Motivation: Assuring Timing Constraints

**Input/Output Timing Constraint**

- Ensuring **timing constraints** is important in developing embedded systems “Given input generated from the environment, a system shall produce output within a bounded delay”

  Pacemakers  
  Brake system in Cars

**Background : Model-Based Implementation**

From a high-level **model**, an **implementation is systematically built to make sure the timing constraints are satisfied**

1. **Timing Requirement**  
2. **Model**  
3. **Software**  
4. **Software Platform**

**Platform-Independent Abstraction**

- **Reason 1**: Abstraction of timing information for efficient model verification
- **Reason 2**: Lack of platform-specific timing information during the modeling phase

**Challenges**

**T/L Abstraction between the model and the implementation**

- The behavior of an implementation is affected by the **complex interactions** among various platform-specific sources that appear at different system boundaries

**Problem Statement**

Propose a way to formally verify the timed behavior of an implementation that has been developed from a platform-independent model

**Example: Infusion Pump Systems**

Platform-Independent Model (PIM)

- **Start** infusion  
- **Stop** infusion

**Case Study: Comparison of Delay in Testing and Verification**

**Comparison of Delay**

- **Bolus-Request Scenario**

<table>
<thead>
<tr>
<th>Test Value</th>
<th>Input Delay</th>
<th>Output Delay</th>
<th>Buffer Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1430ms</td>
<td>490ms</td>
<td>X</td>
</tr>
<tr>
<td>GPCA</td>
<td>1740ms</td>
<td>375ms</td>
<td>X</td>
</tr>
</tbody>
</table>

**Usage of the PSM**

- One can check whether the delay between an input and an output of an implementation is **bounded or unbounded** by formally verifying the system constraints (e.g., buffer overflow occurrence).
- If the delay is bounded, then one can find the upper bound of the input-output delay, and determine how much this bound deviates from the expected timing constraints.

**Examples of Implementation Schemes**

**MC Boundary** implementation schemes characterize the timing information required to implement the interaction between the platform and the environment

**Examples of MC-Boundary Interaction**

- Whenever a patient requests a bolus, the signal is **sustained** [20ms, 100ms]
- A platform **polls** the button every 10ms to detect input from the patient
- The detected input requires input processing time [100ms, 300ms] until it is ready to be read by the platform-independent code

**IO Boundary** implementation schemes characterize the timing information required to implement the interaction between the platform-independent code and the platform

**Examples of IO-Boundary Interaction**

- The platform-independent code reads all inputs from a buffer of size 5 upon each invocation

**Platform-Specific Model (PSM) Transformation**

Transformation from PIM to PSM

- For each implementation scheme, modeling patterns are pre-defined to capture the platform-specific delay occurring at the two system boundaries.
- The PSM is obtained by systematically composing the PIM with the newly constructed platform-specific part through their interfaces

This research is supported in part by NSF CNS-1035715 (CPS:Large: Assuring the Safety, Security and Reliability of Medical Device Cyber Physical Systems) and DGIST Research and Development Program at CPS Global Center and James S. McDonnell Foundation Postdoctoral Fellowship Award.