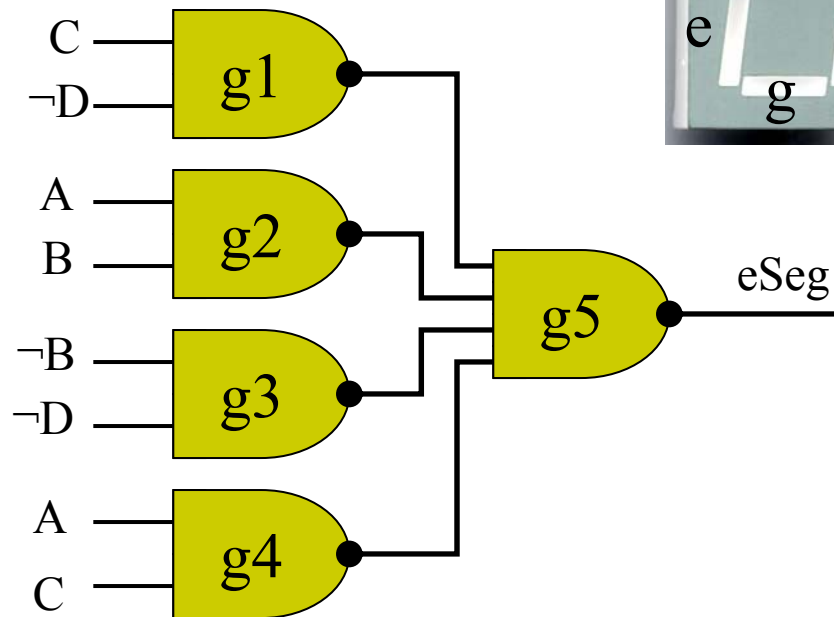
- HDL, tools, ...

# Hardware Description Language

- Verilog, VHDL, PALASM, CUPL, ABEL
- behavioral design
- logic structure
- timing
- hierarchy based on logic gates (no transistors)

# Verilog Example

```
module binaryToESeg;  
  wire eSeg, p1, p2, p3, p4;  
  reg A, B, C, D;  
  
  nand #1  
    g1 (p1, C, ~D),  
    g2 (p2, A, B),  
    g3 (p3, ~B, ~D),  
    g4 (p4, A, C),  
    g5 (eSeg, p1, p2, p3, p4);  
endmodule
```



example from Thomas & Moorby 2002

# Verilog Example – Structure

```
module binaryToESeg  
  (input A, B, C, D,  
   output reg eSeg);  
  
  wire p1, p2, p3, p4;  
  
  nand #1  
    g1 (p1, C, ~D),  
    g2 (p2, A, B),  
    g3 (p3, ~B, ~D),  
    g4 (p4, A, C),  
    g5 (eSeg, p1, p2, p3, p4);  
  
endmodule
```

definition = declaration, implementation, simulation

type of net

delta time

declaration and instantiation of five NAND gates

example from Thomas & Moorby 2002

# Verilog Example - Simulation

```
module binaryToESeg
  ...
  initial
    begin
      $monitor($time,,, “%b, %b, %b, %b = %b”,
                A, B, C, D, eSeg);
      #10 A = 0; B = 0; C = 0; D = 0;
      #10 D = 1;
      #10 C = 1; D = 0;
      #10 $finish;
    end
endmodule
```

example from Thomas & Moorby 2002

# Verilog Example - Behavior

```
module binaryToESeg
```

```
...
```

```
always @(A, B, C, D) begin
```

```
    eSeg = 1;
```

```
    if (~A & D)          eSeg = 0;
```

```
    if (~A & B & ~C)    eSeg = 0;
```

```
    if (~B & ~C & D)    eSeg = 0;
```

```
end
```

```
endmodule
```

whenever A, B, C, or D changed...



synthesis

behavioral description → → → → → structural description

example from Thomas & Moorby 2002

# Verilog

- hierarchical module definitions
- structural descriptions (gate level)
- simulation
  
- behavioral descriptions (procedural)
  
- synthesis (behavior  $\rightarrow$  structure)
- mix-and-match structure and behavior
- timing

# VHDL

- hierarchical module definitions
- structural descriptions (gate level)
  
- dataflow descriptions
- behavioral descriptions (procedural)
- simulation
- synthesis (behavior  $\rightarrow$  structure)
- mix-and-match structure, dataflow, and behavior
- timing

# VHDL

- and what, please, does VHDL stand for?

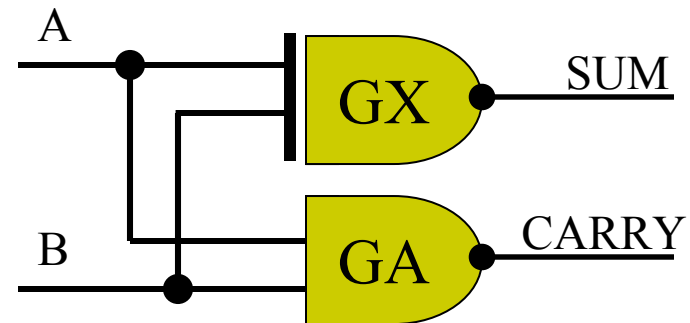
VHSIC HDL

- very nice, but...

Very High Speed Integrated Circuit

# VHDL Example – Declaration

```
entity HALF_ADDER is  
  port (A, B: in BIT; SUM, CARRY: out BIT);  
end HALF_ADDER;
```



example from Bhasker 1999

# VHDL Example – Structure

architecture HA\_STRUCTURE of HALF\_ADDER is

  component XOR2

    port (X, Y: in BIT; Z: out BIT);

  end component;

  component AND2

    port (L, M: in BIT; N: out BIT);

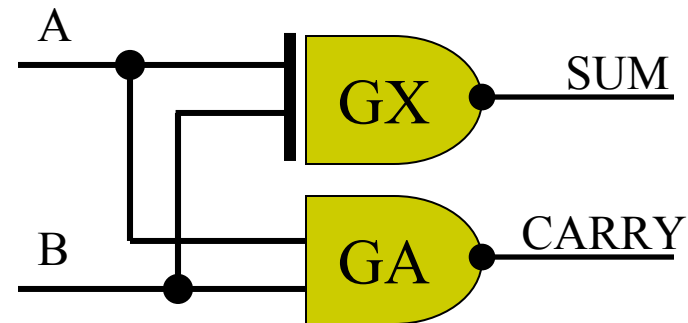
  end component;

begin

  GX: XOR2 port map (A, B, SUM);

  GA: AND2 port map (A, B, CARRY);

end HA\_STRUCTURE;



example from Bhasker 1999

# VHDL Example – Dataflow

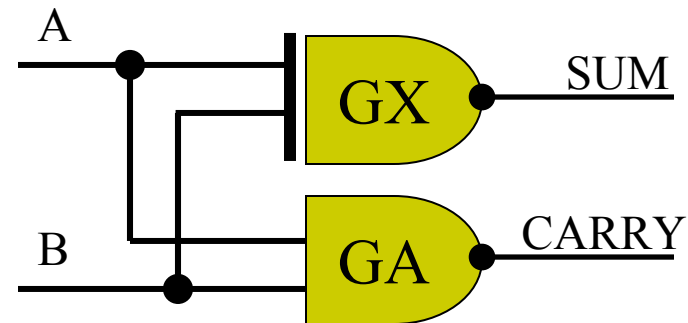
architecture HA\_DATAFLOW of HALF\_ADDER is

begin

SUM <= A xor B after 8ns;

CARRY <= A and B after 4ns;

end HA\_DATAFLOW;



example from Bhasker 1999

# VHDL Example – Behavior

architecture HA\_SEQUENTIAL of HALF\_ADDER is

begin

  process (A, B);

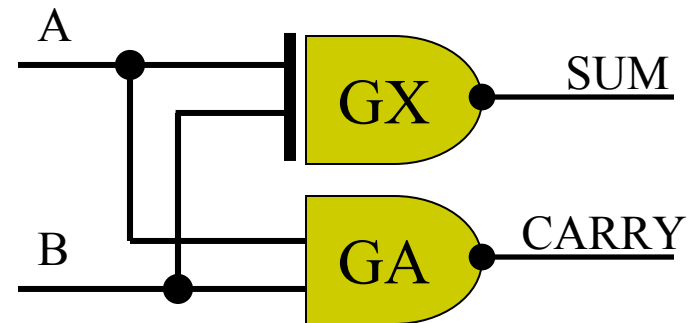
  begin

    SUM := A xor B;

    CARRY := A and B;

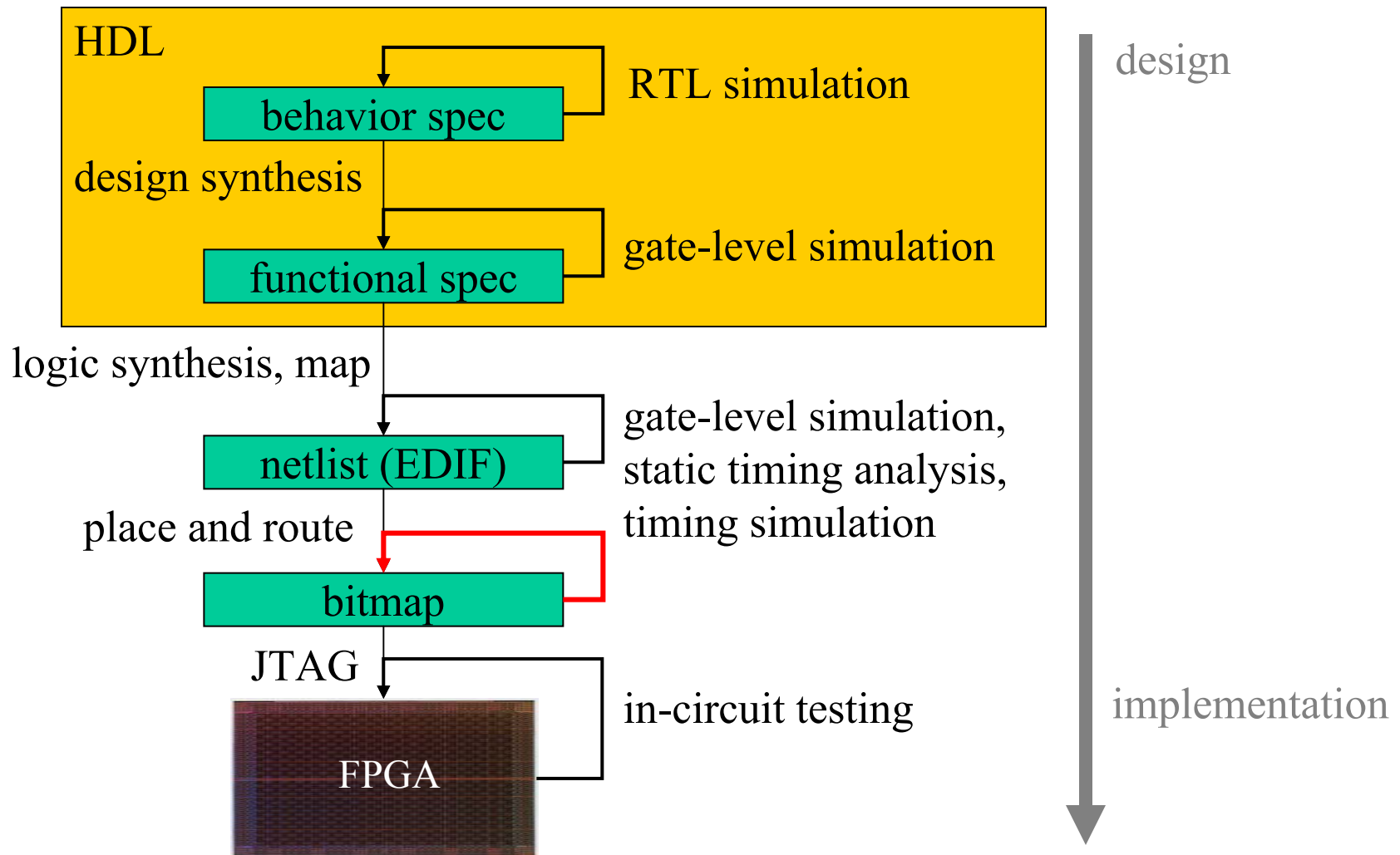
  end process;

end HA\_SEQUENTIAL;



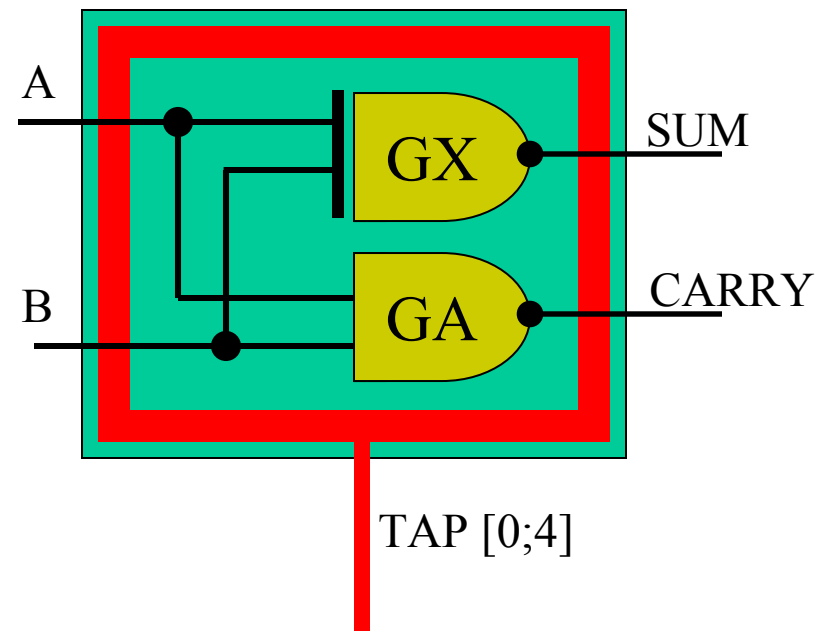
example from Bhasker 1999

# FPGA Design Flow



# JTAG, IEEE 1149.1

- boundary scan
- test access port
- ICs and printed boards
- printed board wires
- now: access to core
  - sw, data, registers
- now: routing



- <http://www-s.ti.com/sc/psheets/ssya002c/ssya002c.pdf>
- <http://www.embedded.com/story/OEG20021028S0049>

# Electronic Design Automation

- complete toolkits, integrating all steps
- includes hardware libraries/macros
  - memory, signals, multipliers, ..., soft DSP/GPP cores
- EDAs available from FPGA manufacturer and 3<sup>rd</sup> parties
- might use higher-level languages
  - sequential, procedural
  - compiled into HDL
  - optimization: parallelization, pipelining, loop unrolling

# EDA Tool Examples

- GNU General Public License tools gEDA (including ghdl);
- Streams-C (LANL);
- Altera SOPC Builder;
- Xilinx Platform Studio and iMPACT;
- Wishbone standard? (open-source);
- Celoxica Handel-C and SystemC;
- Impulse Accelerated Technologies Impulse C;
- Annapolis Micro Systems CoreFire FPGA Design Suite;
- Aldec Active HDL (a simulator);
- Mentor Graphics Catapult C and FPGA Advantage;
- SystemVerilog;
- SystemVHDL;
- Tanner S-Edit;
- Texas Instruments Code Composer Studio (CCStudio);
- Matlab/Simulink can target hardware platforms as output, can talk directly to CCStudio.

# FPGA Comments?

# MobileEye EyeQ

- CMOS ASIC, two 32-bit ARM946E GPPs with DSP extensions
- four 64-bit "Vision Computing Engines" (VCE)
- 110Mhz
- 8 DMA channels
- 288KB RAM
- UART
- two HDR CMOS camera interfaces

## VCE

- slave processors with dedicated function such as image scaling, warping, optical flow, convolution, pyramid creation, edge detection, and color histogram creation.

# Sarnoff Acadia

- ASIC 1M gates
- core at 108MHz
- 576k bit SRAM
- modular layout around cross switch
- digital video I/O (no codecs)

# Other Hardware

- Mobile Phones
  - TI C55xx DSPs
  - Qualcomm MSM6550
- Framegrabbers
  - Matrox
  - Data Translation
- Smart Cameras
  - Wayne Wolf
  - Bernhard Rinner
- GPU
  - commodity
  - highly parallel
  - not (yet) meant for non-CG computation
    - hard to program, need hacks
    - no ideal data paths

# InterSense IS-1200 VisTracker

Foxlin, Naimark, VR'03

# Tutorial Overview

- Motivation
- Media Processors
- FPGAs
- ASICs
- Porting to Embedded Platforms
- Algorithmic Considerations
- Resources, Conclusions, Future...

# Porting to an Embedded Platform

- algorithmic considerations
  - is it the right one? keep options open!
- code optimization
  - loop unrolling
  - minimize branching
  - choice of data type
  - optimize wisely
- profiling
- algorithm simulation
- architecture simulation
  - CPU issue width
  - function units busy?
- careful split: low-level, multi-purpose hardware (reuse, economy of scale!), high-level software

# Algorithmic Considerations

suitability and potential speedup when ported to embedded platform

- high data rates, few instructions
- streams/sequential data access, as opposed to random access
- multiple, largely independent streams of data
- data packet size is fixed, or at least bounded by a small number
- stream computations can be broken up into pipeline stages, i.e. the same (possible parameterized) computations independent of prior stage's output
- fixed-precision data (integer or fix-point fractions)
- parallizable at instruction and module level, that is, little between-stream communication necessary

# Data Interdependency, Image Space

pixel processing	n-pass	fixed-size block access	data-independent, global access	data-dependent, random access
<ul style="list-style-type: none"> <li>•LUT</li> <li>•threshold</li> <li>•color space conversion</li> <li>•brightness correction</li> <li>•arithmetic operations*</li> <li>•logic operations*</li> </ul>	<ul style="list-style-type: none"> <li>•count, min, max, avg, stddev</li> <li>•histograms</li> <li>•Hough transforms</li> </ul>	<ul style="list-style-type: none"> <li>•morphology*</li> <li>•convolution, filtering*,</li> <li>•pyramids</li> </ul>	<ul style="list-style-type: none"> <li>•Viola-Jones</li> <li>•warping, remapping</li> </ul>	<ul style="list-style-type: none"> <li>•flood fill</li> <li>•contour?</li> </ul>

•for multiple images: need to consider additional memory accesses; similarly, operations that can not be trivially performed in-place need extra care

# Porting to Embedded Platform

- Fast Algorithms
  - FFT, fast convolution and correlation, fast morphology
- Reduced dimensionality
  - Principal Component Analysis (PCA)
  - Pythagorean trick for PCA
- Fixed-Point Arithmetic
  - Fixed-Point PCA
- Reduced data transfers
  - Quadruple memory

# Real-Time Tricks: Fixed-Pt PCA

- Eigenspace projections are dot-products
- Great for fixed-point implementation
- Speed-up on TI DM-642: 70x
- Example: From 17 ms down to 240  $\mu$ s
- Scale, round, do work, rescale back
- Scale factor can be found off-line

$$\tilde{\phi}_k = \text{round} \left[ \frac{127\phi_k}{\max_i |\phi_k^{(i)}|} \right] \quad (k = 1, \dots, r)$$

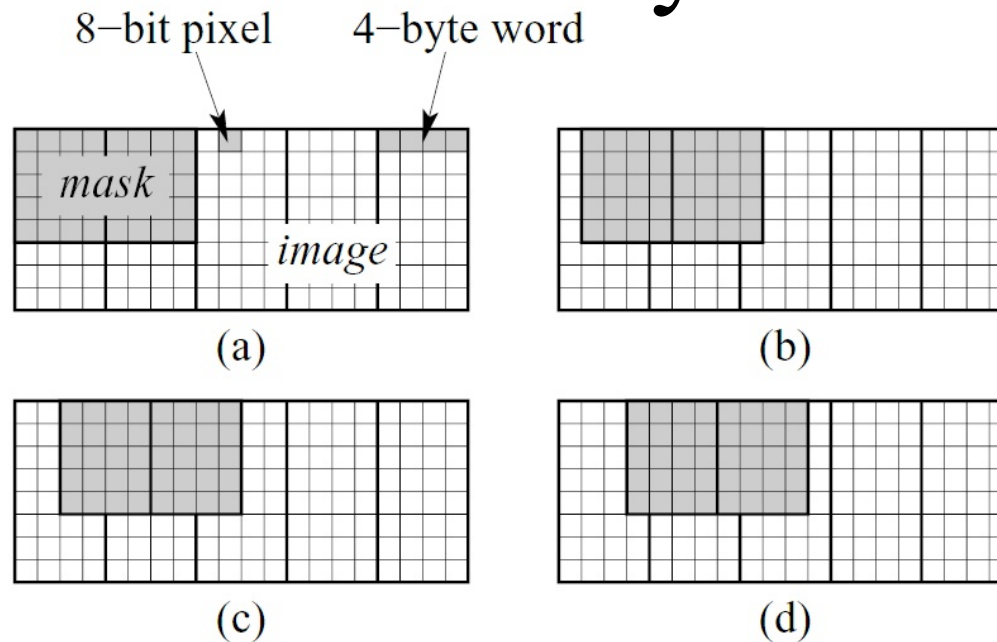
# Real-Time Tricks: Pythagorean Reconstr.

$$e^2 = \|x - \hat{x}\|^2 = \sum_{i=1}^m (x_i - \hat{x}_i)^2$$

$$e^2 = \|x - \mu\|^2 - \sum_{k=1}^r w_k^2$$

- Reconstruction Error calculation
- Roughly 2x faster
- Total PCA speed-up:  $\approx 140x$

# Real-Time Tricks: Quadruple Memory



- SIMD ops require data to be aligned
- Reduced data transfers from 2.5 MB to 25 KB
- Speed-up: From 16.5 ms to 0.75 ms

# Real-Time Tricks: Fast Morphology

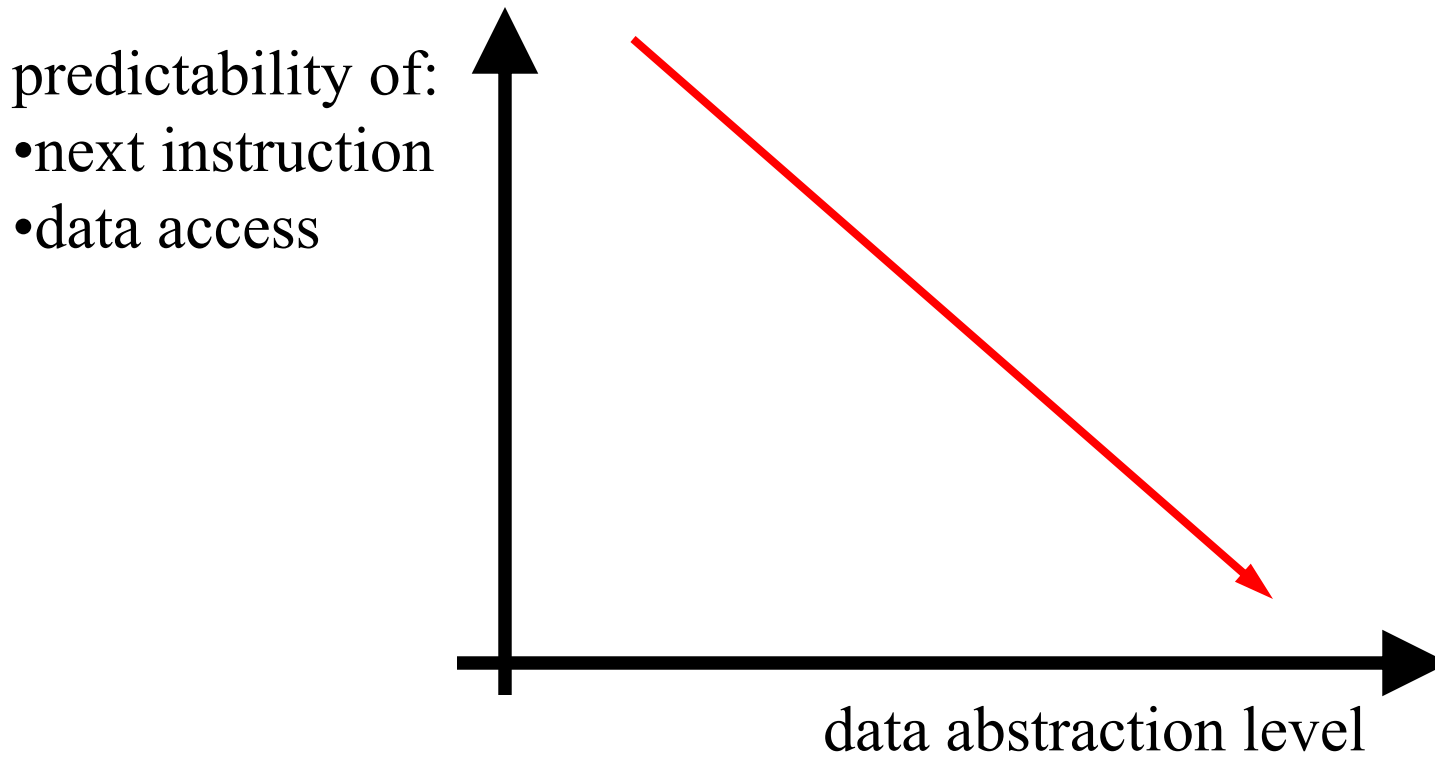
$$A \oplus S = \text{bin}_{1/2}(A * S)$$

- Media processors are not good with nonlinear ops
- Dilation in terms of convolution and thresholding
- Example: 30 ms down to 1 ms

# Tutorial Overview

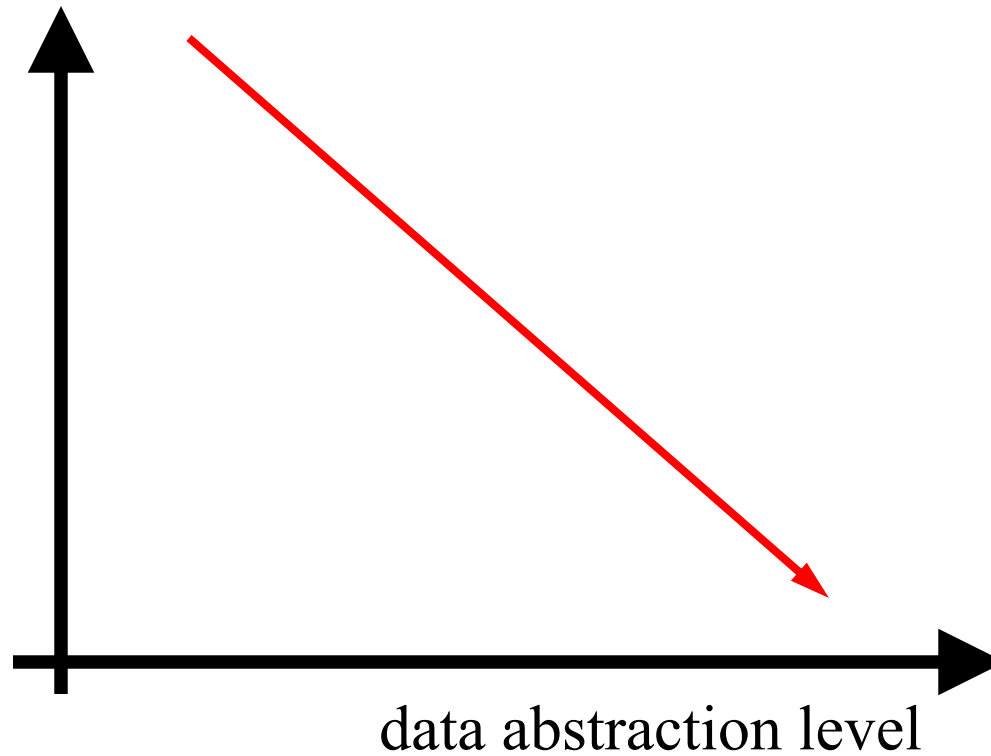
- Motivation
- Media Processors
- FPGAs
- ASICs
- Porting to Embedded Platforms
- Algorithmic Considerations
- Resources, Conclusions, Future...

# Abstraction and Predictability

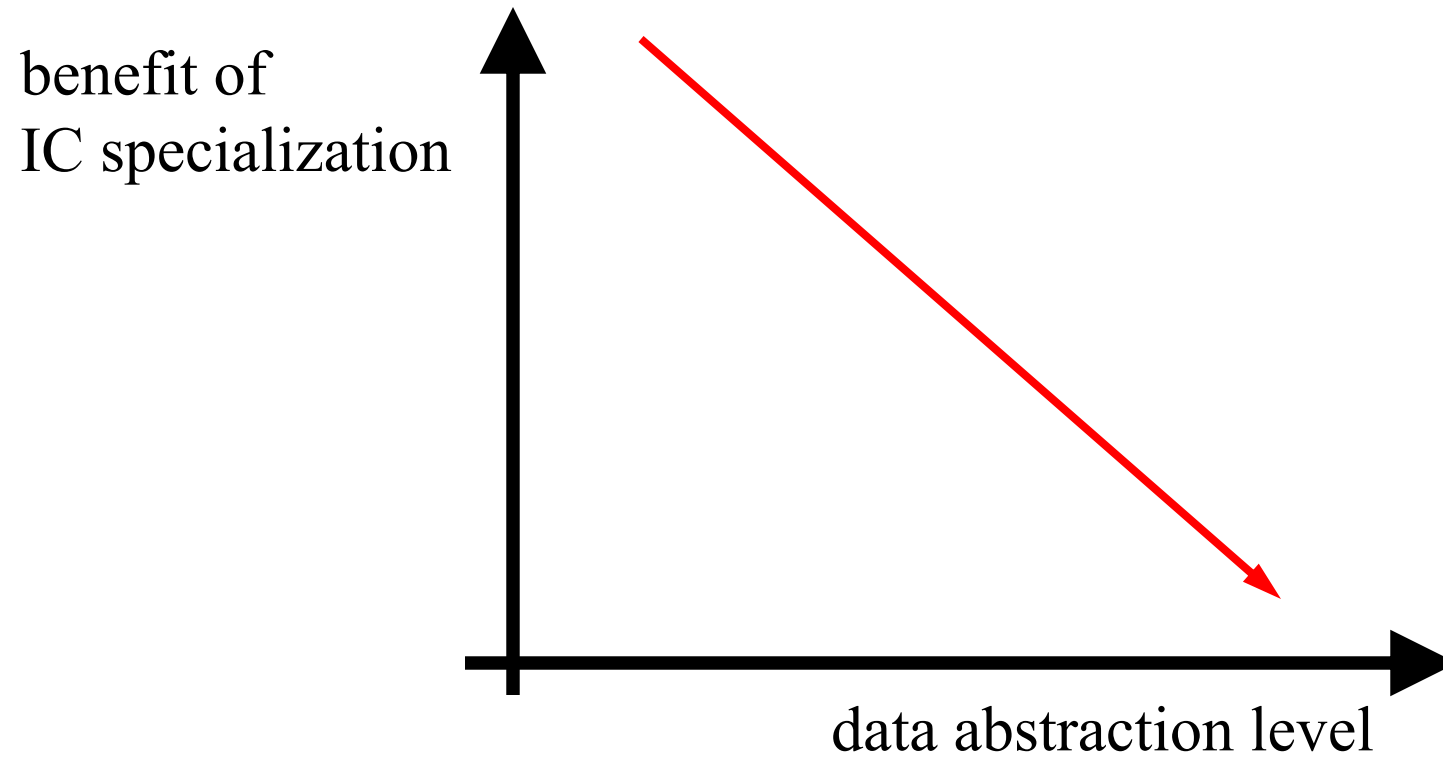


# Abstraction and Repetitiveness

instruction  
repetitiveness  
& uniformity  
(parallelism,  
SIMD...)



# Abstraction and Microcontrollers



# HW and Algorithms

- Porting vs. development
- Exploit synergy between algorithms and HW
- Problem: iterative process, no convergence guaranteed
- More art than science
- Rewards are great

# resources, where to go from here

- if you are a professional
  - professional training workshops (specific to chips usually)
  - where to find skilled employees (CS/CE/EE) – need team?
  - offer a job mail list?
- if you are a student
  - resources, software libraries, contacts for research-minded industries, internships, joint projects, universities known to do CV on embedded, books, textbooks, online material

➔ show resources

# Vision for the Future

- IP/CV instructions on GPU, on GPP, some embedded
- vision chips (Acadia-style) more common
- CMOS imagers with computation in image plane
  - but: conflicting area usage, co-chip good enough?
- wireless digital video transmission
- unleashing CV applications to consumers
  - still image processing (stitching, 3D models, ...)
  - HDR images and video
  - video post-processing
  - automatic “content” creation
- frameless imaging pipeline → compression, speed, resolution

# Acknowledgments

- Mark Belding
- Andreas Doblender
- Ryan Kastner
- Vladimir Pavlovic
- Bernhard Rinner

*fin*

# Processor Overview

manufacturer & name	type	(max) speed	ROM	RAM	power	cost
Altera Excalibur	ARM9+FPGA	?	?	?	?	?
Analog Devices Blackfin BF533	16-DSP/i	756MHz	?	80kB I + 64kB D	?	\$20
Centrality Atlas	ARM9+DSP	240MHz+120MHz	?	?	?	?
Cirrus Logic Maverick EP9307	ARM9+DSP/if	200MHz	?	16kB I + 16kB D	?	?
Freescale i.MX31	ARM11+DSP*/f	665MHz	?	16kB I + 16kB D + 128kB L2	?	?
Freescale MSC7110	16-DSP/i	266MHz	8kB	64kB	?	\$12.65
Freescale MSC8144	16-DSP/i	4x1GHz	96kB	512kB+10MB	?	\$180
Intel XScale	ARM9+DSP	800MHz	?	32kB I + 32kB D	40-900mW	?
Intel PXA27x	ARM9+DSP	624MHz	?	32kB I + 32kB D	500mW	?
Infineon TC1775	32-DSP	?	?	?	?	?
Lucent Venus?	DSP	?	?	?	?	?
TI OMAP2420	ARM11&DSP/i	330MHz	?	32kB I + 32kB D + 5Mb	?	?
TI TMS320C6416T	32-DSP/i	600MHz	?	16kB I + 16kB D	2W	?
TI TMS320C67x	32-DSP/f	300MHz	?	?	?	\$13-100
Xilinx Virtex 4	FPGA	?	-	1,392kb dis + 6,048kb blk	?	?
Xilinx Virtex 5	FPGA	550MHz	-	3,420kb dis + 10,368kb blk	?	?
Zilog Z86295	8-Z8&16-DSP	40MHz	?	256kB	?	?