kantlipsumhow-many164

Departamento de Engenharia Eletrotécnica e de Computadores

Mestrado em Engenharia Eletrotécnica e de Computadores

Laboratório 1

Co-design e Sistemas Reconfiguráveis

Alunos:
João Pedro Antunes - **70380**
Júlio Lopes- **70512**
Marco Romao - **71348**

Docentes:
Aniko Costa
Filipe Moutinho

1º Semestre 2025/2026

Conteúdo

1 Introdução

1.1 Contextualização

Este trabalho insere-se no âmbito da unidade curricular de Co-design e Sistemas Reconfiguráveis, focando-se no desenvolvimento e implementação de controladores para sistemas de automação industrial. O caso de estudo considerado é um sistema de produção composto por 3 células em cascata, onde cada célula integra um braço de robô e um tapete rolante.

A complexidade destes sistemas justifica a utilização de metodologias formais de modelação e verificação, permitindo validar o comportamento do controlador antes da sua implementação física. Adicionalmente, a possibilidade de implementar o controlador de forma centralizada ou distribuída oferece diferentes trade-offs entre complexidade de implementação, desempenho e escalabilidade do sistema.

1.2 Objetivos

O presente trabalho tem como objetivo principal o desenvolvimento completo de um controlador para um sistema de automação com três células em cascata (N=3), desde a fase de modelação até à implementação física em hardware reconfigurável.

Os objetivos específicos incluem:

- Modelar o comportamento do sistema utilizando redes de Petri Input-Output Place-Transition (IOPT), explorando as capacidades de composição modular através de operações de adição e fusão de redes;
- Validar e analisar o modelo através de simulação e análise do espaço de estados, verificando propriedades comportamentais críticas do sistema;
- Implementar o controlador em plataformas de hardware distintas (FPGA e Arduino), avaliando a abordagem centralizada de controlo;
- Desenvolver uma arquitetura de controlo distribuído através da decomposição do modelo global, explorando paradigmas de execução síncrona e GALS (Globally Asynchronous Locally Synchronous);
- Analisar o impacto de comunicações não-instantâneas entre controladores distribuídos, introduzindo atrasos pseudo-aleatórios;
- Estender o modelo para suportar buffers de capacidade finita superior a uma unidade, aumentando a flexibilidade do sistema;
- Comparar as diferentes abordagens de implementação, avaliando vantagens, desvantagens e adequação a diferentes contextos aplicacionais.

2 Análise Teórica

2.1 Redes de Petri

As redes de Petri constituem uma ferramenta formal de modelação adequada para sistemas concorrentes e distribuídos. Uma rede de Petri é definida por uma quádrupla $PN = (P, T, F, M_0)$, onde P representa lugares, T transições, F arcos direcionados, e M_0 a marcação inicial. A dinâmica baseia-se no disparo de transições, que removem tokens dos lugares de entrada e adicionam aos de saída.

Redes IOPT (Input-Output Place-Transition) estendem as redes clássicas com capacidade de interagir com o ambiente através de:

- Eventos de entrada associados a sinais externos (sensores);
- Sinais de saída para controlo de atuadores;
- Guardas e prioridades para resolução de conflitos;
- Semântica temporal para modelar delays.

Redes de Petri Coloridas permitem representar de forma compacta sistemas com estrutura repetitiva. Os tokens possuem "cores"(tipos de dados), permitindo que um único modelo represente múltiplas instâncias de subsistemas idênticos. No sistema estudado, tokens $\langle 1 \rangle$, $\langle 2 \rangle$ e $\langle 3 \rangle$ identificam cada célula.

Composição Modular: A construção de sistemas complexos utiliza operações de:

- *Merge Nets*: união de múltiplas redes mantendo nós distintos;
- *Fusion Sets*: identificação de nós a fundir criando sincronização;
- *Net Addition*: combinação de merge e fusão num modelo integrado.

2.2 Execução síncrona vs. GALS

Em contexto síncrono, todos os componentes operam com um relógio global comum. Oferece simplicidade de design e determinismo, mas apresenta limitações de escalabilidade, consumo energético elevado (relógio sempre ativo) e dificuldades na distribuição do sinal de relógio (clock skew) em sistemas grandes.

Já num paradigma GALS, este divide o sistema em domínios síncronos locais que comunicam assincronamente. Cada domínio opera com relógio independente, comunicando através de protocolos de handshaking ou FIFOs assíncronas.

Vantagens do GALS incluem escalabilidade, modularidade, eficiência energética e tolerância a falhas. As desvantagens são a maior complexidade de interface, latência variável na comunicação e necessidade de tratamento de metastabilidade nas fronteiras entre domínios.

2.3 Implementação em FPGA e Arduino

Utilizar-se-á uma **FPGA (Field-Programmable Gate Array)**. A descrição do sistema é feita em VHDL, uma linguagem de descrição de hardware que permite especificar comportamento concorrente. Recorrer-se-á também a um microcontrolador **Arduino**.

A framework IOPT-Tools permite geração automática de código VHDL para FPGA e código C para Arduino, facilitando a comparação entre as duas abordagens de implementação.

Critério	**FPGA**	**Arduino**
Paralelismo	Hardware real	Software (sequencial)
Tempo de resposta	ns	ms

Tabela 1: Comparação entre plataformas FPGA e Arduino

hyperref

3 Redes de Petri

3.1 IOPT-Tools

3.1.1 Rede de Petri — Tapete Singular

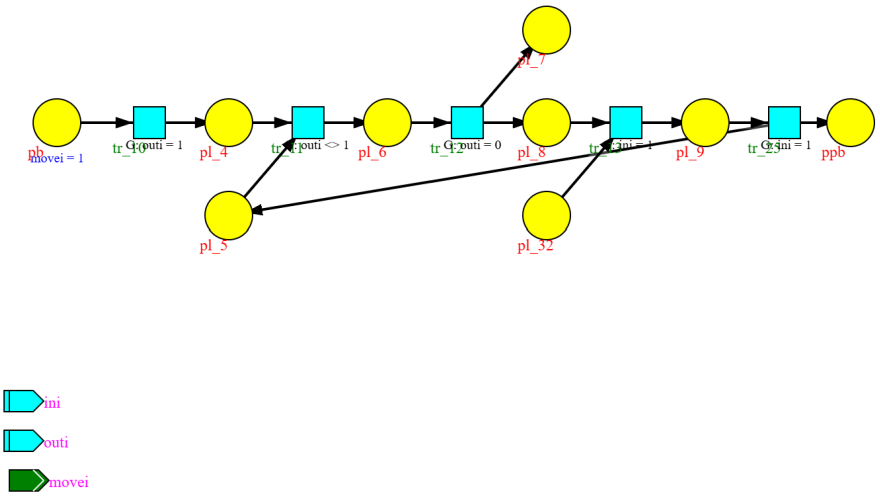

Figura 1: Rede de Petri — Tapete Singular

3.1.2 Rede de Petri — Modelo Completo

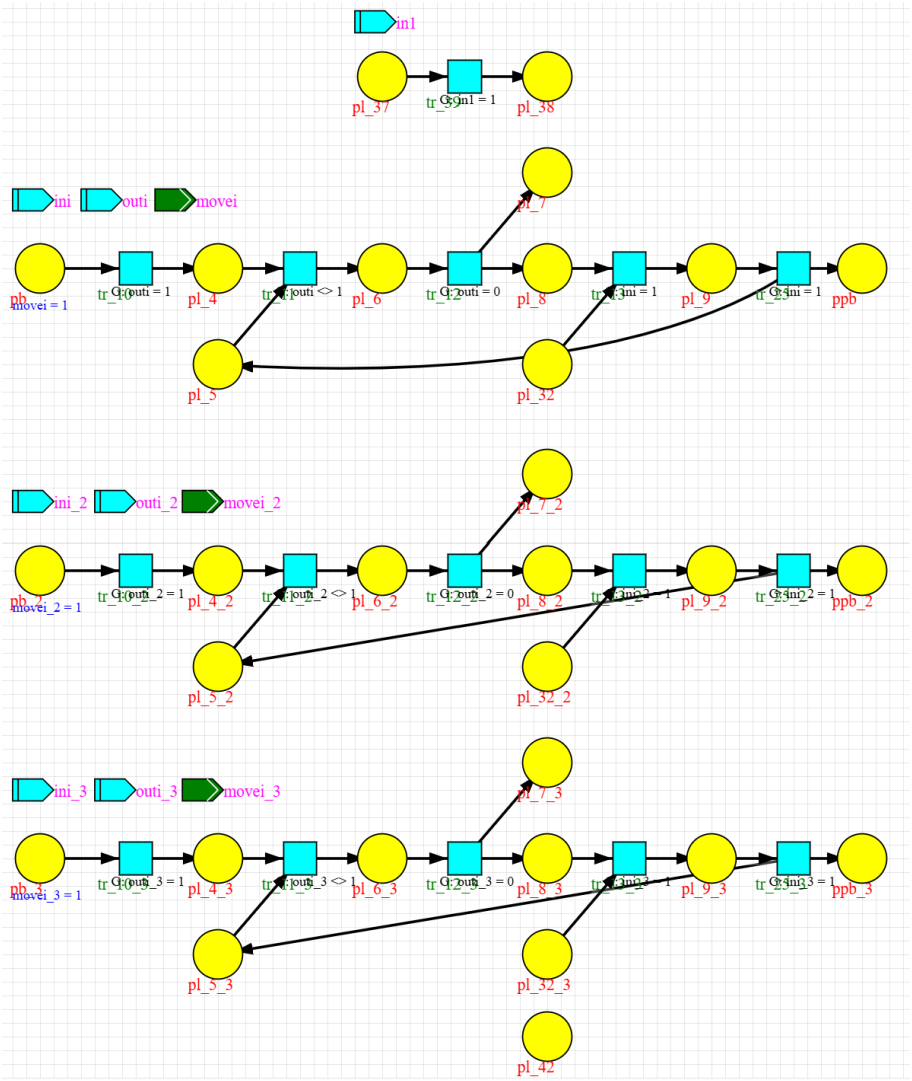

Figura 2: Rede de Petri — Modelo Completo

3.1.3 Rede de Petri — Fusion set

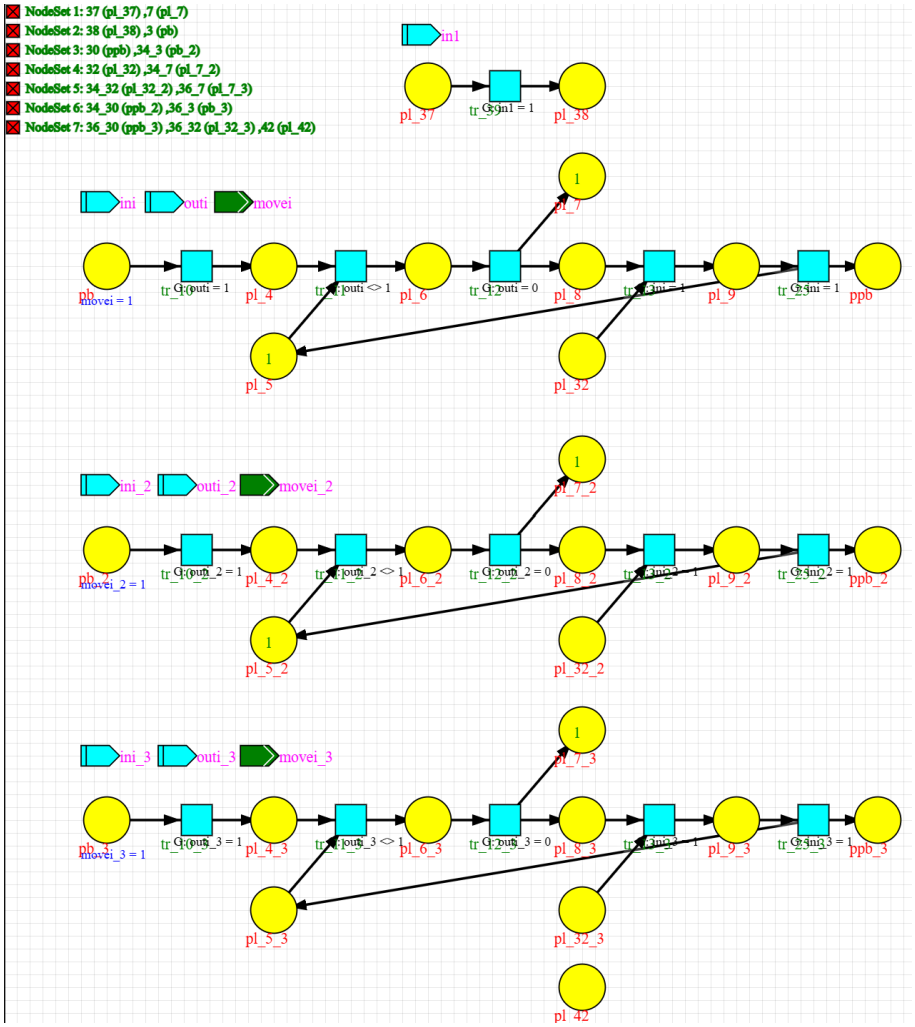

Figura 3: Rede de Petri — Fusion set

3.1.4 Rede de Petri — Node set

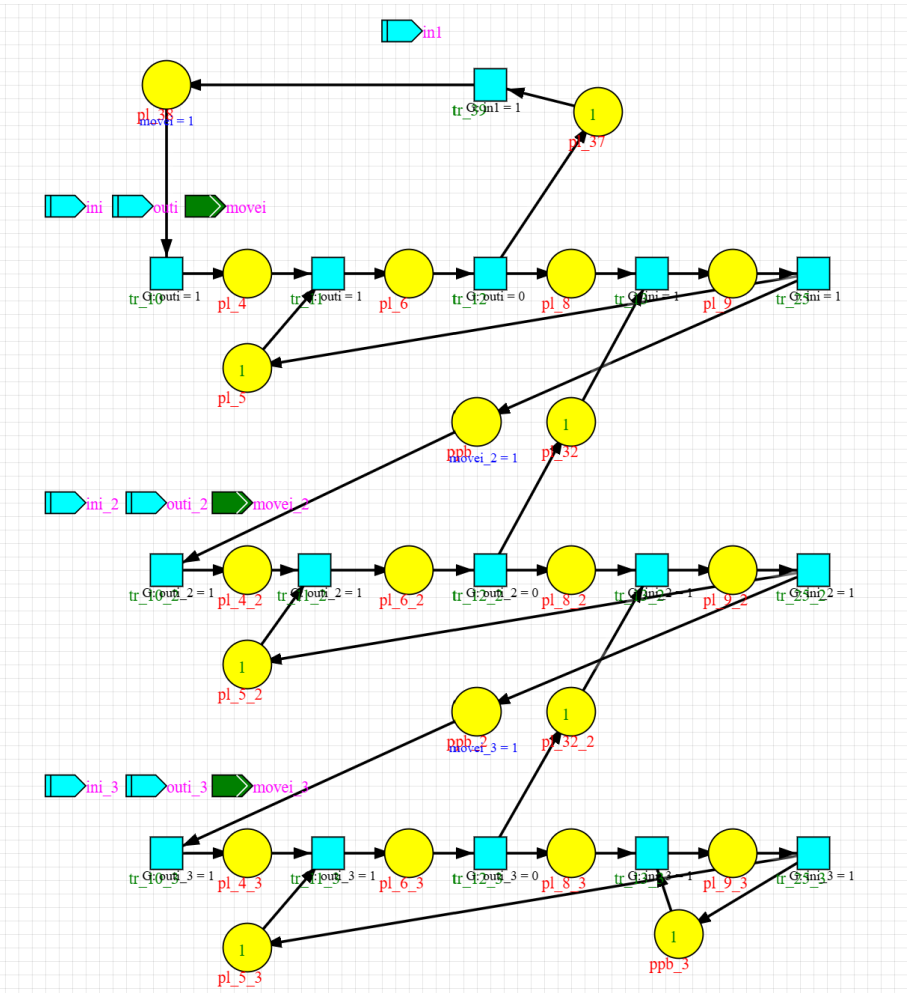

Figura 4: Rede de Petri - Node set

3.2 Simulação e Análise

3.2.1 Simulação com token-player

Neste link tem acesso ao vídeo da simulação com token-player:

https://unlpt-my.sharepoint.com/:v:/g/personal/jafe_lopes_fct_unl_pt/EUvkbe_KlbRFrrdI0gsSh98Bu2URieTdJFFzwphaj5HS6A?e=i5XMNg&nav=eyJyZWZlcnJhbEluZm8iOns3D

3.2.2 Simulação temporal

3.2.3 Geração e análise do espaço de estados

```
586884 Started state-space generation.

Cycle 1: 1 states + 0 links
Cycle 2: 2 states + 0 links
Cycle 3: 3 states + 0 links
Cycle 4: 4 states + 0 links
Cycle 5: 5 states + 0 links
Cycle 6: 8 states + 0 links
Cycle 7: 13 states + 4 links
Cycle 8: 18 states + 10 links
Cycle 9: 25 states + 18 links
Cycle 10: 34 states + 30 links
Cycle 11: 51 states + 48 links
Cycle 12: 70 states + 98 links
Cycle 13: 103 states + 174 links
Cycle 14: 146 states + 306 links
Cycle 15: 211 states + 510 links
Cycle 16: 304 states + 808 links
Cycle 17: 351 states + 1396 links
Cycle 18: 432 states + 1852 links
Cycle 19: 474 states + 2393 links
Cycle 20: 524 states + 2769 links
Cycle 21: 566 states + 3145 links
Cycle 22: 578 states + 3373 links
Cycle 23: 598 states + 3485 links
Cycle 24: 606 states + 3601 links
Cycle 25: 614 states + 3657 links
Cycle 26: 622 states + 3713 links

MIN Bounds: pl_32=0 pl_32_2=0 pl_37=0 pl_38=0 pl_4=0 pl_4_2=0 pl_4_3=0 pl_5=0 pl_5_2=0 pl_5_3=0 pl_6=0 pl_6_2=0 pl_6_3=0 pl_8=0 pl_8_2=0 pl_8_3=0 pl_9=0 pl_9_2=0 pl_9_3=0 ppb=0 ppb_2=0 ppb_3=0

MAX Bounds: pl_32=1 pl_32_2=1 pl_37=1 pl_38=1 pl_4=1 pl_4_2=1 pl_4_3=1 pl_5=1 pl_5_2=1 pl_5_3=1 pl_6=1 pl_6_2=1 pl_6_3=1 pl_8=1 pl_8_2=1 pl_8_3=1 pl_9=1 pl_9_2=1 pl_9_3=1 ppb=1 ppb_2=1 ppb_3=1

##################
Total States:   622
Total Links:    3741
##################

Executing queries...
Done: found 0 query matching states.

Generation time (sec): 0.00 (when 0.00 it is smaller than 0.01sec)

Generating output file.
Done.

##################
Total States:   622
Total Links:    3741
Deadlock count: 0
Conflict count: 0
Invalid count:  0
##################
```

Figura 5: Rede de Petri — Espaço de Estados

4 Implementação em FPGA

4.1 Geração do código VHDL

4.2 Deployment na FPGA

4.3 Configuração de entradas/saídas

4.4 Testes experimentais

4.5 Comparação com resultados de simulação

5 Implementação em Arduino

5.1 Geração do código C

5.2 Deployment no Arduino

5.3 Configuração do ambiente

5.4 Testes

5.5 Comparação com simulação e implementação FPGA

6 Controlador Distribuído em regime Síncrono

6.1 Identificação do conjunto de corte (cutting set)

6.2 Decomposição usando SPLIT

6.3 Modelo com canais síncronos

6.4 Simulação e validação

6.5 Modelo com canais assíncronos

6.6 Simulação e Análise

6.7 Decomposição GALS - três controladores separados

7 Implementação em FPGA - Distribuição Síncrona

7.1 Geração do código VHDL

7.2 Interconexão dos componentes

7.3 Deployment na FPGA

7.4 Testes e análise de resultados

8 Buffers de Capacidade Finita

8.1 Modificação do modelo para múltiplas peças

8.2 Interconexão dos componentes

8.3 Módulo de visualização do número de objetos

8.4 Simulação e validação

8.5 Implementação centralizada em FPGA

8.6 Análise de resultados

9 Distribuído com Buffers (Síncrono)

9.1 Aplicação síncrona ao buffer de capacidade finita

9.2 Decomposição e implementação distribuída

9.3 Execução síncrona global

9.4 Testes e análise

10 Distribuído com Buffers (Assíncrono)

10.1 Aplicação assíncrona ao buffer de capacidade finita

10.2 Comunicações não-instantâneas com buffers

10.3 Deployment e testes

10.4 Análise comparativa

11 Comparação de abordagens

11.1 Centralizado vs Distribuído

11.2 Síncrono vs Assíncrono

11.3 Impacto dos atrasos de comunicação

11.4 Vantagens e desvantagens de cada abordagem

11.5 Análise de escalabilidade

12 Conclusões

12.1 Resultados alcançados

12.2 Dificuldades encontradas e soluções adotadas