Example of prototyping algorithm using MATLAB/SCILAB and ECLIPSE environment

Frédéric Bard

Eclipse IoT Days - 2017
Introducing myself

WITH VEYADO COMPANY: RESEARCH ENGINEERING AND CONSULTING FOR FUNCTIONAL SAFETY, QA, CWE, DEVELOPMENT OF PROFESSIONAL TERMINAL OR IOT DEVICE, HOMOLOGATION, PRODUCTION AND MASS DEPLOYMENT TO SUPPORT VARIOUS SERVICES.

CURRENT WORK IN THE FIELDS OF ENERGY MANAGEMENT, COMMUNICATION FOR ELECTRIC DEVICES, TRANSPORTATION AN MEDICAL SERVICES.

BEFORE, DEPLOYMENT OF SERVICES OVER IP/MPLS (QUADRUPLES PLAY, GEOLOCALIZATION, M2M, UC, VOIP, VOD, IPTV, TÉLÉPRÉSENCE, VISIOCONFÉRENCING) WITH MAJOR TELECOM PROVIDERS AND ANALYTICS OF UNIFIED COMMUNICATIONS: STATISTICS, ROI CALCULATIONS, TRANSPORTATION AND TRAVEL COSTS, GREEN HOUSE GAS (GHG) EMISSION EVALUATION AND ECO2 REDUCTIONS CALCULATION

Frédéric Bard
VEYADO SAS

Email: bardf@veyado.fr
Cél: 0689821845

15 years in DSP and electronics
15 years in services over IP and network
PARISTECH graduated engineer
Mastère ParisX – IFG ICG – Management
CEE/IEEE auditor
ISEP lecturer

Eclipse IoT Days - 2017
Developing an IoT system is so simple: on your kitchen table.
Eclipse allows to prototype in MATLAB/SCILAB, to generate and validate C/C++ code and debug on embedded platform.
IoT devices development

Functional Tests
- Uses Cases
  - Proof concept, breadboard
  - Research Equipment
  - Tools: Labview

Laboratory

Product Design

Unitary, Integration, Loading Tests
- System model and manual testing, first non regression tests
- Development processes, patent
  - Tools: Matlab

Choice of an embedded platform, SW partitioning 1st choice
- HW: PCB routing, components choice
- SW: drivers, application code
- Tests: unitary, functional, integration
- QA referential and tools
  - Tools: SDK, C language

Proto 1

Bluetooth, ISO13485 Certification, Field Usage

Proto 3
- HW rework, SW partitioning 3rd choice
- HW: PCB routing, components choice
- SW: drivers, application code
- Tests: unitary, functional, integration, network and server testing
- QA referential and tools
  - Tools: SDK, C language, JAVA, C#

Proto 2
- Add of communication chip, SW partitioning 2nd choice
- HW: PCB routing, components choice
- SW: drivers, application code
- Tests: unitary, functional, integration, network and server testing
- QA referential and tools
  - Tools: SDK, C language, JAVA, C#

Product development: a stairway with many challenges

Eclipse IoT Days - 2017
Importance of Quality Control, testability during the Life Cycle: from Use cases to Product assessment

Requirements: Needs, New projects

- Needs and Use Cases
- Design inputs and Requirements identification

Project setup, functional design specification

1st Design Review

Validation

Implementation and Design specification / Product Backlog

Unitary Test, HW drivers and integration Test

Coding

Dossier technique

1st Design Review

Functional and Integration tests, Field Usages

Customer review, Tests, Design Changes, Coding review

Specification Qualification and QA requirements fulfillment

3rd Design review

Customer Needs Satisfaction

Release & homologation

α, β and commercial products delivery

Importance of Quality Control, testability during the Life Cycle: from Use cases to Product assessment
Passing from the Lab demo to an embedded POC, translating Signal Processing Model to C code
Basics for simplicity:

Have an simple HIM to develop and signal processing validation (Scilab, Labview, MatLab, Octave)

Distribute it to all users (cheap license)

Use open standard for code generation

Run laboratory and process (ex medical, biological, optics, …), tune lab device and functional tests that will serve to tune end products

If needed, compliance to standard EN62304 (medical), ISO26262 (vehicle)
Simulating on a Model (SCILAB) on core i7 PC with Windows 10, improving performances can have strange outcome
SCILAB parallel_run : parallel calls to a SCILAB function

```
timer();
for i = 1:nCourb
  // waitbar(i/nCourb,winH);
  myfile_bin = MyfilesDIR(i,1)+'\correl_data.bin';
  courbes_dispos = [courbes_dispos string(i)];
  courbe_desc = 'courbu('+string(i)+');';
  courbes_o = [courbes_o courbe_desc];
  corr_plot(i,myfile_bin,myfile_txt);
end
Temps_calcul_boucle = timer();

------ critical loop (in sec.) : 
2402.921875
```

```
function corr_PAR_plot(i)
  // waitbar(i/nCourb,winH);
  myfile_bin = MyfilesDIR(i,1)+'\correl_data.bin';
  courbes_dispos = [courbes_dispos string(i)];
  courbe_desc = 'courbu('+string(i)+');';
  courbes_o = [courbes_o courbe_desc];
  corr_plot(i,myfile_bin,myfile_txt);
Endfunction

timer();
//for i = 1:nCourb
  PAR_test = parallel_run(1:nCourb, corr_PAR_plot);
Temps_calcul_boucle = timer();

------ critical loop (in sec.) :
2481.625000
```

Eclipse IoT Days - 2017

Regular to parallel loop SCI

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>97.4%</td>
</tr>
</tbody>
</table>
SCILAB parallel_run : calls to a SCILAB or C compiled function

```scilab
link_name = "CalCorr_FindCoagTime"; // to the call table
exec loader.sce;
...
[Test_retour] = call("CalCorr_FindCoagTime",mydata,1,"d",lengthC,2,"i",myTcoagSCI,3,"d","out",[1,3],4,"d");

-------------- critical loop (in sec.): 82.140625

function [Tcoag,Error] = Coag_FindCoagTime_EOLANE(Corr,T)
...
Endfunction

[myTcoagSCI,myErrorSCI] = Coag_FindCoagTime_EOLANE(mydata,mytime);

-------------- critical loop (in sec.): 2402.921875

C to SCI ratio
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3,65 %</td>
</tr>
</tbody>
</table>

Eclipse IoT Days - 2017
SCILAB parallel_run: parallel calls to a function + compiled C code call

```scilab
function corr_PAR_plot(i)
    // waitbar(i/nCourb,winH);
    myfile_bin = MyfilesDIR(i,1)+'correl_data.bin';
    courbes_dispos = [courbes_dispos string(i)];
    courbe_desc = 'courbu('+string(i)+');';
    courbes_o = [courbes_o courbe_desc];
    corr_plot(i,myfile_bin,myfile_txt);
end
```

```scilab
PAR_test = parallel_run(1:nCourb, corr_PAR_plot);
Temps_calcul_boucle = timer();
```

----- critical loop duration:
82.140625

Regular to parallel loop C

<table>
<thead>
<tr>
<th>Function</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td></td>
</tr>
<tr>
<td>Parallel</td>
<td>98.8%</td>
</tr>
</tbody>
</table>

----- critical loop duration:
87.890625

Eclipse IoT Days - 2017
Embedded computing bring other constraints, an example with of STM32 (Cortex Mx core):

- floating point and fixed point calculation optimization (FPU and CPU)
- No MMU but faster memory access through data cache or local memory (compare to flash RAM)
Smart use of a « simple » architecture
Correlation critical loop

Calculs en float :
/// long int SumAA; /* Sum des A^2 */
/// SumAA = SumAA + ( ((float)(pI1[(iIn*m) + iIm])) - val_moy_1 ) * 
/// ( ((float)(pI1[(iIn*m) + iIm])) - val_moy_1 );

LDRB    R0,[R0, R11]
VMOV    S1,R0
VCVT.F32.U32 S1,S1
VSUB.F32 S1,S1,S17
VMLA.F32 S19,S0,S1

Calculs en int :
/// long int SumAA;
/// SumAA = SumAA + (((pI1[(iIn*m) + iIm]) - val_moy_1 )^2); //KK2 = 11^2
= 121 donc 8b de dynamique perdue, mais (28b)^2 on sature la dynamique 32b
LDR    Ro,[SP, #+16]
LDR    R1,[SP, #+844]
LDR    R2,[SP, #+852]
MLA    R2,R2,R4,R10
LDRB   R1,[R2, R1]
LDR    R2,[SP, #+8]
MULS   R1,R2,R1
LDR    R2,[SP, #+20]
SUBS   R1,R1,R2
EORS   R1,R1,#0x2
MULS   R1,R2,R1
ADDS   R0,R1,R0
STR    R0,[SP, #+16]

Eclipse IoT Days - 2017
Ideally, direct generation of C/C++/C# from the Model Language AND optimization for the embedded/multicore platform.