Executable models for Embedded Controllers Development — A Cloud Based Development Framework

Fernando Pereira*†‡, Filipe Moutinho*, João Paulo Barros*§, Aniko Costa*‡, and Luís Gomes*‡
* UNINOVA - CTS, Portugal
† ISEL, Instituto Superior de Engenharia de Lisboa, Portugal
‡ Universidade Nova de Lisboa - Faculdade de Ciências e Tecnologia, Portugal
§ Instituto Politécnico de Beja, ESTIG, Portugal
Email: fjp@deea.isel.ipl.pt, {fcm, jpb, akc, lugo}@uninova.pt

***Abstract*—We present IOPT-Tools, a tool framework for the development of digital controllers based on graphical executable models. The framework supports edition, simulation, verification through state-space querying, and code generation for several hardware platforms, most notably microcontrollers (e.g. Arduino, PIC, and Raspberry Pi) and FPGAs. The tool framework uses a class of Petri nets and is cloud based: the development process is performed using a browser.**

I. Motivation and Goals

Digital controllers are often developed and programmed using textual languages. Due to the well-known suitability of Petri nets to graphically specify sequence, concurrency, and synchronizations, specific classes of Petri nets have been designed to support the specification of this type of systems (e.g., [1]–[4]). Yet, somehow surprisingly, the "Petri Nets Tool Database" [5] shows a notable scarcity of tools able to support the design and, especially, the generation of code for digital controllers.

This work presents a web tool framework that provides a higher level language for the development of digital controllers. It is based on graphical executable Petri net models that use a non-autonomous class of Petri net. Besides supporting the modeling, simulation, state space exploration, and code generation for digital controllers, the framework is totally web based, thus providing multi-platform support and avoiding the need to install software.

II. Non-autonomous Petri nets

Petri nets have their origin in the PhD. Thesis of Carl Adam Petri [6]. Yet, numerous Petri net classes exist, some of them with associated tools [5]. These classes of nets usually share a small set of base characteristics that we summarize next:

- Petri nets are directed graphs with two types of nodes: places and transitions;
- Places are the "passive" part, are represented by circles or ellipses, and are typically associated to states, resources, or conditions;
- Transitions are the "active" part, are represented by bars or rectangles, and are typically associated to changes of state, actions, message passing, or consumption of resources;
- Places can only be connected to transitions, and vice-versa;
- Places and transitions can have any quantity of input and/or output arcs.

In its most well-known form (Place/Transition nets or P/T nets [7]) Petri nets provide support for very abstract models, which are autonomous in the sense that they do not depend on external entities. Yet, it is a well-established fact that an effective modeling of digital controllers requires or benefits very significantly from the addition of non-autonomous concepts, namely the explicit modeling of external input and output signals and events. Additionally, as P/T nets are non-deterministic, the designer must have a way to remove non-determinism from the models. To that end, as presented next, the IOPT nets class adds priorities to transitions, thus providing a way to remove effective conflicts.

IOPT-Tools [8] is based on a non-autonomous class of Petri nets, which adds several well-known and established concepts to P/T nets. The net class is named Input-Output Place-Transition Petri nets, a name that emphasizes the supported explicit specification of input and output signals and events in the net models. The precise syntax and semantics of this net class are presented elsewhere [9], [10]. Here, we briefly present the main additions to P/T nets. Compared to P/T nets, IOPT nets add the following constructs:

- Explicit modeling of external input and output signals and events;
- Guards (function of input signals) and priorities in transitions;
- Test arcs;
- Input and Output events in transitions;
- Output actions in places, as well as in transitions.

The metamodel for the Input-Output Place-Transition nets reuses many of the elements of the PNML metamodel for Place/Transition nets, such as places, transitions, and arcs, and extends it to include specific concepts. The IOPT metamodel is

described in RelaxNG, as well as in MOF and Ecore formats, putting IOPT in the context of MDA artifacts and allowing one to take benefit from the MDA infrastructure [10].

IOPT nets have a single server maximal step cycle accurate semantics, which together with priorities in transitions, allows the development of deterministic models. In this semantics, all transitions that can fire will fire. To that end each fireable transition has to be *enabled* (sufficient tokens in input places) and *ready* (external signals and events allow the transition to fire).

III. The IOPT-Tools Framework

IOPT-Tools is a framework that integrates a set of tools supporting the development of digital controllers. The framework is also being used for teaching [11], which serves as a validation of its usability and simplicity. Presently, it integrates the following tools in a web-based interface allowing cloud based storage of the Petri net models being created [12]:

- Edit Model — Enter in the IOPT Model Editor tool;
- Simulate the models — executing the ”token game” [13];
- Generate C Code — Invoke the automatic software C code generator [14] ;
- Synthesize VHDL — Invoke the automatic hardware VHDL code generator [15];
- Generate State Space — Execute the State-space generation tool;
- Query Editor — Open the Query-editor tool to specify state-space queries;
- Query Results — View query results after the end of state- space calculation;
- Download Model File — Download the current model file to the users PC;
- Export Snoopy C — Convert the current model to the Snoopy/IOPT editor file format, using C syntax for math expressions;
- Export Snoopy VHDL — Convert the current model to the Snoopy/IOPT editor file format, using VHDL syntax for math expressions;
- Decompose GALS — Decompose the selected model into several sub-models according to specified GALS time-domains [16], [17];
- Model List — Select other model from the list of models stored in the server’s user account.

Each model (component) can be seen as a sub-model to be mapped into an hardware or software platform assuming hardware-software co-design techniques and specific metrics, as power consumption, performance, among others. FPGAs have been used in conjunction with IOPT-Tools framework allowing exploring and exercising solutions in hardware-software solution space. This allows the development of globally asynchronous, locally synchronous (GALS) systems.

Figure 1 shows the login web page. After, Figure 2 shows an IOPT net model for a quad encoder model, ready to be selected among several available, and Figure 3 shows the screen for the simulator tool.

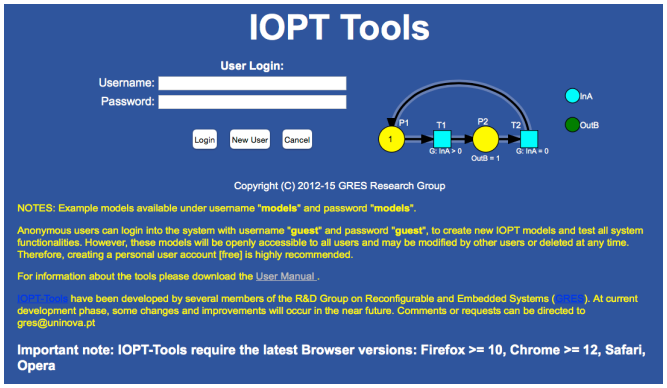

Fig. 1. Login window.

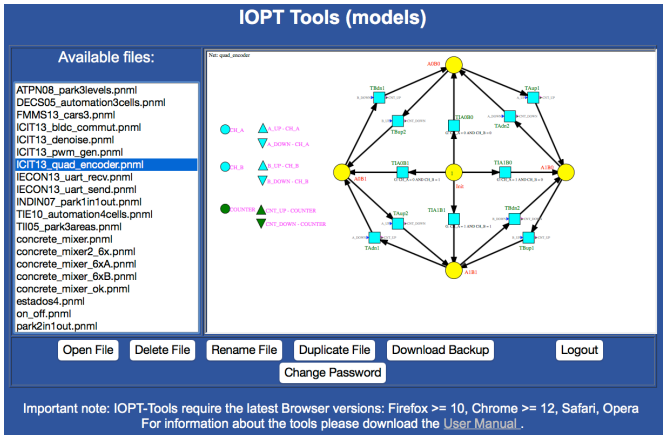

Fig. 2. Selecting one model to open, while presenting the one currently selected.

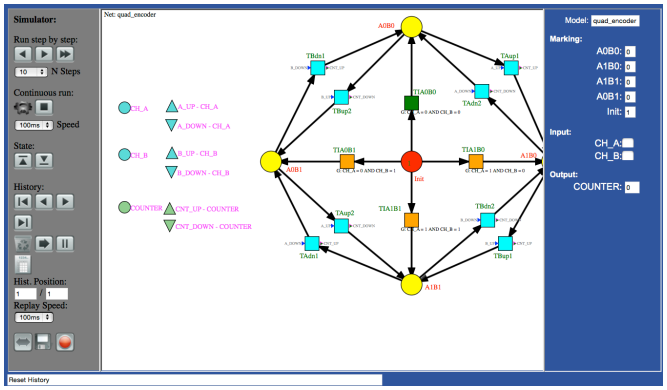

Fig. 3. Simulation tool window.

For the generation of the state-space, the user specifies the initial marking and the state-space is generated using the maximal step semantics of IOPT nets, taking in consideration the values of input signals and events. The state space can then be queried. Each query is a logical expression that can use place markings and event output signals associated to transitions. These queries allow the verification of some forms of reachability.

Listing 1 and Listing 2 show part of the generated C and VHDL code, respectively, which can be directly deployed into implementation platforms, such as FPGAs for VHDL descriptions, and current popular platforms (e.g., Arduino and Raspberry PI) for C based implementations.

Listing 1. Part of the generated C code

```
/* Net quad_encoder - IOPT */
/* Automatic code generated by IOPT2C XSLT transformation. */

#include <stdlib.h>
#include "net_types.h"

int trace_control = TRACE_CONT_RUN;

extern void webServer_init();
extern void webServer_getRequest();
extern void webServer_sendResponse();
extern void webServer_disconnectClient();
extern void webServer_checkBreakPoints();

static quad_encoder_NetMarking marking;
static quad_encoder_InputSignals inputs, prev_inputs;
static quad_encoder_PlaceOutputSignals place_out;
static quad_encoder_EventOutputSignals ev_out;

void setup()
{
    createInitial_quad_encoder_NetMarking( &marking );
    init_quad_encoder_OutputSignals( &place_out, &ev_out );
    quad_encoder_InitializeIO();
    quad_encoder_GetInputSignals( &prev_inputs, NULL );
#ifdef HTTP_SERVER
    webServer_init();
#endif
}

void loop()
{
#ifdef HTTP_SERVER
    webServer_getRequest();
#endif

    if( trace_control != TRACE_PAUSE )
        quad_encoder_ExecutionStep( &marking, &inputs, &prev_inputs, &place_out, &ev_out );
    else quad_encoder_GetInputSignals( &inputs, NULL );
    if( trace_control > TRACE_PAUSE ) --trace_control;

#ifdef HTTP_SERVER
    webServer_sendResponse();
#endif

    quad_encoder_LoopDelay();

#ifdef HTTP_SERVER
    webServer_disconnectClient();
    webServer_checkBreakPoints();
#endif
}

int main()
{
    setup();

    do loop();
    while( quad_encoder_FinishExecution( &marking ) == 0 );

    return 0;
}
```

Listing 2. Part of the generated VHDL code

```
-- Net quad_encoder - IOPT
-- Automatic code generated by IOPT2VHDL XSLT transformation.
-- by GRES Research Group - 2014


Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;


Entity quad_encoder IS
Port(
    Clk : IN STD_LOGIC;
    CH_A : IN STD_LOGIC;
    CH_B : IN STD_LOGIC;
    COUNTER : OUT INTEGER RANGE 0 TO 255;
    Enable : IN STD_LOGIC;
    Reset : IN STD_LOGIC
);
End quad_encoder;

Architecture Structural OF quad_encoder IS

    Signal p_2: INTEGER RANGE 0 TO 1 := 0;
    Signal p_3: INTEGER RANGE 0 TO 1 := 0;
    Signal p_4: INTEGER RANGE 0 TO 1 := 0;
    Signal p_5: INTEGER RANGE 0 TO 1 := 0;
    Signal p_30: INTEGER RANGE 0 TO 1 := 1;

    Signal prev_CH_A: STD_LOGIC := '0';
    Signal prev_CH_B: STD_LOGIC := '0';

    Signal event_A_UP: STD_LOGIC := '0';
    Signal event_A_DOWN: STD_LOGIC := '0';
    Signal event_B_UP: STD_LOGIC := '0';
    Signal event_B_DOWN: STD_LOGIC := '0';

    Signal s_COUNTER : INTEGER RANGE 0 TO 255 := 0;

    -- Array implementation:

Begin
    -- Selected array items:

    proc_in_events: PROCESS( Clk, Enable ) IS
    Begin
      If falling_edge(Clk) Then
(continues)
```

IV. Related Work

Petri nets have been used for a long time for the description of hardware systems and for the design of digital controllers (e.g. [18]). For example, a Petri net variant called STG (State-transition graphs) has been widely used in the field of logic systems design.

Another Petri Net inspired formalism is NCES/TNCES (Net Condition Event System/ Timed Net Condition Event System), which has been used for system modeling, mostly in the

automation area [19]. Due to its structure, NCES can be used to represent module interface abstraction and internal behavior, and can be easily mapped onto IEC61499 function blocks. The NCES editor ViEd has several associated tools as ViVe Visual Verifier that contain model builder, simulator, time diagram generator among others.

When comparing IOPT-Tools with other Petri net editors it is possible to conclude that there are several/many editors also freely available, however only a very few of them offer the possibility of code generation. Compared to other Petri net based code generators for hardware/software systems, the tools in IOPT-Tools have a significant advantage, mainly due to the modeling capabilities of IOPT nets, but also due to its web based platform.

V. Additional Information

The IOPT tools can be freely used at http://gres.uninova.pt/IOPT-Tools. A user manual is available from the login webpage http://gres.uninova.pt/iopt_usermanual.pdf and a list of related publications can be found at http://gres.uninova.pt/iopt_publications.html. The developers can be contacted using the GRES Research Group email: gres@uninova.pt.

A self-explanatory short video presentation, without sound, is available at https://goo.gl/0cnRkf.

The editor employs a plug-in architecture to simplify the addition of third party extensions [20].

VI. Conclusions

IOPT-Tools have been successfully used to design digital controllers, targeting both software and hardware platforms [21], and has been extensively validated within several master courses at the department of Electrical and Computer Engineering at Nova University of Lisbon [11]. The cloud-based architecture and web user interface requires no local software installation and can be used from any PC or tablet, thus simplifying data sharing and collaborative work [8].

A Javascript code generator (used within the simulator tool), as well as a MatLab/Simulink code generator (currently in beta testing phase) are expected to be publicly available in the near future.

Ongoing and future work also include a new web-based Animator tool, to implement user interfaces for simulation, and tools to import/export model files from/to other Petri-net tools, as well as better support for modularity and net operations (namely net addition and net splitting operations allowing composition and decomposition of models). Finally, a configurator tool is currently being developed to allow semi-automatic assigning of platform I/O pins, definition of platform specific options, and code deployment into specific implementation platforms.

ACKNOWLEDGMENT

This work is financed by National Funds through Portuguese Agency "FCT -Fundação para a Ciência e a Tecnologia" in the framework of project PTDC/EEI-AUT/2641/2012.

References

[1] R. David and H. Alla, *Petri Nets & Grafcet; Tools for Modelling Discrete Event Systems*. Prentice Hall International (UK) Ltd, 1992.

[2] L. E. Holloway, B. H. Krogh, and A. Giua, "A Survey of Petri Net Methods for Controlled Discrete Event Systems," *Discrete Event Dynamic Systems*, vol. 7, pp. 151–190, 1997.

[3] G. Frey and M. Minas, "Editing, Visualizing, and Implementing Signal Interpreted Petri Nets," in *Proceedings of the AWPN 2000, Koblenz*, Oct. 2000, pp. 57–62.

[4] H.-M. Hanisch and A. Lüder, "A Signal Extension for Petri Nets and its Use in Controller Design," *Fundamenta Informaticae*, vol. 41, no. 4, pp. 415–431, 2000.

[5] "Petri nets tool database," accessed on 2015/07/30. [Online]. Available: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html

[6] C. A. Petri, "Kommunikation mit Automaten," Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn, Dissertation, Schriften des IIM 2, 1962.

[7] W. Reisig, *Petri nets: an Introduction*. Springer-Verlag New York, Inc., 1985.

[8] L. Gomes, F. Moutinho, and F. Pereira, "IOPT-tools – a web based tool framework for embedded systems controller development using Petri nets," in *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, Sept 2013, pp. 1–1.

[9] L. Gomes, J. Barros, A. Costa, and R. Nunes, "The Input-Output Place-Transition Petri Net Class and Associated Tools," in *Proceedings of the 5th IEEE International Conference on Industrial Informatics (INDIN'07)*, Vienna, Austria, Jul 2007.

[10] J. Ribeiro, F. Moutinho, F. Pereira, J. Barros, and L. Gomes, "An Ecore based Petri net Type Definition for PNML IOPT Models," in *Proceedings of the 9th IEEE International Conference on Industrial Informatics (INDIN 2011)*. Lisbon, Portugal: IEEE, jul 2011, http://dx.doi.org/10.1109/INDIN.2011.6034992.

[11] L. Gomes and A. Costa, "Cloud based development framework using IOPT Petri nets for embedded systems teaching," in *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, June 2014, pp. 2202–2206.

[12] F. Pereira, F. Moutinho, and L. Gomes, "IOPT-Tools - Towards cloud design automation of digital controllers with Petri nets," in *Proceedings of the 2014 IEEE International Conference on Mechatronics and Control (ICMC 2014)*. Jinzhou, China: IEEE, jul 2014.

[13] F. Pereira and L. Gomes, "Cloud based IOPT Petri net simulator to test and debug embedded system controllers," in *Technological Innovation for Cloud-Based Engineering Systems*, ser. IFIP Advances in Information and Communication Technology, L. M. Camarinha-Matos, T. A. Baldissera, G. Di Orio, and F. Marques, Eds. Springer International Publishing, 2015, vol. 450, pp. 165–175.

[14] R. Campos-Rebelo, F. Pereira, F. Moutinho, and L. Gomes, "From IOPT Petri nets to C: An automatic code generator tool," in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, July 2011, pp. 390–395.

[15] F. Pereira and L. Gomes, "Automatic synthesis of VHDL hardware components from IOPT Petri net models," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Nov 2013, pp. 2214–2219.

[16] L. Gomes, F. Moutinho, F. Pereira, J. Ribeiro, A. Costa, and João Paulo Barros, "Extending Input-Output Place-Transition Petri Nets for Distributed Controller Systems Development," in *Proceedings of the 2014 IEEE International Conference on Mechatronics and Control (ICMC 2014)*. Jinzhou, China: IEEE, jul 2014.

[17] F. Moutinho and L. Gomes, "Asynchronous-channels within Petri net-based gals distributed embedded systems modeling," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 4, pp. 2024–2033, Nov 2014.

[18] A. Yakovlev, L. Gomes, and L. Lavagno, *Hardware Design and Petri Nets*. Kluwer Academic Publishers, 2000.

[19] V. Vyatkin, "Modelling and verification of discrete control systems," 2007. [Online]. Available: http://www.fb61499.com/valid.html

[20] F. Pereira, F. Moutinho, J. Ribeiro, and L. Gomes, "Web based IOPT Petri net Editor with an extensible plugin architecture to support generic net operations," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 6151–6156.

[21] F. Pereira and L. Gomes, "FPGA based speed control of Brushless DC Motors using IOPT Petri Net models," in *Industrial Technology (ICIT), 2013 IEEE International Conference on*, Feb 2013, pp. 1011–1016.