CE653 – Asynchronous Circuit Design

Instructor: C. Sotiriou

http://inf-server.inf.uth.gr/courses/CE653/

Contents

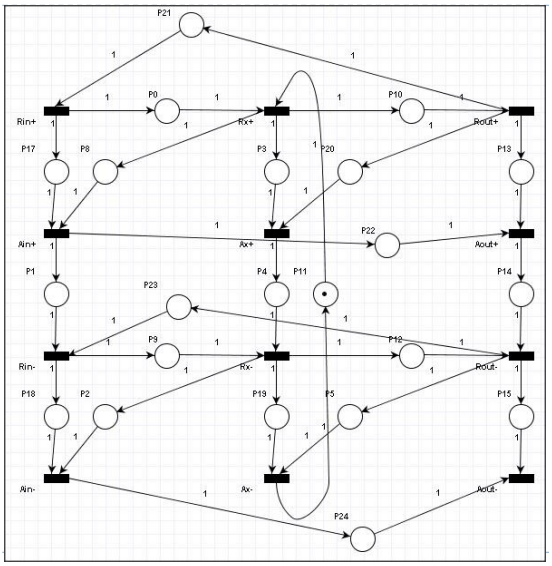
Micropipeline Pitfalls

Micropipeline Rings

- So far focused on Linear Pipelines
- A Latch Controller Ring is a very useful structure
  - Data tokens go around the ring
  - Represents basic iterative computation
  - May include entry/exit points
  - How do I build a ring?
    - **Connect Rout/Aout of RHS controller to Rin/Ain of a LHS controller**
- Does it *always* work with micropipelines?

Micropipeline 2-Stage Ring

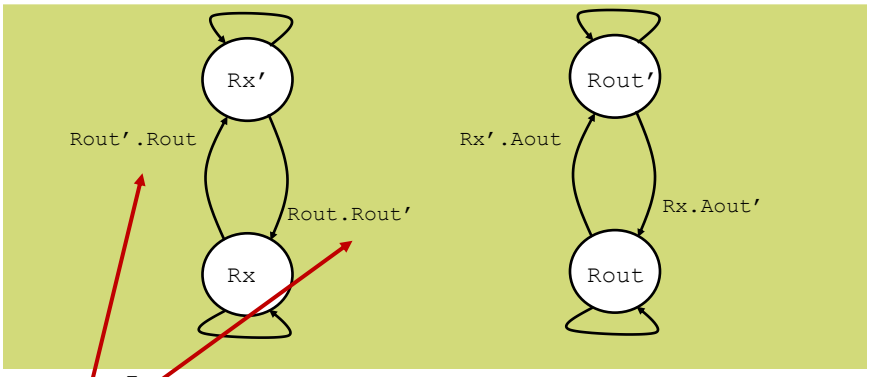
- PTnet Cycles with no tokens ➔ **Deadlock**!!!
- Commoner's Theorem:
  - **Deadlocked systems include an unmarked cycle**
- *Can I find a valid marking to make it **live**?*



Micropipeline Analysis - MSFSMs

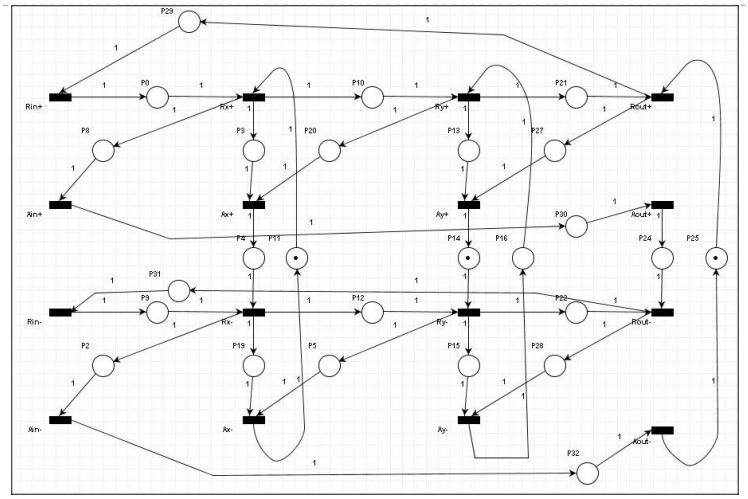
- Two-stage Micropipeline FSMs:

First stage FSM **Second stage FSM**



- *Ring* ➔ `Rin` = `Rout`, `Ain` = `Aout`
- ***Never LIVE!!!***

Micropipeline 3-Stage Ring

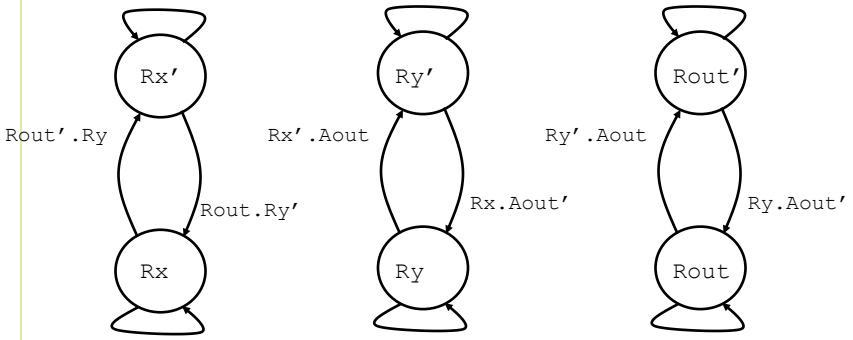


- Deadlocked at initial (original) marking

Micropipeline Analysis - MSFSMs

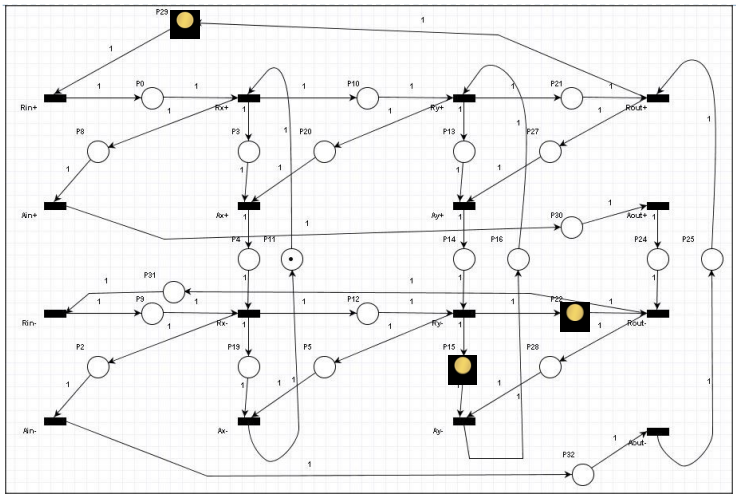
- Three-stage Micropipeline Ring FSMs:

First stage FSM	Second stage FSM	Third Stage FSM



- *Now:* `Rin` = `Rout`, `Ain` = `Aout`
- *Live when?* `Rout'`, `Ry'`, `Rx` *initial states*

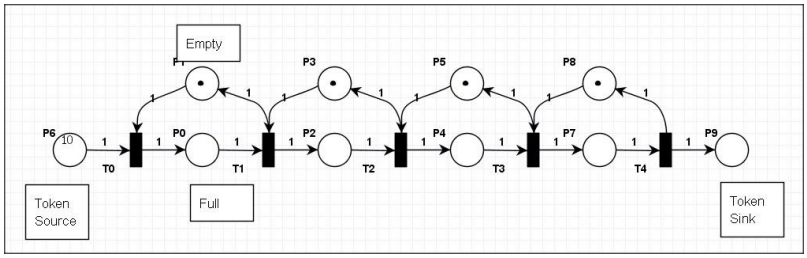
Live Micropipeline 3-Stage Ring with Alternate Marking



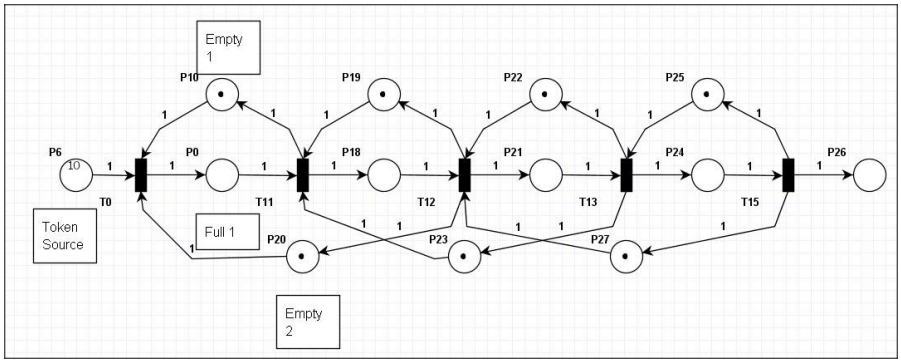
- Live Marking Shown: `Rin+`, `Rx+`, `Ry-`

Token Occupancy – Half vs. Full Buffers

Full-Buffer PTnet Model



Half-Buffer PTnet Model



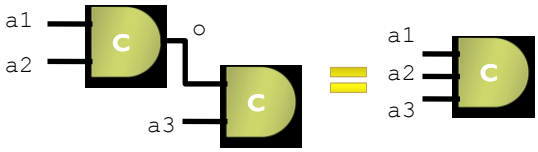
C Element Extensions and Generalisations

Boolean Function with Feedback

- **Define**:
  - `SET`, `KEEP` and `RESET` functions which are feedback free
  - `RESET` = U – `SET` (OFF-set of `SET`), **or**
  - `RESET` ∩ `SET` = ∅
  - `KEEP` = `RESET`'
  - **Sole feedback of `f` is on the `f` line** (feeds back to input)
- Form 1 (Set and Keep):
  - **`f`** = (`SET` function) + **`f`** (`KEEP` function)
- Form 2 (Set and Reset):
  - **`f`** = (`SET` function) + **`f`** (`RESET` function)'
- **Example**
  - Asymmetric C Element:
  - **`f` = `bc` (SET) + `f` (`ab`)' (RESET)'**

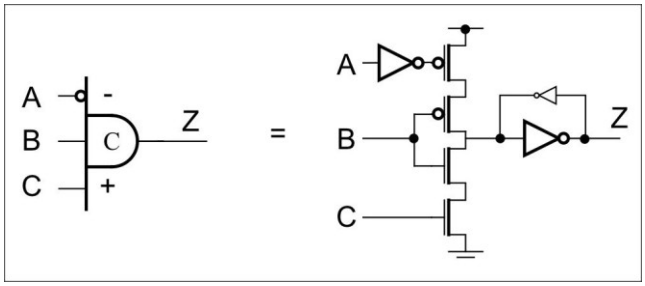
Composing C Elements

- Based on the C-Element Boolean Equation:
  - 2-input C-Element's Logic Function is:
    $c = ab + c(a + b) = ab + bc + ac$
  - n-input C-Element is:
    $c = a_1a_2...a_n + c(a_1 + a_2 + \ldots + a_n)$
- It can be shown that C gates are composable using their ***set***, ***keep*** functions:
  - $c3_{set} = a1a2a3$, $c3_{keep} = a1 + a2 + a3$
  - $c2_{set} = a1a2$, $c2_{keep} = a1 + a2$
    - $o_{set} = a1a2$, $o2_{keep} = a1 + a2$
    - $output_{set} = (o_{set} \cdot a3)$, $output_{keep} = (o_{keep} + a3)$
- Thus:
  - $\mathbf{o_{set} = c3_{set}}$, $\mathbf{output_{keep} = c3_{keep}}$



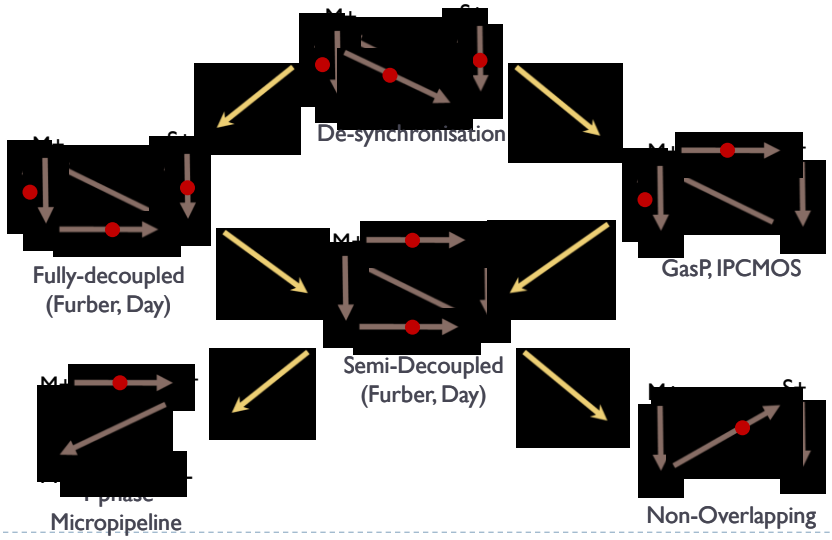
Asymmetric C Elements

- In certain cases, the Set and Reset logic of a C element is not identical
  - Obvious from PTnet specification of a controller
- Asymmetric C elements are an extension of the basic C
- Equivalent to SR Latch or Boolean feedback circuit as well

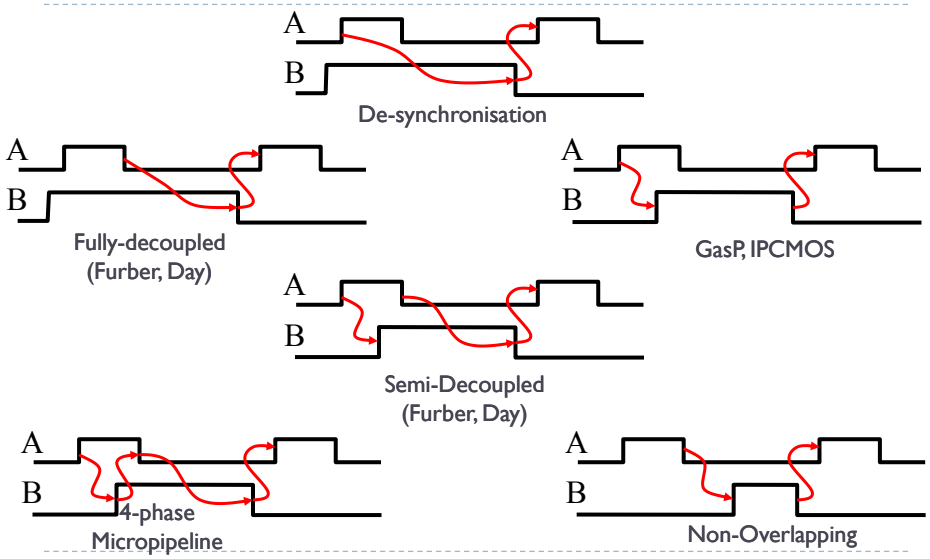


Taxonomy of Latch Controllers

Taxonomy of Latch Controllers - Signals

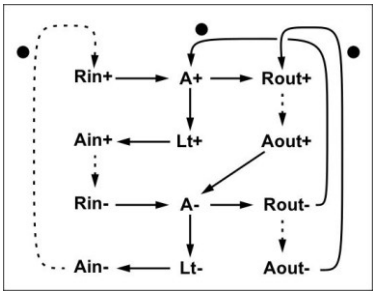


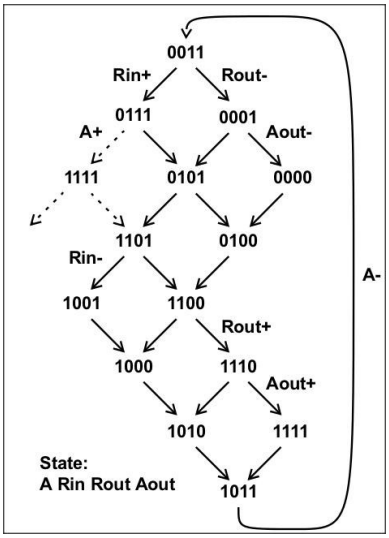
Taxonomy of Latch Controllers - Timing



Semi-Decoupled Latch Controller

Semi-Decoupled Latch Controller – STG and SG

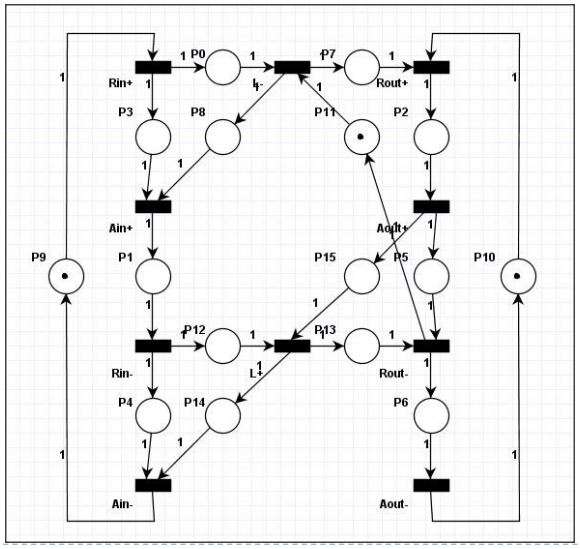




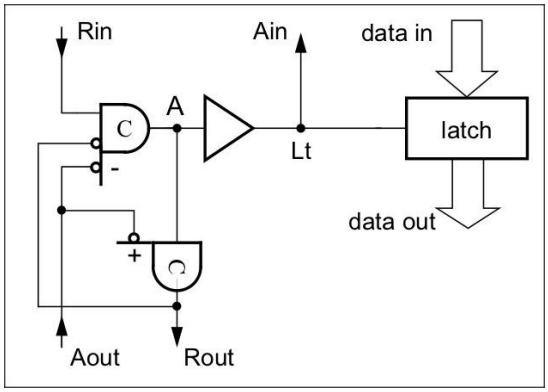
- A is redundant – ignore
  - pre-buffered Lt
- Lt is for active-low latch,

Semi-Decoupled Latch Controller PTnet

- Latch Control
  - L
  - L-, L+ in PTnet

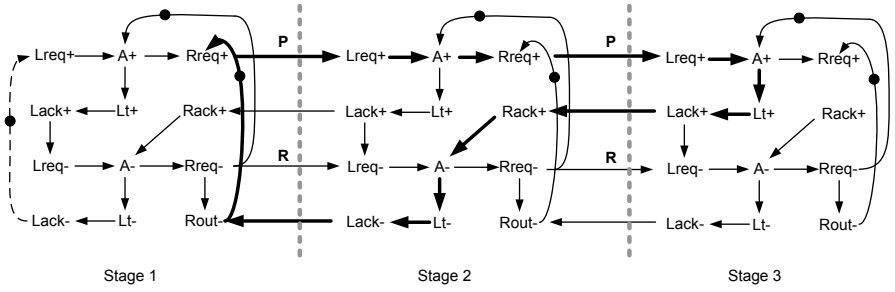


Semi-Decoupled Latch Controller Circuit



- C-gate Set, Reset functions may be inferred from STG/PTnet specification of the controller

Semi-Decoupled – PTnet for 3 Stages



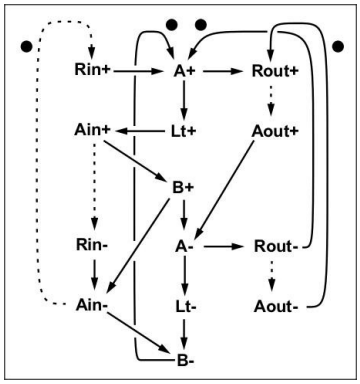
Notice how critical cycle contains two "P"s

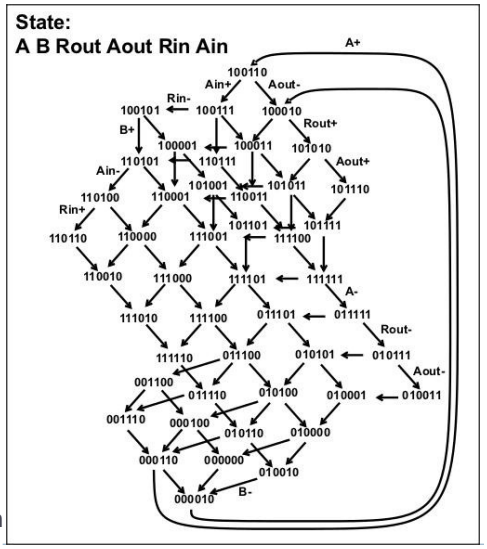
(i.e., processing steps involving delay lines)

- **Is this an issue? Why?**

Fully-Decoupled Latch Controller

Fully-Decoupled Latch Controller – STG and SG

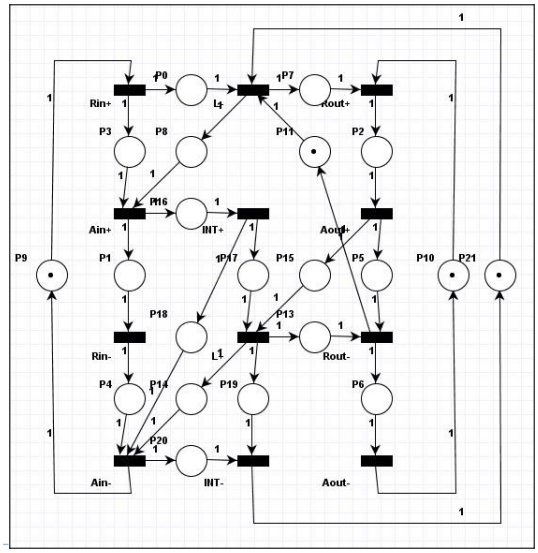




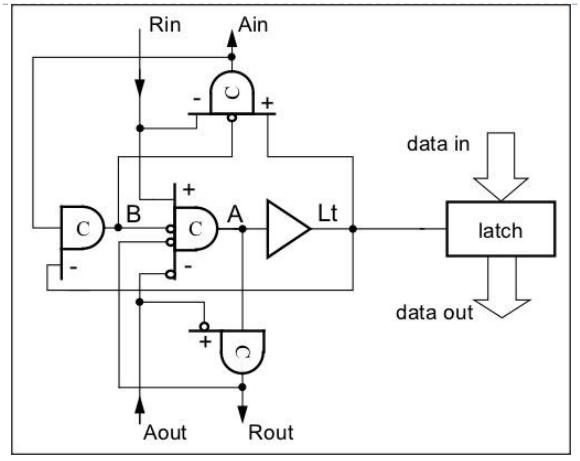
- `A` again is redundant
- `Lt` is for active-low latch,
- `B` is internal signal
  - Needed for implementation

Fully-Decoupled Latch Controller PTnet

- Latch Control
  - L
  - L-, L+ in PTnet
- Signal INT,
  - Transitions INT+/INT-
  - Internal
    - For implementation purposes only



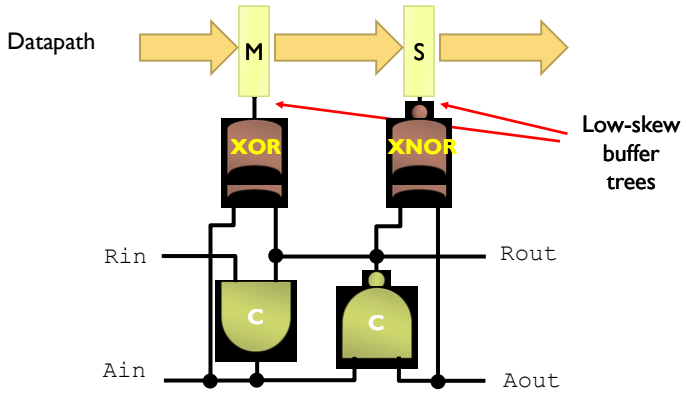
Fully-Decoupled Latch Controller Circuit



- Note Tradeoff between Concurrency and Circuit Complexity
- ***PTnet concurrency is not confluent with circuit***

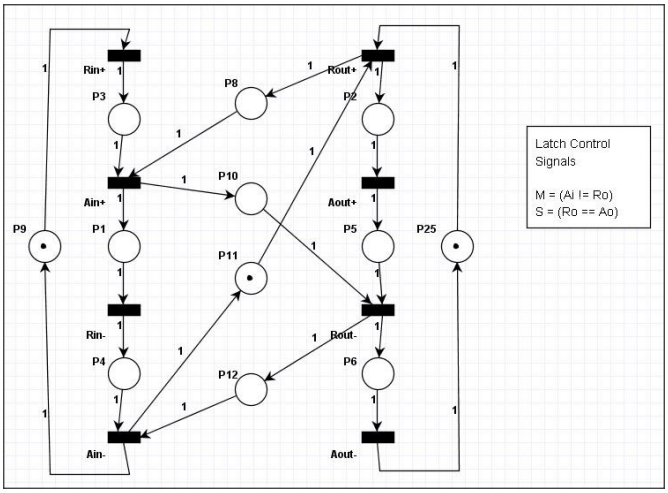
2 C-Element De-synchronisation Controller

2 C-Element De-Synchronisation Controller



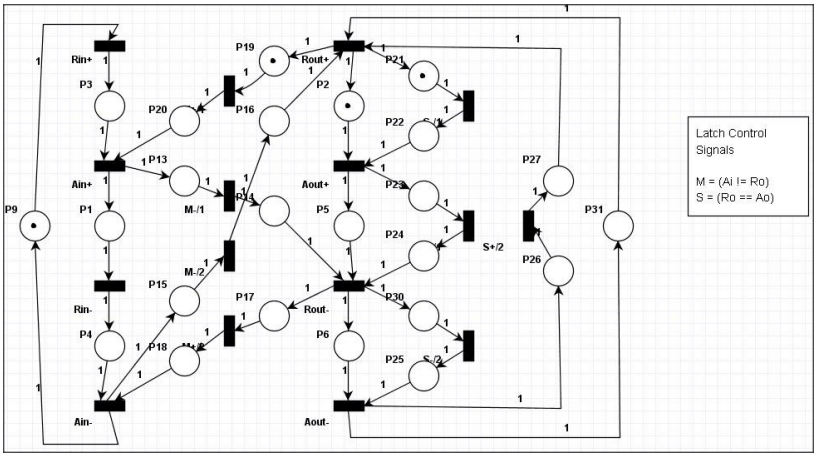
- Master Latch Enable: `M = (Ain != Rout)` [XOR gate]
- Slave Latch Enable: `S = (Rout == Aout)` [XNOR gate]

2 C-Element De-Synchronisation Controller – Handshake Signals PTnet



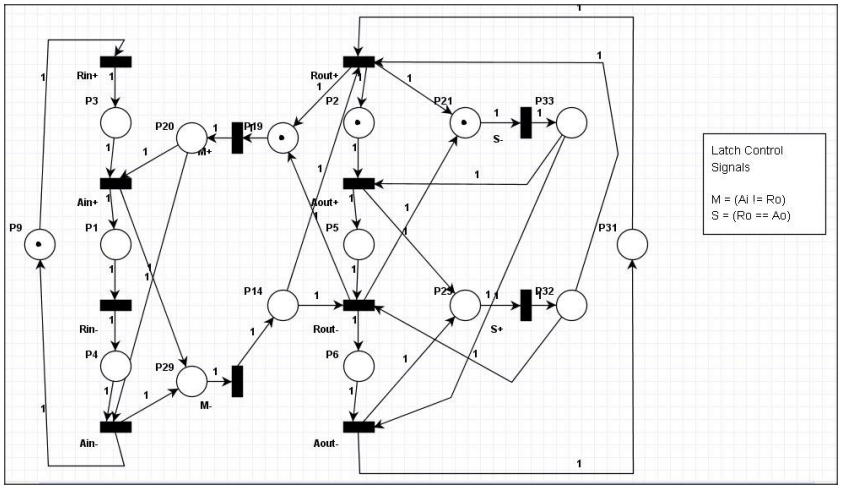
- C elements obvious from PTnet Signal dependencies

2 C-Element De-Synchronisation Controller



- Need multiple instantiations of M, S signals
  - Per control signal transition

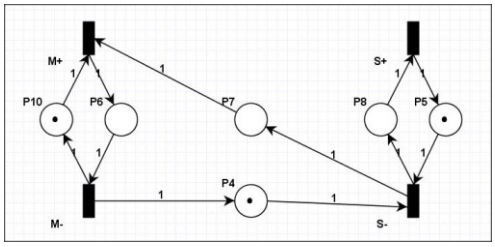
2 C-Element De-Synchronisation Controller – How to Merge M, S Signals using Choice



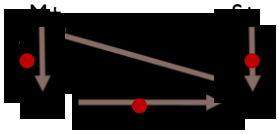
- Note: PTnet is now AC

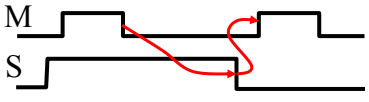
2 C-Element De-Synchronisation Controller Analysis - PTnets

- Reduction to Latch Control Signals (Verify)



- Characteristic Pattern:





De-Synchronisation Maximum Concurrency Controller

De-Synchronisation Maximum Concurrency Controller