ISWCS’07

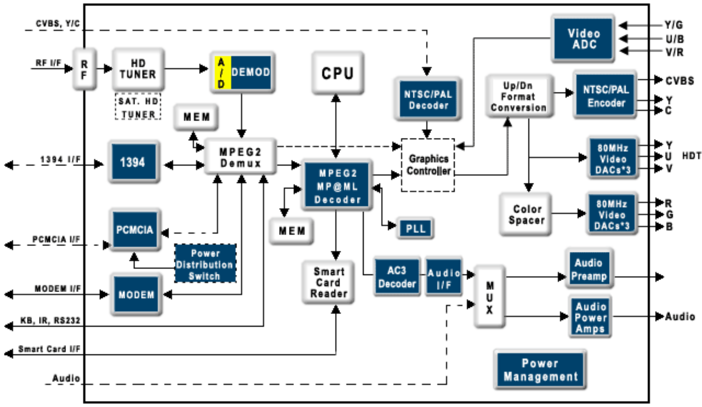
Development of complex wireless systems requires new development technologies

Dr. Hans Martin Ritt
Senior Teamleader Application Engineering
Martin.Ritt@mathworks.com

The MathWorks GmbH
Adalperostr. 45
D-85737 Ismaning (Munich)

Industry Trends:
Wireless Communication Systems

- Multiple target technologies
  - Digital and analogue hardware
  - Embedded software/firmware
  - Shifting partitioning boundaries

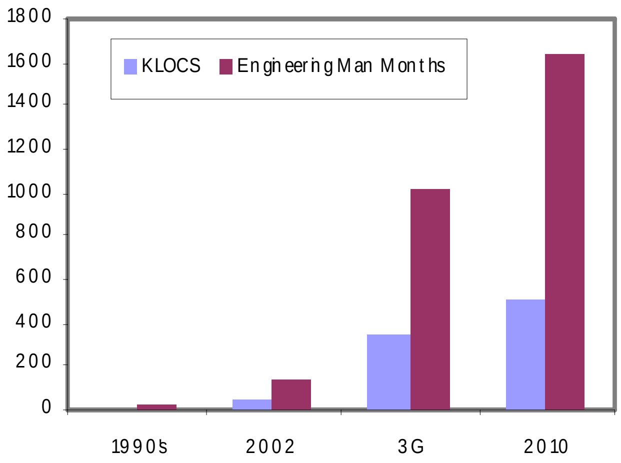


- Performance, cost, development time trade-offs

- Process challenges
  - Iterate between algorithms and implementation
  - IP portability and reuse
  - Increasing cost of design flaws

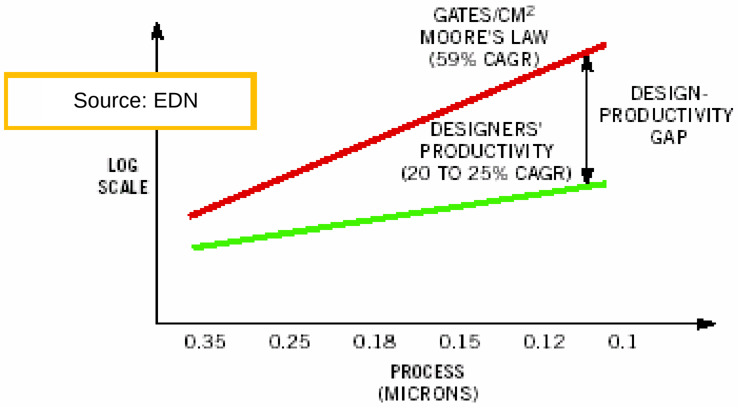
Systems Are Becoming More Complex

Software is becoming more complex



Source: Bob Frankel, Chief SW strategist & TI Fellow

Hardware is becoming more complex



The compound growth rate of semiconductor manufacturing continues to outpace designers' productivity.

- Just adding more head count doesn’t help:
  - “Brooks' Law”, often summarized as:
    “Nine women cannot have a baby in one month”

Theses

1. Increasing the level of abstraction increases the productivity
   - Do not reinvent the wheel
   - Challenge: trade-off development effort against optimal performance
2. Start to adapt the "computer" to the application
   - Single source cannot mean single language
   - There is not a single best "language" for everything
3. Iterative development
   - Early verification
   - Flexible partitioning
   - Challenge: Moving between abstraction levels

Increase level of abstraction
MATLAB Vs C/C++

- **C Code**

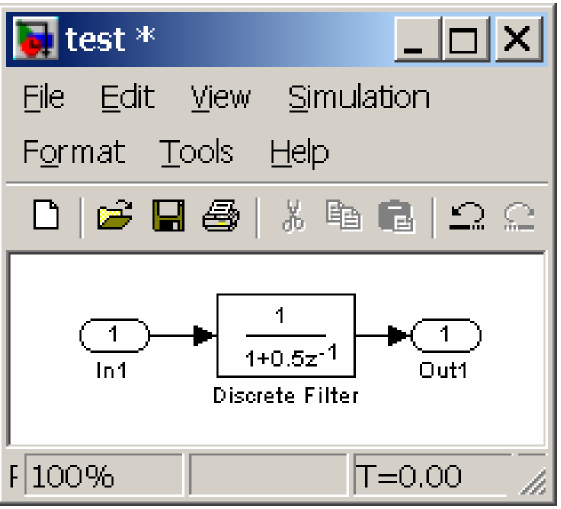
```
Bits spread=addChips(diffOut[slice(i,1)]);

Bits
IEEE802_11b_Transmitter::addChips(const Bits& input) {
    Bits spreadOut(input.size()*Ns,false);
    for (int i=0;i<input.size();++i){
        for(int j=0; j<11; ++j) {
            spreadOut[i*Ns+4*j]= m_chip[j]^input[i];
        }
    }
   return spreadOut;
}
```

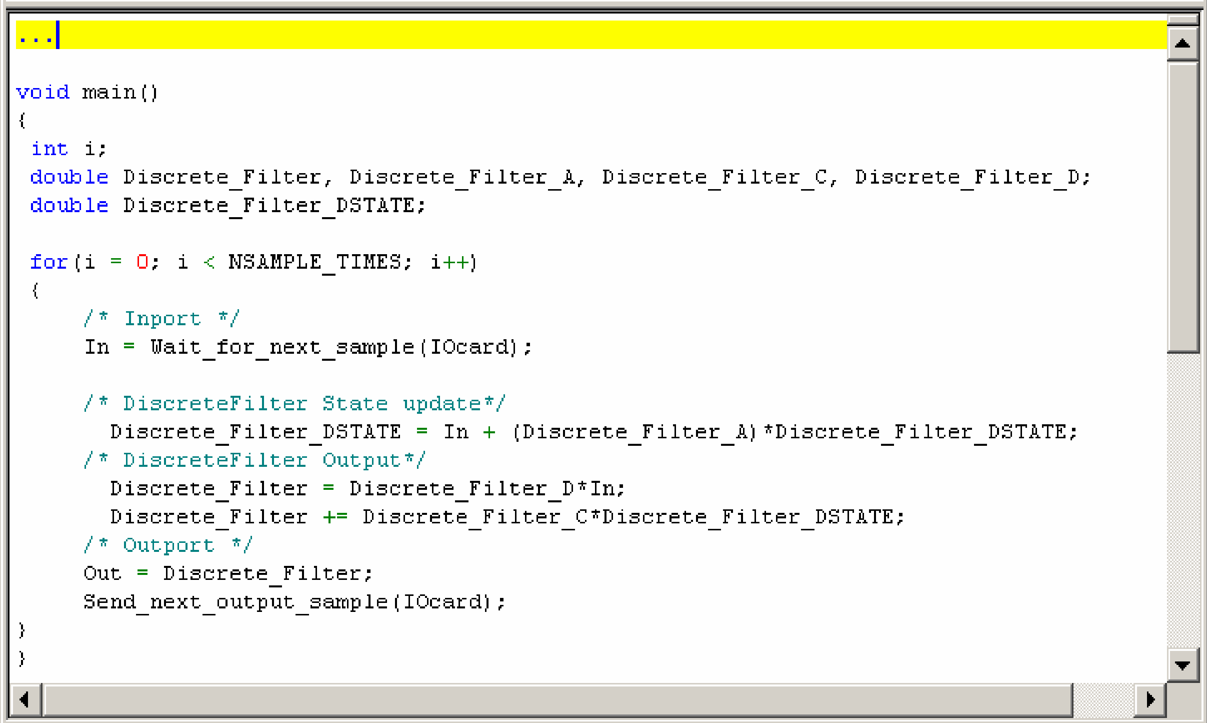
- **M Code**

```
Tx_chips=reshape(Barker*Tx_symbols',[],1);
Tx_samples(1:Samples_per_chip:end)=Tx_chips;
```

Abstraction: Graphical Design vs. Hand-coding

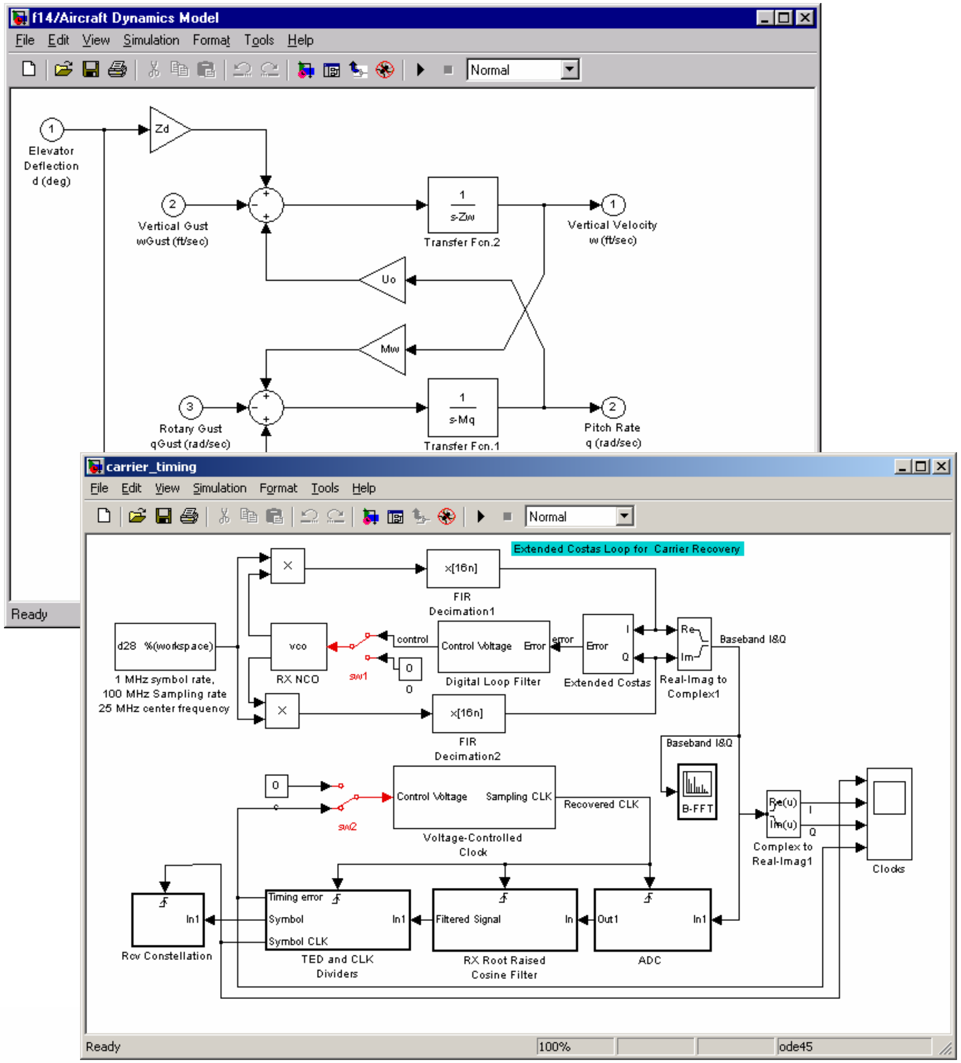
- Simulation of graphical model
  - automatic synchronisation of calculations
  - handling of signals/data
- Easy start with prebuild standard-functions





```
...

void main()
{
 int i;
 double Discrete_Filter, Discrete_Filter_A, Discrete_Filter_C, Discrete_Filter_D;
 double Discrete_Filter_DSTATE;

 for(i = 0; i < NSAMPLE_TIMES; i++)
 {
     /* Inport */
     In = Wait_for_next_sample(IOcard);

     /* DiscreteFilter State update*/
       Discrete_Filter_DSTATE = In + (Discrete_Filter_A)*Discrete_Filter_DSTATE;
     /* DiscreteFilter Output*/
       Discrete_Filter = Discrete_Filter_D*In;
       Discrete_Filter += Discrete_Filter_C*Discrete_Filter_DSTATE;
     /* Outport */
     Out = Discrete_Filter;
     Send_next_output_sample(IOcard);
}
}
```

Complex Timing and Concurrency


f14/Aircraft Dynamics Model
File Edit View Simulation Format Tools Help
Normal
1 Elevator Deflection d (deg)
Zd
2 Vertical Gust wGust (ft/sec)
1/s-Zw Transfer Fcn.2
1 Vertical Velocity w (ft/sec)
Uo
Mw
3 Rotary Gust qGust (rad/sec)
1/s-Mq Transfer Fcn.1
2 Pitch Rate q (rad/sec)
Ready
carrier_timing
File Edit View Simulation Format Tools Help
Normal
Extended Costas Loop for Carrier Recovery
x[16n] FIR Decimation1
d28 %(workspace)
1 MHz symbol rate, 100 MHz Sampling rate 25 MHz center frequency
vco RX NCO
sw1
control
0
Control Voltage Error Digital Loop Filter
error
Error I Q Extended Costas
Re Im Real-Imag to Complex1
Baseband I&Q
x[16n] FIR Decimation2
Baseband I&Q
B-FFT
0
sw2
Control Voltage Sampling CLK Voltage-Controlled Clock
Recovered CLK
Re(u) Im(u) I Q Complex to Real-Imag1
Clocks
In1 Rcv Constellation
Timing error Symbol Symbol CLK In1 TED and CLK Dividers
Filtered Signal In RX Root Raised Cosine Filter
Out1 In1 ADC
Ready
100%
ode45

- Complex timing
  - Feedback
  - Asynchronous edge triggered blocks
  - Multi-rate digital with arbitrary sample rates
- Concurrency
  - True expression of parallelism
  - Important for whole system or hardware sub-system design

Possible pitfalls

- Model-Based Design including graphical entry is more than graphical programming

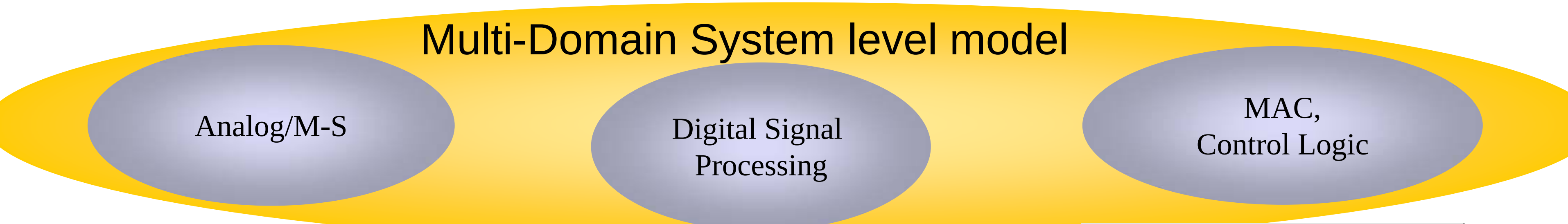
- There is no productivity gain, if you
  - draw what you would write in a program
  - try to tweak the code generator to generate the code you already have in your mind

- Long time goal of this next abstraction level is to eliminate the need to review in detail code in C, HDL, etc…(like today ASM, Gate-level,…)

Theses

1. Increasing the level of abstraction increases the productivity
   - Do not reinvent the wheel
   - Challenge: trade-off development effort against optimal performance
2. Start to adapt the "computer" to the application
   - Single source cannot mean single language
   - There is not a single best "language" for everything
3. Iterative development
   - Early verification
   - Flexible partitioning
   - Challenge: Moving between abstraction levels

Model Different Components different

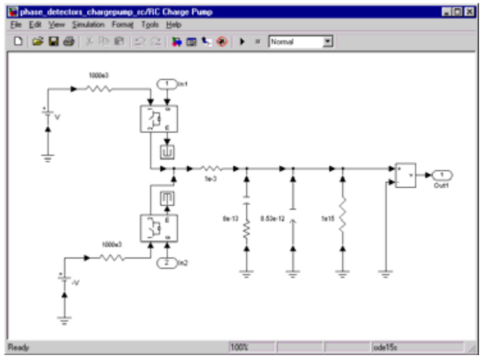


Multi-Domain System level model

Analog/M-S

Digital Signal Processing

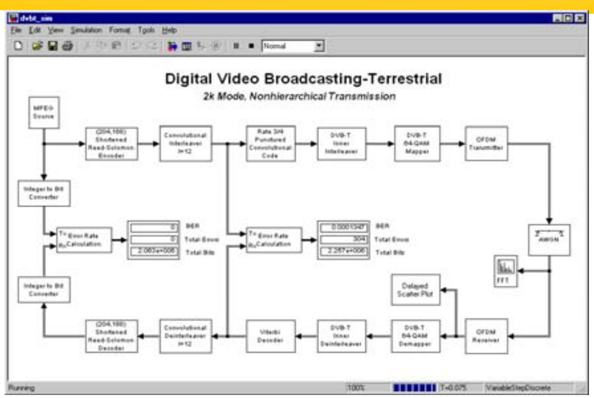
MAC, Control Logic



phase_detectors_chargepump_sc/RC Charge Pump
File Edit View Simulation Format Tools Help
Normal
1000e3
1 In1
V
1e-3
8e-13
8.53e-12
1e15
1 Out1
1000e3
2 In2
-V
Ready
100%
ode15s

Analog/Mixed-Signal

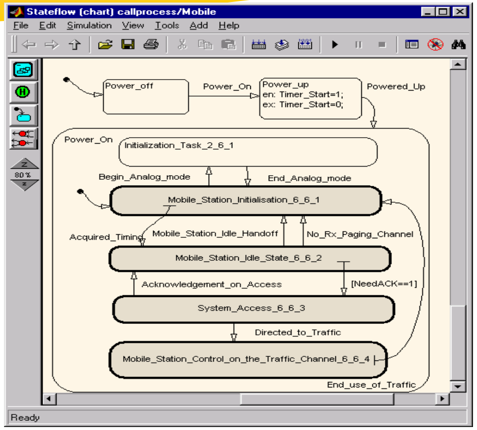
- PLLs, data converters
- Continuous time, variable-step ODE solvers



dvbt_sim
File Edit View Simulation Format Tools Help
Normal
Digital Video Broadcasting-Terrestrial
2k Mode, Nonhierarchical Transmission
MPEG Source
(204,188) Shortened Reed-Solomon Encoder
Convolutional Interleaver I=12
Rate 3/4 Punctured Convolutional Code
DVB-T Inner Interleaver
DVB-T 64-QAM Mapper
OFDM Transmitter
Integer to Bit Converter
Tx Rx Error Rate Calculation
0
0
2.063e+006
BER
Total Errors
Total Bits
Tx Rx Error Rate Calculation
0.0001347
304
2.257e+006
BER
Total Errors
Total Bits
AWGN
FFT
Delayed Scatter Plot
Integer to Bit Converter
(204,188) Shortened Reed-Solomon Decoder
Convolutional Deinterleaver I=12
Viterbi Decoder
DVB-T Inner Deinterleaver
DVB-T 64-QAM Demapper
OFDM Receiver
Running
100%
T=0.075
VariableStepDiscrete

DSP Baseband

- Discrete time, fast frame-based processing. Bit-true cycle accurate.

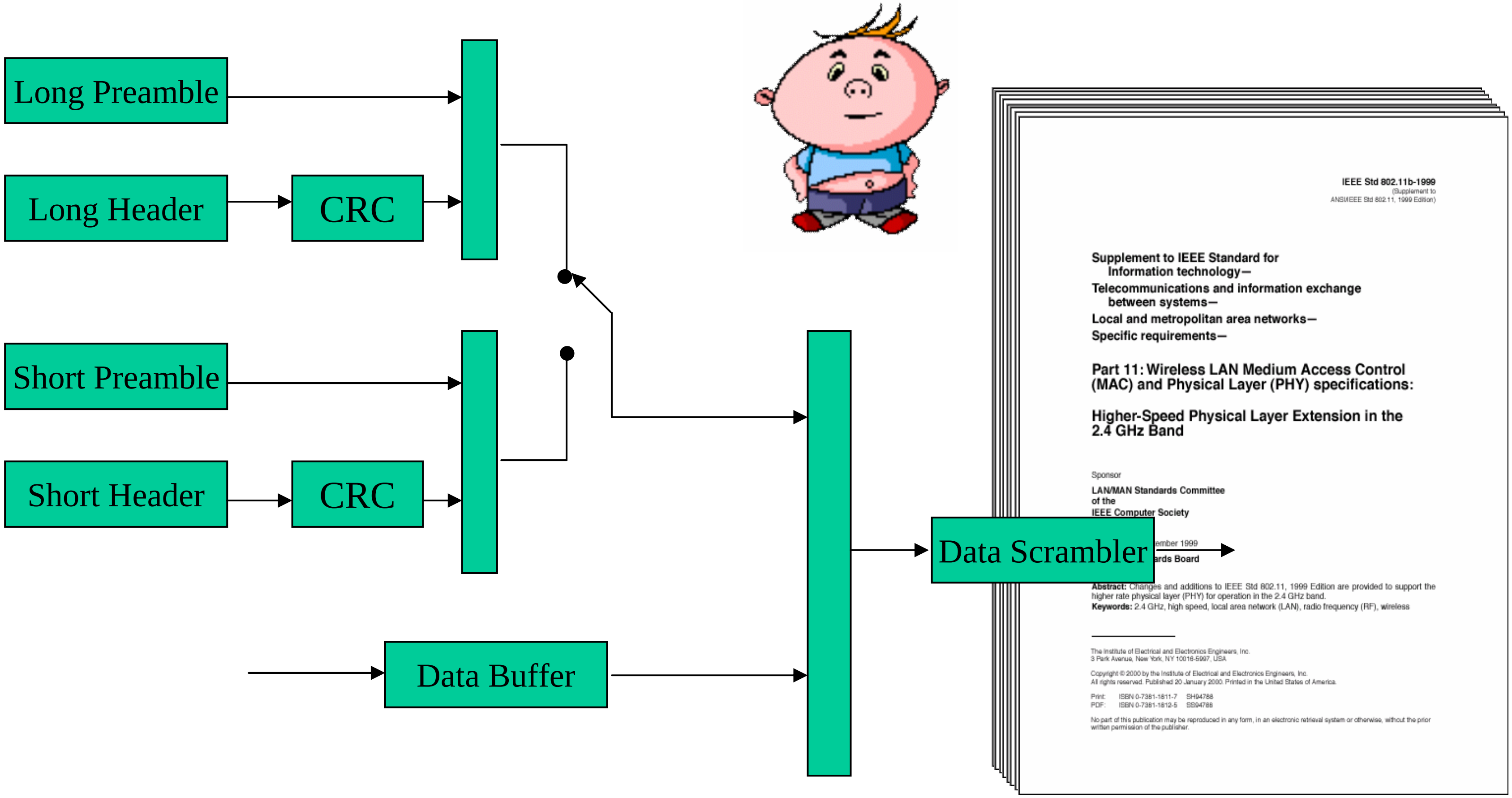


Stateflow (chart) callprocess/Mobile
File Edit Simulation View Tools Add Help
80 %
Power_off
Power_On
Power_up
en: Timer_Start=1;
ex: Timer_Start=0;
Powered_Up
Power_On
Initialization_Task_2_6_1
Begin_Analog_mode
End_Analog_mode
Mobile_Station_Initialisation_6_6_1
Acquired_Timing
Mobile_Station_Idle_Handoff
No_Rx_Paging_Channel
Mobile_Station_Idle_State_6_6_2
Acknowledgement_on_Access
[NeedACK==1]
System_Access_6_6_3
Directed_to_Traffic
Mobile_Station_Control_on_the_Traffic_Channel_6_6_4
End_use_of_Traffic
Ready

MAC layer/Data Link Layer

- Simple protocols, acknowledgement schemes
- Reactive or event driven state machines

System Architect: “For some pieces a block diagram is appropriate…”



IEEE Std 802.11b-1999
(Supplement to
ANSI/IEEE Std 802.11, 1999 Edition)

Supplement to IEEE Standard for
Information technology—
Telecommunications and information exchange
between systems—
Local and metropolitan area networks—
Specific requirements—

Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications:

Higher-Speed Physical Layer Extension in the 2.4 GHz Band

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

ember 1999

rds Board

Abstract: Changes and additions to IEEE Std 802.11, 1999 Edition are provided to support the higher rate physical layer (PHY) for operation in the 2.4 GHz band.
Keywords: 2.4 GHz, high speed, local area network (LAN), radio frequency (RF), wireless

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2000 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 20 January 2000. Printed in the United States of America.

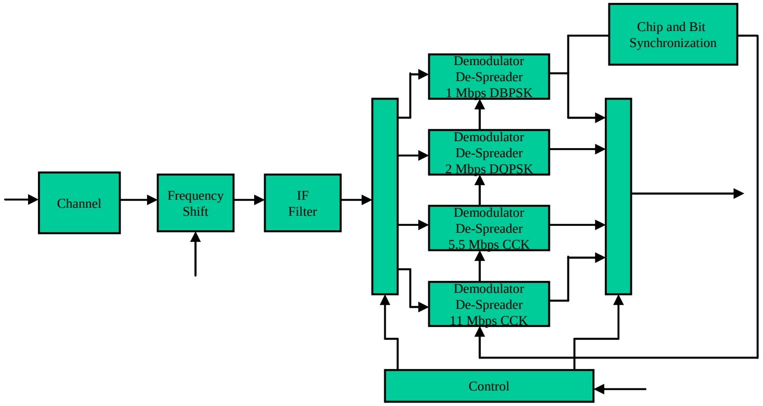
Print: ISBN 0-7381-1811-7 SH94788
PDF: ISBN 0-7381-1812-5 SS94788

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

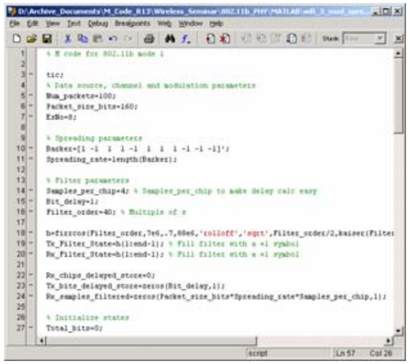
R. Durrant, Intel, 802.11b system block diagram, Jan 2002.

Limitations of C and M for System Design

- No architecture information
  - Can only model a pipeline
  - Can't describe a real system
- No timing information
  - Can only model uniform Fs
  - Difficult to model delays
  - Must manually handle state
  - Can't model A/M-S
  - Difficult to model Rx algorithms
- For system level models this is critical

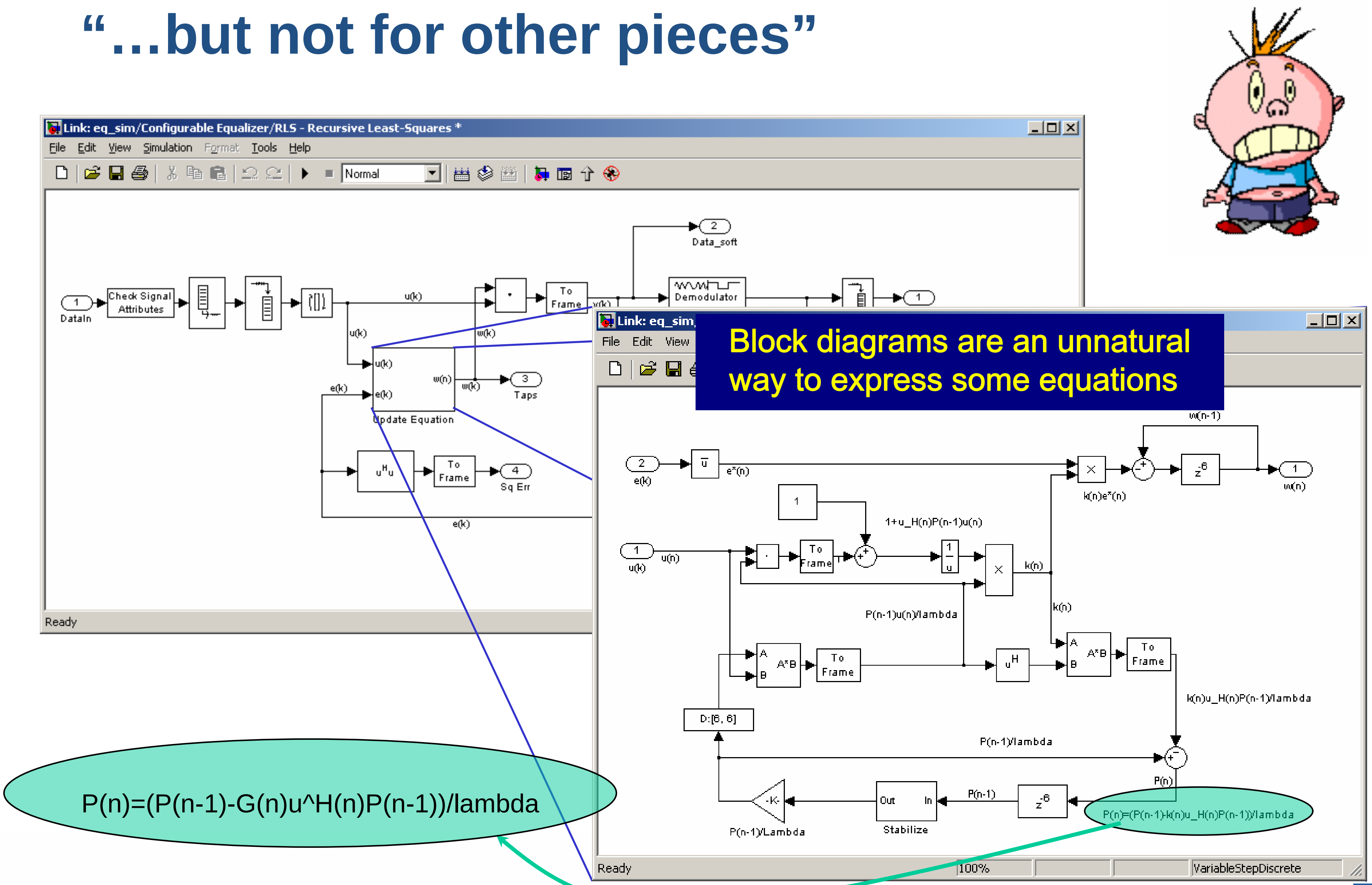

Channel
Frequency Shift
IF Filter
Demodulator De-Spreader 1 Mbps DBPSK
Demodulator De-Spreader 2 Mbps DQPSK
Demodulator De-Spreader 5.5 Mbps CCK
Demodulator De-Spreader 11 Mbps CCK
Chip and Bit Synchronization
Control

||


D:\Archive_Documents\M_Code_R13\Wireless_Seminar\802.11b_PHY\MATLAB\
File
Edit
View
Text
Debug
Breakpoints
Web
Window
Help
Stack
script
Ln 57
Col 26

```
% M code for 802.11b mode 1

tic;
% Data source, channel and modulation parameters
Num_packets=100;
Packet_size_bits=160;
EsNo=8;

% Spreading parameters
Barker=[1 -1  1  1 -1  1  1  1 -1 -1 -1]';
Spreading_rate=length(Barker);

% Filter parameters
Samples_per_chip=4; % Samples_per_chip to make delay calc easy
Bit_delay=1;
Filter_order=40; % Multiple of 8

h=firrcos(Filter_order,7e6,.7,88e6,'rolloff','sqrt',Filter_order/2,kaiser(Filte
Tx_Filter_State=h(1:end-1); % Fill filter with a +1 symbol
Rx_Filter_State=h(1:end-1); % Fill filter with a +1 symbol

Rx_chips_delayed_store=0;
Tx_bits_delayed_store=zeros(Bit_delay,1);
Rx_samples_filtered=zeros(Packet_size_bits*Spreading_rate*Samples_per_chip,1);

% Initialize states
Total_bits=0;
```

"…but not for other pieces"



Link: eq_sim/Configurable Equalizer/RLS - Recursive Least-Squares *

Normal

Ready

Ready 100% VariableStepDiscrete

Block diagrams are an unnatural way to express some equations

P(n)=(P(n-1)-G(n)u^H(n)P(n-1))/lambda

“For these pieces, equations are better…”

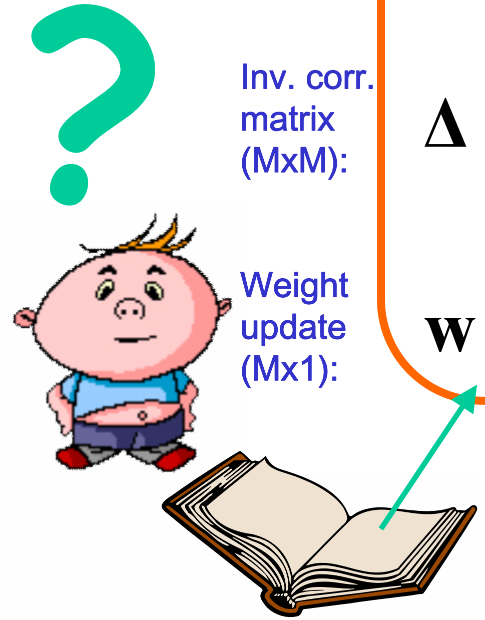
Output: $$y_n = \mathbf{w}^H \mathbf{u}$$

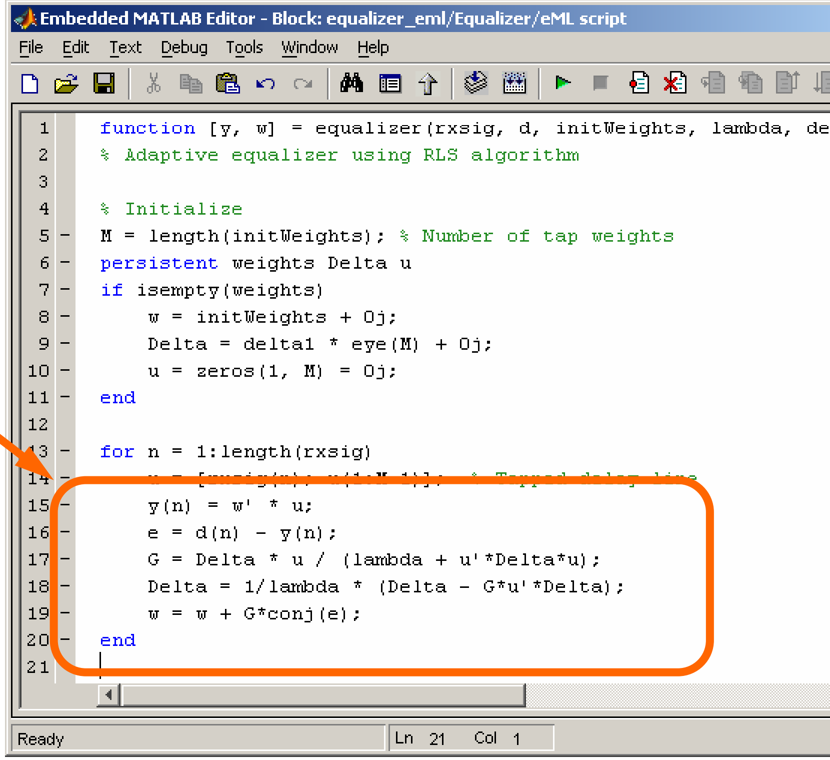
Error: $$e = d_n - y_n$$

Gain vector (Mx1): $$\mathbf{G} = \frac{\Delta \mathbf{u}}{\lambda + \mathbf{u}^H \Delta \mathbf{u}}$$

Inv. corr. matrix (MxM): $$\Delta \leftarrow \frac{1}{\lambda}\left(\Delta - \mathbf{G}\mathbf{u}^H \Delta\right)$$

Weight update (Mx1): $$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{G}e^*$$



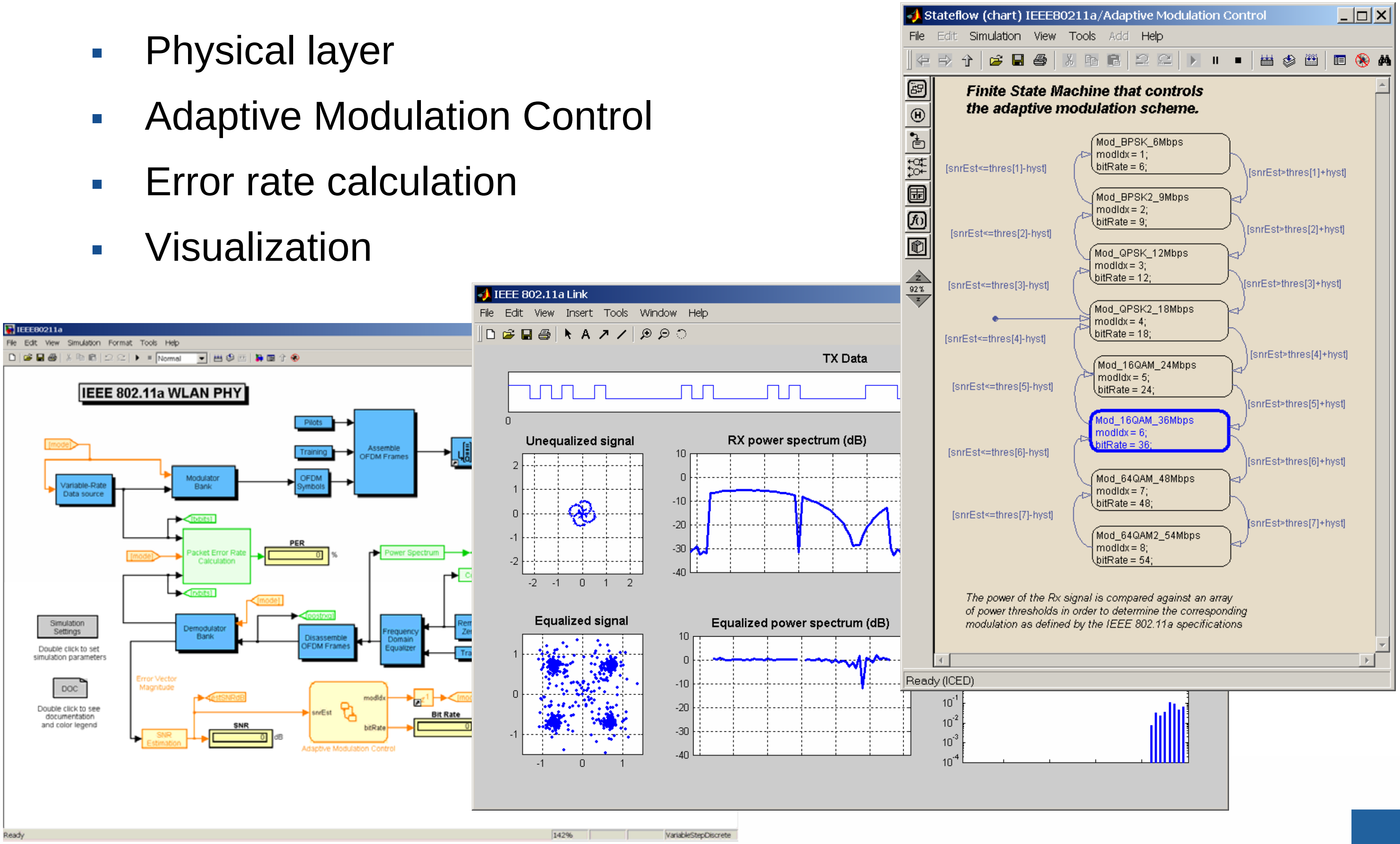


Embedded MATLAB Editor - Block: equalizer_eml/Equalizer/eML script
File Edit Text Debug Tools Window Help

```
function [y, w] = equalizer(rxsig, d, initWeights, lambda, de
% Adaptive equalizer using RLS algorithm

% Initialize
M = length(initWeights); % Number of tap weights
persistent weights Delta u
if isempty(weights)
    w = initWeights + 0j;
    Delta = delta1 * eye(M) + 0j;
    u = zeros(1, M) = 0j;
end

for n = 1:length(rxsig)
    u = [rxsig(n); u(1:M-1)];  % Tapped delay line
    y(n) = w' * u;
    e = d(n) - y(n);
    G = Delta * u / (lambda + u'*Delta*u);
    Delta = 1/lambda * (Delta - G*u'*Delta);
    w = w + G*conj(e);
end
```

Statechart: 802.11a Adaptive modulation

- Physical layer
- Adaptive Modulation Control
- Error rate calculation
- Visualization



Stateflow (chart) IEEE80211a/Adaptive Modulation Control

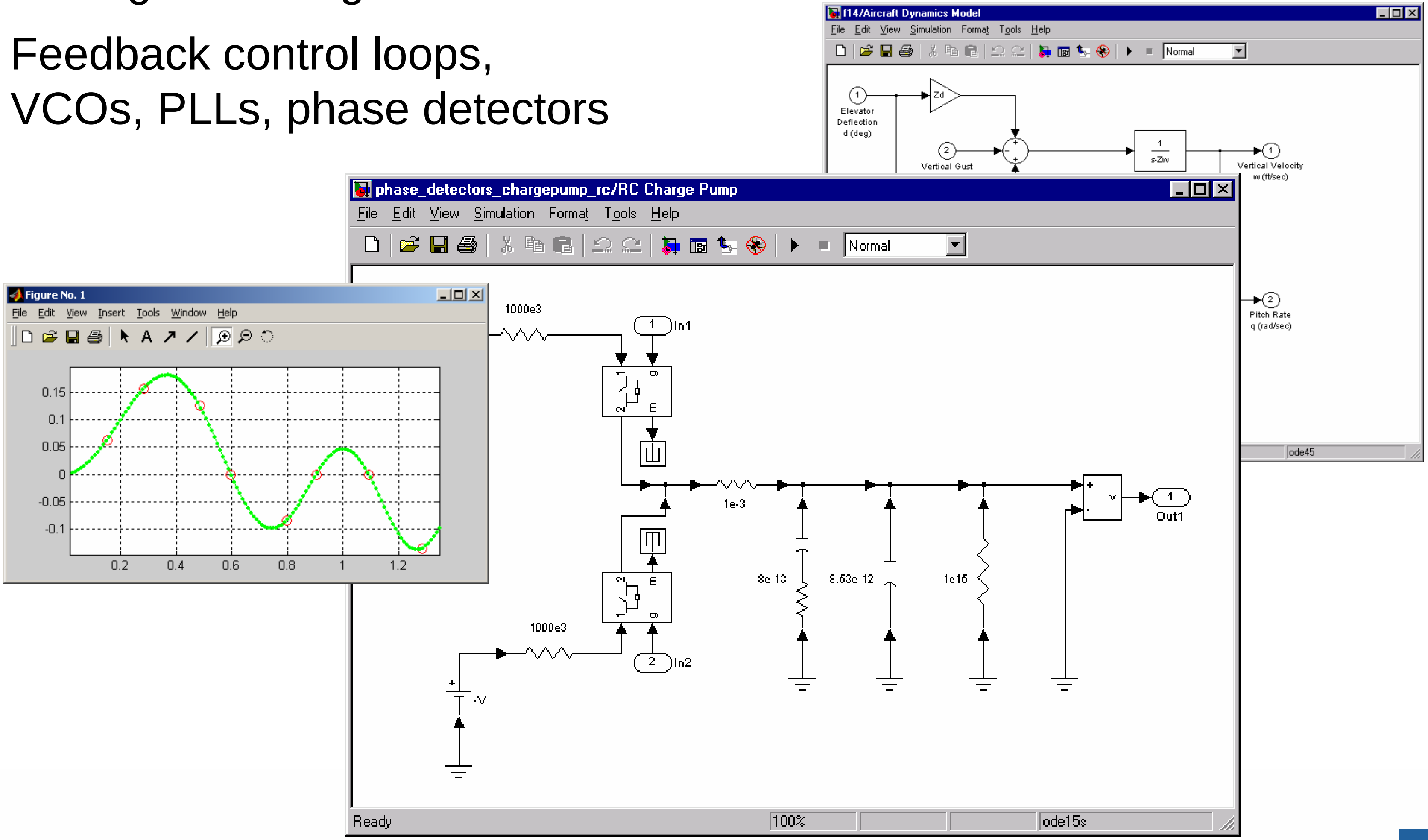
Finite State Machine that controls the adaptive modulation scheme.

The power of the Rx signal is compared against an array of power thresholds in order to determine the corresponding modulation as defined by the IEEE 802.11a specifications

Ready (ICED)

Circuit diagram

- Analog/Mixed Signal
- Feedback control loops, VCOs, PLLs, phase detectors


f14/Aircraft Dynamics Model
File Edit View Simulation Format Tools Help
Normal
1 Elevator Deflection d (deg)
Zd
2 Vertical Gust
1/s-Zw
1 Vertical Velocity w (ft/sec)
2 Pitch Rate q (rad/sec)
ode45
phase_detectors_chargepump_rc/RC Charge Pump
File Edit View Simulation Format Tools Help
Normal
1000e3
1 In1
1e-3
8e-13
8.53e-12
1e15
v
1 Out1
1000e3
2 In2
-V
Ready
100%
ode15s
Figure No. 1
File Edit View Insert Tools Window Help
0.15
0.1
0.05
0
-0.05
-0.1
0.2
0.4
0.6
0.8
1
1.2

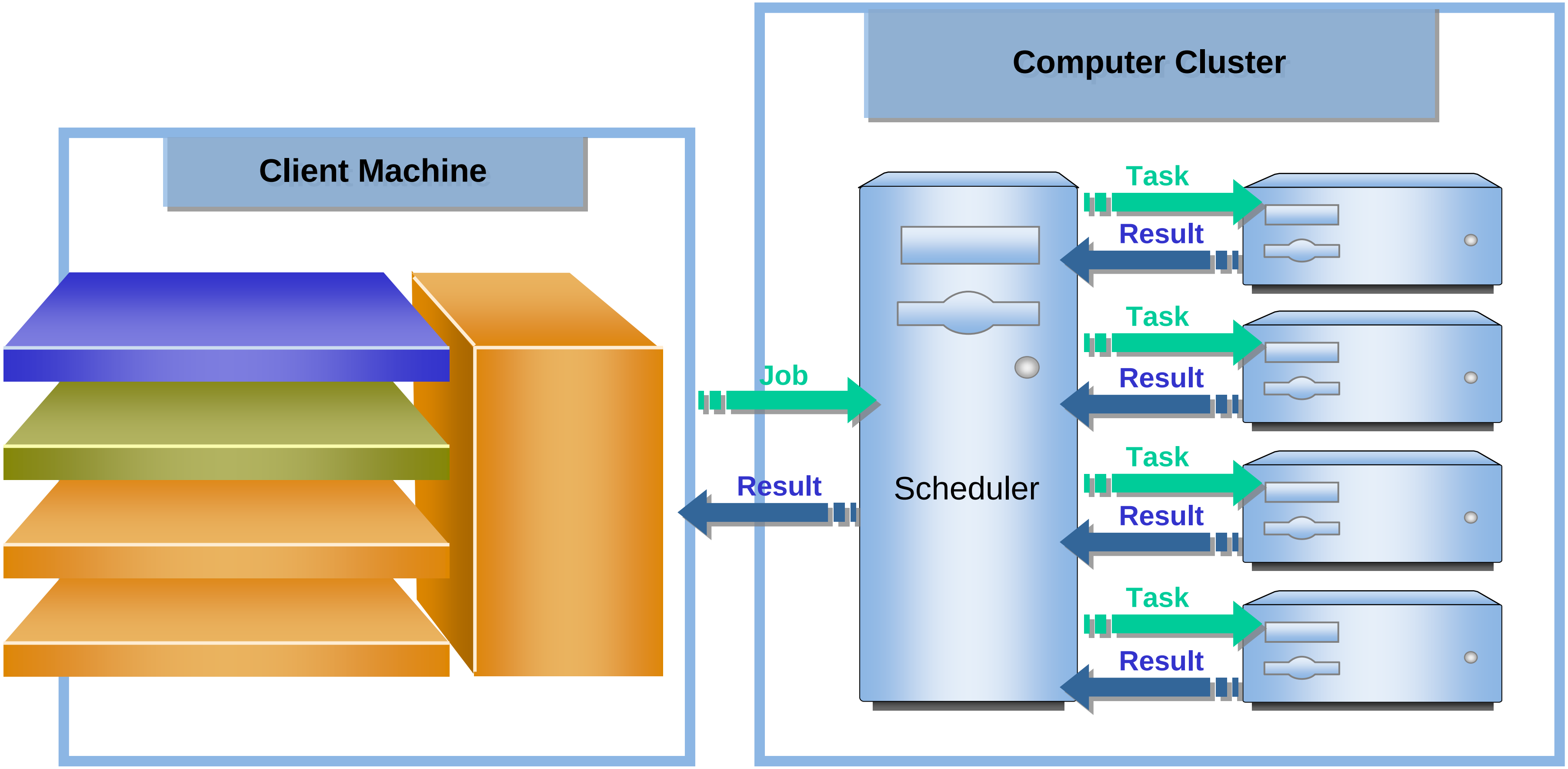
Thesis

- The competence people have developed in "programming"/coding make them sceptical reagarding alternative entries

- Software development tools will further reduce the need to transform system descriptions from the "human style" to the "computer style"
- This is only possible if we leave behind the ideal of a single language for all

- Single source in various languages

Distributed and parallel computing



Writing a parallel application

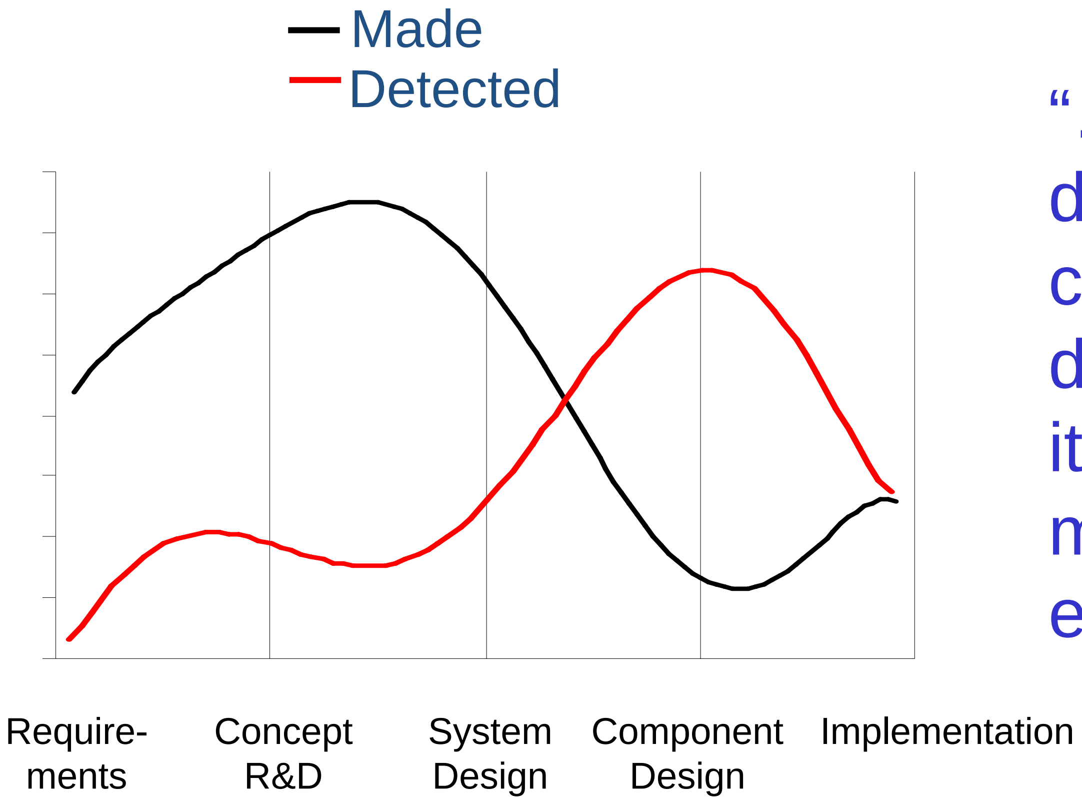
- Parfor
- parallel for loop to run in MATLAB or a matlabpool

```
>> matlabpool
   clear A
   parfor (i = 1:8)
      A(i) = i;
   end
   A
   matlabpool close
```

Theses

1. Increasing the level of abstraction increases the productivity
   - Do not reinvent the wheel
   - Challenge: trade-off development effort against optimal performance
2. Start to adapt the “computer” to the application
   - Single source cannot mean single language
   - There is not a single best “language” for everything
3. Iterative development
   - Early verification
   - Flexible partitioning
   - Challenge: Moving between abstraction levels

How to catch errors early?



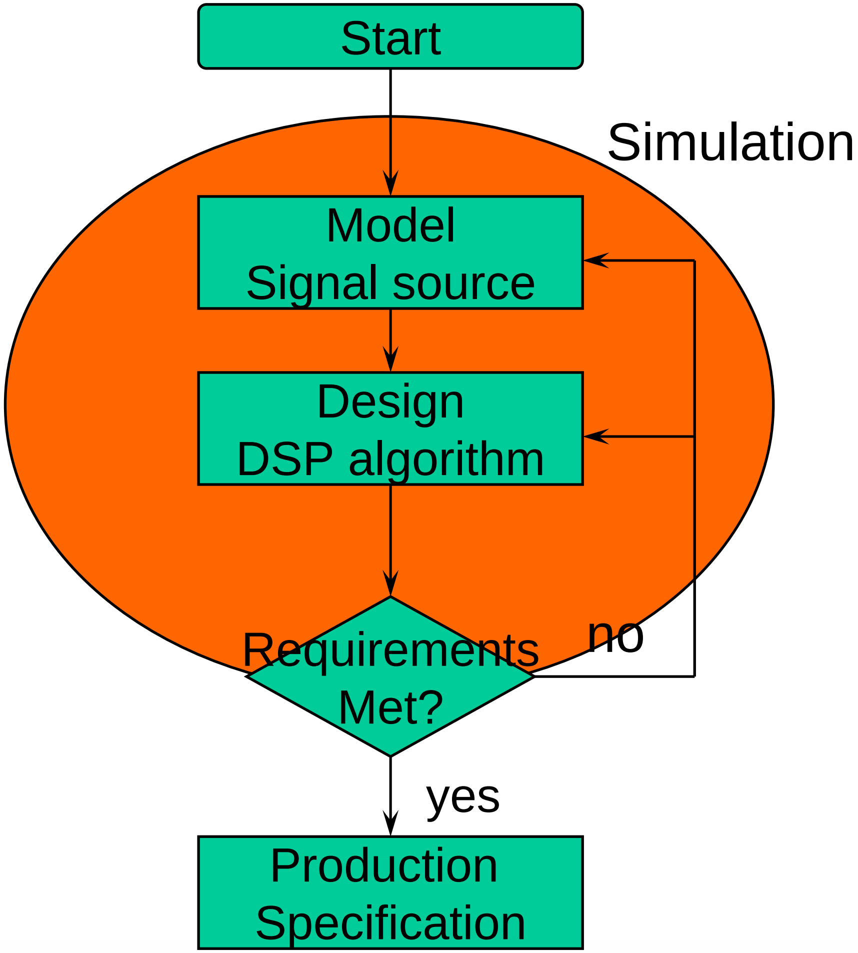
"…each delay in the detection and correction of a design error makes it an order of magnitude more expensive to fix…"

Clive Maxfield and Kuhoo Goyal
"EDA: Where Electronics Begins"
TechBites Interactive, October 1, 2001
ISBN: 0971406308

Source: "Migration from Simulation to Verification with ModelSim®" by Paul Yanik. *EDA Tech Forum*, 2004 Mar 11, Newton MA

Early verification

- Consider algorithm in its implementation environment
- Use tools to optimise algorithm conversion
- Speeds up the design cycle
- Easy to switch back and forth
- Separate the algorithm from the implementation details

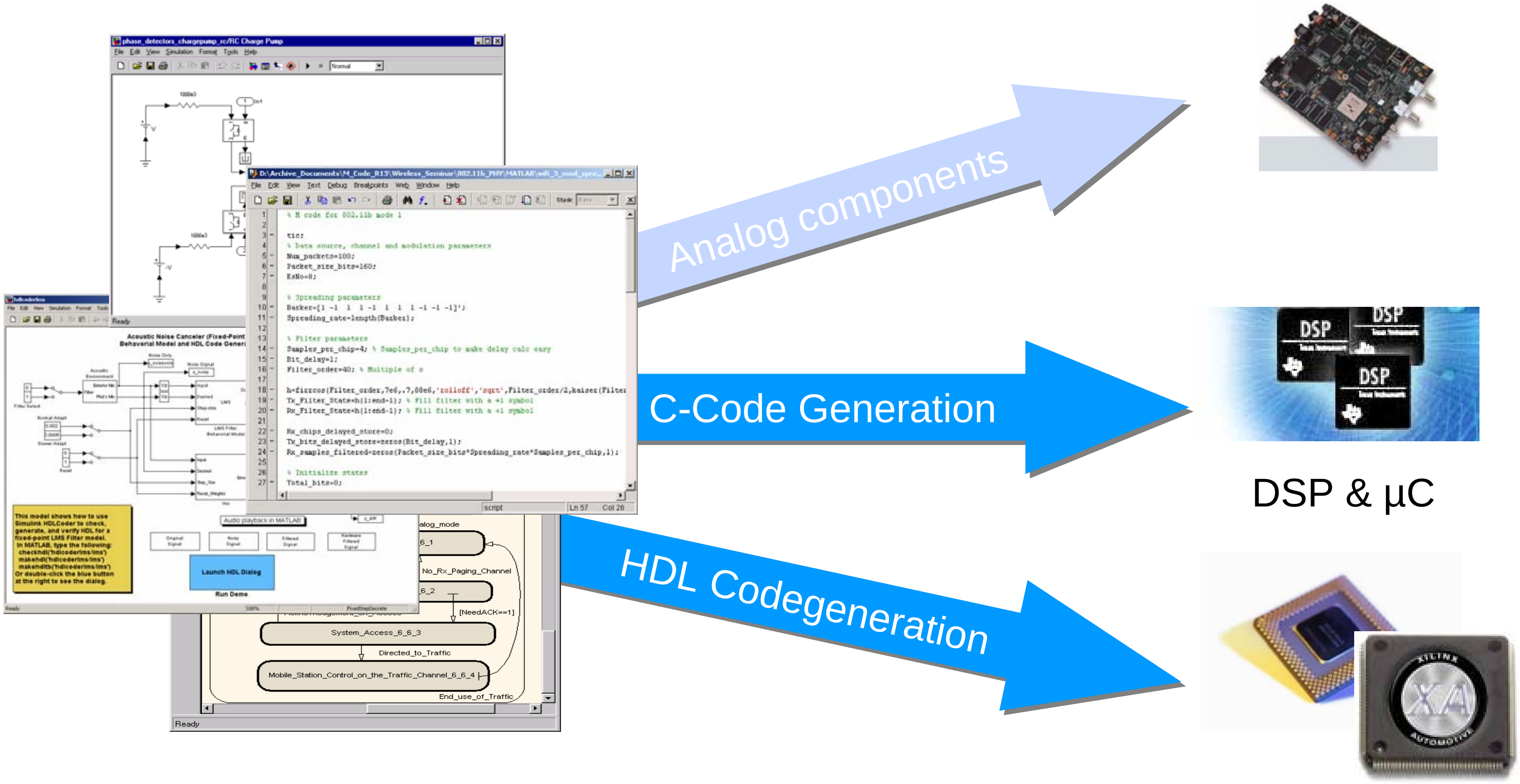


Iterative Design flow

- Top-Down
  - Rapid prototyping
  - Optimize by parameterizing the code generation
- Bottom-Up
  - Reuse optimized IP

- Design flow:
  - Create a system model
  - Generate for a module
    - C Code - Analyze performance
    - HDL Code - Analyze performance
  - Decide on the implementation method per module
  - Optimize the performance per module
    - Adapt code generation
    - Manually optimize and reintegrate the code – Bottom-up

Flexible partitioning



```
% M code for 802.11b mode 1

tic;
% Data source, channel and modulation parameters
Num_packets=100;
Packet_size_bits=160;
EsNo=8;

% Spreading parameters
Barker=[1 -1  1  1 -1  1  1  1 -1 -1 -1]';
Spreading_rate=length(Barker);

% Filter parameters
Samples_per_chip=4; % Samples_per_chip to make delay calc easy
Bit_delay=1;
Filter_order=40; % Multiple of s

h=firrcos(Filter_order,7e6,.7,88e6,'rolloff','sqrt',Filter_order/2,kaiser(Filter
Tx_Filter_State=h(1:end-1); % Fill filter with a +1 symbol
Rx_Filter_State=h(1:end-1); % Fill filter with a +1 symbol

Rx_chips_delayed_store=0;
Tx_bits_delayed_store=zeros(Bit_delay,1);
Rx_samples_filtered=zeros(Packet_size_bits*Spreading_rate*Samples_per_chip,1);

% Initialize states
Total_bits=0;
```

This model shows how to use Simulink HDLCoder to check, generate, and verify HDL for a fixed-point LMS Filter model. In MATLAB, type the following:

```
checkhdl('hdlcoderlms/lms')
makehdl('hdlcoderlms/lms')
makehdltb('hdlcoderlms/lms')
```

Or double-click the blue button at the right to see the dialog.

DSP & µC

FPGA & ASIC

Single source in various “languages”

Implementation trade-off

Blocks can have more than one implementation

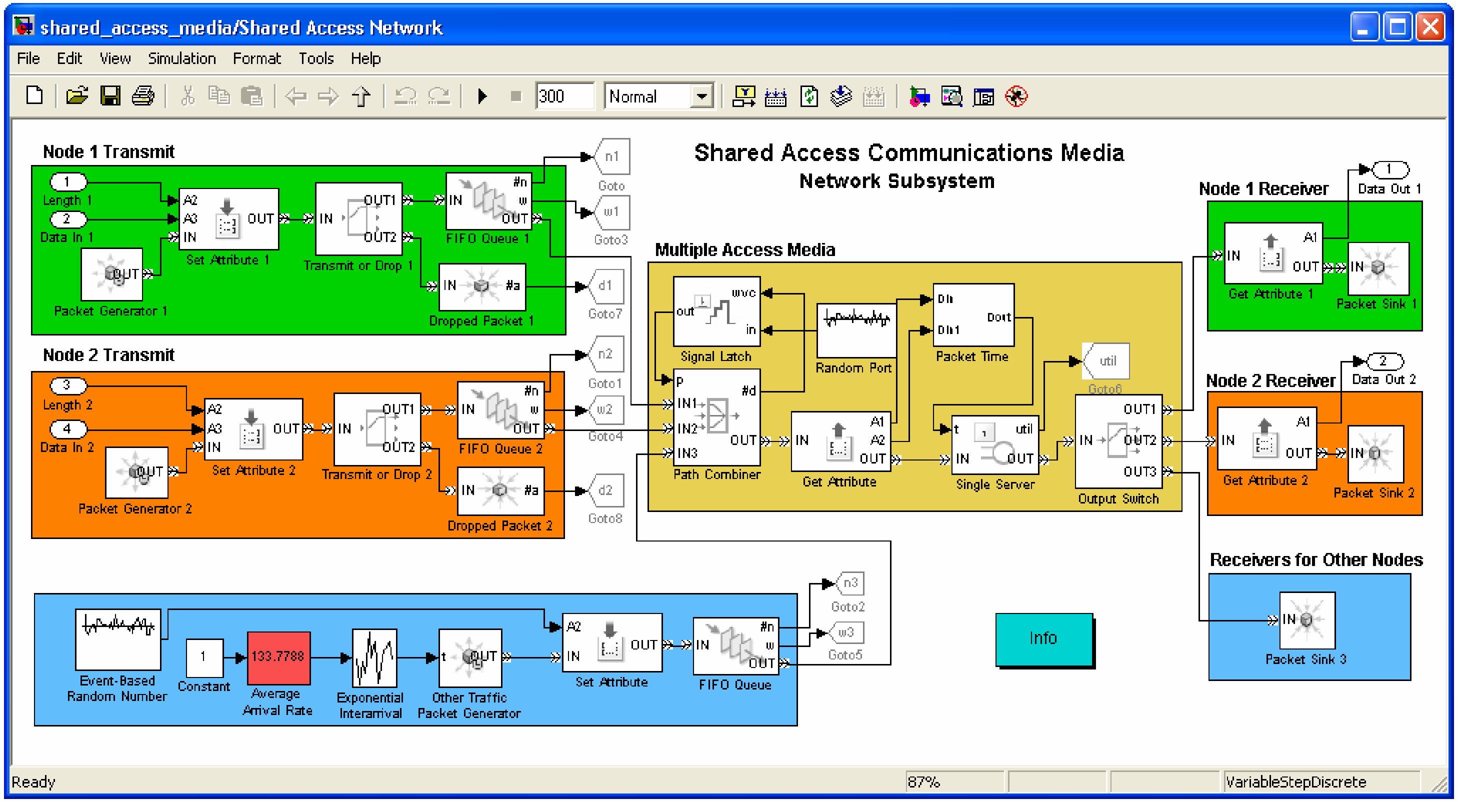
- Gain
  - hdldefaults.GainMultHDLEmission
  - hdldefaults.GainFCSDHDLEmission
  - hdldefaults.GainCSDHDLEmission

- Lookup Table
  - hdldefaults.LookupHDLInstantiation
  - hdldefaults.LookupHDLEmission

- …

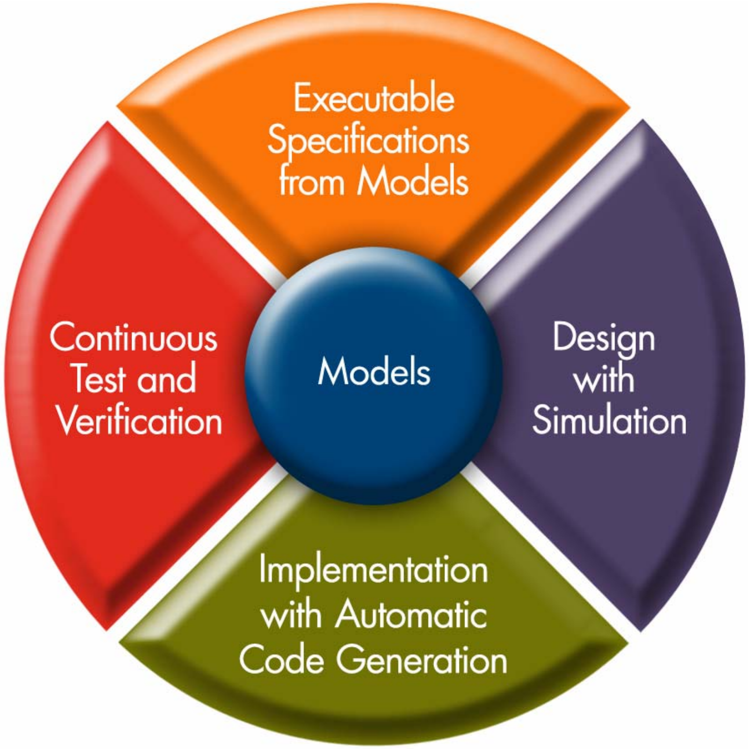
Limitations: Abstract Modeling

- Efficient but abstract
- Challenge: Simulation: There is no way to easily switch between detailed and abstract model



Conclusion

- Model-Based Design puts modeling and simulation at the center of system design
  - Increased abstraction = Increased productivity
  - Make the computer understand the "human" input
  - Iterative design for optimized system+process performance


Executable Specifications from Models
Continuous Test and Verification
Models
Design with Simulation
Implementation with Automatic Code Generation

Thank you for your attention

Questions?

See us at our booth