An Experiment to Introduce Interrupts in SDL

alain.clouard@st.com
emmanuel.gaudin@pragmadev.com

PragmaDev

- French software editor based in Paris
- Dedicated to the development of modeling and testing tools for event driven applications
- Underlying technologies are:
  - SDL
  - SDL-RT
  - TTCN-3
- Modeling capabilities:
  - Simulation
  - Model checking
  - Test generation
  - Code generation

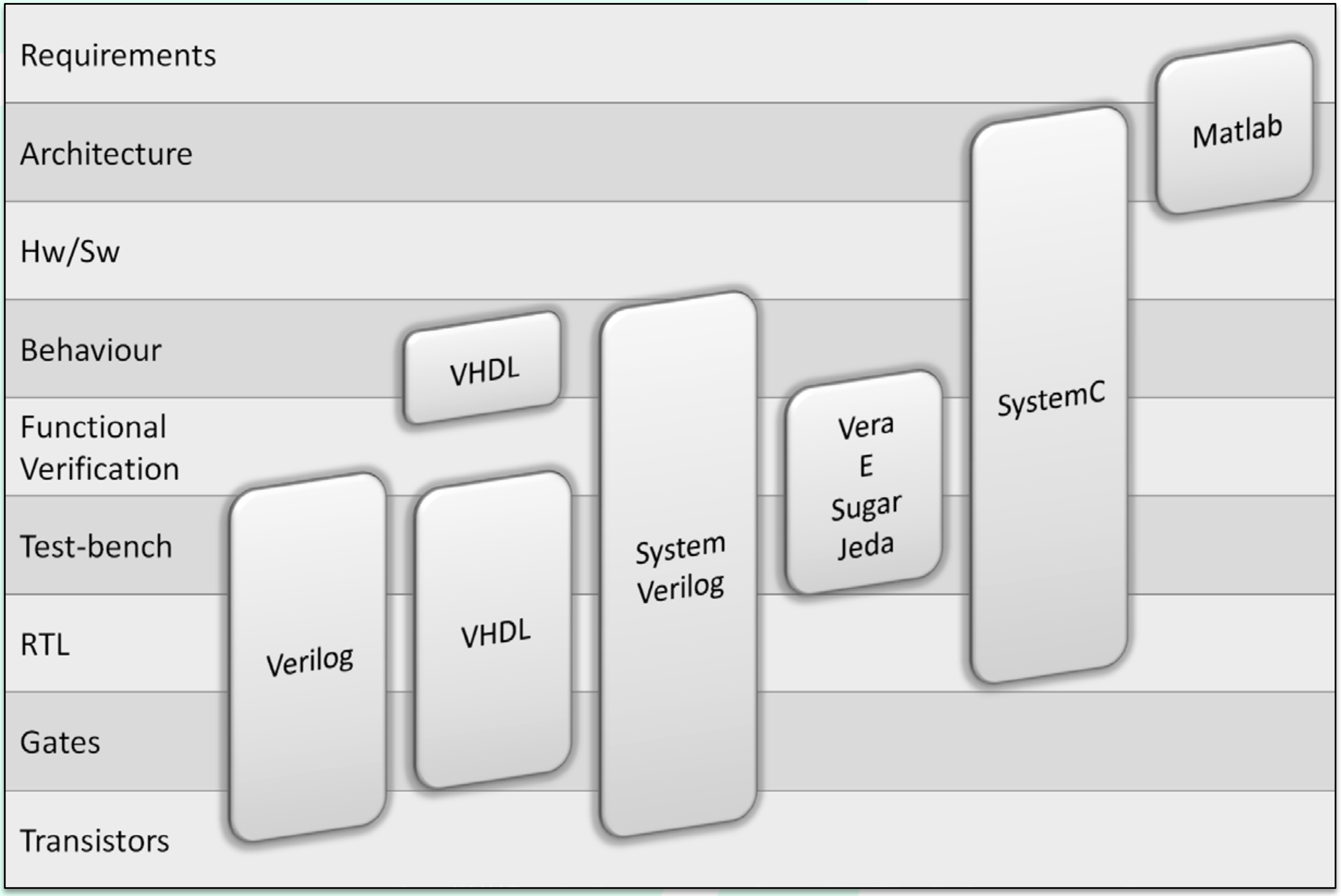
ST Microelectronics

- A world leader in providing the semiconductor solutions that make a positive contribution to people’s lives, both today and in the future
- Two main product categories:
  - Embedded processing
  - Sense & Power and Automotive Products

Introduction

- Specific modeling for hardware design
  - Synthesizable
  - Cycle accurate
  - Bit accurate
- Abstraction is very low, simulation requires a lot of processing, close to executing the real implementation

Introduction



Needs

- Need for a higher abstraction level during the initial conceptual stage of a product development
- SDL models
  - Asynchronous semantic of execution
  - Executable => verifiable
- Hardware models
  - Naturally clock based => synchronous
  - But system level synchronisation is based on asynchronous events such as interrupts
- The use of SDL to describe an interrupt mechanism needed to be investigated.

Interrupts

- Can occur at any time
- Interrupt the flow of execution
- Can modify the values of variables used in the standard flow of execution
- Because of that, interrupts can be masked to guarantee valid values of variables

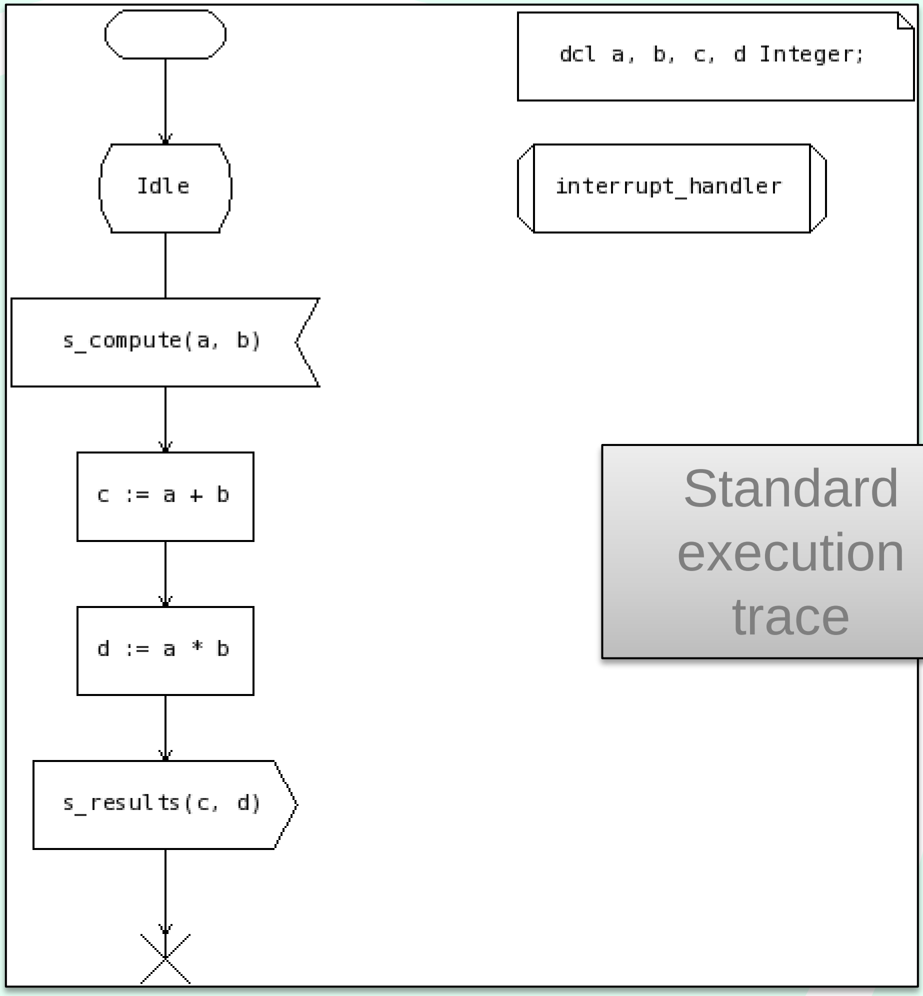
SDL procedures

- Can not occur at any time, they are explicitly called
- Interrupt the flow of execution until procedure returns
- Procedures might have parameters and return value
- Procedures might have states
- Can modify the values of variables of the caller

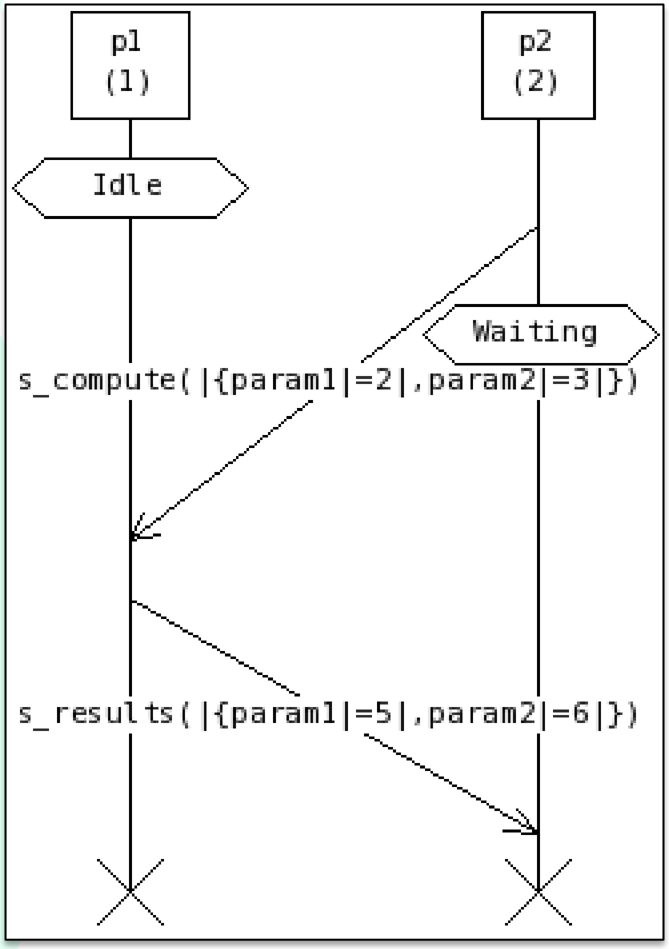
Matching the concepts

- A procedure call without any parameter, nor return values, nor states could be used to represent an interrupt.
- Need a way to call the interrupt at any time
  - Slight twist in PragmaDev SDL Simulator to be able to call a procedure at any time
  - Execution must be stopped
  - A specific command is called

Example

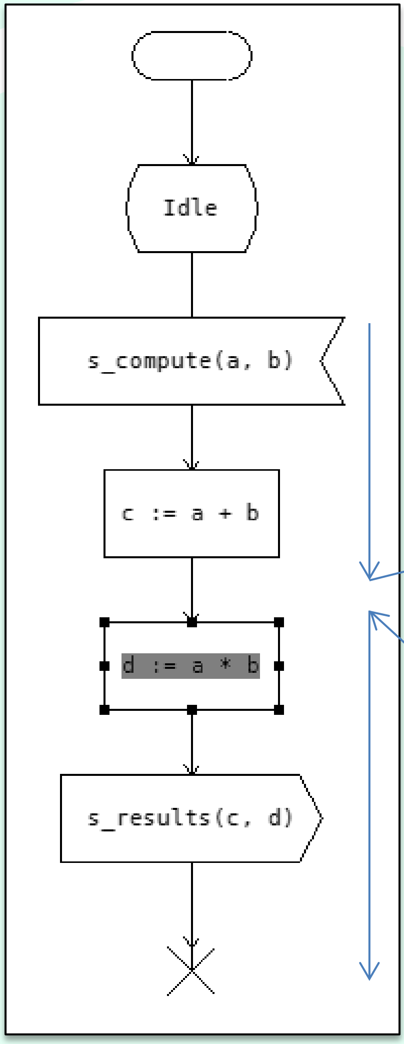


Standard execution trace

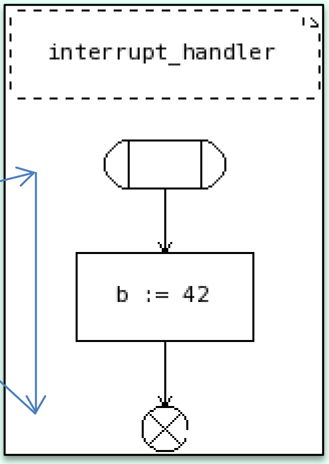


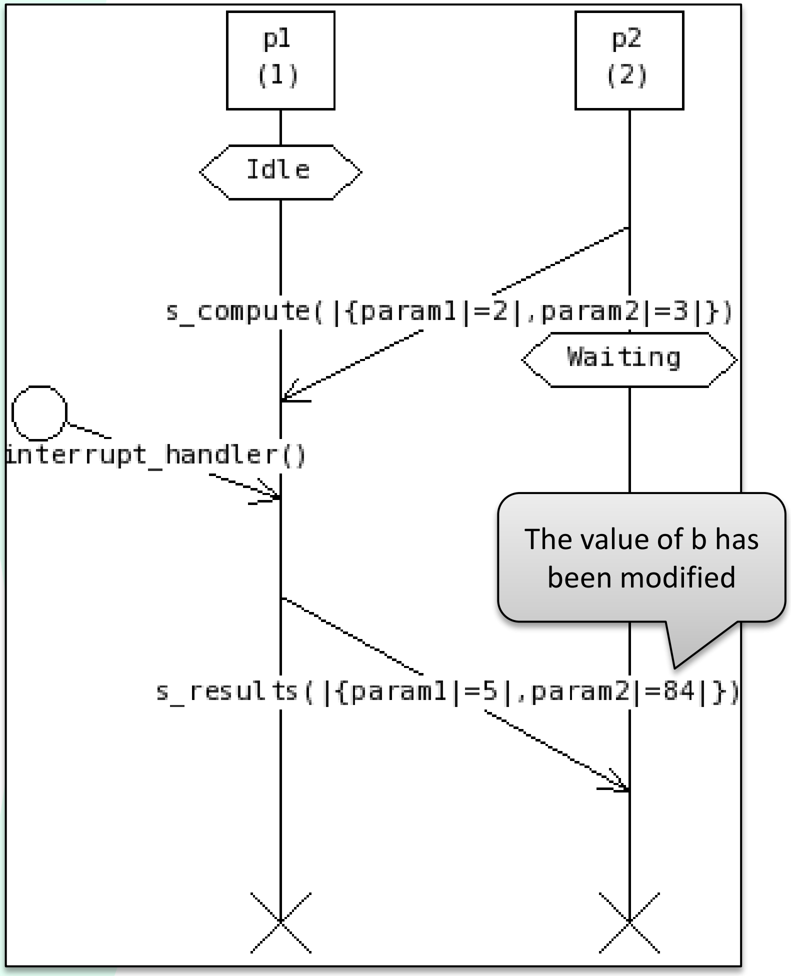
p1
(1)
p2
(2)
Idle
Waiting
s_compute(|{param1|=2|,param2|=3|})
s_results(|{param1|=5|,param2|=6|})

Example



- Breakpoint stop
- Procedure call
- Resume execution




p1
(1)
p2
(2)
Idle
s_compute(|{param1|=2|,param2|=3|})
Waiting
interrupt_handler()
The value of b has been modified
s_results(|{param1|=5|,param2|=84|})

Conclusion

- SDL can be used to describe interrupts from a functional point of view.
- The twist to execution semantic is very light
- Needs to be presented to potential users for feedback