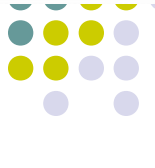
MANCHESTER 1824

The University of Manchester

Newcastle University

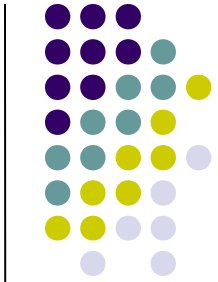


Desynchronisation Technique using Petri Nets

Sohini Dasgupta
University of Manchester

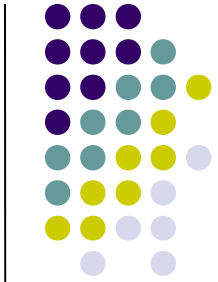
Alex Yakovlev
Newcastle University

Overview of the talk



- Introduction
- Background
- Notion of Desynchronisation
- Pre-requisites for Locality formation
- Desynchronisation Methodology
- Partitioning Correctness
- GALS implementation
- Conclusion and Future Work

Introduction



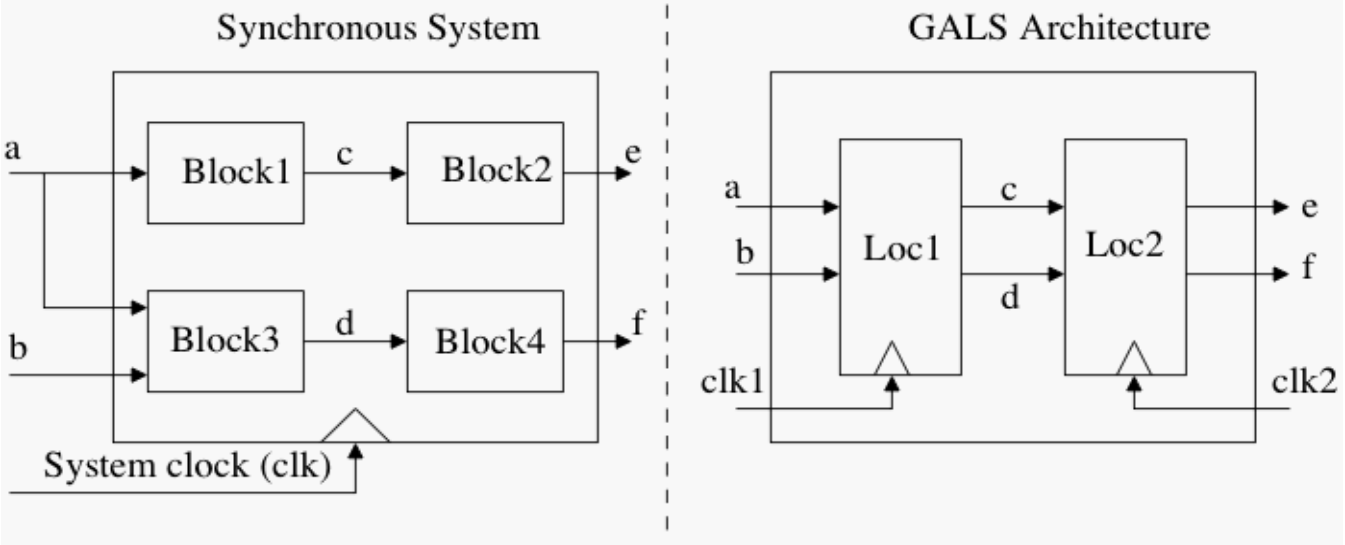
- Synchronous Model – Single clock
  Synchronous assumption: associate globally synchronous paradigm with maximal firing parallelism

- GALS Model – Multiple clock
  GALS assumption: inputs may arrive in any order and at any instant of time.

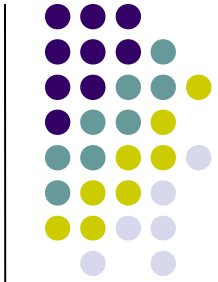
- Moving from Synchronous-> GALS-> Asynchronous
  - In order to deal with systems efficiently we require to handle heterogeneity (unrelated clocks).
  - Support multi-core architectures
  - Avoid global clock distribution in complex systems

Desynchronisation Technique

- The notion of synchrony/asynchrony is applied in our framework through semantics of signal transition execution.
- Partitioning of blocks and refinement of interfaces to handle asynchronous communication.
- Allocation of localities that form locally synchronous blocks using proposed method of desynchronisation.

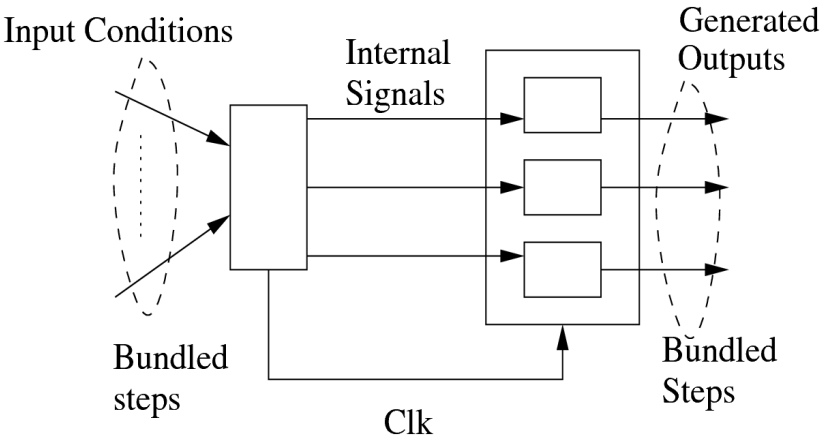


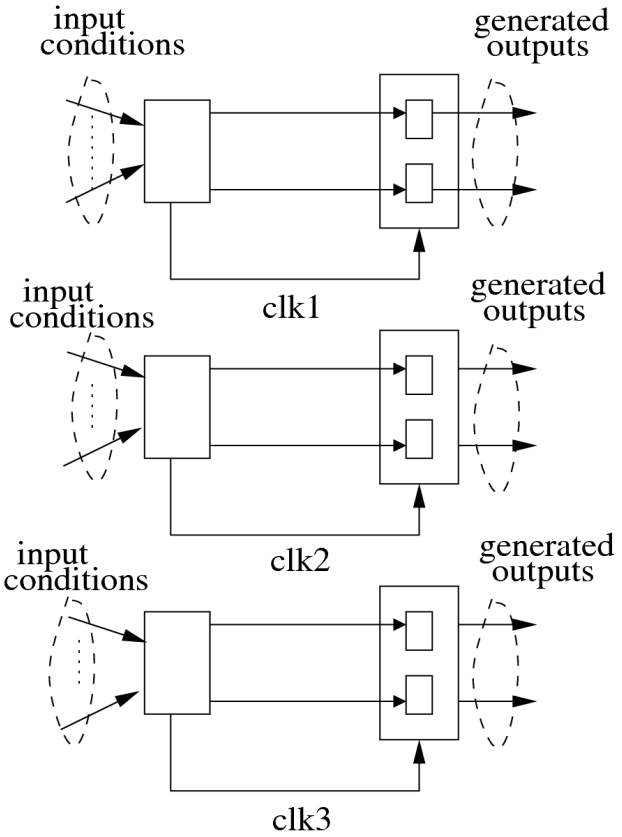
Notion of Localities



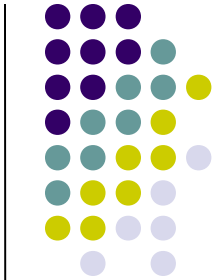
- Localities refer to components that comprise a synchronous subsystem and their associated actions as individual blocks.
- Incorporate additional ordering constraints on the input and output signals.
- Each individual block will behave like an independently clocked block.
- Global clock can be eliminated as timing is reconstructed from internal actions.

Notion of Localities





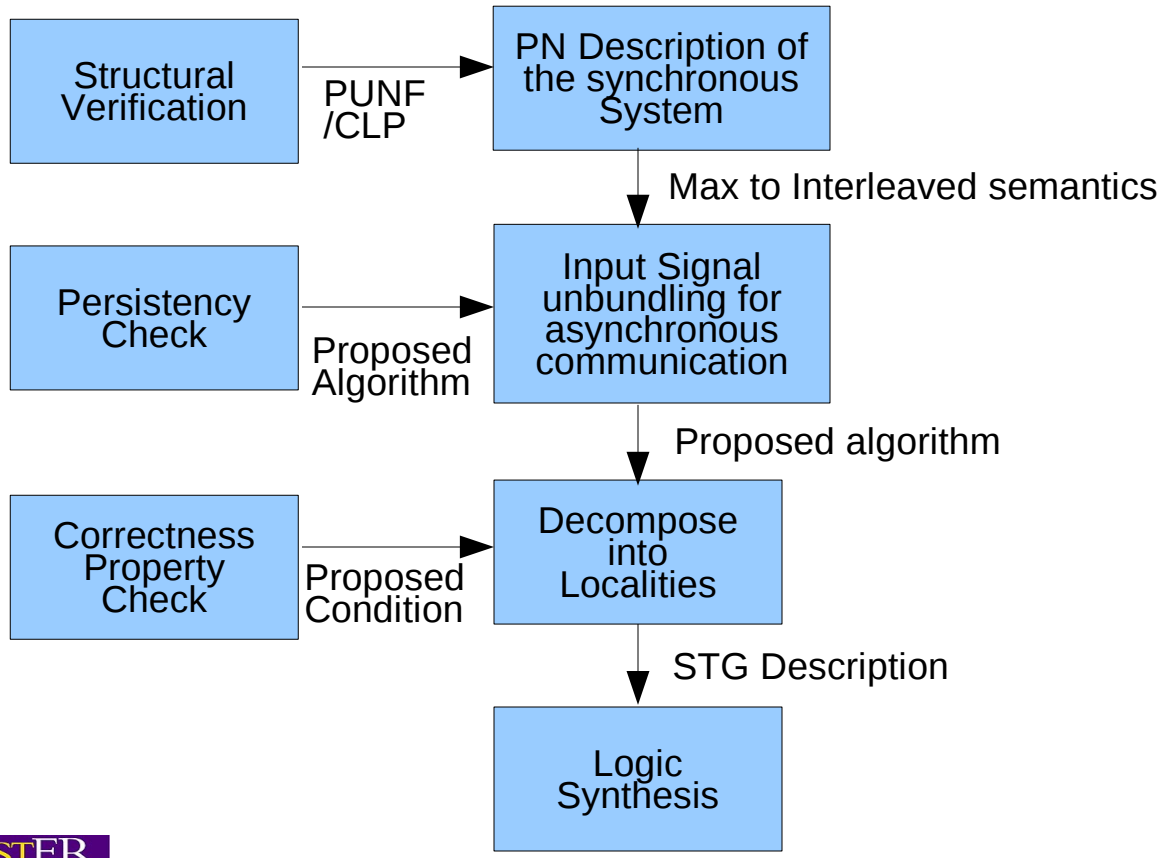
Correctness properties

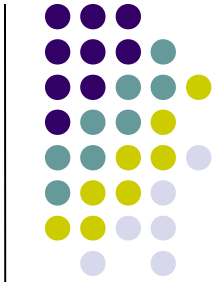


- The technique to obtain a distributed architecture from a globally synchronous system must satisfy two essential correctness properties, namely,
  - Semantics preservation of the original synchronous system
  - Prevention of deadlocks

Desynchronisation Method Flow

- Presentation of a formal framework for the desynchronisation of globally synchronous systems





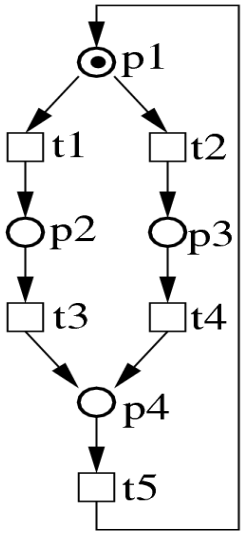
Background

Background

- ***Petri Net***

  A *Petri net* is a quadruple $PN = P, T, F, \mu_0$ where, *P* is a set of *places*, *T* is a set of *transitions*, *F* is an *arc* that denotes the flow relation:

$$F \subseteq \{(P \times T) \cup (T \times P)\}$$

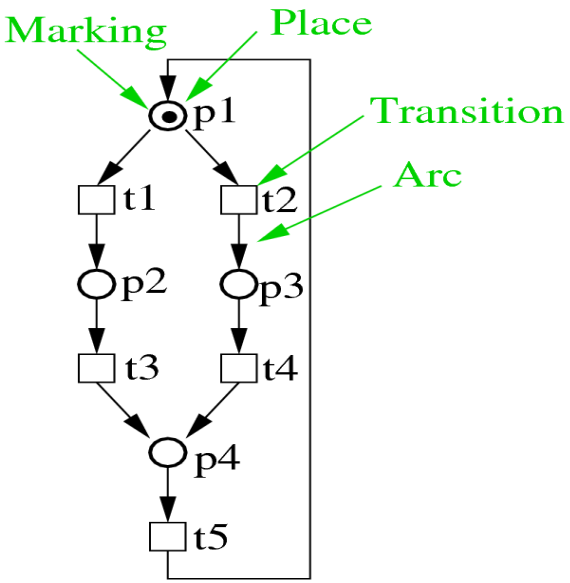


Background

- ***Petri Net***

  A *Petri net* is a quadruple $PN = P, T, F, \mu$. where, *P* is a set of *places*, *T* is a set of *transitions*, *F* is an *arc* that denotes the flow relation:

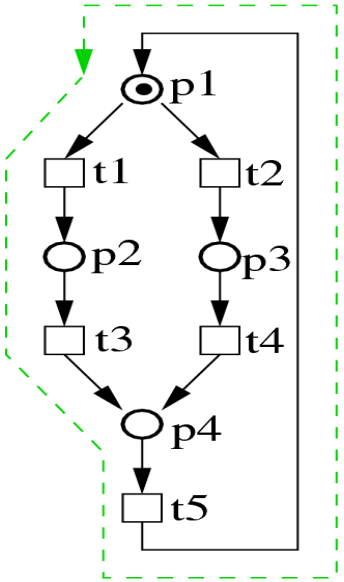
$$F \subseteq \{(P \times T) \cup (T \times P)\}$$

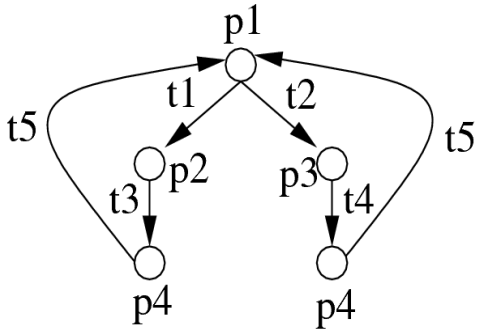


Background

- ***Reachable marking***

  A *marking* μ_1 can be reached from μ_0 if there exists a *firing sequence* that will yield μ_1.





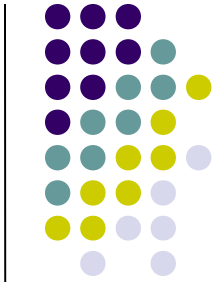
Background Definitions

- ***Step***

  A *step* is a multiset of transitions $U : T \rightarrow \mathbb{N}$, where $\mathbb{N}$ is a set of natural numbers.
  In a *maximal step semantics* a PN model evolves through the concurrent firing of transition sets, given the associated external conditions are true.

- ***Persistency***

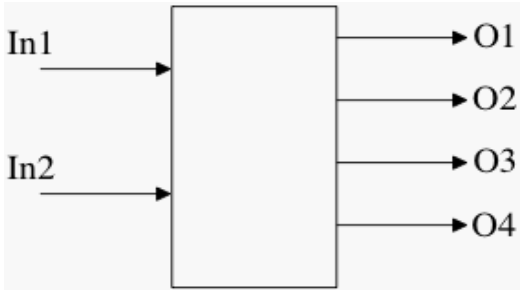
  A Petri net (Σ) is *persistent* if for any two different transitions t_1, t_2 of Σ and any reachable marking μ if t_1 and t_2 are enabled at μ then the occurrence of one cannot disable the other.

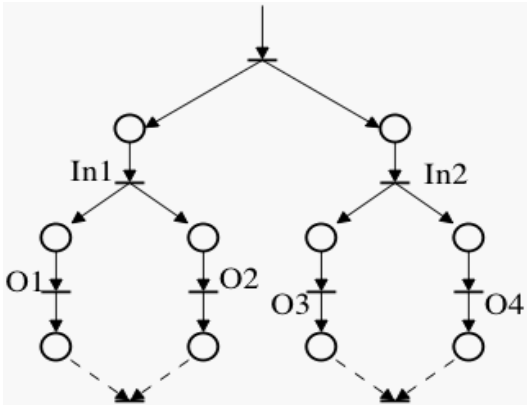


Notion of Desynchronisation

Synchronous System Description Model

- Block and PN representation of a synchronous system.
- A firing sequence of the input and output signals of the synchronous system is modelled by the PN.





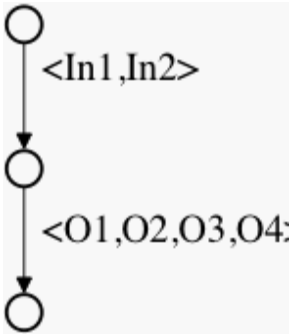
Firing Semantics

- *Interleaved* vs *maximal step semantics*

- *Interleaved semantics* requires to execute in every marking all possible subsets of enabled transitions if they are not in conflict.
- *Maximal step semantics* requires that at each step a maximal set of concurrent firable transitions are allowed to fire.

Firing semantics

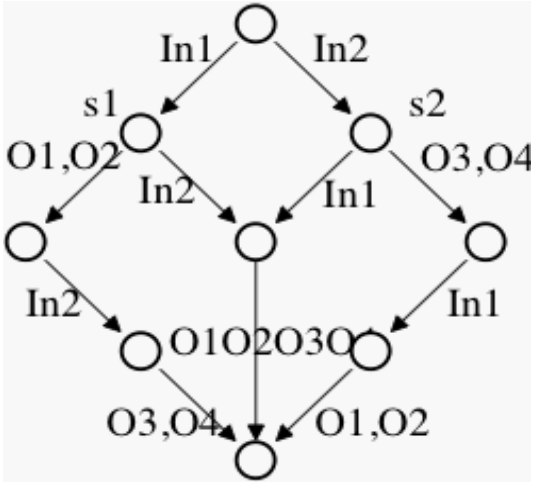
- A state graph of the synchronous system is denoted by maximal input and maximal output steps (synchronous assumption):



- Analogy with Burst mode circuits:
  - Allows multiple signal changes on a single transition
  - Takes into account I/O causality

Moving from maximal Parallelism to maximal Concurrency – Persistency Violation

- The input signals are required to be unbundled to enable inputs to arrive in any order to enable desynchronisation.
- The output signals are fired in bundles or maximal output steps.



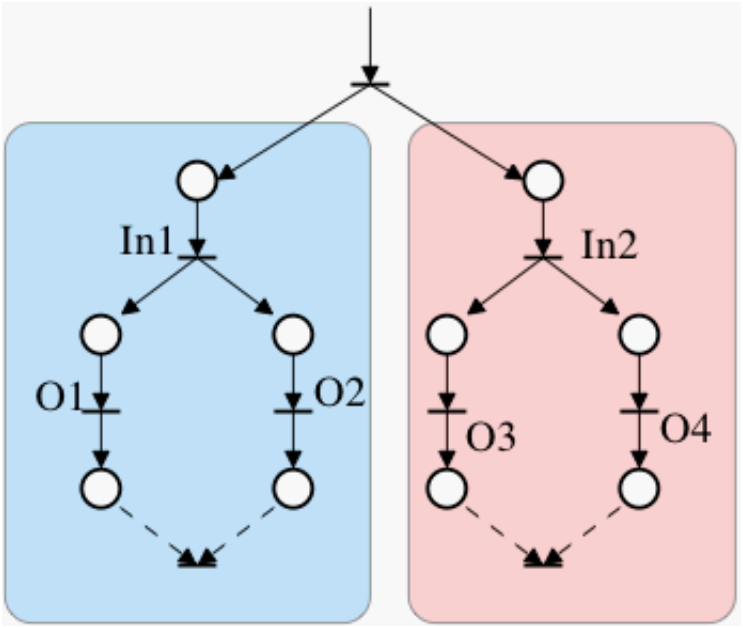
Steps violating *Persistency* property are:

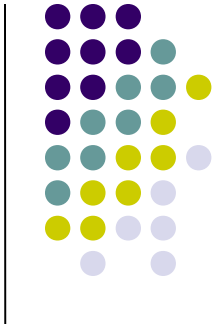
$<\text{In}_1>$ and $<O_3, O_4>$
$<\text{In}_2>$ and $<O_1, O_2>$

- Bundles altered "on the fly" are: $\{O_1, O_2, O_3, O_4\}$ and $\{O_1, O_2\}$

Solution...

- Partition the net into blocks or *localities* to avoid *persistency* violation.
- *Persistency* condition is valid in each block.
- This leads to the realisation of a globally synchronous system into a *GALS* architectures

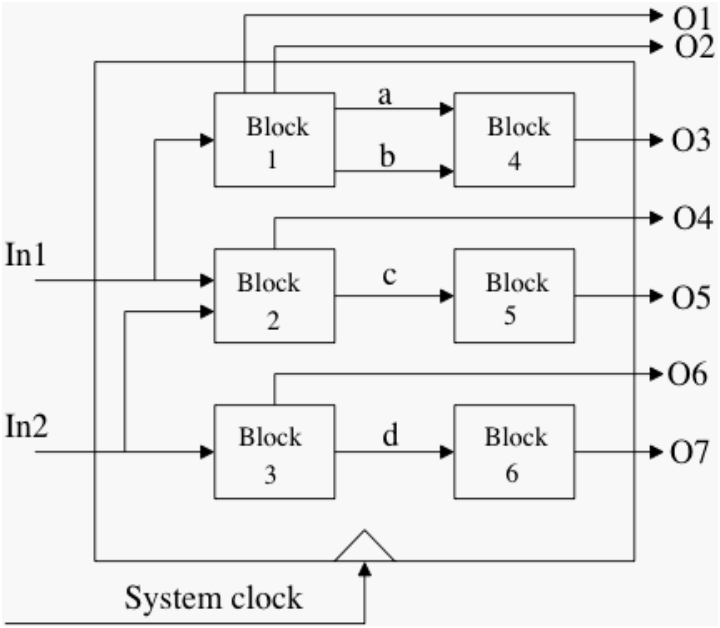




Pre-requisites for Locality Formation

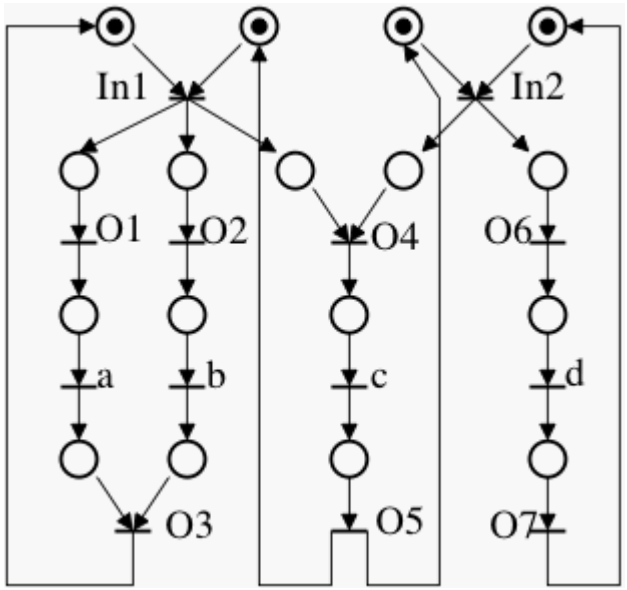
Pre-requisites for locality formation

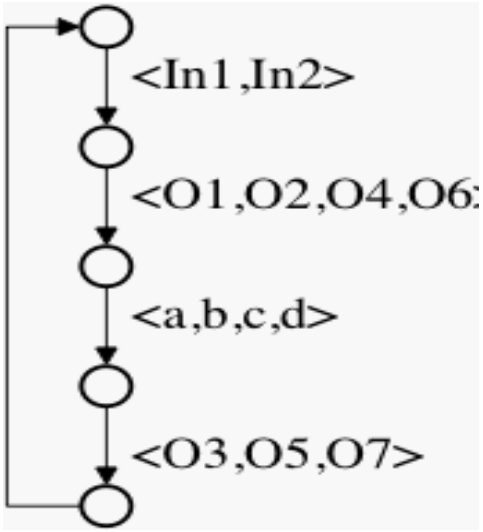
- Each synchronous block can be sub-divided into smaller blocks which would constitute a GALS system.



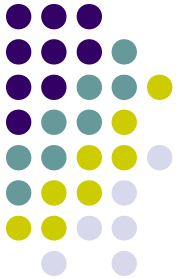
Pre-requisites for locality formation

- The PN model and the state graph depicting the ordering sequence of the input/output signals





Transition Splitting and Signal Insertion



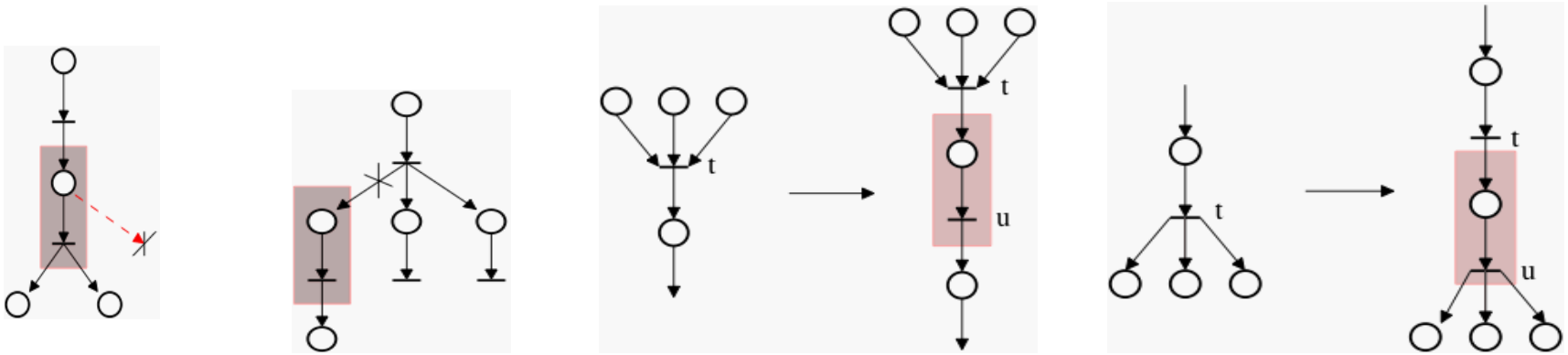
- Transformation of the internal signals between interconnected blocks of a system into intermediate input output signals that would constitute a GALS system to aid the localisation process.

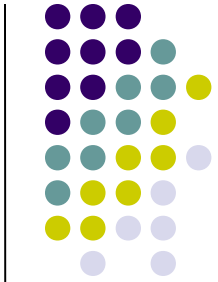
- Transformation is in the form of transition splitting and signal insertion.

- Transition splitting is only applied on transitions $T_{\text{int}} \notin I \cup O$ where *I* and *O* are sets of Input and Output signals of the system under consideration.

Conditions of valid transformation

- The place cannot have the token stole by another transition in conflict.
  Avoid: transition stealing token and running one locality into deadlock
- If the signal has fanouts, signal should be inserted before the fanout.
  Avoid: multiple signals being inserted leading to the formation of unnecessary localities.

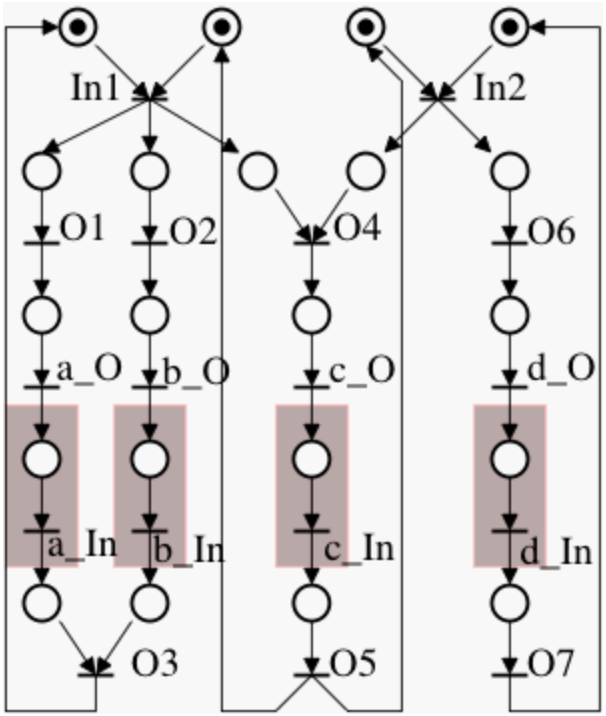


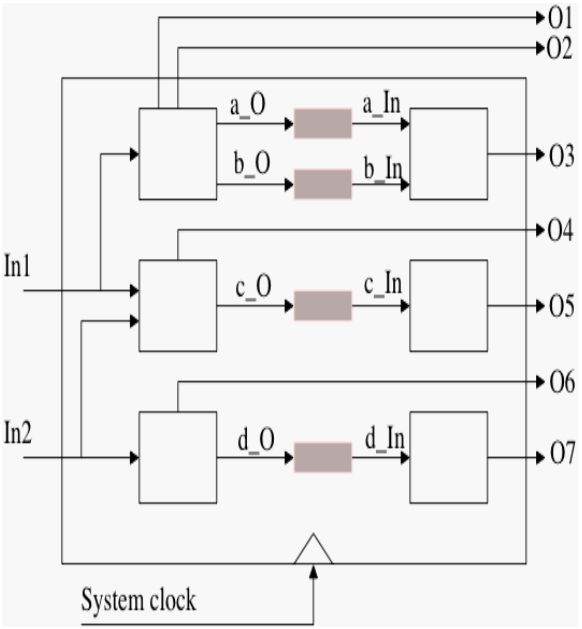


Desynchronisation Methodology

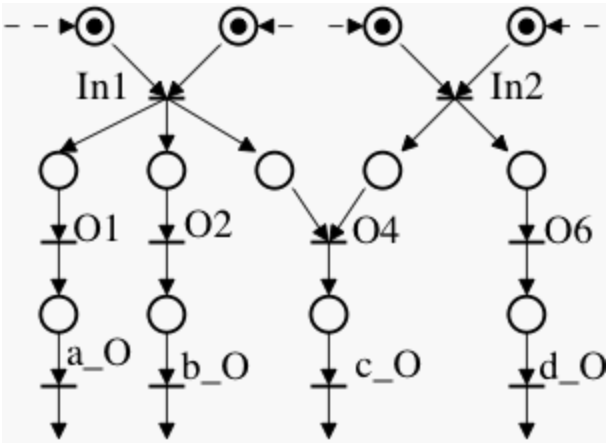
Net Transformation

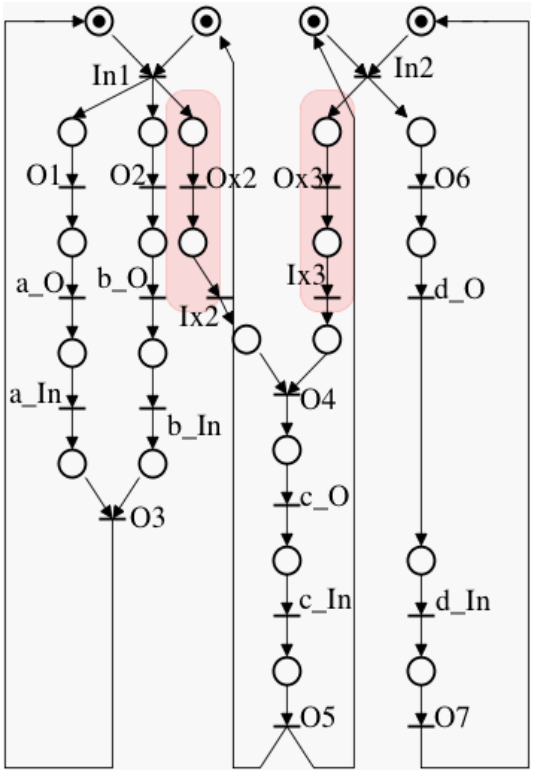
- Signal insertion following the assumptions presented in the previous slides



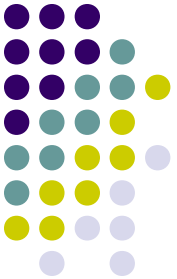


Persistency Check





Theory of Locality formation



- Let $\Sigma = P, T, F, \mu_0$ be an elementary net system Localisation leads to the division of net into *n* smaller nets denoted by

$$\Sigma_i = (P_i, T_i, F \cap (P_i \times T_i \cup T_i \times P_i), P_i \nabla \mu_0)$$

for $i = 1$ to n, where n is a set of integers, each $T_i \subseteq T$ so that $(T_1 \cap T_2 \cap .... T_n) = 0$ and each $P_i \subseteq P$ $(P_1 \cap P_2 \cap ... P_n) \neq \emptyset$

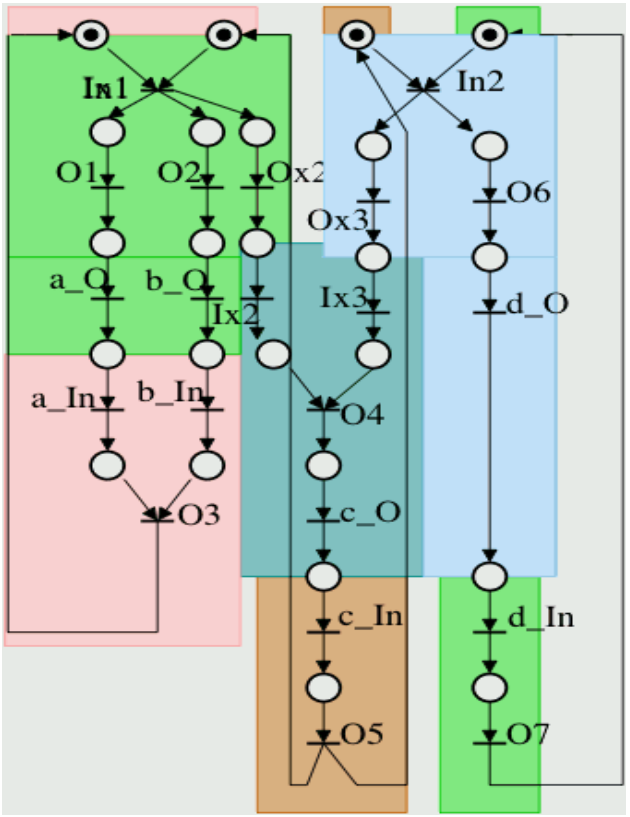
$P_i \nabla \mu_0$ is defined by the following:

if , $\mu_0 : P \rightarrow 0,1$ then,

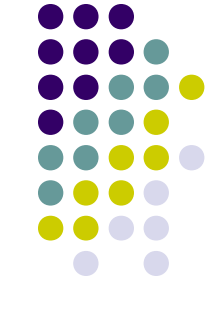
$$\forall\, p \in P_i, \mu_{0i} : P_i \rightarrow 0,1 \vee \mu_{0i}(p) = \mu_0(p)$$

Locality formation

- The partitioning algorithm is applied on the transformed net to obtain required localities.
- The final output of the running example is depicted below:



Notion of Partioning Correctness



- Let $\Sigma = P, T, F, \mu.$ be an elementary net.

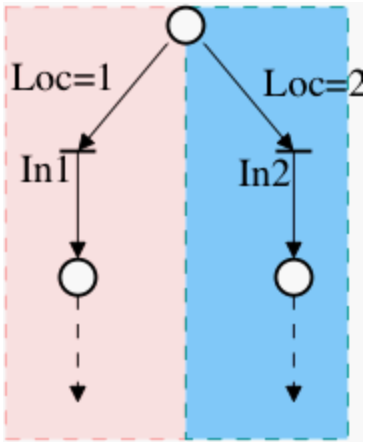
  Partitioning $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \ldots \Sigma_n$ each belonging to localities $L_1, L_2 \ldots L_n$ is correct at a marking μ iff for all steps of transitions $U_1 \subseteq T_1, \ldots U_n \subseteq T_n$, where $U_{1,} \ldots U_n$ are enabled in $\Sigma_1, \ldots \Sigma_n$ respectively, the combined step $U_1 \cup U_2 \cup \ldots U_n$ is enabled in Σ . This is denoted by,

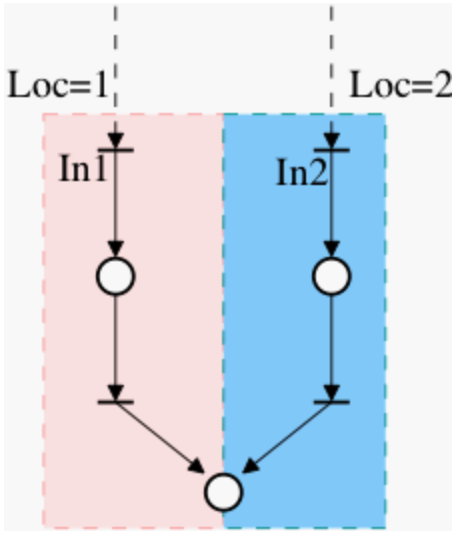
$$(\mu \nabla P_1)[U_1 >_{\Sigma_1} \wedge (\mu \nabla P_2)[U_2 >_{\Sigma_2} \wedge \ldots (\mu \nabla P_n)[U_n > \Sigma_n \rightarrow \mu[U_1 \cup U_2 \cup \ldots U_n >_{\Sigma}$$

Correctness Criterion

- The transitions should not share pre- and post-conditions

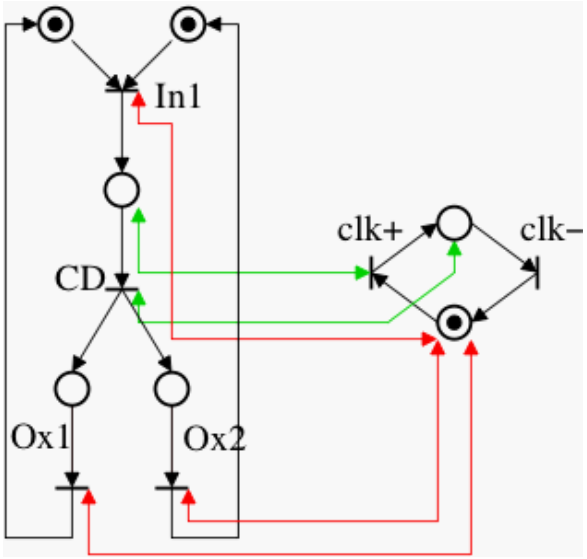
$$\bullet T_1 \cap \bullet T_2 \cap \ldots \bullet T_n = T_1 \bullet \cap T_2 \bullet \cap \ldots T_n = \emptyset$$





GALS implementation

- Each of the localities formed can be either implemented asynchronously or by its own internal clock.
- For the latter, appropriate wrapper can be built that will generate local clock enables and can be synthesised using existing PN based synthesis tools.
- A simple example of such wrappers is shown below:



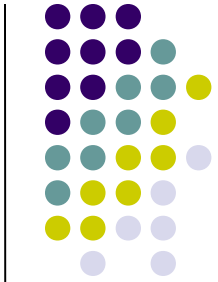
Conclusion and Future Work

- Work addressed the problem of synthesising a GALS system by desynchronisation methodology using PN as a model of abstraction.
- The granularity of the desynchronised systems are small and hence easy to automate and apply to large complex circuits.
- The desynchronisation approached presented a clear route to synthesis, while preserving the I/O behaviour of the synchronous system.

Future Work

- Automate the proposed methodology to reduce design time and designer intervention.
- The locality allocation can be further optimised to meet various criteria, e.g, minimise interconnection between localities, increase component speed.
- Possible to apply other ways of unbundling transitions and obtain different conditions for persistent steps.

Thank you



Questions?