20 YEARS OF EMBEDDING MATHWORKS WORKFLOWS
HOW AND WHY WE USE THEM – AND WHY NOT!

Stäfa, 9. Juni 2015, H. Roeck, Director Key Technologies
HEARING LOSS

IT’S ALL ABOUT UNDERSTANDING, NOT HEARING!
KEY TAKE AWAY

Know / learn
– the properties of the tools and flows you use
– the different demands of the users of these tools and flows
– how that fits together with your modeling and development flow

Adapt
– accordingly to achieve appropriate efficiency gains
BROADEST OFFERING: HEARING INSTRUMENTS, COCHLEAR IMPLANTS …

… AND PROFESSIONAL AUDIOLOGICAL SERVICES
GLOBAL PRESENCE

Well established global infrastructure and network

Strong local presence – Supporting customers worldwide
– For 10 years now, we generate ca. 2/3 of our turnover with products which are less than 2 years in the market
– At least 100 individual new products per year (incl. all price points, form factors, private labels etc.)
– Only possible through a platform based technology development with re-usage of key components in many products
– DSP platform engaging *model based design*
GLOBAL R&D TEAM

DIFFERENT SITES
– Stäfa: ca. 50 Matlab licenses + Simulink, fixed-point, statistics, comm, neural network, …
– Murten
– Kitchener (Canada)
– Valencia (California)
– Hannover

DIFFERENT USER GROUPS
– DSP Researchers => “function developers” => Physicists, Audiologists, EE
– SW Implementers => “platform developers” => SW engineers
– Other R&D => information / understanding

… all access the same common source repository
DIFFERENT INTERESTS

RESEARCHERS
– Good libraries (DSP toolboxes), quick prototyping capabilities (=> RTW / Simulink coder), functional overview, no hassle => scripts, no SW (e.g. ‘OO’) complexity
– Function development
– Could work with Matlab/Simulink e.g. Release 14 Service Pack 3

IMPLEMENTERS
– Structure, maintainability, non-functional performance, compatibility, testability, quick automated test & release mechanism, automated configurability on model and parametrization level, …

OTHER R&D
– Simple access, quick functional overview, readability without being a DSP / Simulink specialist, usage of special toolboxes
What is a ‘model’?

– *Abstraction* of ‘the real thing’, keeping specific properties thereof.

– *Simplified* description, especially a *mathematical* one, of a *system* or *process*, to assist *calculations* and *predictions*.

– Contains numerical, physical and other limitations, disregards properties of the real thing intentionally or not so intentionally

– Models are faster developed then the real thing

– Models might be used to draw conclusions within their validity range
Essentially, all models are wrong, but some are useful.

George E. P. Box (1987)
% modification 24.6.97, WK:  
% removed error for sign calculation of e_bf

%% compute beamformer output %
if (BF_ON == 1)  % beamformer mode
    e_bf = cf_bf - beta_bf * cb_bf - y_hat_fc;  

%% LMS update only when beamformer is active  
if (BF_ADAPTIVE == 1)
    Ecb_bf = sum(abs(oh_bf)) + Pa_lms_bf;  % frame output power of back cardioid + const
    Fcb_bf = lambda_lms_bf * (Fcb_bf - Ecb_bf) + Ecb_bf;
    signe_bf = sign(e_bf) + (sign(e_bf) == 0);  % trick out Matlab function sign to set sign(0) = 1
    E_y_ob_bf = sum(signe_bf .* cb_bf);  % frame cross correlation estimation
    beta_bf = beta_bf + mu_lms_bf * E_y_ob_bf / Fcb_bf;  % LMS update of beta

%% limit beta
    beta_bf = min(beta_bf, MAX_BETA_BF);  % upper bound of beta_bf
    beta_bf = max(beta_bf, MIN_BETA_BF);  % lower bound of beta_bf
end;  % omni mode
else
    e_bf = cf_bf - y_hat_fc;
end;
... OVER TIME ...

2001

Control signals:
Observables & Block outputs
Note: the implementation uses the AltGain BEFORE the repeat blocks in Gain Interpolator.

5

BOut FP

4

BOut CL

3

TBus MiM

2

TBus CL

1

Out FP

CL

lin2dB

10^(u/20)

dB2lin

TBus CL

TBus MiM

Block out CL

Block out FP

TBus & BOut

Synthesis

EnableLP

EnableHP

Out

BOut

Sound Generator

In

Threshold

Out

Peak Clipper

NS

Noise Shaper

cb
cf

cb avg
cf avg

DSP Int

TBus

Mic Matching

TBus_FCO

BOut_DCR0

TBus_DCR1

BOut_DCR0

BOut_DCR1

BOut_Mult

BOut_OLIM_CL

BOut_CF

BOut_CB

BOut_Synthesis

TBus_MIC

BOut_SOGE

BOut_OLIM_FP

BOut_FNF

BOut_ANF

TBus_MicMatchDSPInt

TBus_GINT

BOut_GainLin

BOut_dB2lin

TBus_GainLog

BOut_NC

BOut_GLIM

BOut_MBC

TBus_NC

TBus_GLIM

TBus_BF

BOut_FCO

Gain Mult
dB in

Gain

TBus

Gain Model

Gain_dB

Spectrum_dB

Out dB

TBus

BOut

Gain Limiter & MPO Control

GInt

Out

TBus

Gain

Interpolator

[Enable_NegateOutput]

[Enable_NS]

[Enable_FCO]

[Enable_Gain]

[Enable_OLIM]

[Enable_SOGE_HP]

[Enable_SOGE_LP]

[Enable_SOGE]

[Enable_NC]

[Enable_GLIM]

[Enable_DCR]

[Enable_DCR]

[Enable_FixedNotch]

[Enable_AdaptNotch]

In

Out

BOut

Fixed Notch

PAR.MBC.FixGain

Fix Gain

BP Sig

FreqCoef

RadZeroSqProb

TBus

BOut

Feedback Detection

Enable Flags

Enable

Out

TBus

DCR 2

DC-Remover1

In

Enable

Out

TBus

DCR 1

DC-Remover0

In

Enable

Out

TBus

OLIM

PAR.GINT.AltGain

AltGain

In

Threshold

TBus

BOut_CL

Adaptive Threshold

In

Frequency

Probability

Out

BOut

Adaptive Notch Filter

In

Channels
f1k6..10k

BOut

5 Channel WDF Filterbank
... BECAME MORE COMPLEX ...
CHALLENGE: FIXED-POINT PROPERTIES

FIXED-POINT TOOLBOX (WITH ADVISER / OPTIMIZER):

- Depends on simulation results (max/min values for all signal nodes, precision)
  (Attention: => dynamic range/sign of stimuli, parameters, dead code)
- Does not know about relevant details
  - Input to log10 negative ?
  - What if simin is never negative in simulation ?
  - What about the dynamic range of the parameter ‘gain’ ?
- All discontinuous operations are critical:
  - Max, min, if..else, switch/case, …

An algorithm designer must know about the relevant properties of ALL parameters & input signals!

We use our fixed-point license e.g. for our sigma-delta DAC!

MATLAB:
If simin > 0
   Result = log10(Simin);
else
   Result = -gain * sqrt(abs(simin));
end
MULTI CORE DSP IN A HEARING AID

ARCHITECTURE

– Multiple cores (like in every Smartphone)
– Multiple instructions per clock cycle
– Co-processors for specialized tasks
– Native support in instruction set for specific data types
– …

POWER CONSUMPTION

– Two cores at half the clock frequency are more power efficient than one core at the full clock frequency IFF
– the overhead to synchronize and schedule is sufficiently low

Challenge: => How to achieve efficient parallelism?
CHALLENGE: PARALLEL COMPUTING

TYPES OF PARALLELISM

- **Instruction** level parallelism
  - Decent way of writing code => Target compiler
- **Data** level parallelism
  - SIMD data types => Target compiler
- **Task** level parallelism
  - Multi core scheduler => OS, dynamic, static

PARALLEL COMPUTING TOOLBOX

- Ideal and tailored for *huge* processing needs, on host computer within MATLAB environment
- not for distributing a few hundred cycles here and there to a multi-core embedded target (as we have to do)
CHALLENGE: PRODUCT QUALITY TARGET CODE

SIMULINK CODER
– Requires target data types => might generate less readable model
– Our researchers do not want to care too much about rounding and saturation behavior, bit widths
– Far too slow simulation speeds (for V&V as well as real-time simulation)
=> We use it not for target code generation, but for RT simulation on a PC

ALTERNATIVE:
– Insert saturation and quantization blocks only at few critical signal nodes
– Bit exact models only of nonlinear function approximations (e.g. log, exp, 1/x)
– Difference to target HW simulation results: 0 – 2 LSB
– Write C code manually with full knowledge of underlying ISA and compiler properties
– Schedule our code snippets (‘grains’) semi-automatically!

19 June 2015
DEVELOPMENT LANDSCAPE

Matlab
Simulink
Parameter Tool
'trafo'
Simulink Parameter set
Toolboxes
Brand & Product line specific Applications
Grains
Parameter Tool 'trafo'
Fitting SW platform (C#/.net)
Hearing aid
Fitting SW
Target Parameter set
Data base
Production Support Tool
Brand A
Fitting SW
Brand B
Fitting SW
Fitting Parameter set
Simulink Parameter set
Scheduler
Profile data
Profile
Single core Simulink, Profiler
Block Dependencies
Speedgoat platform
Viewer
Fast prototype
RTW / Simulink coder
Matlab
Toolboxes
Scripts
Test cases
Verification suite (==?)
Test results
Multi core Simulator, Profiler
Target Parameter set
Fast prototype
Target C code
Compiler
Profile
Profile data
Scheduler
Asm/ linker
Block Dependencies
Dependency extractor
Simulink
Target Parameter set
Brand specific Components
Brand & Product line specific Applications
Test results
Test cases
19 June 2015
MULTI CORE SCHEDULE

– Optimal distribution of even small grains to different cores without extensive task setup
– Scheduling according to processing rate (=> thread) and dependency to other grains
– Using profiler generated execution time data per grain => write code to have signal independent execution time !
– Goal: Balance load on all cores as evenly as possible with as few ‘empty’ cycles in between as possible.

19 June 2015
CONCLUSION

– Over the last 20 years, we use Mathworks based solutions quite successfully

– A fool with a tool is still a fool though

– By learning
  – where and to which extent it makes sense to use which tools and automate flows,
  – which properties to model
  – (and automating a few things which have nothing to do with Mathworks solutions 😊)

– we increased even in Stäfa only our “product quality DSP algorithm output” from ca. 1/year in ~2000 to 10/year in 2014

– while manyfolding the complexity of the models and
– with only doubling the # of developers
Q&A