LibGALS: A Library for GALS Systems Design and Modeling

Wei-Tsun Sun*, Zoran Salcic and Avinash Malik

Department of Electrical and Computer Engineering
University of Auckland
New Zealand

Email : wsun013@aucklanduni.ac.nz*

January 18 2009

What is LibGALS ?

- A library implemented based on C to support GALS MoC
  - GALS – Globally Asynchronous Locally Synchronous
  - Provide programming interface to describe GALS Systems
  - Programming constructs are intuitive – no low-level details on communications and synchronizations
  - Not only for single processor systems but also for multi-core/multi-processing architectures

A GALS software system (program)

- A system can be described and modeled by a number of concurrent sequential behaviors

- Behaviors can run at the same pace or different speeds

- Behaviors require communications and synchronizations :
  - Communications used to exchange information
  - Synchronizations to maintain integrity of the concurrency

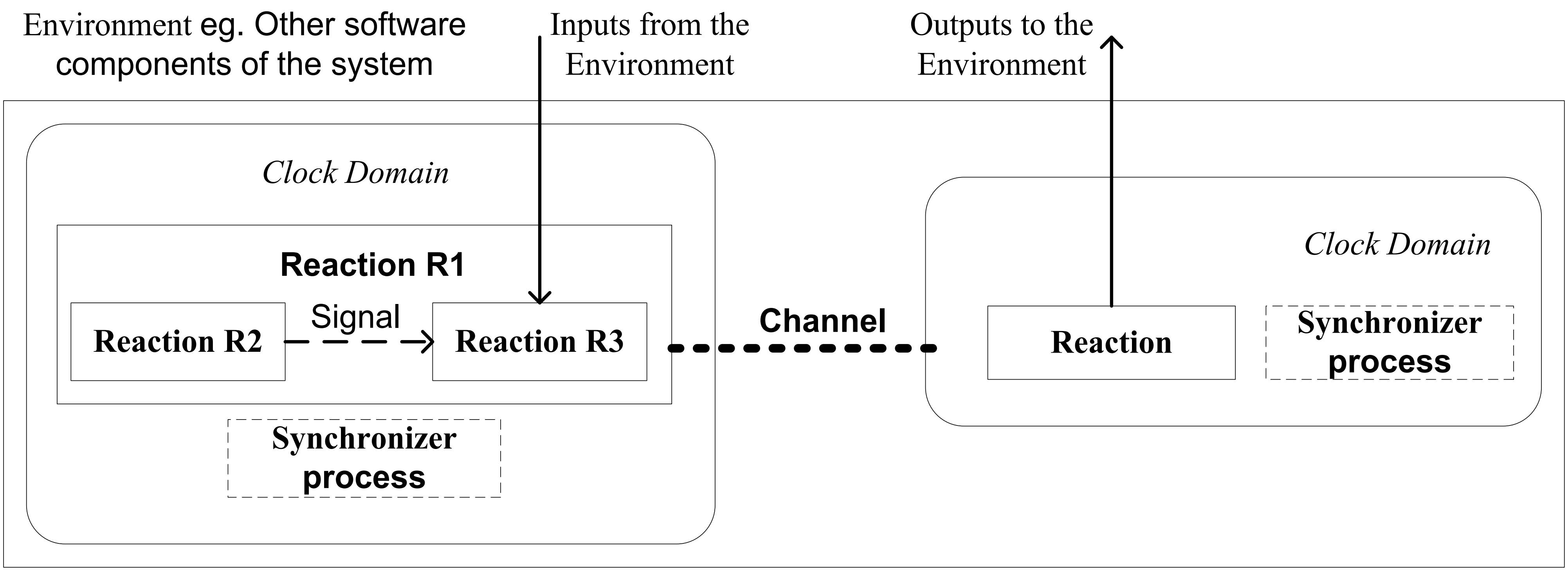
What is a GALS system in a LibGALS program ?

- A program that describes a GALS system using LibGALS is called a LibGALS program

- Entities of a LibGALS program consists of basic building blocks including:
  - Clock domains – asynchronous behaviors
  - Reactions – synchronous behaviors
  - Channels – communication between asynchronous behaviors
  - Signals – communication between synchronous behaviors

What is a GALS system in a LibGALS program ?

- A GALS systems is described with these building blocks



What is a GALS system in a LibGALS program ?

- A *clock domain* is an entity in a GALS system which may consists of one or more reactions

- A GALS program can include one or more clock domains

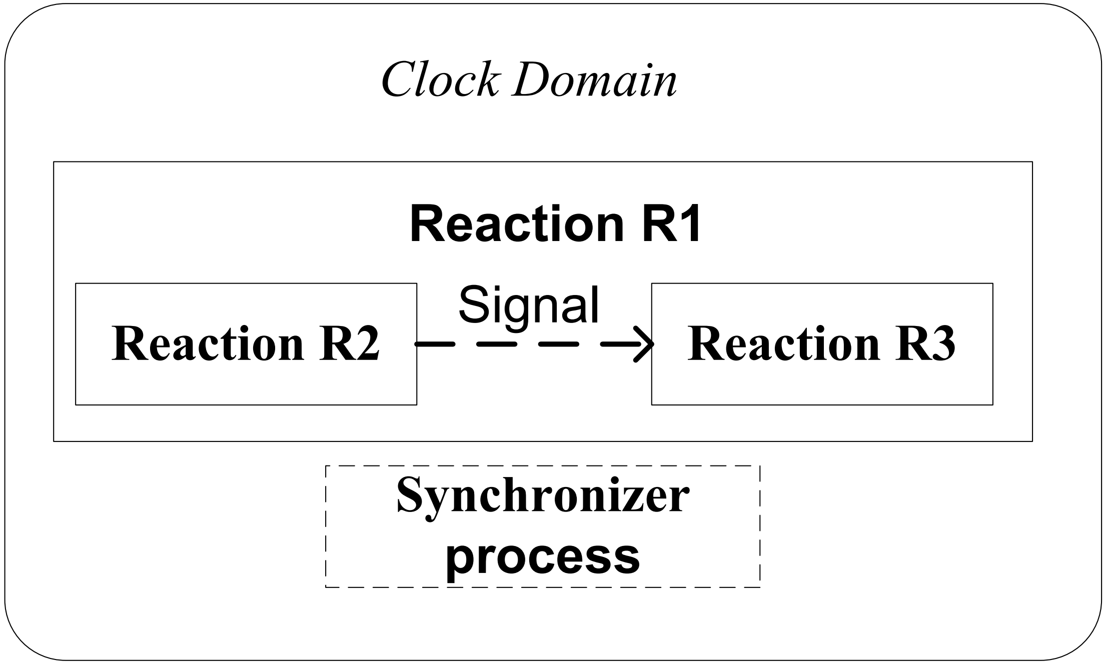
- Reactions are acting synchronously
  - Follows logical ticks – barrier synchronization
  - Allow creation of children reactions

- Clock domains are acting asynchronously to each other

What is a GALS system in a LibGALS program ?

- A reaction can be a composition of other reactions known as *children reactions*

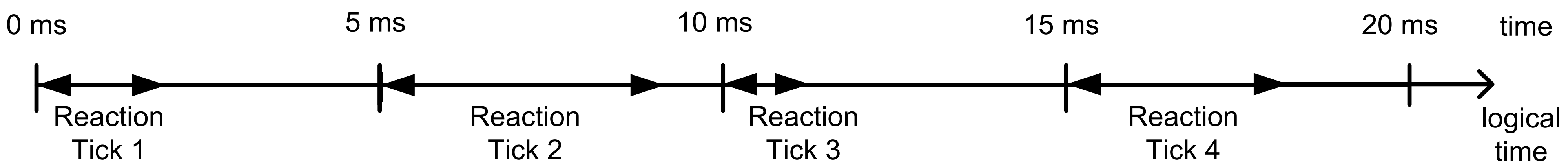
- As illustrated, reaction R2 and R3 are children reactions of reaction R1



What is a GALS system in a LibGALS program ?

- Reactions are execute in lock-steps called ticks

- Ticks are logical times, which can be of different length in real time units



- Reactions of the same clock domain share one common tick, a reaction will be in the same tick as long as all reactions of the same clock domain finishes their ticks

What is a GALS system in a LibGALS program ?

- Reactions communicate and synchronize with each other via *signals*

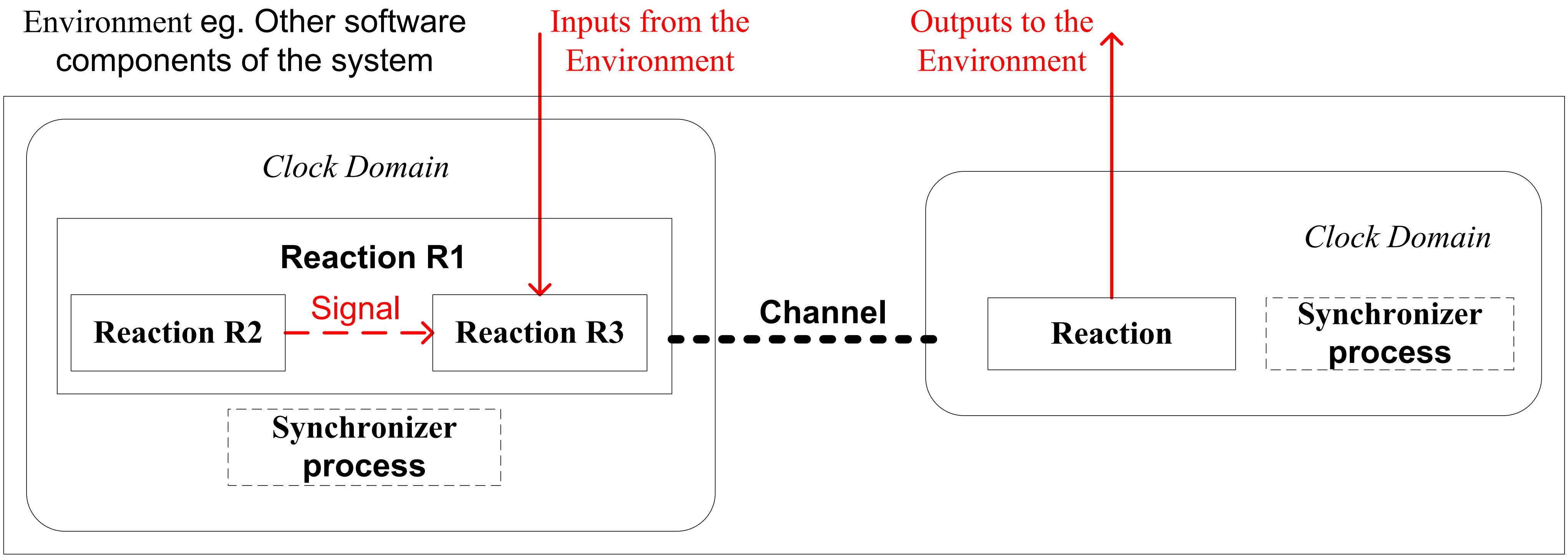
- Two types of signals – *pure signals* and *valued signals*
  - A pure signal can either be present or absent
  - A valued signal is a pure signal with extra value attribute
  - The value of a value signal is persistent

- Signal can be made present through *emission* by a reaction

- The presence of signal is broadcast within a clock domain at the current tick. The same as Esterel and SystemJ

What is a GALS system in a LibGALS program ?

- Signals in GALS system



What is a GALS system in a LibGALS program ?

- A signal is used within a clock domain

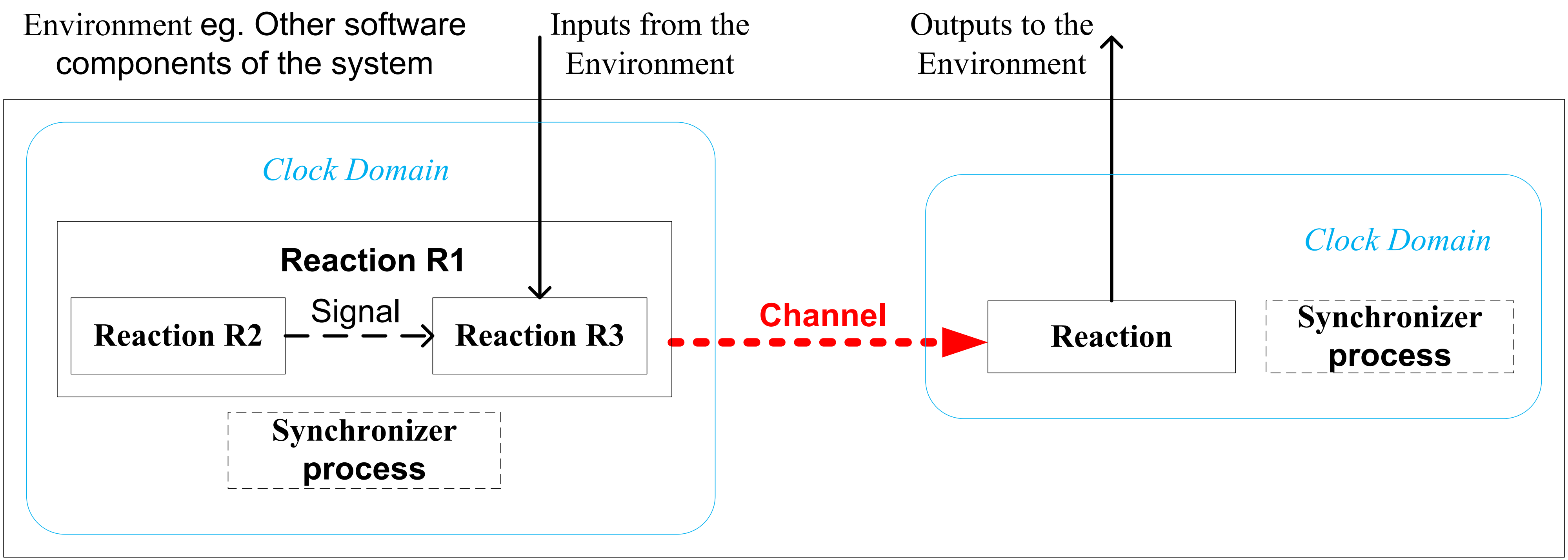
- Signals are also used as inputs and outputs of a clock domain
  - To sample environment as input to the clock domain
  - Similar, to generate outputs to the environments
  - Inputs and outputs are occurred according to ticks

- Communication between clocks domains are via *channels*
  - Channels are point-to-point and uni-directional
  - Data sent by channel are *copied* instead of shared

- Channels synchronized by rendezvous as in CSP

What is a GALS system in a LibGALS program ?

- Channels in GALS system



How LibGALS is implemented ?

- A set of data structures is established to "bookkeep" the status of clock domains, reactions, channels, and signals
- Clock domains are containers which link with relevant reactions, and signals, and is registered with channels
- Each reaction is implemented as a process/thread. *Pthread* is used at the current implementation of the LibGALS
- To resolve dependencies between reaction process-es/threads , a *synchronizer process* is introduced
- LibGALS program is *multi-thread* in nature

How LibGALS is implemented ?

- Each clock domain will have a synchronizer process
  - Synchronizer process is formed automatically with the creation of a clock domain
  - Synchronizer is programmer invisible – abstracts the details out

- Semaphores are used in libGALS internally as part of its data structure
  - Only process and semaphore operations are essential to every OS and are required by the LibGALS
  - Hence LibGALS is highly portable!

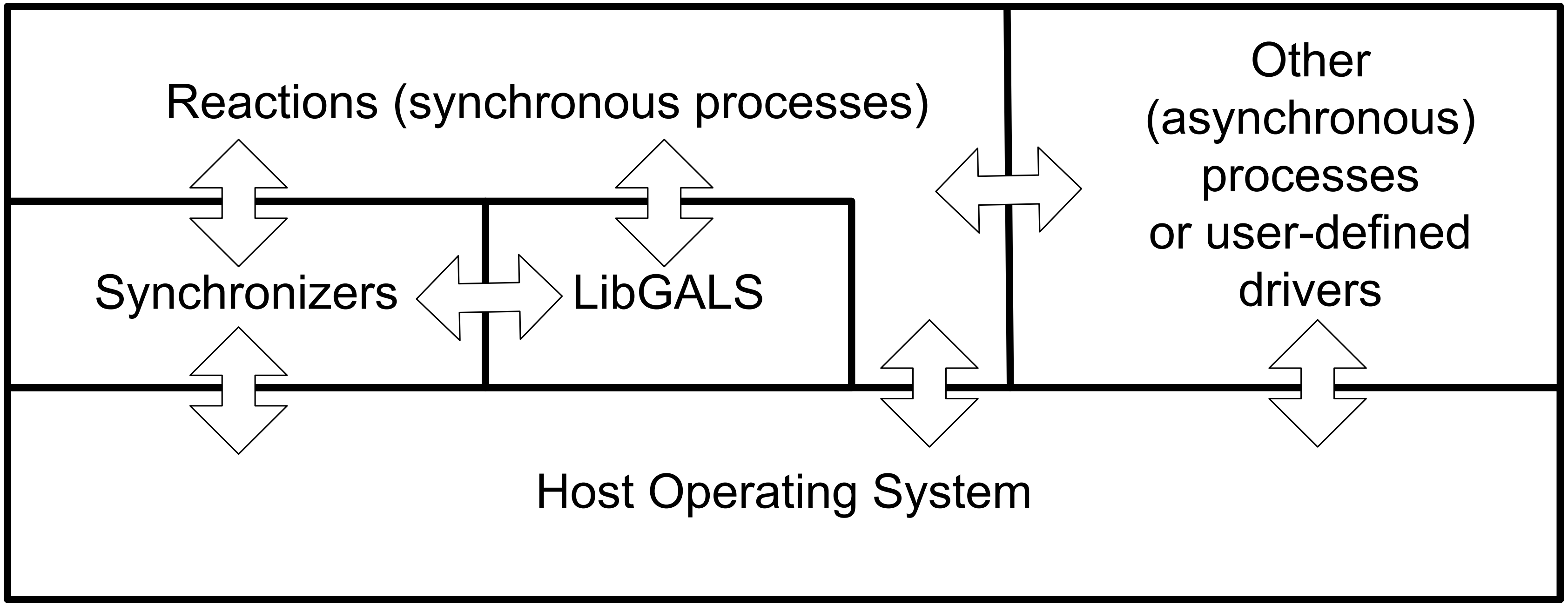
Inputs and outputs of a clock domain

- Inputs and outputs are implemented as functions registered with clock domains
  - They are known as input/output functions
  - They are activated at tick edges

- Inputs and outputs functions are interfaces to other programs and device drivers outside of the LigGALS program

How LibGALS is positioned ?

- LibGALS is implemented by using OS services
- Reactions and synchronizers are implemented based on LibGALS and other OS services
- Other software communicate with reactions via I/O functions



How to use LibGALS

- LibGALS provides a set of application programming interfaces (APIs) which are intuitive and easy to use

API Function Name	Description
createClockDomain	Create a clock domain
createReaction	Create a reaction within a clock domain
create[Signal | Trap]	Create an instance of a signal or a trap
startClockDomain	Start running a clock domain
initReaction/ endinitReaction	Initialize a reaction and end initialization of the reaction
getArgument	Get an argument passed to the reaction
endReaction	End a reaction, called if the reaction is not a child reaction

How to use LibGALS

register[Emitter|Trap]	Register a process as a signal emitter or a trap thrower
emit | sustain	Emit/broadcast (or sustain) a signal
present	Check if a signal is present
pause	Enforce end of tick for a reaction
await	Wait for the presence of a signal
[strong|weak] abort/endAbort	Start and end of a preemption block; preempt if monitored signals are present
suspend/endSuspend	Suspend a reaction by one tick if a monitored signals are present
setTrap/endTrap	Set and end the scope of the trap

How to use LibGALS

AND,OR,NOT,REP	Form a combined signal expression
value	Acquire the value of a signal
pre[Value]	Get the presence status and value of a signal in the previous tick
createChannel	Create a channel connecting two clock domains
send/receive	Send and receive data between reactions in different clock domains via a channel

How LibGALS is implemented ?

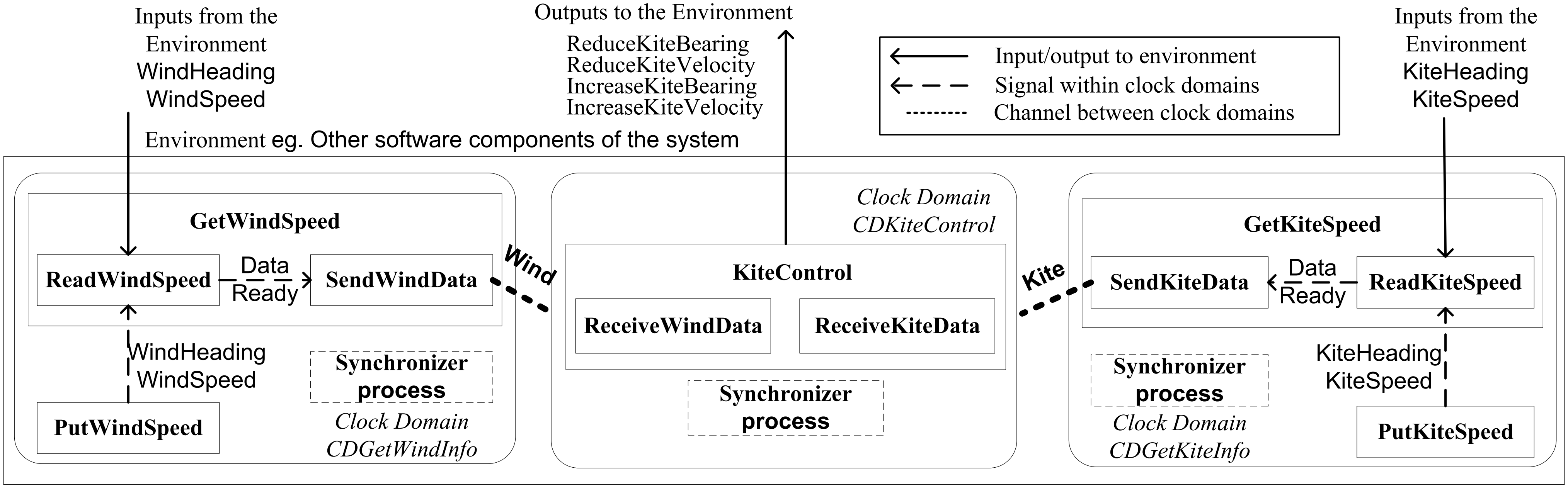
- APIs are implemented by using data structures and basic building blocks of LibGALS

- For example, abort is implemented with checking presence of *signals* with *goto* statements built-in as *macros*

- Another example, *traps* behave similarly to signals, whose activation is triggered by mechanism similar to *signal emission*
  - A trap is a variant (more restricted version) of a signal

Uses of LibGALS

- Less than 200 lines of code (and most of them are in this paper) are required to describe a power kite controller system shown below



Uses of LibGALS

- Allows to implement existing languages which only support single threaded-implementation to use multiple threads:

SystemJ Statements	**Mappings with LibGALS**
present S	if (Present(S))
emit S;	emit(S);
pause;	pause();

abort (S)	strongAbort(S, AbortName);
	…
	endAbort(S, AbortName);

Advantages of multi-threaded approach

- LibGALS adapts process/thread approach to implement reactions hence reactions can perform when signal dependencies allow – faster in computation time, dependencies resolved dynamically

- LibGALS does not require JVM as SystemJ – smaller code size

Example	Average tick time (µs)		Code Size (Bytes)	
	LibGALS	SystemJ	LibGALS	SystemJ
2CD Freq Relay	27.67	75.23	33,865	101,469
2CD KiteController	11.37	27.16	9,431	59,296
2CD Async Proto	48.37	16.25	13,078	52,800
2CD Data Comp	18.23	26.37	865	10,920
3CD Data Comp	17.72	39.28	975	11,944
4CD Data Comp	17.43	56.62	1,085	13,010

Conclusions

- LibGALS enables designer to describe GALS systems easily
  - GALS systems are collections of concurrent processes

- LibGALS APIs can be used to abstract out details of communication and synchronization
  - Less error-prone than using traditional threading libraries
  - No need to play around with low level constructs

- LibGALS programs utilize the advantage of multi-processing/ multi-core architecture
  - Better performance !

Future developments

- Dynamic creations of clock domains
  - Clock domains and their reactions can be migrated from one machine to another
  - Possible load sharing or distributed computing to achieve even better performance
  - At this moment, a plug-in system for LibGALS has been developed

- Integrate with other researches to establish a framework to enable system designs
  - Simulate LibGALS programs and SystemC is possible and working under progress

Questions ?

Thank you

&

Questions ?