HW accelerated Ultra Wide Band MAC protocol using SDL and SystemC

Malek Haroud, Ljubica Blažević
STMicroelectronics, AST, 39, Ch. du Champ des Filles, 1228 Plan-les-Ouates, Switzerland
{malek.haroud,ljubica.blazevic}@st.com
Armin Biere
Computer Systems Institute, ETH Zürich, RZ H15 Clausiusstr. 59, 8092 Zürich, Switzerland, biere@inf.ethz.ch

***Abstract*— In this paper we present a novel method for designing and validating a HW accelerated MAC controller for Ultra Wide band systems. Guaranteed response time and low power consumption are the two main drivers for the proposed HW/SW partitioning. We propose to use the SDL formalism in a way that facilitates the refinement verification of the SDL model against its derived SystemC implementation.**

I. Overview of standard IEEE 802.15.3

The upcoming ultra-wide-band radio technology holds great promises in revolutionizing wireless communications. UWB technology offers very high-bit rates with a very low-power operation. In the network of UWB devices, medium access control (MAC) coordinates transmission access to the channel which is shared among all devices. IEEE 802.15.3 MAC [1] is adapted as a good candidate of MAC for UWB PHY.

We distinguish two classes of MAC functionalities. Those with hard real time constraints that should be implemented in HW, whereas functionalities with soft timing constraints are implemented in software.

IEEE 802.15.3 MAC protocol is centrally coordinated, with a PicoNet Coordinator (PNC) which synchronizes the devices (DEVs) and allocates the communication resources. Even if the MAC protocol is a centralized one, the topology is ad-hoc and piconet communications are in peer to peer mode. Timing within a piconet is based on the superframe, which is illustrated in Figure 1.

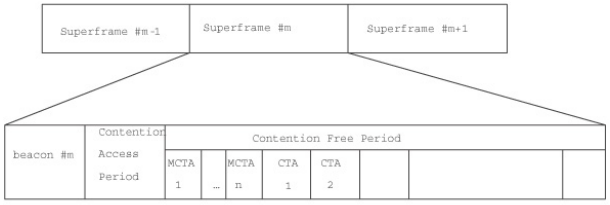

Figure 1: Super frame structure

The superframe is composed of three parts: a beacon, a Contention Access Period (CAP) and a Contention Free Period (CFP). The beacon frame is sent by the PNC at the beginning of every superframe. It is used to time-synchronize all DEVs to the PNC's clock, to set the superframe timing allocations, as well as to communicate management information for the piconet. The Contention Access Period (CAP) is used to communicate commands and non-stream asynchronous data. During CAP, DEVs access the channel using CSMA/CA. PNC divides the CFP into channel time allocation (CTA) slots. CFP is used for asynchronous and isochronous data streams. Channel access in the CFP is based on a TDMA method. Each CTA has guaranteed start time and duration within the CFP.

II. MAC HW/SW partitioning: Rationale

In order to meet the real time requirements of 802.15.3 MAC with UWB PHY, a partition of the MAC into MAC software and MAC hardware is necessary. Examples of time critical MAC functions that are to be executed in HW:

- In the case of immediate ACK policy (known at the receiver at frame reception), the receiver should send ACK frame back to the source if the frame is correctly received in a very short time. Frame verification and creation and transmission of ACK frame is done in HW.
- Beacon frame reception and decoding is another time critical operation. Since the beacon frame payload sets the superframe timing information, upon beacon reception, the device should parse its content, and be ready for frame transmitting or receiving just after the beacon.

III. MAC HW Architecture

We introduce the following MAC HW architecture for IEEE 802.15.3 MAC.

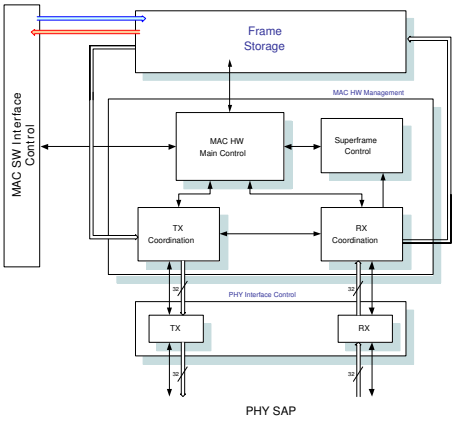

Figure 2: HW MAC architecture

The main MAC HW building blocks that handle the CFP period are:

1) MAC HW management module that comprises the following submodules:
   - Superframe Control: decodes beacon frames, identifies RX/TX opportunities during a CTA period and manages the superframe clock timer.
   - MAC HW Main Control: initiates transmission or reception process (CTA level).
   - TX Coordination: controls the transmission of one or more MAC frames in CTA (data flow control, timing, ACK policy).
   - RX Coordination: controls reception of one or more MAC frames in CTA and forwards the beacon to the Superframe Control block
2) PHY interface block is in charge of transmitting/receiving frames to/from the physical layer.
3) Frame storage is a memory storage of frames that are under control of MAC HW and that are transferred from MAC SW.

IV. Design Methodology

A. Flow

Figure 3 depicts the proposed flow and can be summarized as follows:

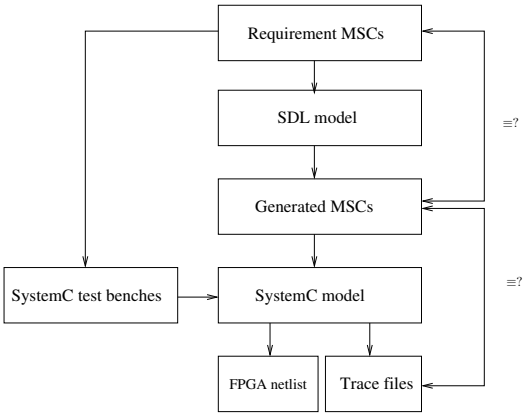

Figure 3: SDL-SystemC equivalence checking

1) specify the main scenarios of execution using MSC
2) develop the SDL model
3) simulate the SDL model and verify that the generated MSC correspond to what was specified in (1).
4) translate each SDL process into an SC_module
5) instrument the SystemC code so that it generates MSC compatible textual trace
6) verify that each timeline of an specification MSC can be reproduced by executing the SystemC model

B. SDL versus SystemC

SDL (i.e.; Specification and Description System) provides a clear practical way to unambiguously specifying, modelling and validating telecommunications system. SDL has a formal semantic and is therefore amenable to formal verification. As a matter of facts, SDL TAU Suite from Telelogic provides a validator that can explore the state space of an SDL model to uncover common problems like deadlocks or the inability to send and receive signals. Moreover, the validator can compare a Message Sequence Chart (i.e; MSC) file against the SDL model. The SDL system behavior is defined as a network of extended finite state machines that communicate with each other and with the environment using asynchronous signals. The later make the use of SDL unsuitable to model synchronous digital circuits.

On the other hand, SystemC [5], [6] is an embedded language based on C++ with an additional library allowing cycle-based simulation of concurrent processes. We limited ourselves to a sub-set of SystemC that is synthesizable and used the Synopsis SystemC compiler to derive the netlist from the 8'000 lines of SystemC source code. Our main objectives are:

1) to take advantage of the SDL Tau suite to formally validate the design
2) to use SystemC as an implementation language
3) to ensure correctness of the refinement between SDL and SystemC using input/output trace equivalence

C. Constraints

The four following constraints have been imposed in order to use the proposed flow:

- the SDL structure must reflect the actual HW structure (i.e.; one-to-one mapping between the SDL process instance set and the SystemC (i.e.; SC) module instance set)
- Each SDL signal type has a unique destination process instance that can be computed statically.
- SystemC and SDL models should have identical signal encoding (i.e.; name matching)
- SystemC module instances and the corresponding SDL process instances should have the same state encoding.

With these constraints the SDL scheduling becomes completely deterministic and allows us to define safely an input/output trace equivalence relation between SDL processes and SystemC modules taken pairwise.

Figure 4 depicts the HW MAC architecture described in SDL.

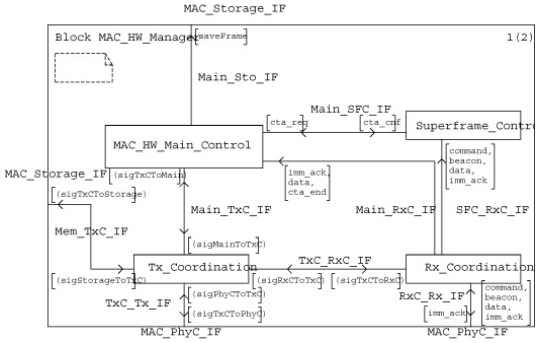

Figure 4: HW MAC SDL model

TX coordination block is further described in figure 5.

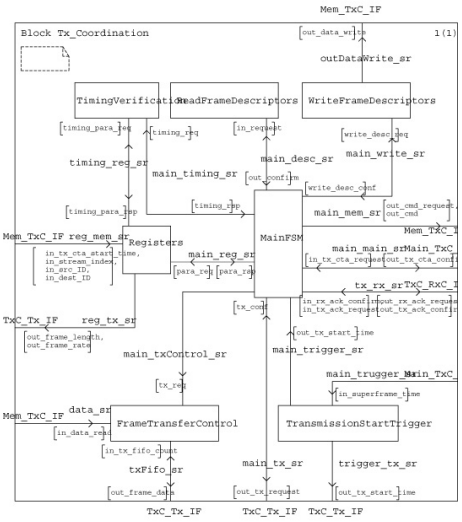

Figure 5: TX coordination structure in SDL.

When simulating the SDL system we obtain a number of message sequence charts (see figure 6). Each MSC corresponds to one test case and can be used as specification that should be automatically cross validated against the SDL model.

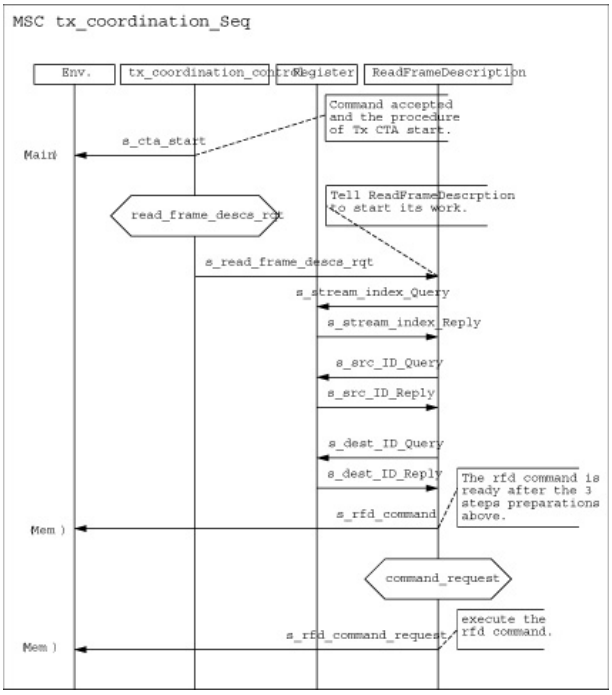

Figure 6: MSC used for specifying and equivalence checking

Message Sequence Charts also have a textual representation (see figure 7) that enable us to further validate the SystemC code that refines the SDL model just by annotating the code with the corresponding traces.

```
msc tx_coordination_Seq;
tx_coordination_control: instance;
Env.: instance;
Register: instance;
ReadFrameDescription: instance;
tx_coordination_control:
  out s_cta_start,1 to Env.
Env.:
  in s_cta_start,1
   from tx_coordination_control;
tx_coordination_control:
  condition read_frame_descs_rqt;
tx_coordination_control:
  out s_read_frame_descs_rqt,2
      to ReadFrameDescription;
.....
endmsc;
```

Figure 7: Textual representation of an MSC

In code fragment represented in figure 5, the textual form (i.e.; ITU Z.120) of one message sequence chart is presented. This MSC starts with the declaration of the instances involved in the scenario. A message within an MSC is a relation between an output and an input. The output may come from either the environment or an instance. A message exchanged between two instances can be split into two events: the message input and the message output. The correspondence between message outputs and message inputs has to be defined uniquely. This trace can be split to produce one trace file representing one trajectory in the SDL process instance. When the instrumented SystemC code is run, the execution should produce one trace file per SC_module instance that must correspond to the SDL process trace discussed previously.

D. Translating SDL into SystemC

At the lowest part of the SDL system description hierarchy, we find SDL processes that implement the actual functionality of the model. The following code represents part of the implementation of one SDL process.

```
Process ReadFrameDescriptors;
start ;
  nextstate init;
  state init;
    input in_request;
    nextstate read_first_word;
    ...
endprocess ReadFrameDescriptors;
```

The graphical representation is depicted in figure 8.

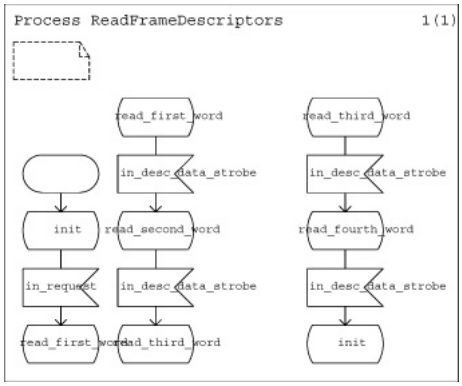

Figure 8: Part of the SDL process for reading the frame description

An SDL process is basically an extended finite state machine that can be translated very easily to SystemC since

communication services are already part of the language (See figure 7).

```
SC_MODULE (ReadFrameDescriptors){
  enum  RFD_States {init,read_first_word,..};
  sc_signal<RFD_States> current_state;
  sc_in<bool> in_request;
  SC_CTOR (ReadFrameDescriptors){
    SC_METHOD(get_Next_State)
      << current_state
      << in_request;
  }
}
ReadFrameDescriptors::get_Next_State (){
  switch (current_state){
    case init:
       if (in_request.read()){
         current_state=read_first_word;
       }
    break;
  case read_first_word:
    ...
    break;
  }
}
```

Figure 7: SystemC synchronous implementation of an SDL process

If we do not want to break the asynchronous hypothesis imposed by the semantic of the SDL language, we can still use the SC_fifo to simulate the input queue associated to the SDL process. However, the SC_fifo is not synthesizable (See figure 7).

```
SC_MODULE (ReadFrameDescriptors){
  enum  RFD_States {init,read_first_word,..};
  sc_signal<RFD_States> current_state;
  sc_fifo<RFD_signal_t> in_fifo;
  SC_CTOR (ReadFrameDescriptors){
    SC_METHOD(get_Next_State)
      << current_state
      << in_fifo;
  }
}
ReadFrameDescriptors::
get_Next_State (){
  RFD_signal_t recv_sig;
  recv_sig = in_fifo.read();
  switch (current_state){
    case init:
      if (recv_sig.id==in_request){
         if (in_request.read()){
           current_state=read_first_word;
         }
     }
    break;
    case read_first_word:
      ...
    break;
  }
}
```

Figure 7: SystemC asynchronous implementation of an SDL process

V. Conclusion

In this paper, we presented the development process of our 802.15.3 MAC HW architecture. Tight timing constraints and low power consumption requirements guided the HW/SW partitioning. The use of SDL from the TAU environment allows rigorous protocol modeling and verification. On the other hand, the use of SystemC enables FPGA netlist synthesis while providing an easy target language for the manual translation from the SDL specification. Finally, input/output trace equivalence was presented in order to establish the correctness of the manual translation from SDL to SystemC. Based on our experience in WLAN technology, we conclude that the use of SDL and SystemC is a good combination for developing and validating robust HW implementation of medium access controllers.

Acknowledgment

The authors would like to thank Renato Villan for producing SystemC code and also Zehning Peng for developing the SDL/MSC models.

References

[1] IEEE Standard 802: Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPAN), 2003.

[2] ITU-T: ITU-T Recommendation Z.100 (11/99). SDL: Specification and Description Language, 1999.

[3] Telelogic AB: Telelogic Tau SDL Suite, 2003. Available from http://www.telelogic.com/products/tau/sdl

[4] S. Leue and Ph. Oechslin: From SDL Specifications to Optimized Parallel Protocol Implementations. In: M. Ito and G. Neufeld (eds.), Workshop Proceedings of the Fourth International IFIP Workshop on Protocols for High Speed Networks, pages 308-328, 1994.

[5] Open SystemC Initiative
Synopsys Inc, CoWare Inc, Frontier Inc. SYSTEM C Version 1.0 User s Guide, 2000.

[6] The Simulation Semantics of SystemC
Wolfgang Mueller C-LAB/Paderborn University Paderborn, Germany Juergen Ruf, Dirk Hoffmann, Joachim Gerlach, Thomas Kropf, Wolfgang Rosenstiehl University of Tuebingen Tuebingen, Germany

[7] High-Level Behavioral SDL Model for the IEEE 802.15.3 MAC Protocol Daniel Dietterle, Irina Babanskaja, Kai Dombrowski, Rolf Kraemer