DISSERTAÇÃO DE MESTRADO Nº 1038

ESTUDO DE DESEMPENHO DO CLUSTER TOOL - ABORDAGEM BASEADA NA TEORIA DE CONTROLE SUPERVISÓRIO

Márcio Júnior Nunes

DATA DA DEFESA: 20/02/2018

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

ESTUDO DE DESEMPENHO DO CLUSTER TOOL - ABORDAGEM BASEADA NA TEORIA DE CONTROLE SUPERVISÓRIO

Márcio Júnior Nunes

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientadora: Profa. Patrícia Nascimento Pena

Belo Horizonte - MG

Fevereiro de 2018

Nunes, Márcio Júnior.
N972e Estudo de desempenho do cluster tool – abordagem baseada na teoria de controle supervisório [manuscrito] / Márcio Júnior Nunes. - 2018.
88 f., enc.: il.

Orientadora: Patrícia Nascimento Pena.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 70-88.

Bibliografia: f. 68-69.

1. Engenharia elétrica - Teses. 2. Sistemas de tempo discreto - Teses. 3. Teoria do controle - Teses. I. Pena, Patrícia Nascimento. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Agradecimentos

Os agradecimentos principais são direcionados a meus pais, Márcio e Norma, meu irmão, Guilherme, e minha noiva, Bruna, por sempre estarem presentes e pelo apoio em todos os momentos. Agradeço à Deus pelas bençãos e por ter me proporcionado mais essa conquista. Agradeço também à Lucas Alves, por todo o auxílio prestado, e à minha orientadora Patrícia Pena, por ter me acolhido e pela valiosa orientação.

Agradecimentos especiais são direcionados ao Laboratório de Análise e Controle de Sistemas Discretos [1] da Universidade Federal de Minas Gerais (LACSED), em particular aos colegas Michel Rodrigo e Gustavo Caetano, pela caminhada conjunta.

[1] <http://www.lacsed.eng.ufmg.br/>

"Todas as vitórias ocultam uma abdicação."
(Simone de Beauvoir)

Resumo

Este trabalho propõe um estudo sobre os parâmetros que influenciam o desempenho do *cluster tool*. Consideram-se quatro layouts diferentes, sendo dois do tipo radial, com 1 e 2 robôs manipuladores, e dois do tipo linear, com entrada e saída única, e entrada e saída separadas. Para comparar esses layouts, utilizamos como métricas de desempenho o *makespan* e o Paralelismo Acumulado Relativo, sendo este último um indicador proposto neste trabalho. Os valores destas métricas são obtidos aplicando o algoritmo de Máximo Paralelismo com Restrições Temporais ao supervisor responsável pelo controle da planta, que é gerado por meio da Teoria de Controle Supervisório. Este algoritmo busca a sequência com maior paralelismo, considerando o tempo de execução das tarefas, e retorna o *makespan* e o paralelismo acumulado desta sequência. A partir do paralelismo acumulado, utilizamos o número de eventos para produção do wafer e o número de equipamentos do layout considerado para calcular o Paralelismo Acumulado Relativo. Para cada layout, verificou-se o efeito do número de wafers, dos tempos de processamento e da quantidade de câmaras de processo sobre essas duas métricas. Variou-se o número de wafers de 6 a 50, o tempo de processamento de 1 a 200 segundos e o número de câmaras de 4 a 6. A análise é feita comparando o *makespan* e o Paralelismo Acumulado Relativo em cada layout nas diversas condições, sendo o *makespan* uma medida de velocidade de produção do *cluster tool*, e o Paralelismo Acumulado Relativo uma medida de eficiência.

Palavras-chaves: Sistemas a Eventos Discretos, Cluster Tool, Sistemas de Manufatura, Máximo Paralelismo, Teoria de Controle Supervisório

Abstract

This work presents a study of the parameters that influence the cluster tool performance. Four different layouts are considered, two of the radial type, with 1 and 2 handler robots, and two of the linear type, with single input-output and separated input-output. To compare these layouts, the makespan and the Relative Accumulated Parallelism were used as performance metrics, the latter being an index proposed in this work. The values of theses metrics are obtained applying the Maximum Parallelism with Time Constraints algorithm to the supervisor responsible for the control of the plant, generated with the Supervisory Control Theory. This algorithm searches for the sequence with larger parallelism, considering the tasks execution times, and the result is the makespan and accumulated parallelism of this sequence. From the accumulated parallelism, the number of events to produce the wafer and the number of equipments in the considered layout was used to calculate the Relative Accumulated Parallelism. For each layout, the effect on these two metrics of the number of wafers, the processing time and the number of equipments in the layout was verified. The number of wafers was varied from 6 to 50, the processing times from 1 to 200 seconds and the number of chambers from 4 to 6. The analysis is performed comparing the makespan and the Relative Accumulated Parallelism in each layout, the makespan being a measure of quickness of the cluster tool, and the Relative Accumulated Parallelism being a measure of efficiency.

Key-words: Discrete Event Systems, Cluster Tool, Manufacturing Systems, Maximum Parallelism, Supervisory Control Theory

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SED	Sistema a Eventos Discretos
CI	Circuito Integrado
TCS	Teoria de Controle Supervisório
PAR	Paralelismo Acumulado Relativo
PM	Process Modules
MPL	Máximo Paralelismo Lógico
MPT	Máximo Paralelismo com Restrições Temporais
FOUP	Front Opening Unified Pod
ASL	Application Specific Link
CCT	Controlador do Cluster Tool
IC	Intervalo de Confiança

Lista de símbolos

G	Autômato
S	Supervisor
Σ	Conjunto de Eventos
Σ_u	Conjunto de Eventos Não Controláveis
Σ_c	Conjunto de Eventos Controláveis
Σ^*	Fechamento de Kleene
$\hat{\delta}$	Função de Transição
q_0	Estado Inicial
Q_m	Conjunto de Estados Marcados
$\mathcal{L}(G)$	Linguagem Gerada do Autômato G
$\mathcal{L}_m(G)$	Linguagem Marcada do Autômato G
$\cup$	Operação de União
$\cap$	Operação de Interseção
$\subseteq$	Subconjunto
$\|\|$	Operação de Composição Paralela
$\in$	Pertence
$\notin$	Não Pertence
S/G	S Controlando G
$Sup\,\mathcal{C}(K,G)$	Máxima Sublinguagem Controlável de K em Relação a G
F_{TA}	Função Acumulativa de Tarefas Ativas
f_T	Função Temporal
K	Índice de Temperatura do Wafer
T_R	Tempo de Rotação do Robô
T_P	Tempo para Pegar/Depositar o Wafer no PM
T_L	Tempo para Pegar/Depositar o Wafer no *Load Lock*

Sumário

1 Introdução

Neste capítulo os objetivos principais e a motivação do trabalho são apresentados, fazendo sua contextualização. Além disso, detalha-se a forma como esta dissertação foi organizada.

1.1 Contexto e Motivação

A indústria eletrônica, responsável pela manufatura de circuitos integrados (CIs), tem se expandido rapidamente nos últimos anos. A produção dos discos semicondutores que são a base desses circuitos, chamados *wafers*, requer um sistema de manufatura sofisticado e de alto valor associado. As estruturas construídas em um *wafer* semicondutor têm dimensões cada vez menores, sendo que apenas um destes pode dar origem a algumas centenas de circuitos integrados. O número de máquinas necessárias, da mesma forma, é cada vez maior, para conseguir atender todas as complexas operações necessárias. Estas máquinas chegam a custar milhões de dólares.

O processo de fabricação de *wafers* é responsável por mais de 75% do tempo de ciclo na manufatura de circuitos integrados, e é também um dos maiores componentes de custo (MÖNCH et al., 2011). Por esses motivos, há um esforço constante pela melhoria operacional nestes processos, reduzindo os custos e aumentando a produtividade, simultaneamente. A melhoria dos processos é uma das oportunidades mais promissoras para reduzir custos na fabricação de semicondutores (SCHÖMIG; FOWLER, 2000).

Neste contexto, o *cluster tool* têm sido um tipo de equipamento muito utilizado neste processo. Segundo a norma SEMI (2009), o *cluster tool* é "um sistema de manufatura integrado, ambientalmente isolado, consistindo de processos, transporte e módulos cassete mecanicamente ligados em um mesmo conjunto". Nele podem ser executados quase todos os tipos de processos necessários na fabricação de *wafers*, sendo todos integrados em um único equipamento.

Cluster tools têm diversos layouts e requisitos de escalonamento. O layout pode ser radial, linear, ou multi-cluster, de acordo com a configuração dos módulos de processo (PM) (LEE; LEE, 2010).

Devido à diversidade de parâmetros relacionados à especificação de um *cluster tool*, como o número de câmaras, os tempos de processamento do *wafer*, a quantidade de *wafers* a ser produzida, o tipo de transporte e o próprio layout, citando apenas alguns deles, a escolha do modelo mais adequado antes da compra de um novo equipamento se torna difícil. Há alguns modelos que apoiam a tomada de decisão, geralmente fornecidos

pelos próprios fabricantes. Porém, na maior parte das vezes, não é possível comparar a performance de diferentes modelos, devido à diferenças siginificativas na modelagem dos sistemas.

O escalonamento é executado por um sistema computacional, com o objetivo de gerar a sequência de produção e controlar os equipamentos, evitando situações de bloqueio no sistema, chamadas de *deadlock*.

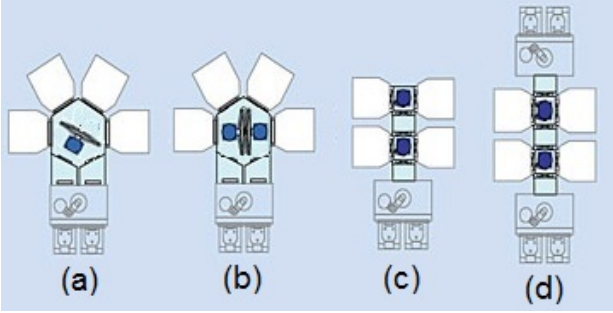
Um problema que pode ocorrer no *cluster tool* é o aparecimento de *deadlocks*. Usualmente, um *deadlock* pode ser resolvido somente interrompendo a operação do *cluster tool*, abrindo-o, e removendo manualmente o *wafer* que gerou o *deadlock*, causando a perda deste *wafer* e diminuição da disponibilidade do equipamento.

Assim, neste trabalho propomos a modelagem e controle do *cluster tool* utilizando a representação baseada em sistemas a eventos discretos (SEDs) por meio da teoria de autômatos e linguagens, considerando o efeito das diversas especificações citadas anteriormente no seu desempenho. Nessa abordagem, um sistema de controle denominado supervisor é construído para controlar e evitar estados indesejáveis da planta, sendo neste caso representada pelo *cluster tool*, utilizando a Teoria de Controle Supervisório (TCS) para isso.

Por meio da TCS, é possível sintetizar um supervisor ótimo, que garante o controle da planta livre de bloqueios, a partir do modelo desta e das especificações de segurança, sendo minimamente restritivo, ou seja, o supervisor permite todas as sequências que não ferem as especificações do sistema e que não o levem a um estado no qual esteja bloqueado. Assim, sua aplicação no *cluster tool* garante que ele opere livre de *deadlocks*, que são causados por falhas na programação e no escalonamento da sequência de produção. Com isso, não há interrupções de funcionamento, que são geradas pelo bloqueio da operação por algum *wafer*.

Além disso, baseado no próprio supervisor, é possível endereçar o problema de sequenciamento por meio de um algoritmo de otimização que faz uma busca em profundidade, garantindo que uma sequência otimizada seja implementada. Assim, podemos aplicar esse algoritmo às diferentes configurações de *cluster tools*, a fim de comparar o desempenho do sistema em várias situações. Dessa forma, temos uma ferramenta que pode ser utilizada ao mesmo tempo para controlar e escalonar o *cluster tool*, e ainda comparar diferentes arquiteturas variando seus principais parâmetros de operação.

Uma vez que a mudança em qualquer parâmetro resulta em diferentes desempenhos operacionais, esta dissertação apresenta uma análise de desempenho do *cluster tool*, comparando o *makespan* e o Paralelismo Acumulado Relativo de quatro layouts diferentes, mostrados na Figura 1, variando o número de câmaras, o tempo de processamento de *wafers* e o número de *wafers*.



(a) Radial com 1 robô, (b) Radial com 2 robôs, (c) Linear com entrada e saída única e (d) Linear com entrada e saída separada

Figura 1 – Layouts Considerados

A metodologia utilizada baseia-se na Teoria de Controle Supervisório (RAMADGE; WONHAM, 1989) de Sistemas a Eventos Discretos, que será utilizada na coordenação do sistema, enquanto o escalonamento será realizado segundo o Critério de Máximo Paralelismo (ALVES; PENA; TAKAHASHI, 2016). A TCS possibilita, ao mesmo tempo, garantir o funcionamento lógico correto e livre de bloqueios, sendo também a base para o uso do algoritmo de escalonamento de Máximo Paralelismo com Restrições Temporais (MPT) (ALVES; PENA; TAKAHASHI, 2016). O MPT executa uma busca no supervisor de modo a obter uma sequência com o maior paralelismo possível, ou seja, com o maior número de máquinas ligadas ao mesmo tempo. O uso desse algoritmo, que é executado sobre o comportamento legal da planta obtido pela TCS, se apresenta como uma alternativa simples de controle e sequenciamento para este tipo de aplicação.

1.2 Objetivo

O objetivo principal deste trabalho é propor um método para analisar e comparar o desempenho do *cluster tool* considerando diferentes parâmetros de operação, como o número de câmaras, o tempo de processamento de *wafers*, o número de *wafers* e o layout. Para tal, utilizaremos a TCS e o algoritmo MPT para controlar e escalonar a sequência de eventos. Entre os objetivos secundários, temos:

- Obter o supervisor para cada configuração.
- Aplicar a TCS no problema de controle.
- Aplicar o algoritmo MPT no problema de escalonamento.
- Verificar o efeito de cada parâmetro no desempenho.
- Incentivar o uso da abordagem de SEDs baseada em autômatos em sistemas de produção.

1.3 Organização do Trabalho

Este artigo está organizado em 7 capítulos, sendo o primeiro esta Introdução.

O capítulo 2 apresenta uma revisão bibliografia sobre o uso da Teoria de Controle Supervisório, sua utilização no escalonamento de tarefas, e a aplicação de abordagens baseadas em SEDs no controle ou escalonamento do *cluster tool* .

No capítulo 3 descrevem-se os conceitos preliminares necessários ao entendimento do trabalho. São apresentadas as definições relacionadas à SEDs, à TCS, linguagens e autômatos. Além disso, são apresentados os conceitos relacionados à produção de *wafers* e a utilização do *cluster tool*. Finalmente, o Critério de Máximo Paralelismo e os algoritmos de Máximo Paralelismo Lógico e Máximo Paralelismo com Restrições Temporais são definidos.

O capítulo 4 apresenta a modelagem do *cluster tool* e a síntese dos supervisores utilizando a TCS.

No capítulo 5 mostra-se a aplicação do algoritmo MPT ao problema e apresentam-se as métricas utilizadas para comparação.

O capítulo 6 traz os principais resultados e a análise dos dados obtidos. As observações finais e conclusões são apresentadas no capítulo 7.

2 Revisão Bibliográfica

Na análise e controle de SEDs são utilizadas várias ferramentas formais que permitem o desenvolvimento, síntese e implementação dos sistemas de controle, como a Álgebra Max-Plus, Teoria das Filas, Redes de Petri e Teoria de Linguagens e Autômatos (CASSANDRAS; LAFORTUNE, 2009).

A Teoria de Controle Supervisório (TCS) (RAMADGE; WONHAM, 1989) proporciona um método formal baseado na teoria de Linguagens e Autômatos para a sintetização de um controlador de sistemas dinâmicos a eventos discretos, denominado supervisor. Esse supervisor é minimamente restritivo, ou seja, ele impede apenas os estados proibidos de serem alcançados, sendo que todos os outros caminhos serão permitidos (CASSANDRAS; LAFORTUNE, 2009). Contudo, a utilização da TCS possui um custo computacional elevado,já que a complexidade envolvida na síntese do supervisor é exponencial no número de componentes do modelo e especificações. Assim, para sistemas de grande porte pode haver a explosão do número de estados inviabilizando o cálculo.

Apesar da TCS fornecer soluções minimamente restritivas, não há informação, à priori, para obter qual seria a melhor sequência, dentre todas as possíveis, para produzir um produto ou um lote deles. Neste sentido, foram propostas diversas abordagens referentes ao escalonamento de SEDs utilizando a TCS (KOBETSKI; FABIAN, 2006; PARK; YANG, 2009; PINHA; QUEIROZ; CURY, 2011; ALVES; PENA; TAKAHASHI, 2016), sendo que quase todas elas focam na minimização do *makespan* para lotes de produção, ou na maximização do *throughput* para um sistema de produção contínuo.

Contudo, a aplicação da TCS em *Cluster Tools* tem pouco respaldo na literatura, principalmente na análise de vários parâmetros de desempenho simultaneamente. Destacam-se principalmente duas abordagens: a análise de um único layout e parâmetros específicos utilizando outras ferramentas de SEDs, como por exemplo as redes de Petri (MURATA, 1989), ou a utilização de outras ferramentas não relacionadas a eventos discretos aplicadas a mais de um tipo de layout.

Os trabalhos do primeiro tipo de abordagem tem sido utilizados com maior frequência. Em Zuberek (2001) foram utilizadas redes de Petri para modelar o comportamento estacionário do *Cluster Tool*, no qual múltiplos robôs são introduzidos para reduzir as limitações quando a operação é *transport bound*, ou seja, o sistema de transporte é um fator limitante do processo, sendo assim o gargalo do *cluster tool*.

Já Shin et al. (2001), utilizou-se máquinas de estado finito para o desenvolvimento de um sistema de escalonamento em tempo real para *cluster tools* com robôs manipuladores do tipo *dual-armed*. O modelo desenvolvido em máquinas de estado finito é utilizado nas

decisões de escalonamento para operações normais e decisões de controle, e em seguida é utilizado para gerar um código de controle executável.

Dentre aqueles que não consideram as ferramentas baseadas em SEDs, diferentes estudos foram propostos para analisar o funcionamento do *cluster tool*.

Em LeBaron e Hendrickson (2000) foi utilizado um modelo para simulação do *cluster tool*, podendo ser utilizado como um emulador ou um modelo de simulação autônomo. A simulação também é usada para estudar o *throughput* e otimização do layout físico por meio de grafos de eventos de *cluster tool* em Nehme e Pierce (1994), Pederson e Trout (2002) e Ding, Yi e Zhang (2006), contudo, esse estudo aborda apenas um único layout. Dümmler (1999) utilizou, além da simulação, algoritmos genéticos para otimizar a operação do *cluster tool*, comparando o resultado com aqueles de abordagens analíticas. Para isso, o autor utiliza um software denominado CluSim, que foi desenvolvido durante o trabalho em linguagem C++.

Em Yi et al. (2007) foi feita uma análise do *throughput* e escalonamento de *cluster tool* lineares. Utilizou-se a formulação matemática por meio do método de decomposição para analisar as configurações de entrada/saída única e entrada/saída separadas, porém, os tempos de deslocamento dos robôs não foram considerados.

Alguns trabalhos se propõem a ampliar a análise para *cluster tools* lineares e radiais simultaneamente, porém, abordando apenas um ou poucos parâmetros. Park e Morrison (2011) usaram a simulação para estudar o efeito do setup e lavagem em ambos os layouts, variando-se o tamanho dos lotes de produção. Por meio de parâmetros reais, foram desenvolvidos modelos *flow line* para três sistemas diferentes. Já Meulen (2007) aponta alguns fatores que poderiam levar a escolhas otimizadas para o processamento de lotes de produção menores que 25 *wafers*, além de mostrar o potencial de diminuição do tempo de ciclo associado com diferentes layouts e tamanhos de lotes. Neste trabalho, verificou-se que pequenos lotes nos sistemas lineares levam a *throughputs* maiores para tempos de processos pequenos.

Outro trabalho que considera diferentes layouts, utilizando as redes de Petri, é proposto por Lee e Lee (2010). Neste artigo, é proposta uma arquitetura de escalonamento aberta para especificação e mudança das regras de escalonamento. Esta considera o layout, a receita e necessidades de escalonamento para projetar e mudar temporalmente as regras de sequenciamento de produção. As redes de Petri foram utilizadas para gerar o escalonamento, definido com uma regra de roteamento de *token* nos lugares de conflito. Essas especificações são implementadas em um arquivo XML de comando no controlador do *cluster tool*.

A abordagem apresentada neste trabalho propõe a utilização da TCS na análise e controle do *cluster tool* simultaneamente. São considerados quatro layouts diferentes de *cluster tools*, e inicialmente o supervisor de cada um deles é sintetizado. A seguir, é aplicado

o método de sequenciamento proposto por Alves, Pena e Takahashi (2016) para encontrar boas sequências de produção. Esse processo é repetido alterando-se vários parâmetros de produção, como o número de *wafers* por lote, o tempo de processamento dos *wafers* nas câmaras e o número de câmaras para cada layout do *cluster tool*. Em cada uma dessas configurações, foram gerados dois parâmetros de desempenho relacionados ao tempo de produção e à eficácia operacional do layout.

Deste modo, cria-se uma metodologia capaz de executar o controle livre de bloqueios e o sequenciamento de produção de todos esses layouts, e ainda comparar cada um deles por meio dos dois parâmetros de desempenho citados em função de cada variável alterada. Isso possibilita dizer qual o layout apresenta melhor desempenho em determinadas condições de produção. Além disso, como a TCS garante um supervisor minimamente restritivo e livre de bloqueios, podemos garantir que não ocorrerão situações de *deadlock* no *cluster tool*, caracterizando mais uma vantagem no uso dessa técnica em relação ao controle e escalonamento. Contudo, convém destacar que teremos um aumento no tempo de processamento à medida que aumentamos o tamanho do problema a ser tratado.

3 Conceitos Preliminares

Nesta seção os principais conceitos tratados nesta dissertação para o desenvolvimento da metodologia proposta são apresentados e definidos. Inicialmente, definimos o conceito de Sistemas a Eventos Discretos. Em seguida, apresentamos a Teoria de Linguagens e Autômatos e a Teoria de Controle Supervisório, com maior ênfase nos supervisores monolíticos. O UltraDES, software utilizado para análise e síntese dos supervisores, é então abordado, e também descrevemos as principais características e conceitos relacionados ao *cluster tool*. Finalmente, o Critério de Máximo Paralelismo e o algoritmo MPT (ALVES; PENA; TAKAHASHI, 2016) são apresentados, sendo estes utilizados no sequenciamento de produção dos layouts considerados.

3.1 Sistemas a Eventos Discretos

Os sistemas a eventos discretos (SEDs) são sistemas dinâmicos regidos por estímulos com o mundo externo, chamados eventos, que ocorrem em instantes de tempo não determinados. Um evento pode indicar o instante de início ou fim de uma tarefa, além de eventos internos. Exemplos de eventos são o início ou fim de uma tarefa do sistema, o acionamento de um sensor em um sistema de manufatura, a chegada de um cliente em uma fila, recepção de mensagens em uma rede de comunicação ou o fim de um evento interno, como uma temporização. Entre a ocorrência de dois eventos consecutivos, o sistema permanece no mesmo estado. São os eventos que causam a mudança destes estados.

Em Cassandras e Lafortune (2009) um SED é definido como um sistema cujo espaço de estados é discreto e a dinâmica é orientada a eventos, ou seja, a evolução dos seus estados depende inteiramente da ocorrência de eventos discretos assíncronos e instantâneos.

Devido a essas características e sua natureza discreta, ferramentas tradicionais de análise de sistemas contínuos ou discretos no tempo, baseadas em equações diferenciais, têm pouca efetividade na análise de SEDs. Deste modo, foram desenvolvidos outros modelos formais para fornecer modelos matemáticos para representar estes sistemas de modo tão apropriado quanto os modelos de equações diferenciais para sistemas de variáveis contínuas.

Os principais modelos formais utilizados que fornecem informações estruturais para SEDs são:

- Álgebra Max-Plus
- Cadeias de Markov
- Redes de Petri

- Teoria das Filas
- Teoria de Linguagens e Autômatos

Dentre estes, o modelo proposto por Ramadge e Wonham (RAMADGE; WONHAM, 1989), baseado na Teoria de Linguagens e Autômatos e denominado "modelo RW", foi utilizado como forma de representação de SEDs nesta dissertação.

Na seção seguinte, apresentamos os principais conceitos da Teoria de Linguagens e Autômatos e da abordagem RW.

3.2 Teoria de Linguagens e Autômatos

A Teoria de Linguagens e Autômatos fornece o ferramental matemático para estudar e representar o comportamento dos SEDs.

3.2.1 Linguagens

Dizemos que um *alfabeto* Σ é formado por um conjunto de eventos finitos. Uma *palavra* ou cadeia s sobre Σ é uma sequência qualquer de eventos deste conjunto, sendo ε a cadeia vazia que não possui qualquer evento. O conjunto de todas as sequências finitas e não-vazias de eventos, tal que cada evento pertencente a Σ, é denotado por Σ^+.

O *fechamento de Kleene* Σ^* é o conjunto de todas as palavras formadas por eventos de Σ, incluindo a palavra vazia ε, tal que:

$$\Sigma^* = \{\varepsilon\} \cup \Sigma^+$$

O comprimento ou tamanho de uma cadeia $|s|$ é igual ao número de eventos que ela contém, incluindo as várias ocorrências do mesmo evento. Assim, se cada ocorrência de evento for denotada por σ_k, sendo k o k-ésimo evento, temos:

$$|\varepsilon| = 0$$

$$|\sigma_1\sigma_2...\sigma_k| = k.$$

A concatenação das cadeias u, v, w forma a palavra $t = uvw$, com t, u, v e $w \cup \ \Sigma^*$, de modo que:

- u é chamado *prefixo* de t
- v é chamado *subcadeia* de t

- w é chamado *sufixo* de t.

Uma linguagem L definida sobre um alfabeto Σ é um subconjunto de cadeias de comprimento finito sobre Σ^*, ou seja, $L \subseteq \Sigma^*$, de tal modo que $\emptyset$, Σ e Σ^* são linguagens. Conforme o exemplo abaixo, diversas linguagens podem ser definidas a partir do mesmo alfabeto.

Para o alfabeto $\Sigma = \{a, b, c\}$, podemos definir as seguintes linguagens:

- $L_1 = \{a, ab, abc\}$, que define uma linguagem formada por três palavras: a, ab e abc;
- $L_2 =$ todas as cadeias de comprimento finito que começam com o evento a;
- $L_3 =$ todas as palavras de comprimento 6 que contenham a subcadeia bc.

Podemos definir algumas operações sobre linguagens, como a união, interseção, diferença e complemento de conjuntos. Outras operações possíveis são a concatenação, potência, o prefixo-fechamento, a projeção natural e a projeção inversa.

A concatenação de duas linguagens L_1 e L_2 sobre um alfabeto Σ é dada por:

$$L_1 L_2 = \{s : s = uv, u \in L_1, v \in L_2\}.$$

Seja L uma linguagem sobre o alfabeto Σ. A potência desta linguagem L^n é definida como $L^0 = \varepsilon$, $L^1 = L$, $L^2 = L \times L$, e assim por diante. O prefixo fechamento $\overline{L}$ de L é uma linguagem que contém todos os prefixos de sequências pertencentes a L, incluindo ε, dado por

$$\overline{L} = \{s \in \Sigma^* \mid su \in L\}$$

Sejam dois alfabetos Σ_1 e Σ_2, tal que Σ_2 é menor ou igual a Σ_1 e $\Sigma_2 \subseteq \Sigma_1$. A projeção natural P_i elimina de uma cadeia $s \in \Sigma_i^*$ os elementos que não pertencem ao alfabeto Σ_i, tal que:

$$P_i(\varepsilon) = \varepsilon$$
$$P_i(\sigma) = \begin{cases} \sigma & \text{se } \sigma \in \Sigma_i \\ \varepsilon & \text{caso contrário} \end{cases}$$
$$P_i(s\sigma) = P_i(s)P_i(\sigma).$$

Para estes mesmos alfabetos, a projeção inversa P_i^{-1} adiciona a uma cadeia s os eventos que não pertencem ao alfabeto Σ_i, mas pertencem ao alfabeto completo $\Sigma_1 \cup \Sigma_2$, tal que:

$$P_i^{-1}(t) = \{s \in (\Sigma_1 \cup \Sigma_2)^* : P_i(s) = t\}.$$

Assim, uma projeção inversa retorna todas as sequências que projetadas formam t. Tanto a projeção natural P_i quanto a projeção inversa P_i^{-1} podem ser estendidas a linguagens, uma vez que linguagem é um conjunto de cadeias.

Para um alfabeto Σ, existe um conjunto de linguagens $F \in 2^{\Sigma*}$, denominadas linguagens regulares, que apresenta as seguintes propriedades:

1. F contém as linguagens $\emptyset$ e σ para $\sigma \in \Sigma$.
2. F é um conjunto fechado sob as operações finitas de união, concatenação e estrela de Kleene.

As linguagens regulares podem ser representadas como autômatos ou expressões regulares, sendo esta uma importante característica. As seguintes condições e operações determinam uma expressão regular:

1. São expressões regulares o conjunto vazio $\emptyset$, o elemento vazio ε e qualquer elemento $\sigma \in \Sigma$ de um alfabeto Σ.
2. Se e_1 e e_2 são expressões regulares, então a concatenação e_1e_2, o fechamento de Kleene e_1^* e_2^*, e a união $e_1 \cup e_2$ são expressões regulares.
3. A aplicação dos itens 1 e 2 por um número finito de vezes forma uma expressão regular.

3.2.2 Autômatos

Um autômato é um modelo matemático capaz de representar linguagens de acordo com regras bem definidas.

Definição 1 *Um autômato finito determinístico é uma quíntupla* $G = (Q,\ \Sigma,\ \hat{\delta},\ q_0,\ Q_m)$, *sendo:*

- Q *o conjunto finito e não-vazio de estados*
- Σ *o conjunto de eventos*
- $\hat{\delta} : Q \times \Sigma \to Q$ *a função de transição*
- $q_0 \in Q$ *o estado inicial*

- $Q_m \in Q$ *o conjunto de estados marcados.*

A representação mais comum do autômato é por meio de um grafo direcionado, no qual os vértices representam os estados e as arestas representam as transições entre esses estados. O estado inicial é indicado por meio de uma seta e os estados marcados por círculos concêntricos. Um exemplo de autômato é mostrado na figura 2.

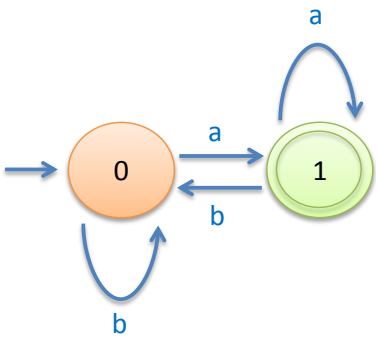


Figura 2 – Exemplo de Autômato Finito Determinístico

Uma transição no autômato G pode ser representada por $x \xrightarrow{\sigma} y$, sendo x o estado de origem, y o estado de destino e σ o evento associado à transição.

A função de transição pode ser estendida para lidar com cadeias sobre Σ^* como $\delta : Q \times \Sigma^* \to Q$. Neste caso, $\delta(q, \epsilon) = q$ e $\delta(q, s\sigma) = \hat{\delta}(\delta(q, s), \sigma)$.

Para um autômato G, temos duas linguagens a ele associadas: a linguagem gerada $\mathcal{L}(G)$ e a linguagem marcada $\mathcal{L}_m(G)$. A linguagem gerada por $G = (Q, \Sigma, \delta, q_0, Q_m)$ é $\mathcal{L}(G) := \{s \in \Sigma^* : \delta(q_0, s)$ é definida$\}$. Esta linguagem representa todas as cadeias do autômato que são formados a partir do estado inicial.

Por sua vez, a linguagem marcada é $\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : \delta(q_0, s) \in Q_m\}$. A linguagem marcada representa todas as cadeias do autômatos que, a partir do estado inicial, terminam em algum estado marcado. Dessa forma, a linguagem marcada é um subconjunto da linguagem gerada, $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$.

Podemos representar um SED por um autômato G, de modo que $\mathcal{L}(G)$ representa o comportamento gerado e $\mathcal{L}_m(G)$ representa o comportamento marcado ou as sequências para alcançar algum estado de interesse ou completar uma tarefa.

3.2.3 Operações sobre Autômatos

Há algumas operações que podem ser aplicadas sobre autômatos. Dentre elas, podemos citar Parte Acessível, Parte Coacessível, Trim, a Composição Paralela e o Produto.

A operação denominada Parte Acessível de um autômato $G = (Q, \Sigma, \delta, q_0, Q_m)$ corresponde a todos os estados acessíveis ou alcançáveis a partir do estado inicial, deste

modo, dizemos que um estado $q \in Q$ é acessível se existe $s \in \Sigma^*$ tal que $\delta(q_0, s) = q$. Assim, esta operação não modifica a linguagem gerada $\mathcal{L}(G)$ e nem a linguagem marcada $\mathcal{L}_m(G)$. Representa-se a Parte Acessível do autômato G por $Ac(G)$.

A operação denominada Parte Co-acessível de um autômato $G = (Q, \Sigma, \delta, q_0, Q_m)$ corresponde a todos os estados que podem alcançar um estado marcado qualquer do autômato G. Neste caso, dizemos que um estado $q \in Q$ é co-acessível se existe $s \in \Sigma^*$ tal que $\delta(q, s) \in Q_m$. A Parte Co-acessível, diferentemente da Parte Acessível, modifica a linguagem gerada $\mathcal{L}(G)$, mas não modifica a linguagem marcada $\mathcal{L}_m(G)$. Representa-se a Parte Coacessível do autômato G por $CoAc(G)$.

A operação Trim consiste em obter a Parte Acessível e a Parte Co-acessível de um autômato G, em qualquer ordem. Deste modo, ela elimina os estados não acessíveis e não co-acessíveis, dando origem a um autômato aparado. Denotamos a operação Trim sobre o autômato G como $Trim(G)$, tal que $Trim(G) = Ac(CoAc(G)) = CoAc(Ac(G))$.

O produto e a composição paralela são operações de composição entre dois ou mais autômatos que permitem representar o comportamento conjunto deles. A composição paralela entre dois autômatos $G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ e $G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$, representada por $G_1||G_2$, é definida por:

$$G_1 \,||\, G_2 = Ac(Q_{1\times 2}, \Sigma_1 \cup \Sigma_2, \delta_{1||2}, (q_{01} \times q_{02}), Q_{m1} \times Q_{m1})$$

tal que:

$$\delta_{1||2}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) \text{ se } \delta_1(q_1, \sigma) \text{ e } \delta_2(q_2, \sigma) \text{ estão definidos} \\ (\delta_1(q_1, \sigma), q_2) \text{ se } \delta_1(q_1, \sigma) \text{ está definido e } \sigma \notin \Sigma_2 \\ (q_1, \delta_2(q_2, \sigma)) \text{ se } \delta_2(q_2, \sigma) \text{ está definido e } \sigma \notin \Sigma_1 \\ \text{indefinido caso contrário} \end{cases}$$

Na composição paralela, se o evento é factível em apenas um dos autômatos, este autômato o executa individualmente. Já se o evento é factível a ambos os autômatos, ele é executado de maneira simultânea, ou seja, ambos executam o evento.

3.3 Teoria de Controle Supervisório

A Teoria de Controle Supervisório (TCS) proposta por Ramadge e Wonham (1989), possibilita, baseado na teoria de linguagens e autômatos, um método formal para o cálculo de supervisores não bloqueantes e minimamente restritivos, tal que a ação de controle sobre a planta seja mínima.

Na TCS, a partir de especificações de segurança e do sistema a ser controlado, denominado planta, gera-se o supervisor, que é o agente controlador, agindo no sistema desabilitando eventos controláveis para limitar o comportamento da planta apenas àquele desejado. Essa estrutura é mostrada na Figura 3.

A planta é modelada por um autômato $G = (Q, \Sigma, f, q_0, Q_m)$, com $\Sigma = \Sigma_c \cup \Sigma_u$, sendo Σ_c o conjunto de eventos controláveis, que podem ser desabilitados pelo supervisor, e Σ_u o conjunto de eventos não controláveis, que não podem ser desabilitados. Esse autômato G é geralmente resultado da composição de vários sub-sistemas que, em conjunto, ditam o comportamento global da planta.

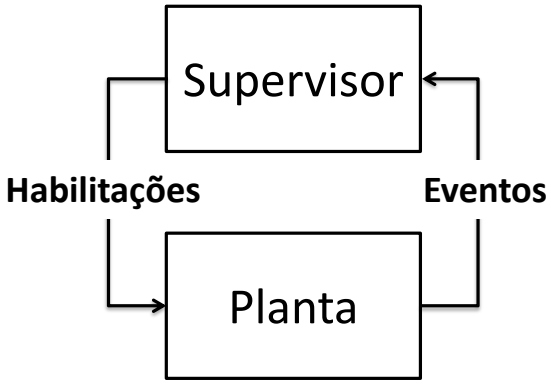


Figura 3 – Estrutura Planta-Supervisor

As especificações de segurança devem garantir que o sistema não alcance estados indesejados, que são aqueles estados que levam a um bloqueio ou representem algum tipo de perigo ao sistema. Cada especificação é dada na forma de um autômato E_j, que podem ser compostas para gerar a especificação global do sistema E.

As linguagens gerada e marcada de uma planta G sobre a ação de um supervisor S são, respectivamente, $\mathcal{L}$(S/G) e $\mathcal{L}_m(S/G) \subseteq \mathcal{L}(S/G)$.

3.3.1 Controle Supervisório Monolítco

Na abordagem monolítica (RAMADGE; WONHAM, 1989), temos apenas um único supervisor responsável por habilitar e desabilitar os eventos controláveis da planta, de acordo com o estado desta, de tal modo que o sistema obedeça as especificações de segurança em malha fechada. Temos um supervisor monolítico representado na Figura 3.

Os eventos permitidos pelo supervisor são todos os eventos não-controláveis e os eventos controláveis habilitados por ele em cada estado observado da planta. Isso visa evitar situações de bloqueio e manter o sistema dentro do comportamento especificado. Um supervisor S é dito não bloqueante quando $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$, ou seja, se o prefixo-fechamento da linguagem marcada pelo supervisor S controlando a planta G é igual à

linguagem gerada pelo supervisor S controlando a planta G, então o supervisor S é não bloqueante.

Sendo G a planta, E uma especificação e K o comportamento desejado, a condição necessária e suficiente para que exista um supervisor S não bloqueante para G, tal que $\mathcal{L}_m(\mathrm{S/G}) = \mathcal{L}(G) \mid\mid E = K$, é que K seja controlável em relação a $\mathcal{L}(G)$ e Σ_u. Diz-se que K é controlável se $\overline{K}\,\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{K}$. Caso não seja, deve-se calcular a máxima sublinguagem controlável da linguagem desejada K em relação a G, denotada por $Sup\mathcal{C}(K, G)$.

Um procedimento para a síntese monolítica de supervisores pode ser descrito por:

1. Modelagem de cada planta local G_i.
2. Obtenção da planta local G pela composição paralela de G_i.
3. Obtenção de especificação genérica E_j
4. Obtenção da especificação global E pela composição paralela de E_j.
5. Obtenção da linguagem desejada K, sendo $K = Trim(E \mid\mid \mathcal{L}_m(G))$.
6. Obtenção do supervisor que implementa a máxima sublinguagem controlável em relação a G contida em K, dado por $S = SupC(K, G)$.

Outras metodologias propostas para síntese de supervisores, como o controle supervisório modular e o controle supervisório descentralizado não serão abordadas neste trabalho, uma vez que utilizamos exclusivamente a abordagem monolítica. Maiores detalhes dessas metodologias são dados em Cassandras e Lafortune (2009).

3.4 UltraDES

Neste trabalho, foi utilizada a ferramenta computacional denominada UltraDES para o desenvolvimento das etapas de cálculo dos supervisores e otimização de sequências utilizando o algoritmo de Máximo Paralelismo.

O UltraDES é uma biblioteca orientada a objetos composta de estruturas de dados e algoritmos para modelagem, análise e controle de sistemas a eventos discretos. Essa biblioteca foi desenvolvida na linguagem *C#*, sendo *.NET Framework* sua plataforma de execução (ALVES; MARTINS; PENA, 2017).

O UltraDES possui classes que representam autômatos, expressões regulares e transições. As operações como união, concatenação e fechamento de Kleene são também definidas como classes.

Um autômato finito determinístico é representado por uma classe definida por uma lista de transições, um estado inicial e um nome, e que possui como propriedades: estados, estados marcados, estado inicial, eventos, nome, transições e funções de transição. As seguintes operações podem ser executadas sobre autômatos determinísticos: composição paralela, produto, parte acessível, parte co-acessível e *trim*. Além disso, pode-se calcular os supervisores monolíticos e o supervisor modular local a partir de uma lista de plantas e especificações modeladas.

Além disso, o UltraDES possibilita, de forma fácil, criar uma interface com outros algoritmos desenvolvidos na linguagem *C#*. O uso do UltraDES neste trabalho deve-se principalmente à utilização do algoritmo de Máximo Paralelismo com Restrição Temporal (MPT), implementado nesta ferramenta. Uma descrição detalhada deste algoritmo é apresentada na seção 3.7.

3.5 Produção de *Wafers*

Nesta seção é explicado de forma breve como os *wafers*, que posteriormente darão origem aos CIs, são produzidos.

A sequência de produção necessária para a manufatura dos CIs consiste basicamente de cinco passos:

1. Preparação do *wafer*
2. Fabricação do *wafer*
3. Testes elétricos
4. Montagem
5. Testes finais

Neste trabalho, focamos na aplicação do *cluster tool* na fabricação do *wafer*, sendo portanto, o único passo que será descrito.

Independente das diferenças entre os *wafers* produzidos, como a estrutura do chip e os materiais básicos, segundo Dümmler (1999), o processo de fabricação consiste da aplicação repetida de quatro processos básicos:

1. Criação de camada
2. Transferência de padrões
3. Dopagem

4. Tratamento térmico

Na criação de camada, material isolante, condutor ou semicondutor é adicionado à superfície do *wafer*, e na transferência de padrões partes deste material são removidas para formar as estruturas geométricas necessárias. Na dopagem, são acrescentados materiais que dão ao *wafer* características eletrônicas desejadas. E, finalmente, no tratamento térmico, o *wafer* é aquecido em temperaturas de 500°C a 1000°C para regularizar suas estruturas cristalinas.

Devido à sua sensibilidade à contaminação molecular e por partículas, a fabricação dos *wafers* deve ser executada em ambiente de sala limpa. Assim, o uso do *cluster tool* reduz o ambiente de controle àquele do equipamento, e não mais de todo o ambiente fabril, sendo, para este último caso, um processo com alto custo.

3.6 *Cluster Tool*

3.6.1 *Conceitos*

Um *cluster tool* consiste de vários módulos de processamento de *wafers* (*PM*), robôs manipuladores, e *loadlocks* para carga e descarga, sendo todos esses equipamentos integrados em um ambiente de sala limpa. Nele, várias etapas do processo de manufatura são integradas em uma única ferramenta, que cobrem todas os tipos de processos de produção necessários. Os *wafers* são transportados em módulos cassetes denominados *Front Opening Unified Pod (FOUP)*, que têm capacidade para até 25 *wafers*. A Figura 4 ilustra uma configuração de *cluster tool* com seus componentes.

O *cluster tool* é operado por um software de controle chamado Controlador do Cluster Tool (CCT). Um subsistema do CCT, responsável pela coordenação das atividades e determinação da sequência de tarefas do robô, é chamado de *escalonador* (SHIN et al., 2001). Este regula a sequência das etapas de produção e regula os recursos, possibilitando reduzir os tempos de espera e transporte.

Do ponto de vista econômico, uma importante propriedade deste tipo de equipamento é que sua modularidade permite seu uso na produção de diferentes produtos finais. Se um novo produto requer diferentes equipamentos de processo, é possível alterar apenas as câmaras de processo específicas, enquanto mantém-se o *mainframe*, o robô manipulador e as outras câmaras de processo (DÜMMLER, 1999).

O *cluster tool* pode operar em duas regiões de operação diferentes. Na primeira, quando ele opera na região denominada *process bound*, o tempo de processamento dos *PMs* domina o tempo de ciclo e o robô deve esperar algum *PM* finalizar a operação. Por

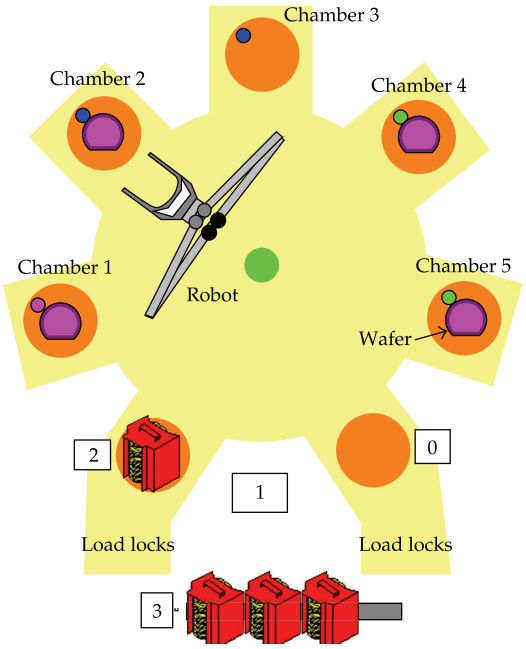


Figura 4 – Exemplo de *cluster tool* (LEE; NI, 2012)

outro lado, na região denominada *robot bound*, o robô está sempre ocupado, transferindo *wafers* entre os *PMs* (YI et al., 2007).

3.6.2 Componentes

A seguir, descrevemos os principais componentes de um *cluster tool* .

Mainframe

O *mainframe* é o componente central do *cluster tool* . Ele abriga o robô manipulador que move os *wafers* entre as câmaras de processo. Basicamente, existem dois principais tipos: com *layout* radial e com *layout* linear, detalhados na próxima seção. A Figura 5 mostra um exemplo de *mainframe* radial.

Câmaras de Processo

As câmaras de processo ou módulos de processo (PM) são responsáveis por realizar o processamento de cada etapa do processo de produção, como os processo físicos e químicos e inspeções.

Os PMs são conectadas ao *mainframe*. Geralmente, as PMs processam apenas um único *wafer* por vez (*single chamber*). Mas, também existem câmaras que processam mais de um *wafer* em paralelo (*batch chambers*), ou uma sequência de processos (*index modules*).

Figura 5 – *Mainframe* radial *(Brooks Automation 2017)*

Neste trabalho, consideramos as PMs do tipo *single chambers*.

Load Locks

Os *load locks* ou módulos cassete, são as interfaces entre o *cluster tool* e o chão-de-fábrica, sendo que há pelo menos um deles em cada *cluster tool*.

Um lote de *wafers* a ser processado é colocado no *load lock* por meio de um operador ou sistema de manuseio de material. O *load lock* é então isolado do ambiente externo, gerando as condições necessárias para o processamento dos *wafers*.

Em alguns equipamentos, há um *load lock* no qual os *wafers* são inseridos no *cluster tool* e outro, independente, no qual eles são retirados.

Robôs Manipuladores

O mecanismo mais comum de transporte dos *wafers* dentro do *cluster tool* é o robô manipulador. A quantidade de robôs manipuladores dependerá da configuração do equipamento, sendo que deve haver pelo menos um em cada *cluster tool*. O robô transporta os *wafers* do *load lock* para as câmaras de processamento, entre as câmaras de processamento, e destas de volta para o *load lock*.

Os robôs manipuladores são categorizados pelo número de *wafers* que eles podem transportar por vez. Robôs *single-armed* carregam apenas um *wafer*, já os *dual-armed* podem carregar dois *wafers* por vez, podendo cada braço ser independente ou ligados mecanicamente com um deslocamento de 180° entre eles. Neste trabalho, utilizaremos robôs *single-armed*. Estes dois tipos são mostrados na Figura 6.

A velocidade de transporte dos robôs manipuladores dependerá se ele está carregado ou não, e se o *wafer* sendo transportando está quente ou não devido ao processamento nas câmaras. O movimento carregado e o *wafer* em temperatura maior fazem com que o robô se desloque com velocidade reduzida.

(a) Single Armed (b) Dual Armed

Figura 6 – Tipos de robôs *(Brooks Automation 2017)*

Controlador do *Cluster tool*

O controlador do *cluster tool* é o módulo responsável pelo controle de todas as atividades e ações dentro do equipamento.

Esse controlador pode ser integrado ao equipamento ou pode estar implementado em um computador que se comanda o *cluster tool* por meio de redes de comunicação.

Suas principais funções são a coordenação e escalonamento de cada componente, o fornecimento da receita para produção dos *wafers*, a adequação das condições ambientais para os *wafers* vindos do *load lock*, a aquisição de dados e o aviso ao operador, na forma de alarmes, da ocorrência de *dead locks*.

3.6.3 Layouts

O layout define a forma como os componentes (robôs manipuladores, câmaras de processos, *load locks*, etc) estão organizados no *cluster tool* . O escalonamento do *cluster tool* depende inteiramente do seu layout.

Basicamente, podemos definir dois layouts diferentes. No primeiro tipo, há um *mainframe* central contendo um ou dois robôs manipuladores, conforme mostrados na Figura 7, no qual as câmaras de processo são acopladas de forma radial. Assim, este tipo é denominado layout radial.

No layout radial, os *wafers*, armazenados no FOUP, são colocados no *load lock*. Um único robô manipulador localizado no centro do *mainframe* é responsável por transportar os *wafers* do *load lock* para as câmaras de processamento, e entre estas, de acordo com a sequência desejada de operações. Assim, que o *wafer* passa por todos os processos necessários, retorna para o FOUP.

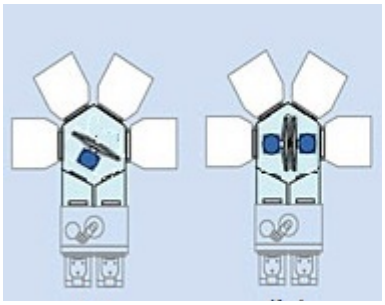


Figura 7 – Layout Radial

No segundo tipo, o *cluster tool* é composto por módulos interconectados em série, sendo que cada módulo possui um robô manipulador, e pode-se acoplar câmaras de processo ou outros módulos, formando um fluxo linear. Nesse layout, pode-se ter apenas um *load lock* no *cluster tool* , no qual os *wafers* são inseridos e retirados, ou pode haver dois *load locks*, um posicionado em uma das extremidades para a inserção dos *wafers*, e um segundo na outra extremidade para retirada. Ambos são mostrados na Figura 8. Este tipo é denominado layout linear.

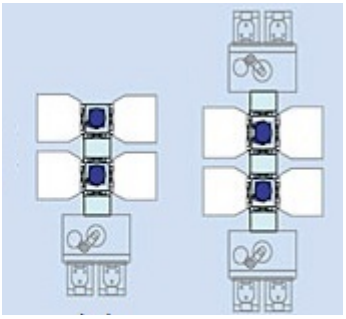


Figura 8 – Layout Linear

No layout linear, os *wafers*, armazenados no FOUP, são colocados no *load lock*. O robô manipulador de cada módulo que compõe o *cluster tool* transporta os *wafers* do *load lock* para as câmaras de processamento do seu módulo, e em seguida, transfere o *wafer* para o módulo seguinte, que também possui um robô manipulador. Quando o *cluster tool* possui apenas um *load lock*, o *wafer* retorna por todo o equipamento até o FOUP, após passar por todos os processos necessários, e dizemos que ele é de entrada/saída única. Já no caso em que há dois *load locks*, o *wafer* é colocado em outro FOUP na extremidade oposta àquela de entrada do *cluster tool* , e dizemos que ele é de entrada/saída separada.

O layout linear nos *cluster tools* é mais flexível que a configuração radial, porém ele não apresenta um tempo de movimentação dos robôs uniforme e minimizado (LEE; LEE, 2010). Essa flexibilidade advém da possibilidade de expansão do *cluster tool* apenas acoplando um conjunto de robô e dois PMs, sem a limitação de número máximo de PMs no *mainframe*, como acontece no caso radial.

Conforme Dümmler (1999), existem duas configurações possíveis para os PMs

dentro de um layout. Na configuração serial, cada câmara de processamento realiza um passo diferente do processo, enquanto na configuração paralela todas as câmaras de processamento são idênticas. Ainda, o *cluster tool* que processa os *wafers* em uma sequência pré-definida é chamado de sequência fixa, enquanto aquele no qual o sistema de controle decide como alocar os recursos disponíveis é chamado de sequência flexível. Quanto ao modo de operação, o *cluster tool* pode ser de modo único, no qual ocorre o processamento de apenas um lote por vez, ou pode ser de modo paralelo, no qual pode-se processar um segundo lote colocado em outro *load lock* em paralelo. Assim, este modo é possível apenas quando há mais de um *load lock* disponível.

De acordo com estas classificações, neste trabalho considerou-se que os *cluster tools* têm uma configuração serial, são de modo único e com sequência fixa, ou seja, cada *PM* é responsável por um passo do processo e cada lote é processado separadamente com uma sequência pré-estabelecida.

3.7 Critério de Máximo Paralelismo

O critério de Máximo Paralelismo, proposto por Alves, Pena e Takahashi (2016), apresenta uma maneira de minimizar o tempo de produção sem levar em conta o próprio tempo, encontrando, ao invés disso, uma solução sub-ótima que maximiza o número de máquinas funcionando em paralelo, sendo o método utilizado determinístico.

Esse método visa encontrar, dentre as execuções permitidas pelo supervisor, aquela que possua o maior número de tarefas acumuladas ao longo de toda a cadeia de eventos, ou seja, mais equipamentos trabalhando ao mesmo tempo. Isso pode ser feito de um ponto de vista totalmente lógico, utilizando apenas o paralelismo e o supervisor para encontrar a sequência maximamente paralela, ou ainda, utilizando informações temporais dos eventos para guiar a geração de sequências que sejam temporalmente factíveis. O primeiro caso denomina-se Máximo Paralelismo Lógico (MPL), e o segundo Máximo Paralelismo com Restrições Temporais (MPT).

Para implementação dos algoritmos de Máximo Paralelismo, é necessário incluir no estado do autômato a informação sobre o número de tarefas ativas ou em execução. Uma tarefa é um conceito subjetivo, que deve ser definido durante a modelagem do sistema, podendo ser o funcionamento de uma máquina ou a execução de uma sub-rotina, por exemplo (ALVES; PENA; TAKAHASHI, 2016). Consideramos neste trabalho que uma tarefa correspondente ao funcionamento de algum componente do *cluster tool* , indicando que o mesmo está executando trabalho.

Podemos definir um autômato modificado G_m conforme a seguir:

Definição 2 *(ALVES; PENA; TAKAHASHI, 2016) Seja um autômato*

$G = (Q, \Sigma, f, q_0, Q_m)$, definimos um autômato modificado como uma quíntupla $G_m = (W, \Sigma, f_m, w_0, W_m)$, sendo $W = (q,i)$ o número de tarefas $i \in \mathbb{Z}$ em execução no estado $q \in \mathbb{Q}$, $f_m \subseteq W \times \Sigma \times W$ a relação de transição entre os estados, $w_0 \in W$ o estado inicial e $W_m \subseteq Q_m \times \mathbb{Z}$ o conjunto de estados marcados.

Deve-se também redefinir a operação de composição síncrona para estados expandidos:

Definição 3 *(ALVES; PENA; TAKAHASHI, 2016) A composição síncrona dos autômatos modificados $G_1 = (W_1, \Sigma_1, f_{m1}, w_{0_1}, W_{m_1})$ e $G_2 = (W_2, \Sigma_2, f_{m2}, w_{0_2}, W_{m_2})$, com $W_1 = (q_1, i)$, $W_2 = (q_2, j)$, $w_{0_1} = (q_{0_1}, i)$ e $w_{0_2} = (q_{0_2}, j)$ é definida como $G_1||G_2 = (W_{12}, \Sigma_1 \cup \Sigma_2, f_{m12}, w_{0_{12}}, W_{m_{12}})$, tal que:*

$W_{12} = \{((q_1, q_2), i + j) : (q_1, i) \in W_1, (q_2, j) \in W_2\}$
$W_{m_{12}} = \{((q_1, q_2), i + j) : (q_1, i) \in W_{m_1}, (q_2, j) \in W_{m_2}\}$
e $w_{0_{12}} = ((q_{0_1}, q_{0_2}), i + j)$

Conforme Alves, Pena e Takahashi (2016), essa nova definição de composição síncrona de autômatos modificados não altera a linguagem marcada do autômato resultante, sendo possível utilizar a TCS sem qualquer alteração adicional para realizar a síntese de supervisores.

No caso dos autômatos que representam as especificações, eles não apresentam nenhuma tarefa ativa, pois não realizam tarefas. Dessa forma, a expansão dos autômatos das especificações deve ser feita com as tarefas em execução com valor nulo em todos os estados.

Para avaliar o paralelismo de uma sequência finita, é necessário definir a função acumulativa de tarefas ativas.

Definição 4 *(ALVES; PENA; TAKAHASHI, 2016) A função acumulativa de tarefas ativas, $F_{TA} : W \times \Sigma^* \to \mathbb{Z}^*$ é definida como:*

$$\begin{cases} F_{TA}((q, i), \epsilon) = i \\ F_{TA}((q, i), \sigma s) = i + F_{TA}(f_m((q, i), \sigma), s) \end{cases}$$

onde $\sigma s \in \Sigma^*$ e $f_m \subseteq W \times \Sigma \times W$ é a relação de transição entre os estados.

Para duas sequências s_1 e s_2, tais que $|s_1| = |s_2|$, aquela com maior valor de F_{TA} tem maior paralelismo.

Com base nessas definições, o MPL e o MPT são descritos nas seções seguintes.

3.7.1 Algoritmo MPL

No algoritmo de Máximo Paralelismo Lógico (MPL), busca-se uma sequência de eventos que maximiza o número de tarefas ativas, sendo esta busca feita de um ponto de vista totalmente lógico, utilizando apenas o paralelismo e o supervisor para encontrar a sequência maximamente paralela.

Sendo n o número de eventos para produção de um lote de produtos e o universo de busca $L = \{s \in \mathcal{L}(S/G) : |s| = n\}$, o problema de otimização pode ser definido como o de encontrar o caminho s^* no autômato do supervisor S que maximiza o número acumulado de tarefas ativas F_{TA}:

$$s^* = \underset{s \in L}{\text{argmax}}\ F_{TA}(s).$$

A resolução desse problema de otimização é por meio de um problema de maior caminho em um grafo acíclico, sendo que o peso de uma transição é o número de tarefas ativas no estado de destino.

Para contornar a presença de ciclos no supervisor é necessário definir uma restrição quanto ao número de eventos que serão executados, ou seja, a profundidade máxima de busca n.

Para tornar o autômato do supervisor um grafo acíclico com profundidade n, é necessário realizar sua composição com o autômato de desenrolamento G_d, mostrado na Figura 9, sendo $\mathcal{L}_m(G_d) = \{s \in \Sigma^* : |s| = n\}$ e Σ o conjunto de eventos de S.

Figura 9 – Autômato de Desenrolamento com profundidade n

O autômato resultante da operação de composição é acíclico e permite a execução de um algoritmo de maior caminho, sendo que, a partir do estado inicial, os demais estados são visitados em ordem topológica.

As entradas para o algoritmo que busca a cadeia de maior paralelismo são:

- conjunto de estados expandidos (W);
- a função de transição de estados (δ);
- a função de eventos ativos (Γ);
- profundidade de busca (*depth*);
- estado inicial (w_0).

Com essas entradas, o algoritmo retorna o mapeamento (*pred*) que fornece o maior caminho, de acordo com o pseudo-código mostrado no Algoritmo 1.

Algoritmo 1: (ALVES; PENA; TAKAHASHI, 2016) Busca da cadeia que resulta no maior paralelismo

```
   // Inicialização
1  foreach state w in W do
2      for i ← 0 to depth do
3          if (w, i) = (w_0, 0) then
4              d[(w, i)] ← 0
5              path[(w, i)] ← ϵ
6          else
7              d[(w, i)] ← −∞
8              path[(w, i)] ← ∅
9          end
10     end
11 end
   // Busca pelo maior caminho
12 F ← (w_0, 0)
13 while F not empty do
14     (w, i) ← F
15     if i = depth then
16         Continue
17     end
18     foreach event σ in Γ(w) do
19         v ← δ(w, σ)
20         if F doesn't contains (v,i+1) then
21             F ← (v, i + 1)
22         end
23         u ← f_ta(v)
24         if d[(w,i)] + u > d[v,i+1)] then
25             d[(v, i + 1)] ← d[(w, i)] + u
26             path[(v, i + 1)] ← path[(w, i)]σ
27         end
28     end
29 end
```

Como vantagem, pode-se citar que o MPL é um algoritmo exato, quanto à maximização do paralelismo e apresenta complexidade polinomial no número de estados. Por outro lado, sua principal desvantagem é ser puramente lógico e não levar em consideração

nenhuma informação quanto ao tempo de execução de cada tarefa, gerando sequências temporalmente infactíveis, ou seja, os eventos não controláveis não ocorrem necessariamente na sequência que o algoritmo gera (ALVES; PENA; TAKAHASHI, 2016).

3.7.2 Algoritmo MPT

O MPT representa uma pequena modificação em relação ao MPL, aplicando restrições temporais, tais que, sendo os tempos de processo determinísticos, pode-se obter sequências que apresentam coerência do ponto de vista temporal. Para adicionar essas informações temporais, é necessário avaliar o tempo decorrido para a execução da sequência a medida em que ela é gerada.

A função temporal f_T avalia o tempo para ocorrência de um evento no supervisor, dada a sequência já executada.

Definição 5 *(ALVES; PENA; TAKAHASHI, 2016) Seja $S = (Q, \Sigma, \delta, q_0, Q_m)$ um supervisor. A função temporal $f_T' : \Sigma^* \times \Sigma \to \mathbb{R}^*$, de S é definida como:*

$$f_T'(s,\sigma) = \begin{cases} t & \text{se } \delta(s,\sigma) \text{ está definido} \\ \infty & \text{caso contrário} \end{cases}$$

tal que, para um evento $\sigma \in \Sigma$ e uma sequência $s \in \mathcal{L}(S/G)$, t é o tempo até que o evento σ ocorra, dado que a sequência s acabou de ser executada.

A função temporal é implementada no algoritmo utilizando a simulação de sistemas a eventos discretos, avaliando a diferença entre o tempo de execução do último evento da sequência s e o tempo de s.

Para a avaliação da informação temporal de uma sequência de eventos, a função temporal deve ser expandida:

Definição 6 *(ALVES; PENA; TAKAHASHI, 2016) Seja $S = (Q, \Sigma, f, q_0, Q_m)$ um supervisor. A função temporal expandida $f_T : \Sigma^* \to \mathbb{R}^*$, de S é definida como:*

$$\begin{cases} f_T(\epsilon) = 0 \\ f_T(s\sigma) = f_T(s) + f_T'(s,\sigma) \end{cases}$$

Assim, busca-se a sequência da linguagem marcada do supervisor que ao mesmo tempo maximize o paralelismo e tenha um tempo de execução menor que infinito, indicando que ela é temporalmente factível.

Sendo n o número de eventos para produção de um lote de produtos e o universo de busca $L = \{s \in \mathcal{L}(S/G) \,|\, |s| = n \wedge f_T(s) \neq \infty\}$, o problema de otimização pode ser

definido como o de encontrar o caminho s^* no autômato do supervisor S que maximiza o número acumulado de tarefas ativas F_{TA}:

$$s^* = \underset{s \in L}{\operatorname{argmax}} \; F_{TA}(s).$$

Esse problema é similar ao problema descrito no MPL, porém, a informação temporal impede que ele seja tratado de maneira exata. Assim, é proposto em Alves, Pena e Takahashi (2016) um algoritmo de melhor caminho que toma um passo guloso pegando o melhor caminho até determinado estado, desconsiderando que este, alcançado de maneiras diferentes, possui diferentes futuros. Assim, o MPT passa a ser um algoritmo heurístico, mas que garante a execução temporal da sequência obtida.

Assim como no MPL, é necessário compor o autômato do supervisor com o autômato de desenrolamento. Além disso, é necessário estabelecer o tempo de operação de cada equipamento.

As entradas para o algoritmo que busca a cadeia de maior paralelismo também são:

- conjunto de estados expandidos (W);
- A função de transição de estados (δ);
- A função de eventos ativos (Γ);
- profundidade de busca (*depth*);
- estado inicial (w_0).

O algoritmo retorna a estrutura *path* que armazena o maior caminho entre o estado inicial (q_0,0) e qualquer outro estado alcançado na busca, de acordo com o pseudo-código

mostrado no Algoritmo 2.

Algoritmo 2: (ALVES; PENA; TAKAHASHI, 2016)Máximo Paralelismo com Restrições Temporais

```
   // Inicialização
1  foreach state w in W do
2      for i ← 0 to depth do
3          if (w, i) = (w_0, 0) then
4              d[(w, i)] ← 0
5              path[(w, i)] ← ϵ
6              time[(w, i)] ← −∞
7          else
8              d[(w, i)] ← 0
9              path[(w, i)] ← ∅
10             time[(w, i)] ← ∞
11         end
12     end
13 end
   // Busca pelo maior caminho
14 F ← (w_0, 0)
15 while F not empty do
16     (w, i) ← F
17     if i = depth then
18         Continue
19     end
20     foreach event σ in Γ(w) do
21         v ← δ(w, σ)
22         t ← f_T(path[(w, i)]σ)
23         if t = ∞ then
24             Continue
25         end
26         if F doesn't contains (v,i+1) then
27             F ← (v, i + 1)
28         end
29         u ← f_ta(v)
30         d_q ← d[(w, i)]
31         d_v ← d[(v, i + 1)]
32         if d_q + w > d_v or (d_q + w = d_v and t < time[(v, i + 1)]) then
33             d[(v, i + 1)] ← d[(w, i)] + u
34             path[(v, i + 1)] ← path[(w, i)]σ
35             time[(v, i + 1)] ← t
36         end
37     end
38 end
```

A ocorrência de eventos geralmente é instantânea, de modo que, quando falamos de temporização em uma sequência, nos referimos ao tempo de permanência nos estados, ou seja, a diferença entre o tempo de ocorrência de dois eventos relacionados, como por exemplo, o evento de máquina ligada e máquina desligada, no qual a diferença de tempo Δt entre estes dois eventos indica o tempo de funcionamento da máquina.

De modo geral, associamos neste trabalho os eventos controláveis ao início de tarefas no *cluster tool* e os eventos não controláveis ao fim dessa tarefa, sendo estes respostas da planta ao eventos controláveis.

No MPT é utilizada uma estrutura chamada Agendador de Eventos que determina o tempo necessário até que um evento ocorra, sendo que este é inicializado com tempo zero para eventos controláveis e tempo infinito para os eventos não controláveis.

O agendador de eventos executa uma simulação, que, ao final, determina o tempo total de execução t da sequência especificada na cadeia s. Se $t = \infty$, a sequência não é temporalmente factível.

Neste trabalho, utilizaremos apenas o MPT, já que este algoritmo garante que as sequências geradas são temporalmente factíveis para intervalos de tempo entre eventos preestabelecidos, sendo esta sua principal diferença e vantagem em relação ao MPL.

4 Modelagem e Síntese dos Supervisores

Neste capítulo é apresentada a modelagem por meio de autômatos dos layouts de *cluster tools* considerados. A obtenção do modelo e dos tempos de duração das tarefas é detalhada. A obtenção dos supervisores a partir dos modelos obtidos, indicando também alguns parâmetros de desempenho do UltraDES na síntese desses supervisores é apresentada.

4.1 Modelagem do Cluster Tool

Neste trabalho, a modelagem do *cluster tool* como um SED é feita por meio de autômatos finitos determinísticos. Para cada layout e suas diferentes configurações, foi gerado o respectivo modelo.

Os quatro layouts de *cluster tools* considerados neste trabalho são:

1. Layout radial com um único robô manipulador *single-armed*
2. Layout radial com dois robôs manipuladores *single-armed*
3. Layout linear com entrada e saída única, com apenas um *load lock* para inserção e retirada de *wafers* no *cluster tool*
4. Layout linear com entrada e saída separadas, sendo um *load lock* para inserção e outro para retirada de *wafers*

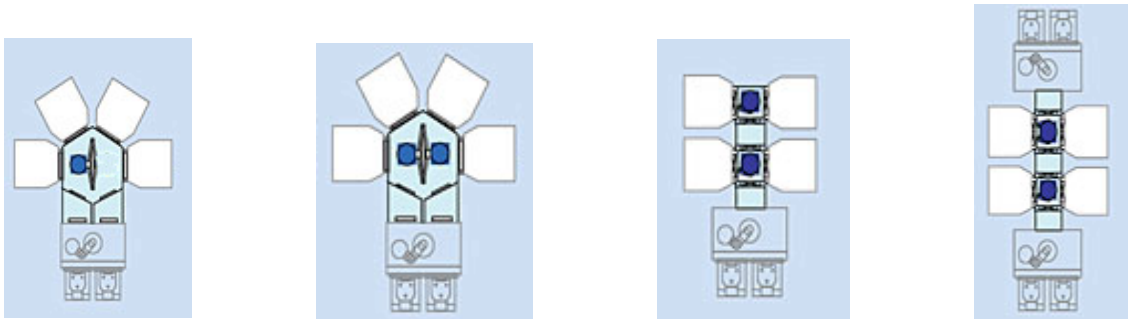
Cada *load lock* possui um robô manipulador e duas *load ports* para armazenar o *FOUP*, onde os *wafers* são transportados.

Sendo um dos objetivos do trabalho verificar a influência dos parâmetros no desempenho do *cluster tool* , em cada um dos layouts variaram-se:

- **Quantidade de câmaras de processamento (*PMs*)**: alterou-se o número de *PMs* em cada layout, utilizando cada configuração com 4, 5 e 6 câmaras do tipo *single chambers*.
- **Tempo de processamento do *wafer* no *PM***: Variou-se o tempo de processamento do *wafer* em cada uma das 30 execuções do algoritmo no conjunto de valores {1, 3, 5, 7, 10, 13, 15, 17, 20, 23, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 125, 150, 175, 200}. Para cada layout a ser comparado, o tempo de processamento de todos os *PMs* é igual.

- **Número de *wafers* por lote**: alterou-se o número de *wafers* em cada situação, sendo um lote podendo ser composto por 6, 12, 25 ou 50 *wafers*, para verificar a influência deste parâmetro.

Assim, temos os layouts com 4 *PMs* mostrados na Figura 10, com 5 *PMs* mostrados na Figura 11 e com 6 *PMs* mostrados na Figura 12. Em cada um deles, variou-se o tempo de processamento dos *PMs* e número de *wafers* conforme descrito anteriormente.



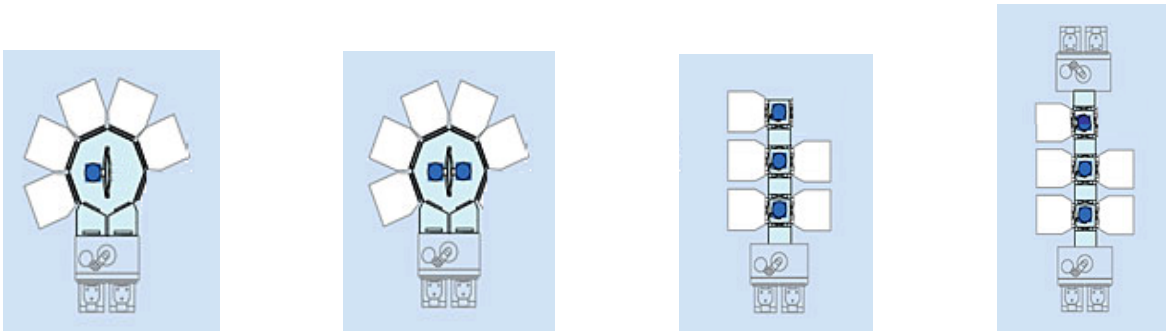
(a) Radial - Robô Único

(b) Radial - Dois Robôs

(c) Linear - Entrada e Saída Única

(d) Linear - Entrada e Saída Separada

Figura 10 – Layouts do *cluster tool* com 4 *PMs*



(a) Radial - Robô Único

(b) Radial - Dois Robôs

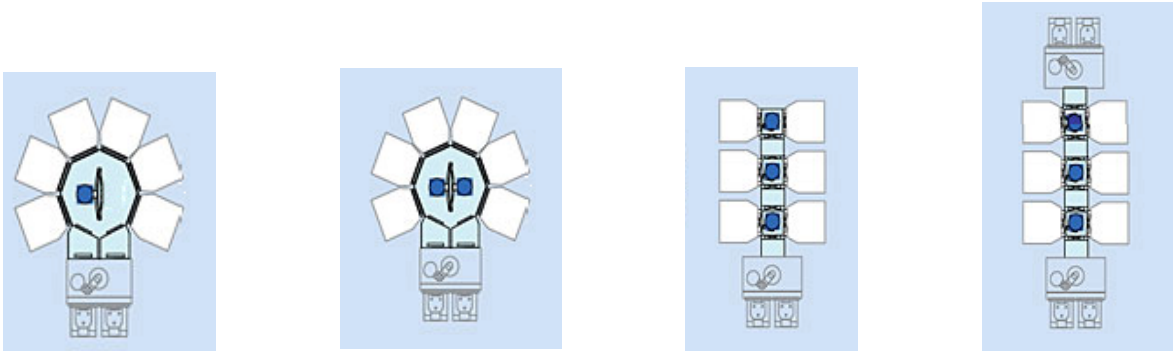
(c) Linear - Entrada e Saída Única

(d) Linear - Entrada e Saída Separada

Figura 11 – Layouts do *cluster tool* com 5 *PMs*

A receita do *wafer*, que é a sequência de passos a ser seguida no *cluster tool*, é única para todos os layouts, considerando um fluxo serial, no qual o *wafer* visita cada *PM* exatamente uma vez de modo sequencial, sendo este completamente responsável por um passo do processo.

A Figura 13 mostra a receita para cada layout, sendo a sequência do processamento do *wafer* identificada pelos números ao lado de cada *PM*. No layout radial com 1 robô, o *wafer* é processado em sentido horário, iniciando pela câmara mais à esquerda, sendo que o robô único realiza o transporte entre todas as câmaras. No layout radial com 2 robôs, a sequência é a mesma, porém, os dois primeiros *PMs* são atendidos por um robô, e os dois



(a) Radial - Robô Único

(b) Radial - Dois Robôs

(c) Linear - Entrada e Saída Única

(d) Linear - Entrada e Saída Separada

Figura 12 – Layouts do *cluster tool* com 6 *PMs*

últimos pelo outro robô. No layout linear com entrada e saída única, o *wafer* é inserido no *cluster tool* e inicialmente percorre-o pelo lado esquerdo, e em seguida pelo lado direito, voltando para o *load load* onde será retirado. Nesse layout, cada robô é responsável pelo transporte no seu *mainframe*. Finalmente, no layout linear com entrada e saída separada, o *wafer* é colocado no *load lock* de entrada, percorrendo os *PMs* de cada *mainframe* em direção ao *load lock* de saída. A mesma sequência é respeitada para layouts com maior número de *PMs*.

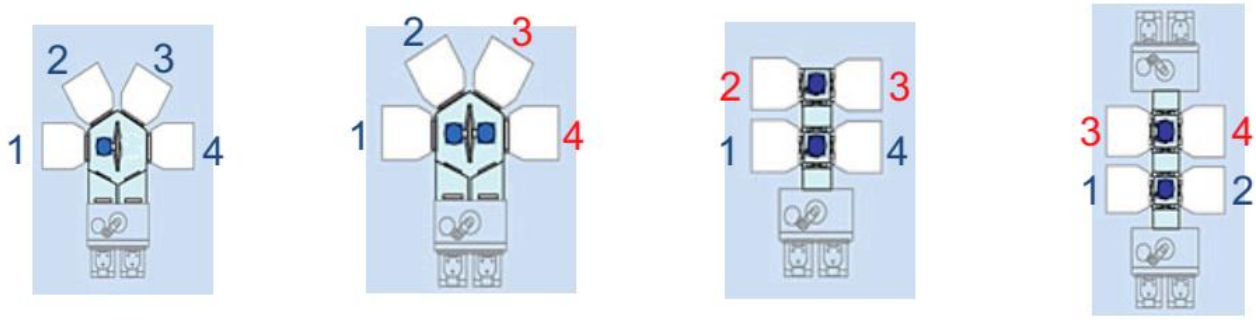
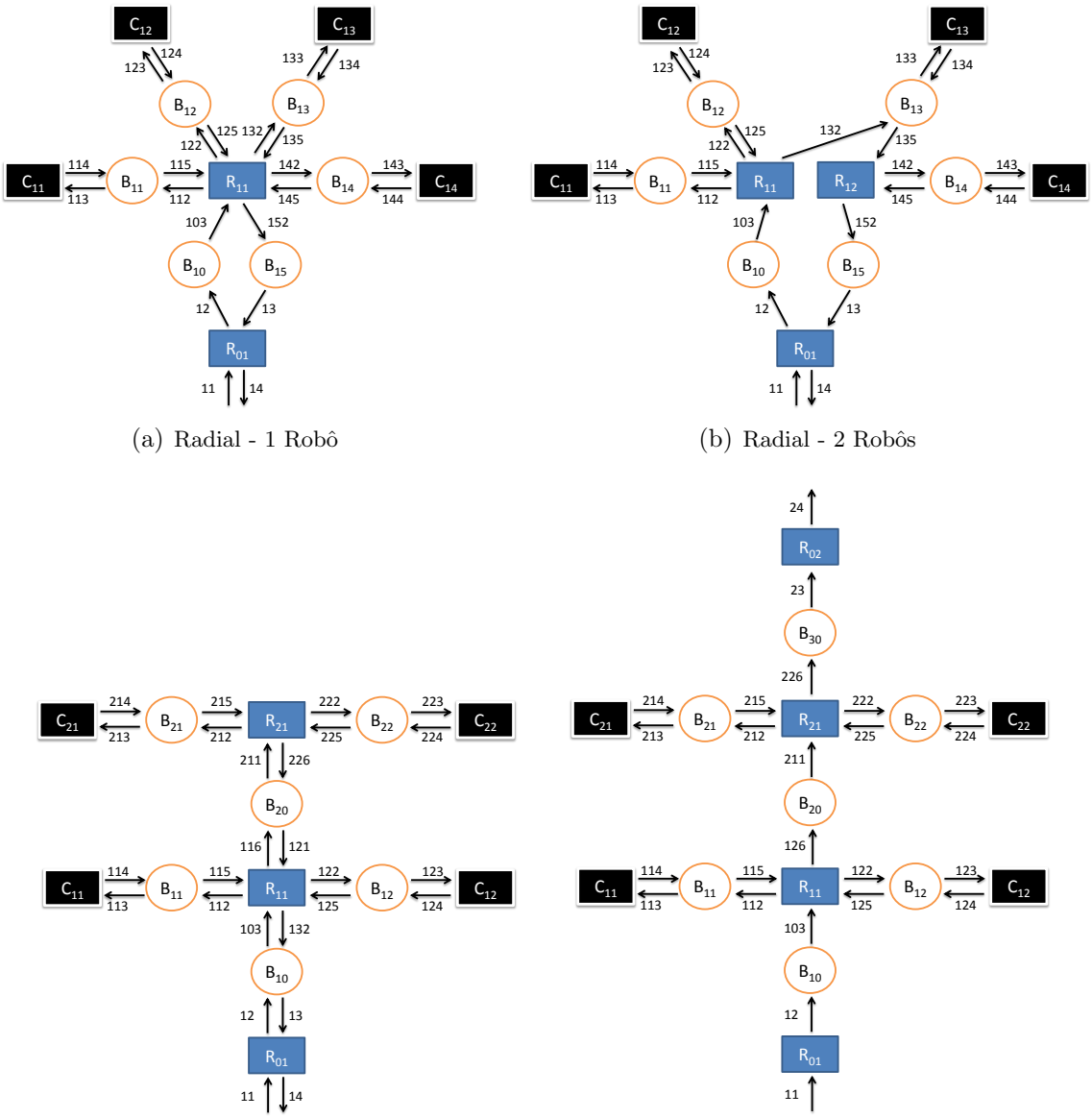


Figura 13 – Receitas

Na modelagem de cada layout as ações dos robôs e as operações nos *PMs* são representadas por eventos. Os eventos rotulados por números ímpares formam o conjunto de eventos controláveis Σ_c, e indicam o início de alguma tarefa, já aqueles rotulados por números pares formam o conjunto de eventos não controláveis Σ_u, e indicam o fim da tarefa correspondente. Deste modo, no MPT, a duração de cada atividade é a diferença temporal entre o evento controlável que indica seu início e o evento não-controlável que indica seu fim.

Para facilitar a compreensão, o rótulo do evento inicia-se com o número do equipamento correspondente. Por exemplo, os eventos 113 e 114 estão relacionados com a produção na câmara de processamento C_{11}, sendo 113 o evento controlável (ímpar) de início de produção e 114 o evento não controlável (par) de fim de produção.

Para ilustrar, os modelos dos layouts com 4 *PMs* são mostrados na Figura 14, com seus respectivos eventos identificados sobre as setas que indicam a movimentação do *wafer*. Os modelos de todos os layouts considerados são apresentados no Anexo A.



(a) Radial - 1 Robô

(b) Radial - 2 Robôs

(c) Linear - Entrada e Saída Única

(d) Linear - Entrada e Saída Separada

Figura 14 – Modelos para Layouts com 4 *PMs*

A sequência de eventos necessários para produção de um *wafer* em cada layout mostrado na Figura 14 é dada a seguir:

- Layout Radial - 1 Robô: 11, 12, 103, 112, 113, 114, 115, 122, 123, 124, 125, 132, 133, 134, 135, 142, 143, 144, 145, 152, 13, 14.
- Layout Radial - 2 Robôs: 11, 12, 103, 112, 113, 114, 115, 122, 123, 124, 125, 132, 133, 134, 135, 142, 143, 144, 145, 152, 13, 14.

- Layout Linear - Entrada e Saída Única: 11, 12, 103, 112, 113, 114, 115, 116, 211, 212, 213, 214, 215, 222, 223, 224, 225, 226, 121, 122, 123, 124, 125, 132, 13, 14.
- Layout Linear - Entrada e Saída Separada: 11, 12, 103, 112, 113, 114, 115, 122, 123, 124, 125, 126, 211, 212, 213, 214, 215, 222, 223, 224, 225, 226, 23, 24.

Inicialmente, para possibilitar a síntese de um supervisor capaz de controlar a planta de modo minimamente restritivo, foram obtidos os modelos em autômatos modificados dos equipamentos que compõem cada layout considerado, e das restrições a serem consideradas no funcionamento desejado. Como exemplo ilustrativo, os autômatos que modelam os componentes e as especificações com 4 *PMs* são mostrados na Figura 15 para o layout radial com 1 robô, na Figura 16 para o layout radial com 2 robôs, na Figura 17 para o layout linear com entrada e saída única e na Figura 18 para o layout linear com entrada e saída separada. Em cada estado q dos autômatos dos robôs e *PMs* foram incluídas o número de tarefas ativas i, de tal forma que cada estado é representado na forma de um estado expandido $W = (q, i)$. Conforme dito, os autômatos modificados das especificações não apresentam tarefas ativas. Os autômatos de todos os layouts considerados são apresentados no anexo B.

Nas modelos em autômatos mostrados nas Figuras 15 a 18 e no anexo B, o estado inicial dos autômatos dos robôs e *PMs* é o estado marcado, uma vez que este indica que alguma tarefa foi completada. No caso dos *PMs*, essa tarefa é o processamento do *wafer*, e para os robôs manipuladores, essa tarefa é o transporte do *wafer*. O autômato de cada PM possui apenas dois estados, que são o estado 0 (estado inicial e marcado), o qual indica que o PM está aguardando um wafer, e por isso tem 0 tarefas ativas, tal que o estado expandido é representado por $W = (0, 0)$, e o estado 1, que indica que o PM está em operação, e por isso tem 1 tarefa ativa, tal que $W = (1, 1)$. No caso do robô manipulador do *cluster tool*, o número de estados depende da quantidade de movimentos que o mesmo executa na movimentação do *wafer*. O autômato do robô foi implementado com a sequência que o robô deve executar na receita, de tal forma que, quando um robô pegar um *wafer*, o único evento possível é depositar o *wafer* na posição correta. Por exemplo, no autômato do robô R_{11} da Figura 15, o robô está inicialmente no estado 0, aguardando transporte, e por isso tem 0 tarefas ativas, tal que $W = (0, 0)$. Assim que o robô executa o evento 103, correspondente à pegar o *wafer* no buffer B_{10}, indo para o estado 1 com 1 tarefa ativa, tal que $W = (1, 1)$, ele pode executar apenas o evento 112, depositando-o no buffer B_{11} e voltando para o estado $W = (0, 0)$. O mesmo ocorre sempre que o robô pegar um *wafer* em diferentes equipamentos do *cluster tool*. Como os robôs manipuladores podem mover o *wafer* para vários pontos dentro do *cluster tool* , o autômato correspondente geralmente apresenta um maior número de estados quando comparado ao autômato do *PM*. As características descritas se mantêm em todos os layouts.

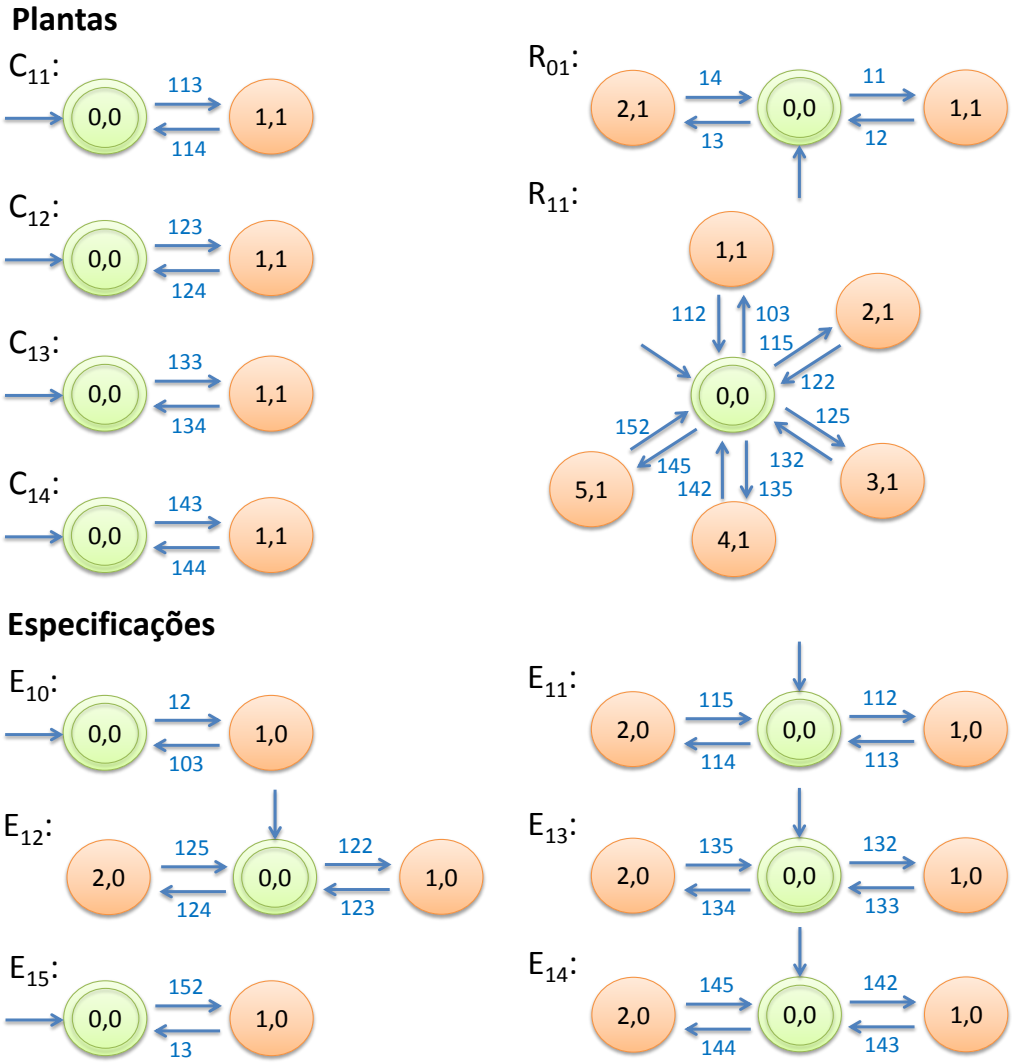


Figura 15 – Autômatos Layout Radial - 1 Robô

Os autômatos das restrições são responsáveis por estabelecer as regras de produção a serem respeitadas, e evitar o *overflow* e *underflow* nos equipamentos do *cluster tool*. Dessa forma, os autômatos das especificações evitam que *wafers* sejam retirados do PM antes de serem processados, que o mesmo *wafer* seja processado mais de uma vez e que mais de um *wafer* seja colocado no PM. Como exemplo, no autômato da especificação E_{11} da Figura 15, quando ocorre o evento 112, indicando que o robô depositou um *wafer* no buffer B_{11}, o autômato estará no estado 1, e o único evento possível então é 113, indicando que C_{11} foi ligado, e o autômato então volta ao estado inicial 0. Da mesma forma, quando acontece o evento 114, indicando que o *wafer* está no buffer B_{11} após o fim do processamento em C_{11}, levando o autômato ao estado 2, o único evento possível é 115, que representa a retirada do *wafer* pelo robô manipulador. Podemos ver que em todos os estados dos autômatos das especificações o número de tarefas ativas indicado em cada estado expandido é sempre 0, tal que $W = (q, 0)$. Isso ocorre pois as especificações não realizam trabalho. Estas características descritas, assim como no caso dos autômatos das

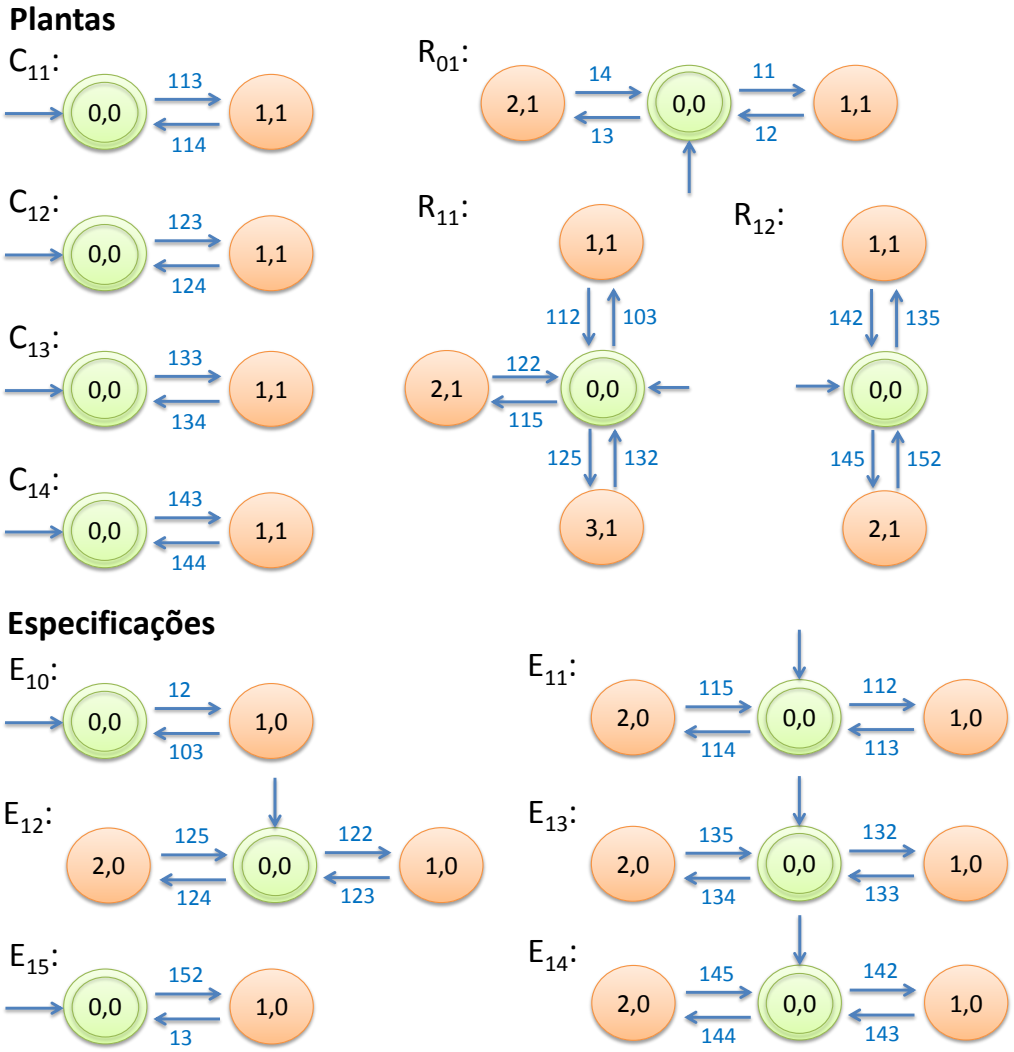


Figura 16 – Autômatos Layout Radial - 2 Robôs

plantas, se mantêm em todos os layouts.

4.2 Síntese de Supervisores

Uma vez obtidos os modelos em autômatos da planta e das especificações, utilizou-se a TCS para calcular os supervisores monolíticos de cada um dos layouts apresentados. O supervisor é responsável pelo controle lógico dos equipamentos no *cluster tool*, garantindo o comportamento desejado de modo e livre de bloqueio, atuando sobre a planta de modo minimamente restritivo . Esse cálculo foi feito utilizando-se a ferramenta computacional UltraDES (ALVES; MARTINS; PENA, 2017). Inseriu-se os modelos em autômatos dos *PMs*, robôs, além das restrições representadas pelos buffers, no UltraDES, que calculou e retornou o supervisor monolítico para cada layout mostrado.

Foram feitas 30 execuções do cálculo do supervisor de cada layout para determinar o tempo médio de sua execução, e seu respectivo intervalo de confiança (IC) para um nível

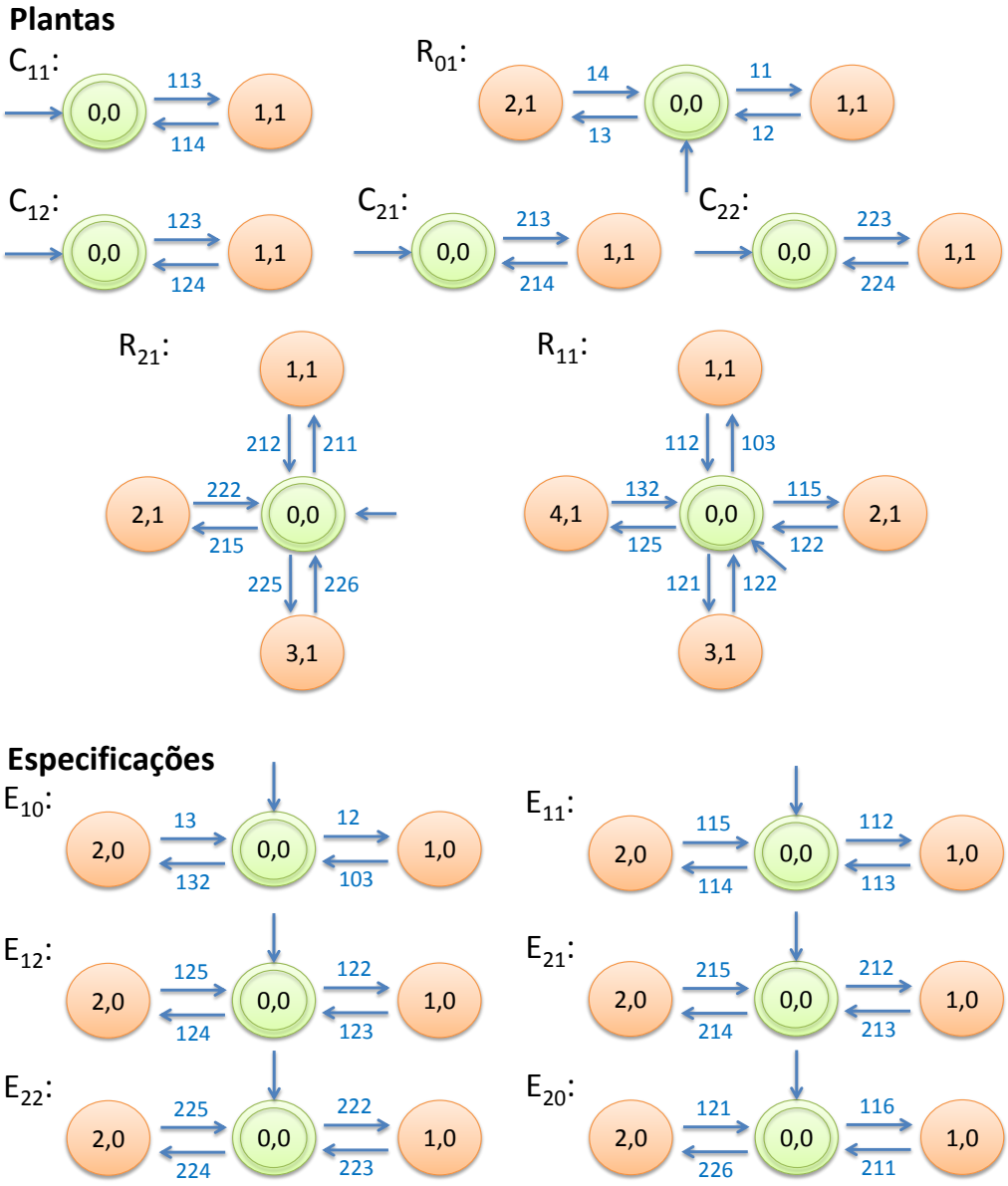


Figura 17 – Autômatos Layout Linear - Entrada e Saída Única

de significância $\alpha = 0,05$. Estas foram feitas de forma aleatória para cada layout, para redução do erro experimental.

O número de estados, transições, o tempo médio de execução do cálculo dos supervisores e o respectivo IC foram registrados na Tabela 1 para os layouts com 4 *PMs*, na Tabela 2 para os layouts com 5 *PMs* e na Tabela 3 para os layouts com 6 *PMs*. O autômato do supervisor é omitido devido ao grande número de estados e transições que ele possui, conforme é observado na tabela.

É objetivo do trabalho comparar o desempenho de diferentes layouts. No entanto, cada layout demanda um número diferente de equipamentos. Apesar da receita ser a mesma em todos eles, devido à diferença entre o número de equipamentos, observa-se uma diferença também no número de eventos de uma sequência para produzir um mesmo lote

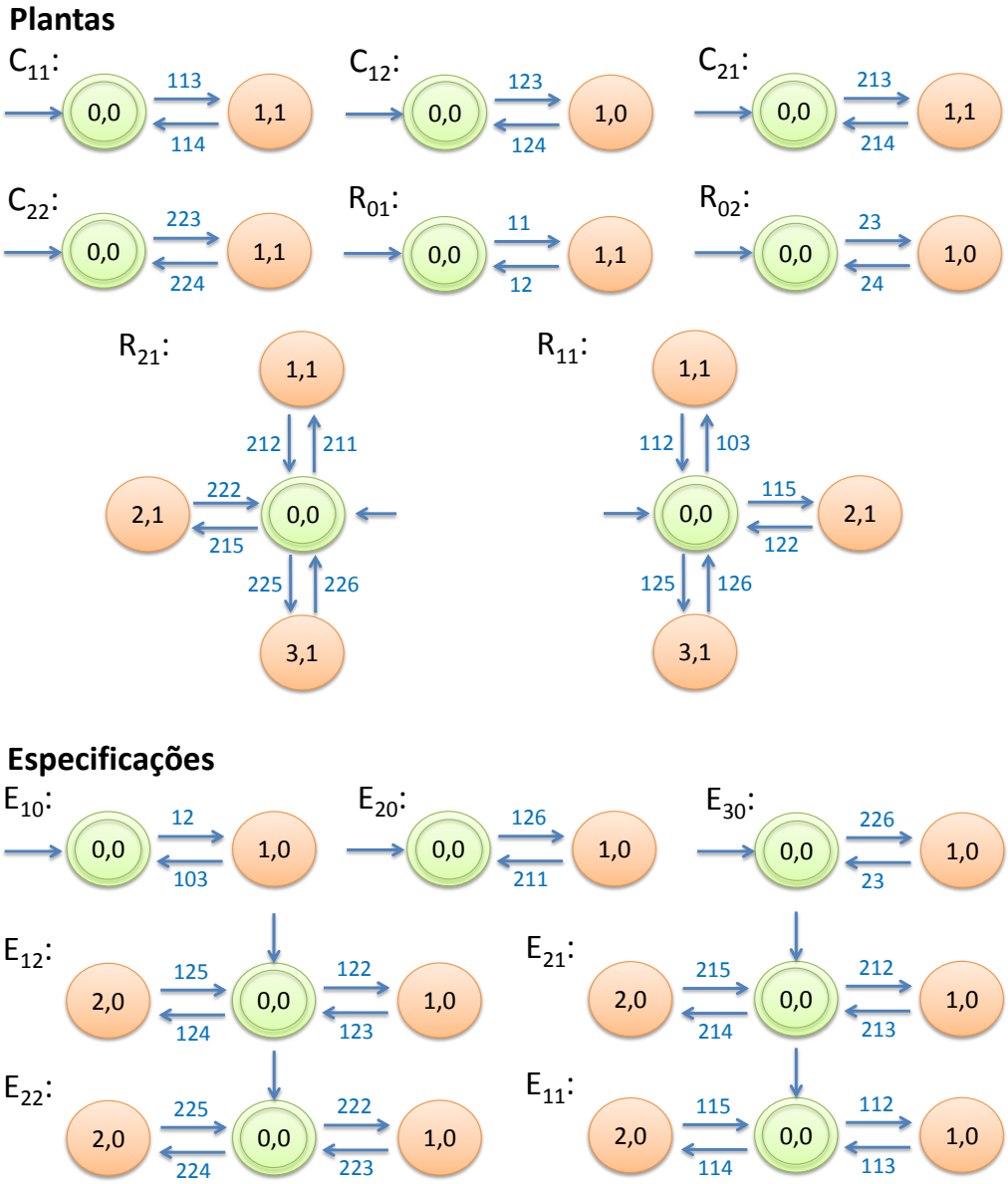


Figura 18 – Autômatos Layout Linear - Entrada e Saída Separada

Tabela 1 – Estados, Transições, Tempo Médio de Cálculo dos Supervisores e Intervalo de Confiança - Layout com 4 PMs

	Supervisor			
Layout	Estados	Transições	Tempo	IC
Radial 1 Robô	3.855	12.951	0,13 s	0,007 s
Radial 2 Robôs	4.684	16.381	0,16 s	0,008 s
Linear Entrada e Saída Única	3.255	9.715	0,16 s	0,010 s
Linear Entrada e Saída Separada	12.696	51.470	0,31 s	0,015 s

Tabela 2 – Estados, Transições, Tempo Médio de Cálculo dos Supervisores e Intervalo de Confiança - Layout com 5 PMs

	Supervisor			
Layout	Estados	Transições	Tempo	IC
Radial 1 Robô	16.141	62.691	0,57 s	0,043 s
Radial 2 Robôs	19.732	79.511	0,67 s	0,029 s
Linear Entrada e Saída Única	28.998	107.237	2,09 s	0,066 s
Linear Entrada e Saída Separada	139.655	679.192	4,54 s	0,088 s

Tabela 3 – Estados, Transições, Tempo Médio de Cálculo dos Supervisores e Intervalo de Confiança - Layout com 6 PMs

	Supervisor			
Layout	Estados	Transições	Tempo	IC
Radial 1 Robô	67.366	296.899	1,66 s	0,063 s
Radial 2 Robôs	82.802	377.545	2,31 s	0,068 s
Linear Entrada e Saída Única	174.983	746.076	10,25 s	0,294 s
Linear Entrada e Saída Separada	584.016	3.150.356	20,96 s	0,375 s

de *wafers*. Como exemplo, a Tabela 4 mostra o número de equipamentos e a quantidade de eventos necessários para produzir um *wafer* em cada layout para 4 *PMs*.

Tabela 4 – Número de eventos e equipamentos para produção de 1 *wafer* para 4 PMs

Layout	Ev.	Equip.
Radial - 1 Robô	22	6
Radial - 2 Robôs	22	7
Linear - E/S Única	26	7
Linear - E/S Separada	24	8

Estas informações serão utilizadas no cálculo do Paralelismo Acumulado Relativo (PAR), que é uma das métricas escolhidas para comparação dos layouts.

5 Aplicação do MPT ao problema

Nesta seção, é detalhada a aplicação do algoritmo de Máximo Paralelismo com Restrições Temporais (MPT) aos supervisores encontrados, além da utilização dos seus resultados para comparar o desempenho dos diferentes layouts estudados neste trabalho.

Para aplicar o MPT nos supervisores encontrados em cada layout, é necessário determinar o tempo de duração das tarefas no *cluster tool* , além de avaliar o tempo até que determinado evento ocorra em algum estado do supervisor.

Neste trabalho, consideramos que os eventos ocorrem de maneira instantânea. Assim, o tempo de duração das tarefas pode ser determinado pela diferença Δt entre o tempo t_1 do evento σ_1 que representa seu início e o respectivo tempo t_2 do evento σ_2 que representa seu fim, tal que $\Delta t = t_2 - t_1$. Como exemplo, se estamos interessados em saber o tempo de processamento de um PM, podemos verificar a diferença temporal entre o evento associado ao início da tarefa no PM e o evento associado ao fim da tarefa no PM.

Para avaliar o tempo até a ocorrência de um evento, dada uma sequência s no supervisor, é utilizada a função temporal, implementada por meio de simulação de sistemas a eventos discretos.

O tempo associado ao processamento do *wafer* em cada PM é um dos parâmetros alterados em cada layout, ou seja, ele varia entre 1 e 200 segundos. Cada valor utilizado é aplicado em todos os PMs do *cluster tool* de modo que tenham o mesmo tempo de processamento.

O tempo de movimentação dos robôs, que também foi considerado, é função do ângulo percorrido e da temperatura do *wafer* a ser transportado. O procedimento para derivação desse tempo é mostrado a seguir.

5.1 Tempo de Movimentação do Robô

Neste trabalho, para calcular os tempos de movimentação dos robôs manipuladores, foram utilizados os mesmos valores descritos por Yi et al. (2007) para os movimentos de 90° e 180°, comuns no *cluster tool* linear. O tempo de rotação T_R do robô é função do ângulo do movimento e do índice K que indica a temperatura do *wafer* transportado. O valor de K é dado por:

$$K = \begin{cases} 0 & \text{se} \quad \text{não há } \textit{wafer} \text{ no robô} \\ 1 & \text{se} \quad \textit{wafer} \text{ está quente} \\ 2 & \text{se} \quad \textit{wafer} \text{ está frio.} \end{cases}$$

Conforme Yi et al. (2007), os tempos de rotação T_R para 90° e 180° são dados por:

$$T_R(K, 90°) = \begin{cases} 0,55s & \text{para} \quad K = 0 \\ 1,11s & \text{para} \quad K = 1 \\ 0,68s & \text{para} \quad K = 2 \end{cases}$$

$$T_R(K, 180°) = \begin{cases} 0,82s & \text{para} \quad K = 0 \\ 1,7s & \text{para} \quad K = 1 \\ 1,03s & \text{para} \quad K = 2 \end{cases}$$

Já o tempo T_P para pegar/depositar o *wafer* no *PM* e o tempo T_L para pegar/depositar o *wafer* nos *load locks* é função apenas da temperatura do *wafer* transportado:

$$T_P(K) = \begin{cases} 2,76s & \text{para} \quad K = 1 \\ 2,1s & \text{para} \quad K = 2 \end{cases}$$

$$T_L(K) = \begin{cases} 2,27s & \text{para} \quad K = 1 \\ 1,75s & \text{para} \quad K = 2 \end{cases}$$

No caso dos layouts radiais, tem-se ângulos de rotação dos robôs manipuladores diferentes de 90° e 180°. Para determinar o tempo de rotação do robô para esses ângulos de movimento diferentes, utilizou-se uma regressão polinomial com os dados anteriores para encontrar a função que relaciona o tempo de rotação (T_R) com o ângulo (θ).

Para o polinômio quadrático do tipo $T_R(\theta) = A\theta^2 + B\theta + C$, obteve-se o valor de erro médio quadrático (*MSE*) igual a 0 para duas casas decimais do resultado, e os coeficientes A, B e C mostrados na Tabela 5 para cada valor de K.

Tabela 5 – Coeficientes do polinômio

Polinômio: $T_R(\theta) = A\theta^2 + B\theta + C$			
K	A	B	C
0	$-1,728 \times 10^{-5}$	0,0076667	0
1	$-3,210 \times 10^{-5}$	0,015222	0
2	$-2,037 \times 10^{-5}$	0,0093889	0

Utilizou-se $K = 0$ quando o robô se movimenta para pegar o *wafer*, e quando retorna à posição inicial. Considerou-se que o *wafer* sempre está quente após qualquer operação no PM em todos os layouts, já que este representa o caso de maior tempo de movimentação para o robô manipulador. Assim, nestas situações, adotou-se $K = 1$ para os tempos de rotação e os tempos de pegar/depositar o *wafer*. Utilizou-se $K = 2$ somente quando o *wafer* é manipulado pelo robô ao entrar no *cluster tool*, antes de passar por qualquer PM.

O tempo de operação total do robô é a soma dos tempos de deslocamento até a posição do *wafer*, o tempo para pegá-lo, executar o movimento de rotação até o destino, colocar o *wafer* e retornar à posição inicial, sendo que cada um desses tempos, dados por T_R, T_P e T_L, devem ser calculados considerando o respectivo parâmetro K. Considerou-se que o robô sempre volta à posição inicial após depositar o *wafer*.

Assim, os tempos de duração de cada tarefa são mostrados nas Tabelas 6, 7, 8 e 9, para os layouts radial com 1 robô, radial com 2 robôs, linear com entrada e saída única e linear com entrada e saída separadas, respectivamente. O tempo dos eventos relacionados à tarefa do processamento do *wafer* foi representado por $*$, já que este é um parâmetro variável.

Tabela 6 – Tempo de duração dos eventos para o layout radial com 1 robô

4 PMs		5 PMs		6 PMs	
$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração
$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s
$103 \rightarrow 112$	4,74 s	$103 \rightarrow 112$	4,54 s	$103 \rightarrow 112$	4,54 s
$113 \rightarrow 114$	*	$113 \rightarrow 114$	*	$113 \rightarrow 114$	*
$115 \rightarrow 122$	7,39 s	$115 \rightarrow 122$	7,00 s	$115 \rightarrow 122$	7,00 s
$123 \rightarrow 124$	*	$123 \rightarrow 124$	*	$123 \rightarrow 124$	*
$125 \rightarrow 132$	7,81 s	$125 \rightarrow 132$	7,41 s	$125 \rightarrow 132$	7,41 s
$133 \rightarrow 134$	*	$133 \rightarrow 134$	*	$133 \rightarrow 134$	*
$135 \rightarrow 142$	7,81 s	$135 \rightarrow 142$	7,68	$135 \rightarrow 142$	7,68 s
$143 \rightarrow 144$	*	$143 \rightarrow 144$	*	$143 \rightarrow 144$	*
$145 \rightarrow 152$	6,90 s	$145 \rightarrow 152$	7,68 s	$145 \rightarrow 152$	7,68 s
$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s
-	-	$155 \rightarrow 162$	7,00 s	$155 \rightarrow 162$	7,41 s
-	-	$153 \rightarrow 154$	*	$153 \rightarrow 154$	*
-	-	-	-	$165 \rightarrow 172$	5,99 s
-	-	-	-	$163 \rightarrow 164$	*

5.2 Métricas Utilizadas

Além da sequência de maior paralelismo, o algoritmo MPT retorna também o *makespan*, que é o tempo total de produção do lote, e o Paralelismo Acumulado (PA) da

Tabela 7 – Tempo de duração dos eventos para o layout radial com 2 robôs

4 PMs		5 PMs		6 PMs	
$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração
$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s
$103 \rightarrow 112$	4,74 s	$103 \rightarrow 112$	4,54 s	$103 \rightarrow 112$	4,54 s
$113 \rightarrow 114$	*	$113 \rightarrow 114$	*	$113 \rightarrow 114$	*
$115 \rightarrow 122$	7,39 s	$115 \rightarrow 122$	7,00 s	$115 \rightarrow 122$	7,00 s
$123 \rightarrow 124$	*	$123 \rightarrow 124$	*	$123 \rightarrow 124$	*
$125 \rightarrow 132$	7,81 s	$125 \rightarrow 132$	7,41 s	$125 \rightarrow 132$	7,41 s
$133 \rightarrow 134$	*	$133 \rightarrow 134$	*	$133 \rightarrow 134$	*
$135 \rightarrow 142$	6,72 s	$135 \rightarrow 142$	7,68	$135 \rightarrow 142$	7,68 s
$143 \rightarrow 144$	*	$143 \rightarrow 144$	*	$143 \rightarrow 144$	*
$145 \rightarrow 152$	6,90 s	$145 \rightarrow 152$	7,41 s	$145 \rightarrow 152$	7,41 s
$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s
-	-	$155 \rightarrow 162$	6,69 s	$155 \rightarrow 162$	7,00 s
-	-	$153 \rightarrow 154$	*	$153 \rightarrow 154$	*
-	-	-	-	$165 \rightarrow 172$	5,96 s
-	-	-	-	$163 \rightarrow 164$	*

Tabela 8 – Tempo de duração dos eventos para o layout linear com entrada e saída única

4 PMs		5 PMs		6 PMs	
$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração
$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s
$103 \rightarrow 112$	5,08 s	$103 \rightarrow 112$	5,08 s	$103 \rightarrow 112$	5,08 s
$113 \rightarrow 114$	*	$113 \rightarrow 114$	*	$113 \rightarrow 114$	*
$115 \rightarrow 116$	7,51 s	$115 \rightarrow 116$	7,51 s	$115 \rightarrow 116$	7,51 s
$211 \rightarrow 212$	6,69 s	$211 \rightarrow 212$	6,69 s	$211 \rightarrow 212$	6,69 s
$213 \rightarrow 214$	*	$213 \rightarrow 214$	*	$213 \rightarrow 214$	*
$215 \rightarrow 222$	8,32 s	$215 \rightarrow 216$	7,51 s	$215 \rightarrow 216$	7,51 s
$223 \rightarrow 224$	*	$223 \rightarrow 224$	*	$223 \rightarrow 224$	*
$225 \rightarrow 226$	6,69 s	$225 \rightarrow 226$	6,69 s	$225 \rightarrow 226$	6,69 s
$121 \rightarrow 122$	7,51 s	$121 \rightarrow 122$	7,51 s	$121 \rightarrow 122$	7,51 s
$123 \rightarrow 124$	*	$123 \rightarrow 124$	*	$123 \rightarrow 124$	*
$125 \rightarrow 132$	6,69 s	$125 \rightarrow 132$	6,69 s	$125 \rightarrow 132$	6,69 s
$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s	$13 \rightarrow 14$	7,06 s
-	-	$311 \rightarrow 312$	6,69 s	$311 \rightarrow 312$	6,69 s
-	-	$313 \rightarrow 314$	*	$313 \rightarrow 314$	*
-	-	$315 \rightarrow 316$	6,69 s	$315 \rightarrow 322$	8,32 s
-	-	$221 \rightarrow 222$	7,51 s	$221 \rightarrow 222$	7,51 s
-	-	-	-	$323 \rightarrow 324$	*
-	-	-	-	$325 \rightarrow 326$	6,69 s

sequência, dado pela função acumulativa de tarefas ativas.

Para efeito de comparação de desempenho dos layouts, o valor do Paralelismo Acumulado (PA) de cada sequência não caracteriza uma boa métrica, já que seu valor

Tabela 9 – Tempo de duração dos eventos para o layout linear com entrada e saída separada

4 PMs		5 PMs		6 PMs	
$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração	$\sigma_1 \rightarrow \sigma_2$	Duração
$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s	$11 \rightarrow 12$	5,35 s
$103 \rightarrow 112$	5,08 s	$103 \rightarrow 112$	5,08 s	$103 \rightarrow 112$	5,08 s
$113 \rightarrow 114$	*	$113 \rightarrow 114$	*	$113 \rightarrow 114$	*
$115 \rightarrow 122$	8,32 s	$115 \rightarrow 122$	8,32 s	$115 \rightarrow 122$	8,32 s
$211 \rightarrow 212$	6,69 s	$211 \rightarrow 212$	6,69 s	$211 \rightarrow 212$	6,69 s
$213 \rightarrow 214$	*	$213 \rightarrow 214$	*	$213 \rightarrow 214$	*
$215 \rightarrow 222$	8,32 s	$215 \rightarrow 222$	8,32 s	$215 \rightarrow 222$	8,32 s
$223 \rightarrow 224$	*	$223 \rightarrow 224$	*	$223 \rightarrow 224$	*
$225 \rightarrow 226$	7,51 s	$225 \rightarrow 226$	7,51 s	$225 \rightarrow 226$	7,51 s
$123 \rightarrow 124$	*	$123 \rightarrow 124$	*	$123 \rightarrow 124$	*
$125 \rightarrow 126$	7,51 s	$125 \rightarrow 126$	7,51 s	$125 \rightarrow 126$	7,51 s
$23 \rightarrow 24$	7,06 s	$23 \rightarrow 24$	7,06 s	$23 \rightarrow 24$	7,06 s
-	-	$311 \rightarrow 312$	6,69 s	$311 \rightarrow 312$	6,69 s
-	-	$313 \rightarrow 314$	*	$313 \rightarrow 314$	*
-	-	$315 \rightarrow 316$	7,51 s	$315 \rightarrow 322$	8,32 s
-	-	-	-	$323 \rightarrow 324$	*
-	-	-	-	$325 \rightarrow 326$	7,51 s

é tanto maior quanto maior o número de eventos ou maior o número de equipamentos para produzir o mesmo *wafer*. Dessa forma, um novo parâmetro denominado Paralelismo Acumulado Relativo (PAR) é proposto, dado por (5.1), para comparar os layouts. Este parâmetro normaliza o Paralelismo Acumulado (PA) de acordo com o número de eventos da sequência para produzir um *wafer* (Ev) e equipamentos do layout ($Equip$).

$$PAR = \frac{PA}{Ev \times Equip} \tag{5.1}$$

Assim, para comparação do desempenho de cada layout, variando o número de PMs, o seu tempo de processamento e o número de *wafers* por lote, utilizaremos como métricas o PAR e o *makespan*.

O PAR é usado como um indicador de eficiência do layout, já que, o layout com maior valor para esse parâmetro produz a mesma quantidade de produtos com um menor número de equipamentos, ou com menor número de eventos.

Já o *makespan* é usado como um indicador de velocidade de produção, uma vez que, comparando dois layouts diferentes, aquele com menor valor de *makespan* irá produzir o lote de *wafers* em menor tempo.

5.3 Tempo de Computação do Algoritmo MPT

Para medição do tempo de computação do algoritmo de escalonamento MPT, foram feitas 30 execuções deste em cada configuração do layout, de forma aleatória, e calculou-se o tempo médio de execução e seu respectivo IC para um nível de significância $\alpha = 0,05$. Os resultados são apresentados na Tabela 10 para um lote de 50 *wafers*, na Tabela 11 para um lote de 25 *wafers*, na Tabela 12 para um lote de 12 *wafers* e na Tabela 13 para um lote de 6 *wafers*.

Tabela 10 – Tempo Médio de Execução do MPT para lotes de 50 *wafers*

Layout	4 PMs		5 PMs		6 PMs	
	Tempo Médio	IC	Tempo Médio	IC	Tempo Médio	IC
Radial 1 Robô	1,73 s	0,183 s	7,05 s	0,261 s	42,39 s	2,077 s
Radial 2 Robôs	1,88 s	0,109 s	8,01 s	0,177 s	55,77 s	2,538 s
Linear Entrada e Saída Única	1,31 s	0,102 s	12,97 s	0,210 s	132,00 s	6,043 s
Linear Entrada e Saída Separada	5,86 s	0,327 s	100,51 s	1,190 s	760,74 s	21,360 s

Tabela 11 – Tempo Médio de Execução do MPT para lotes de 25 *wafers*

Layout	4 PMs		5 PMs		6 PMs	
	Tempo Médio	IC	Tempo Médio	IC	Tempo Médio	IC
Radial 1 Robô	0,84 s	0,048 s	5,31 s	0,270 s	22,83 s	0,979 s
Radial 2 Robôs	0,95 s	0,062 s	5,97 s	0,204 s	29,29 s	0,882 s
Linear Entrada e Saída Única	0,71 s	0,047 s	8,53 s	0,223 s	67,76 s	2,331 s
Linear Entrada e Saída Separada	2,02 s	0,092 s	57,13 s	1,972 s	335,46 s	12,969 s

Observa-se que o layout linear com entrada e saída separada tem o maior tempo de execução para cálculo do MPT em todos os casos. Isso é devido ao tamanho do supervisor para este layout, já que, sendo o maior dentre os 4 layouts considerados, o algoritmo de busca leva mais tempo para encontrar uma solução.

Tabela 12 – Tempo Médio de Execução do MPT para lotes de 12 *wafers*

Layout	4 PMs		5 PMs		6 PMs	
	Tempo Médio	IC	Tempo Médio	IC	Tempo Médio	IC
Radial 1 Robô	0,54 s	0,034 s	2,13 s	0,081 s	11,07 s	0,307 s
Radial 2 Robôs	0,63 s	0,036 s	2,63 s	0,038 s	14,56 s	0,391 s
Linear Entrada e Saída Única	0,46 s	0,026 s	4,12 s	0,105 s	33,81 s	0,703 s
Linear Entrada e Saída Separada	1,78 s	0,088 s	26,45 s	0,405 s	153,17	4,962 s

Tabela 13 – Tempo Médio de Execução do MPT para lotes de 6 *wafers*

Layout	4 PMs		5 PMs		6 PMs	
	Tempo Médio	IC	Tempo Médio	IC	Tempo Médio	IC
Radial 1 Robô	0,30 s	0,011 s	1,63 s	0,085 s	6,74 s	0,146 s
Radial 2 Robôs	0,35 s	0,012 s	2,265 s	0,205 s	8,671 s	0,136 s
Linear Entrada e Saída Única	0,29 s	0,030 s	3,46 s	0,282 s	20,91 s	0,374 s
Linear Entrada e Saída Separada	1,09 s	0,024 s	18,14 s	0,696 s	81,47	2,022 s

6 Análise dos Resultados

Neste capítulo são apresentados os resultados obtidos pela aplicação do algoritmo MPT em cada configuração considerada do *cluster tool*.

Esse resultados são expressos na forma de gráficos para as duas métricas de desempenho: o *makespan* e o PAR. Em cada gráfico são agrupadas as informações dos 4 diferentes layouts para cada variação de número de *wafers* no lote e quantidade de *PMs* no *cluster tool*.

6.1 Resultados Experimentais

Nesta seção, apresentamos os resultados obtidos pela aplicação do algoritmo MPT em cada layout, variando o tempo de processamento dos PMs, o tamanho do lote de produção e o número de PMs no *cluster tool* .

Os algoritmos foram executados em um notebook com processador Intel Core I5-3210M, 12 GB de memória RAM e sistema operacional de 64 bits. Em todos os experimentos apresentados, para obter o tempo médio de execução e os respectivos intervalos de confiança, foram feitas 30 execuções de forma aleatória, considerando um nível de significância $\alpha = 0{,}05$, e alterando o tempo de processamento do PM em cada um dos 30 valores considerados entre 1 e 200 s.

6.1.1 Paralelismo Acumulado Relativo - PAR

O MPT executa uma busca sobre o supervisor, restringindo-o às sequências temporalmente factíveis que produzem o lote de tamanho desejado. Dessa forma, essa busca foi executada para lotes de 6, 12, 25 e 50 *wafers*, que são os tamanhos de lotes considerados. Para cada tamanho de lote, o algoritmo é aplicado sobre os quatro layouts. Uma das saídas do algoritmo MPT é o Paralelismo Acumulado da sequência obtida, a partir do qual obtemos o Paralelismo Acumulado Relativo por meio da equação (5.1). O PAR da melhor sequência obtida é apresentado em função dos tempos de PM, que variam entre 1 e 200 s, da Figura 19 até a Figura 21 para o lote de 6 *wafers*, e no Anexo C para lotes de 12, 25 e 50 *wafers*.

6.1.2 *Makespan*

Da mesma forma apresentada para o PAR, essas buscas foram executadas para lotes de 6, 12, 25 e 50 *wafers*, nos quatro layouts considerados, em função dos tempos de

PM, que variam entre 1 e 200 s. Assim, obteve-se outra saída do algoritmo MPT, que é o *Makespan*, retornado em cada execução.

A partir da Figura 22, até a Figura 24, é apresentada a evolução do *makespan* em função do tempo do *PM*, para o lote de 6 *wafers*. Para lotes de 12, 25 e 50 *wafers*, as figuras são mostradas no Anexo D.

6.2 Análise de Resultados

Quando o tempo do *PM* é menor que o tempo de movimentação do robô, este último sempre estará em operação, o que levará os *wafers* processados a aguardar pelo transporte. Nesta situação, o *cluster tool* é denominado *robot bound*, uma vez que o gargalo do *cluster tool* é o robô.

Por este motivo, observa-se nas Figuras 19 até 21 e no Anexo C que para tempos de processamento de até 15 s com 4 PMs, o layout radial com 2 robôs apresenta maior valor de PAR em todos os tamanhos de lotes de produção, o que significa que é o layout com maior número de operações em paralelo, considerando o número de equipamentos e eventos, ou seja, o mais eficiente do ponto de vista do paralelismo nestas condições. Porém, essa característica se altera à medida que aumentamos o número de PMs. Para 5 PMs temos o layout linear com entrada e saída única e para 6 PMs, o layout linear com entrada e saída separada, como aquele que apresenta maior valor de PAR, considerando os tempos de processamento menores que 15 s. Os layouts linear com entrada e saída única e linear com entrada e saída separada apresentam valores de PAR próximos para tempos do *PM* até 5 s em todas as condições, indicando que a eficiência destes layouts é aproximadamente a mesma nestes casos.

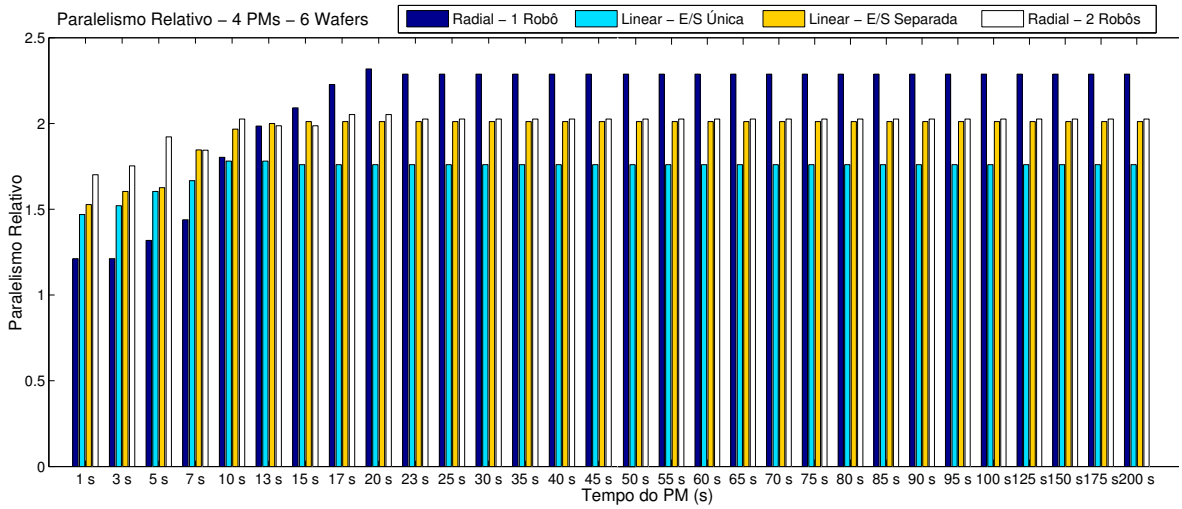


Figura 19 – Paralelismo Relativo: 4 PMs / lote de 6 *wafers*

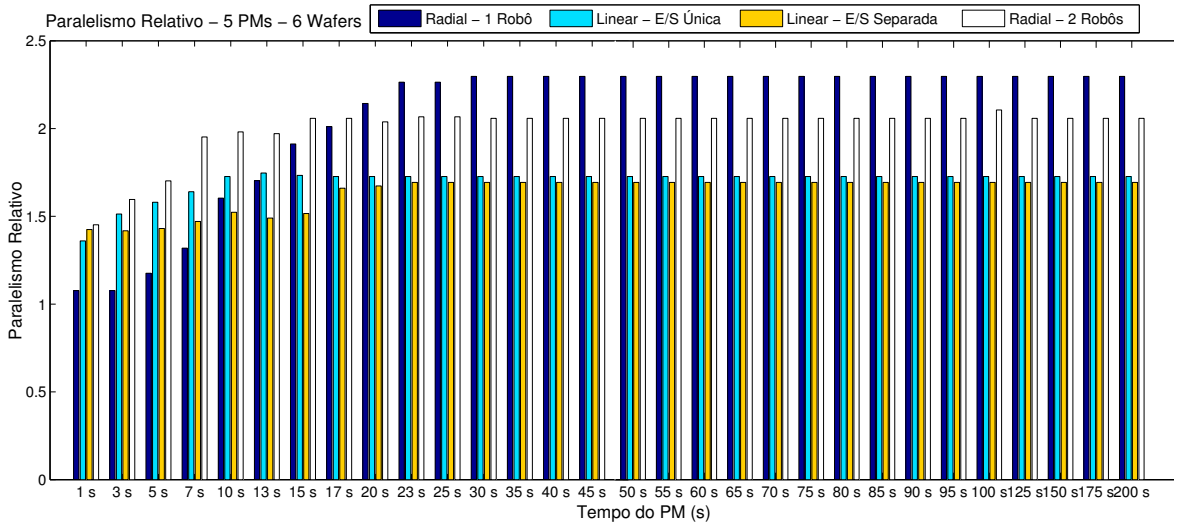


Figura 20 – Paralelismo Relativo: 5 PMs / lote de 6 *wafers*

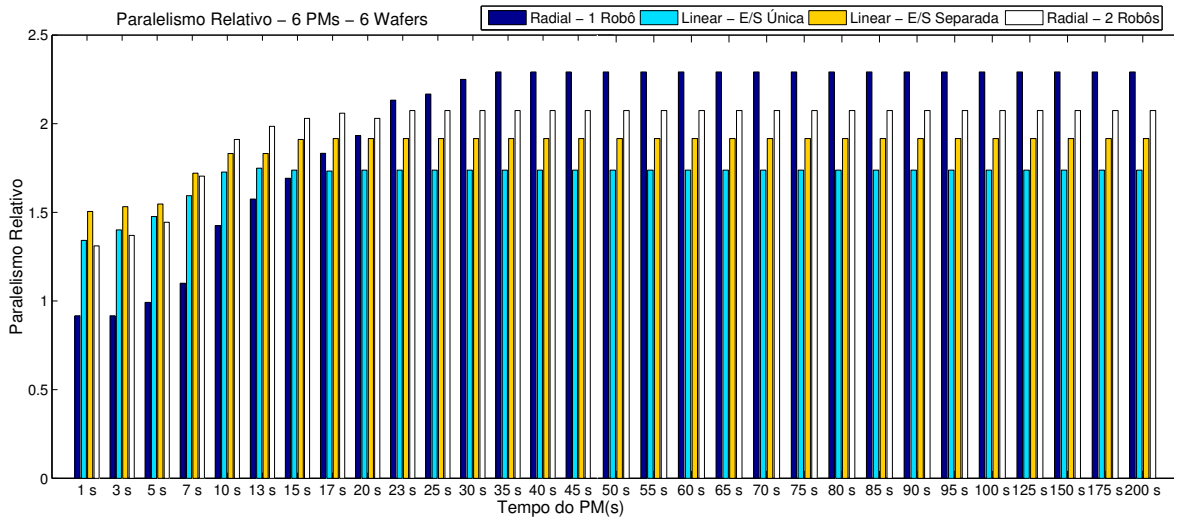


Figura 21 – Paralelismo Relativo: 6 PMs / lote de 6 *wafers*

Para tempos dos *PMs* maiores ou iguais a 15 s, o tempo de movimentação do robô é menor que o tempo de processamento do *wafer*, e o robô ficará ocioso aguardando o fim do processamento dos *wafers*. Assim, o *cluster tool* é denominado *process bound*, uma vez que o *PM* se torna o gargalo.

Observa-se que o layout radial com 1 robô aumenta o valor de PAR com o aumento do tempo de processamento do *PM*. Em todas as condições para valores do *PM* maiores que 25 s, esse layout passa a ter o maior valor de PAR, indicando ser o layout mais eficiente do ponto de vista do paralelismo, enquanto o layout radial com 2 robôs terá o segundo maior valor de PAR. Observa-se também que os layouts linear com entrada e saída separada e o layout linear com entrada e saída única apresentam os menores valores de PAR. Isso ocorre porque a utilização de dois robôs no layout radial ou dois EFEMs no layout linear não traz um paralelismo maior à sequência, já que os equipamentos adicionais ficam ociosos aguardando o processamento no *PM*.

Para tempos dos *PMs* maiores que 25 s, percebe-se que os valores de PAR de todos os layouts se mantêm constantes, pois o aumento do tempo do *PM* apenas aumenta o tempo de espera do robô.

Nas Figuras 22 até 24 e no Anexo D, verifica-se que para tempos de processamento maiores que 10 s, o layout radial com 2 robôs apresenta o menor *makespan*, sendo que a diferença entre este e o layout radial com 1 robô é pequena. Para tempos dos *PMs* menores ou iguais a 13 s, quando o processo é *robot bound*, o layout radial com 1 robô tem o pior desempenho em termos de *makespan*, o que pode ser percebido no detalhe superior das figuras. Isso se justifica pelo fato do robô ser o gargalo do processo para estes tempos de *PM*.

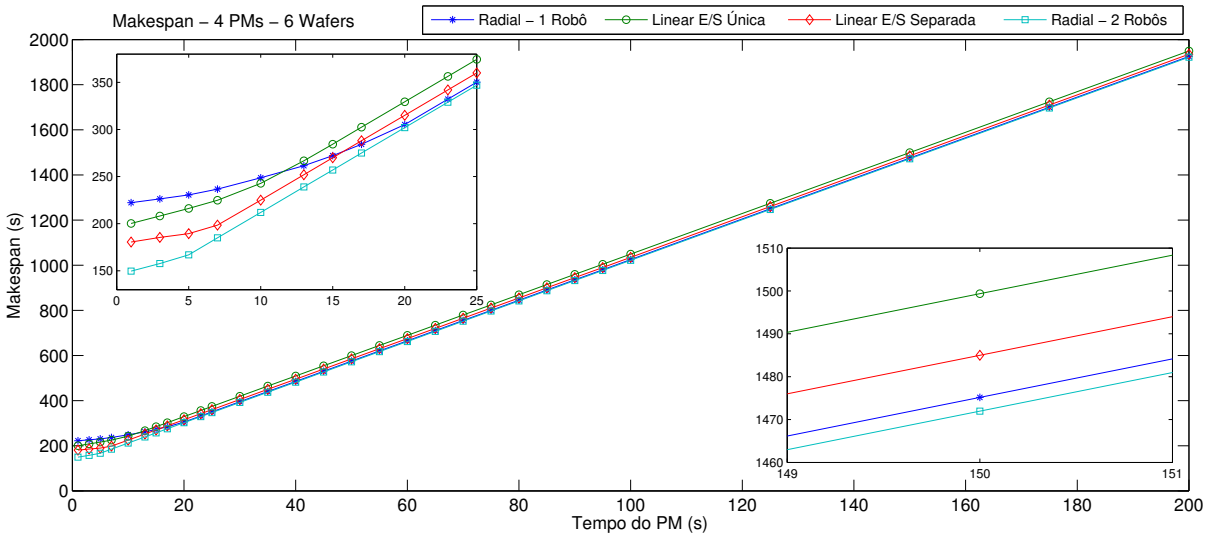


Figura 22 – *Makespan*: 4 PMs / lote de 6 *wafers*

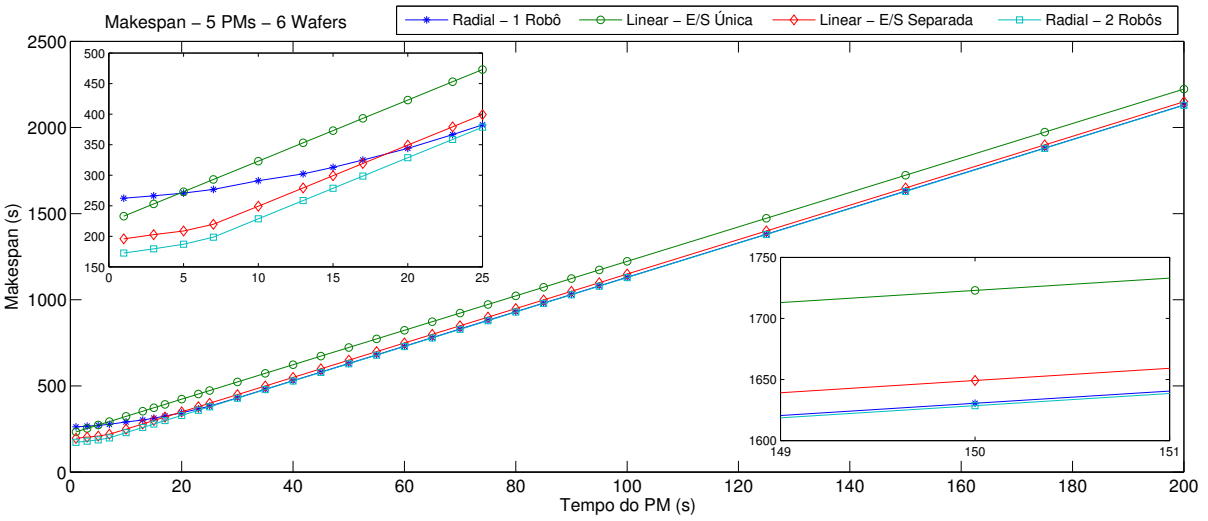


Figura 23 – *Makespan*: 5 PMs / lote de 6 *wafers*

O layout linear com entrada e saída separada tem menor valor de *makespan* na comparação com o layout linear com entrada e saída única, em todas as condições,

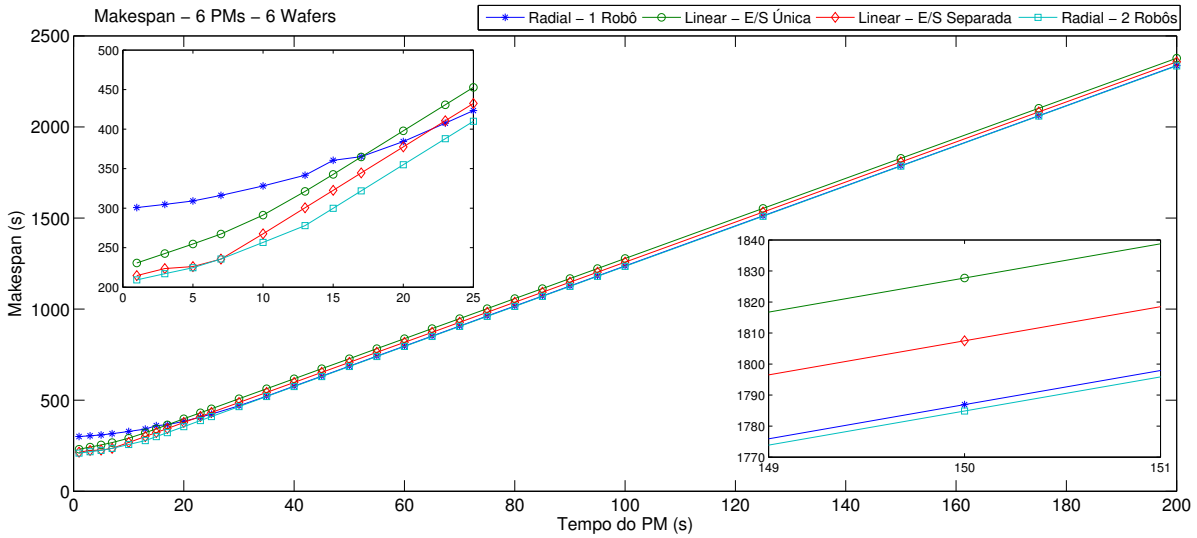


Figura 24 – *Makespan*: 6 PMs / lote de 6 *wafers*

uma vez que o *wafer* não retorna ao início do *cluster tool* ao fim do processamento. É possível observar também que, para tempos de processamento muito pequenos, seu valor de *makespan* não sofre grandes alterações com o aumento do número de PMs, de tal modo que, para 6 PMs e tempo de processamento menores que 10 s, esse layout apresenta o menor valor de *makespan*.

Para tempos dos *PMs* maiores que 30 s, o layout radial com 1 robô tem o segundo menor valor de *makespan*. Neste caso, como o *cluster tool* é *process bound*, há pouca influência do tempo de movimentação do robô no *makespan*, sendo o valor dessa métrica muito próximo para o layout radial com 1 robô e radial com 2 robôs, conforme detalhe inferior das Figuras 22 até 24 e do Anexo D.

Observa-se também nessas figuras que o *makespan* cresce linearmente, com duas inclinações diferentes. Em todos os layouts, a menor inclinação ocorre para os menores valores de tempo do *PM*. Nesta situação, os tempos dos *PMs* são menores que o tempo de movimentação do robô, e enquanto essa condição se mantiver, um aumento neste parâmetro tem pouca influência no *makespan*. O ponto de inflexão, que é diferente em cada situação, variando entre 5 s e 17 s, representa o ponto no qual o *cluster tool* deixa de ser *robot bound* para se tornar *process bound*, sendo esses tempos muito próximos. Após este ponto, um aumento no tempo do *PM* gera um incremento proporcional no *makespan*, pois o tempo do *PM* será maior que o tempo de operação do robô.

Observando simultaneamente os gráficos de PAR e *makespan* podemos ver que existe uma correlação negativa entre essas duas métricas. Quanto maior o PAR de uma sequência em uma das configurações consideradas, observa-se uma tendência de diminuição do *makespan* desta. Isso é esperado, corroborando com o critério de Máximo Paralelismo, que propõe otimizar o tempo de execução de uma sequência, sem contudo utilizar o próprio tempo, ao invés disso maximizando o número de equipamentos funcionando em paralelo.

É interessante observar que o tempo do *PM* influencia no *makespan* de cada layout, como pode ser visto no caso do layout radial com 1 robô, que, para tempos de *PM* menores que 15 s, tem o maior valor de *makespan*, e para tempos maiores que 20 s tem o segundo menor valor.

Dessa forma, vemos que quando o *cluster tool* opera na condição *robot bound*, analisando o *makespan*, o *layout* radial com 2 robôs é preferível , sendo que, a medida que aumentamos o número de *PMs*, o desempenho do *layout* linear com entrada e saída separadas se torna próximo ou mesmo superior. Analisando o PAR, não é possível verificar prontamente qual o melhor *layout*, sendo que isto deve ser observado nos gráficos de acordo com o número de *PMs*, tamanho do lote e tempo de processamento. Já na condição *process bound*, ambos os *layouts* radiais apresentam melhor desempenho, tanto em relação *makespan* quanto ao PAR, sendo o radial com 2 robôs com *makespan* ligeiramente menor e o radial com 1 robô com PAR maior, nas mesmas condições.

7 Conclusão

Neste trabalho foi apresentada uma metodologia para comparação entre diferentes layouts do *cluster tool*. Utilizou-se o algoritmo MPT, baseado na TCS, para gerar duas métricas que são utilizadas para avaliar o desempenho dos layouts. A metodologia permite caracterizar o sistema, classificando-o como *process bound* ou *robot bound*. O conhecimento dos tempos de *PM* e de movimentação do robô são fundamentais para a definição de um layout mais adequado.

Analisando os resultados obtidos, observa-se que o layout radial com 2 robôs apresenta melhor desempenho quando o *cluster tool* é *robot bound* para até 5 PMs, pois seu menor valor de *makespan* e maior valor de PAR indicam que ele produz de forma mais rápida e com maior eficiência, respectivamente. Para 6 PMs, o mesmo é observado para o layout linear com entrada e saída separada. Já quando o *cluster tool* é *process bound*, o layout radial com 1 robô se destaca, pois seu maior valor de PAR indica um layout mais eficiente, já que o valor de *makespan* em relação ao layout que apresentou o menor valor, no caso o layout radial com 2 robôs, é muito próximo.

No caso da aplicação do algoritmo MPT especificamente no *cluster tool*, uma vantagem da metodologia desenvolvida é que ela incorpora simultaneamente o software Controlador do Cluster Tool (CCT) e o *escalonador*, uma vez que essa abordagem fornece ao mesmo tempo um controle minimamente restritivo e uma sequência de escalonamento de produção, evitando assim um grande problema do *cluster tool* que é o *deadlock*.

Uma vez que características e vantagens qualitativas de cada layout não são consideradas, este método não deve ser usado como forma única de seleção de um *cluster tool*, já que considera apenas o *makespan* e o paralelismo da sequência. Porém, esta abordagem mostrou-se adequada para verificar a influência do layout e dos tempos dos *PMs* no desempenho.

7.1 Trabalhos Futuros

Como desdobramentos futuros deste trabalho podemos citar:

- A aplicação dessa abordagem com parâmetros reais de um *cluster tool*.
- A inclusão de informações qualitativas na análise de desempenho, como a facilidade de manutenção, operação e de expansão de cada layout.
- Utilizar supervisores modulares ao invés do supervisor monolítico.

Referências

ALVES, L.; MARTINS, L.; PENA, P. UltraDES - A library for modeling, analysis and control of discrete event systems. In: *Accepted at the 20th World Congress of the International Federation of Automatic Control, IFAC 2017*. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 29 e 50.

ALVES, L.; PENA, P.; TAKAHASHI, R. Planejamento da produção baseado no critério do máximo paralelismo com restrições temporais. *XXI Congresso Brasileiro de Automática (CBA),*, p. 1518–1523, 2016. Citado 10 vezes nas páginas 17, 19, 21, 22, 36, 37, 39, 40, 41 e 42.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to discrete event systems*. [S.l.]: Springer Science & Business Media, 2009. Citado 3 vezes nas páginas 19, 22 e 29.

DING, S.; YI, J.; ZHANG, M. T. Multicluster tools scheduling: An integrated event graph and network model approach. *IEEE Transactions on Semiconductor Manufacturing*, IEEE, v. 19, n. 3, p. 339–351, 2006. Citado na página 20.

DÜMMLER, M. A. Using simulation and genetic algorithms to improve cluster tool performance. In: ACM. *Proceedings of the 31st Conference on Winter Simulation: Simulation—a bridge to the future-Volume 1*. [S.l.], 1999. p. 875–879. Citado 4 vezes nas páginas 20, 30, 31 e 35.

KOBETSKI, A.; FABIAN, M. Scheduling of discrete event systems using mixed integer linear programming. In: IEEE. *Discrete Event Systems, 2006 8th International Workshop on*. [S.l.], 2006. p. 76–81. Citado na página 19.

LEBARON, H. T.; HENDRICKSON, R. A. Using emulation to validate a cluster tool simulation model. In: WINTER SIMULATION CONFERENCE. *Proceedings of the 2000 Winter Simulation Conference*. [S.l.], 2000. p. 1417–1422. Citado na página 20.

LEE, J.-H.; LEE, T.-E. An open scheduling architecture for cluster tools. In: IEEE. *IEEE Conference on Automation Science and Engineering (CASE), 2010*. [S.l.], 2010. p. 420–425. Citado 3 vezes nas páginas 15, 20 e 35.

LEE, S.; NI, J. Genetic algorithm for job scheduling with maintenance consideration in semiconductor manufacturing process. *Mathematical Problems in Engineering*, Hindawi Publishing Corporation, v. 2012, 2012. Citado 2 vezes nas páginas 8 e 32.

MEULEN, P. V. D. Linear semiconductor manufacturing logistics and the impact on cycle time. In: IEEE. *Advanced Semiconductor Manufacturing Conference, 2007. ASMC 2007. IEEE/SEMI*. [S.l.], 2007. p. 111–116. Citado na página 20.

MÖNCH, L. et al. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling,*, Springer, v. 14, n. 6, p. 583–599, 2011. Citado na página 15.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. Citado na página 19.

NEHME, D. A.; PIERCE, N. G. Evaluating the throughput of cluster tools using event-graph simulations. In: IEEE. *Advanced Semiconductor Manufacturing Conference and Workshop. 1994 IEEE/SEMI*. [S.l.], 1994. p. 189–192. Citado na página 20.

PARK, K.; MORRISON, J. R. Cluster tool design comparisons via simulation. In: WINTER SIMULATION CONFERENCE. *Proceedings of the Winter Simulation Conference*. [S.l.], 2011. p. 1877–1887. Citado na página 20.

PARK, S.-J.; YANG, J.-M. Supervisory control for real-time scheduling of periodic and sporadic tasks with resource constraints. *Automatica*, Elsevier, v. 45, n. 11, p. 2597–2604, 2009. Citado na página 19.

PEDERSON, D.; TROUT, C. Demonstrated benefits of cluster tool simulation. In: *Proceedings of the 2002 International Conference on Modeling and Analysis of Semiconductor Manufacturing*. [S.l.: s.n.], 2002. p. 84–89. Citado na página 20.

PINHA, D. C.; QUEIROZ, M. H. de; CURY, J. E. Optimal scheduling of a repair shipyard based on supervisory control theory. In: IEEE. *Automation Science and Engineering (CASE), 2011 IEEE Conference on*. [S.l.], 2011. p. 39–44. Citado na página 19.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. *Proceedings of the IEEE,*, IEEE, v. 77, n. 1, p. 81–98, 1989. Citado 5 vezes nas páginas 17, 19, 23, 27 e 28.

SCHÖMIG, A.; FOWLER, J. Modeling semiconductor manufacturing operations. In: *Proceedings of the 9th ASIM dedicated conference simulation in production and logistics*. [S.l.: s.n.], 2000. p. 55–64. Citado na página 15.

SEMI. Cluster tool module interface: mechanical interface and wafer transport standard. *SEMI E21-94 (reapproved 0309)*, SEMI International Standards, 2009. Citado na página 15.

SHIN, Y.-H. et al. Modeling and implementing a real-time scheduler for dual-armed cluster tools. *Computers in Industry,*, Elsevier, v. 45, n. 1, p. 13–27, 2001. Citado 2 vezes nas páginas 19 e 31.

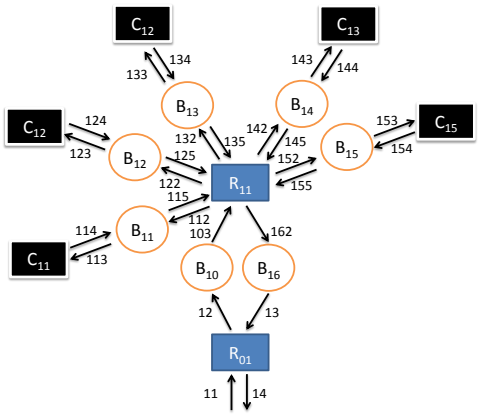
YI, J. et al. Throughput analysis of linear cluster tools. In: IEEE. *IEEE International Conference on Automation Science and Engineering. CASE 2007*. [S.l.], 2007. p. 1063–1068. Citado 4 vezes nas páginas 20, 32, 54 e 55.

ZUBEREK, W. M. Timed petri nets in modeling and analysis of cluster tools. *IEEE Transactions on Robotics and Automation,*, IEEE, v. 17, n. 5, p. 562–575, 2001. Citado na página 19.

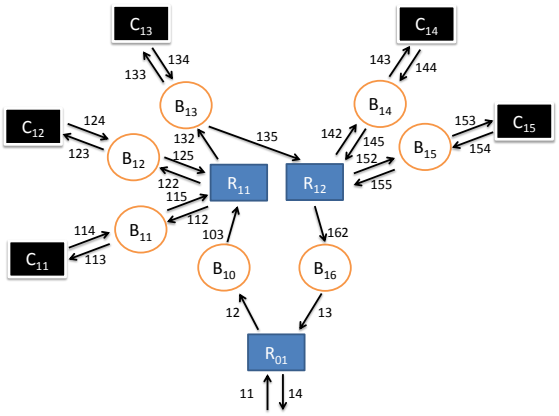
Anexos

ANEXO A – Modelos dos Layouts Considerados

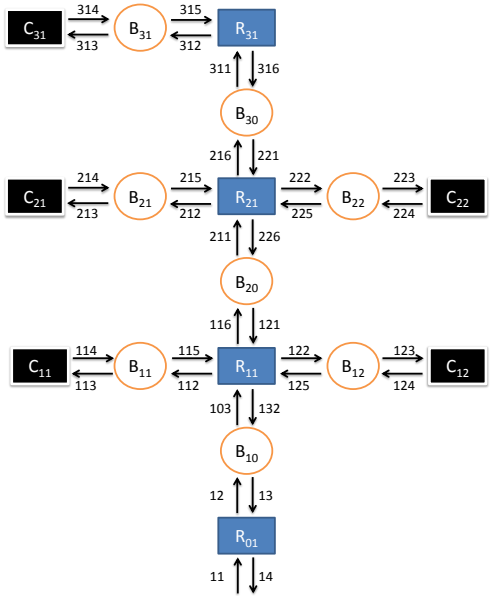
Neste anexo são apresentados os modelos de cada layout considerado. A Figura 25 apresenta o modelo para os layouts considerados com 5 *PMs*.



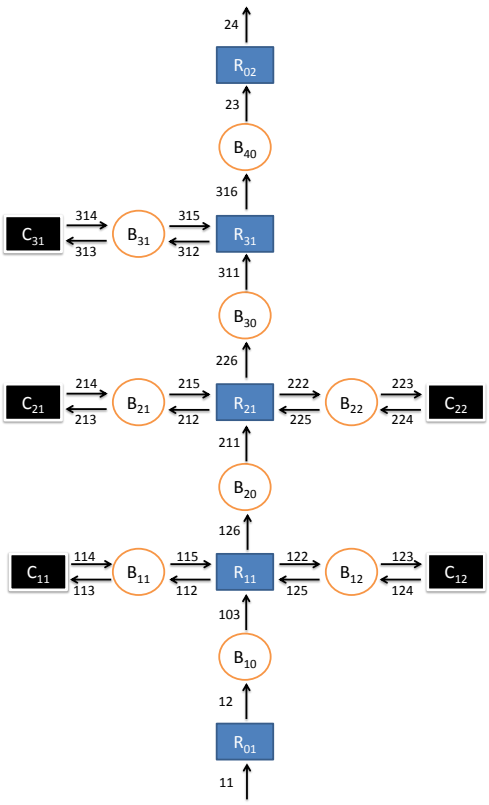
(a) Radial - 1 Robô



(b) Radial - 2 Robôs



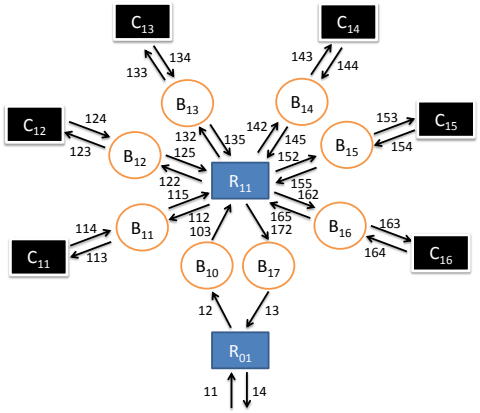
(c) Linear - Entrada e Saída Única



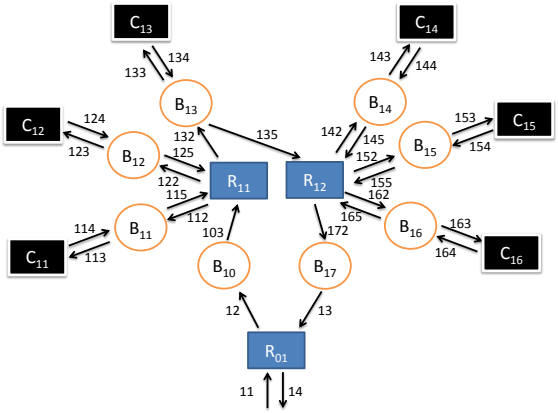
(d) Linear - Entrada e Saída Separada

Figura 25 – Modelos para Layouts com 5 PMs

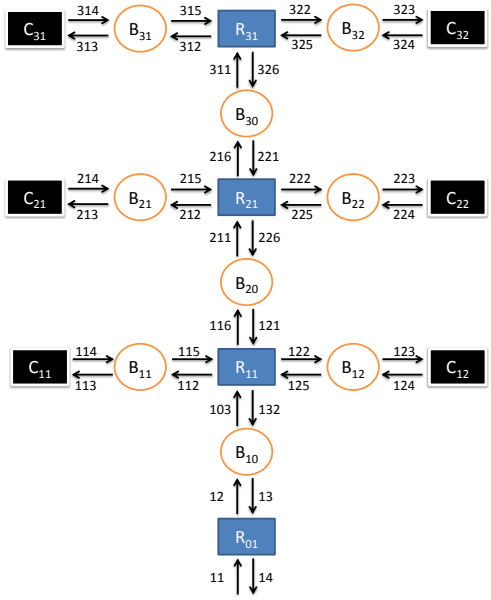
A Figura 26 apresenta o modelo para os layouts considerados com 6 *PMs*.



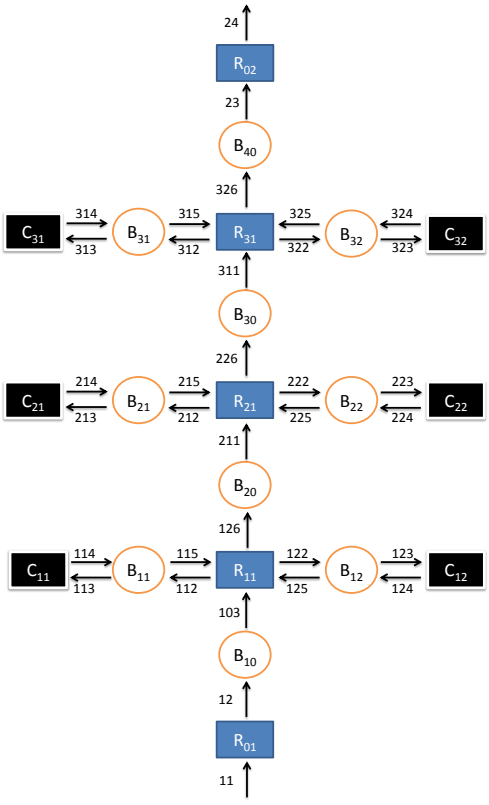
(a) Radial - 1 Robô



(b) Radial - 2 Robôs



(c) Linear - Entrada e Saída Única



(d) Linear - Entrada e Saída Separada

Figura 26 – Modelos para Layouts com 6 PMs

ANEXO B – Autômatos dos Modelos

Neste anexo são apresentados os autômatos dos modelos de cada layouts. A Figura 27 apresenta os autômatos das plantas e especificações do layout radial com 1 robô para 5 *PMs*.

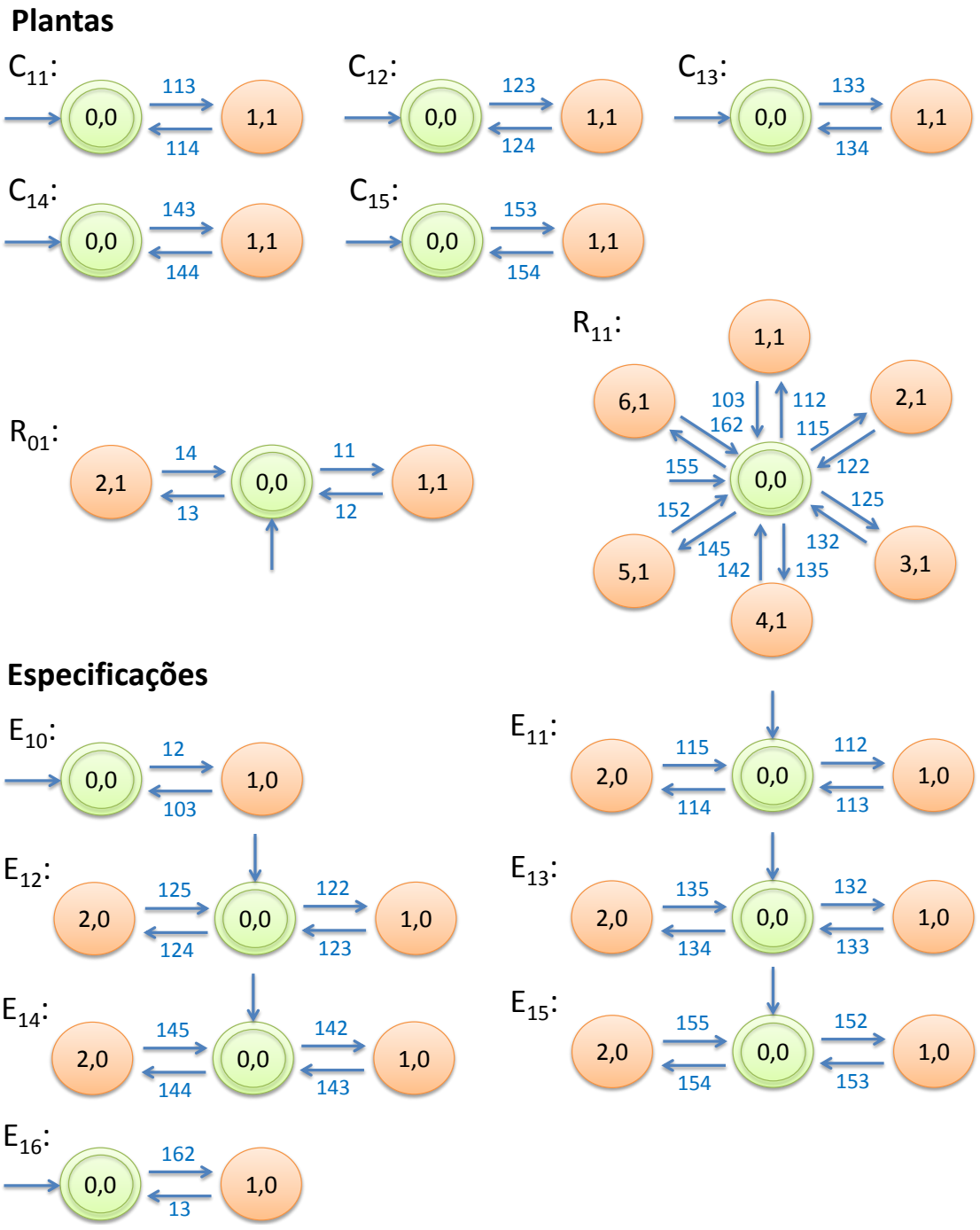


Figura 27 – Autômatos do Layout Radial com 5 PMs - 1 Robô

A Figura 28 apresenta os autômatos das plantas e especificações do layout radial com 2 robôs para 5 *PMs*.

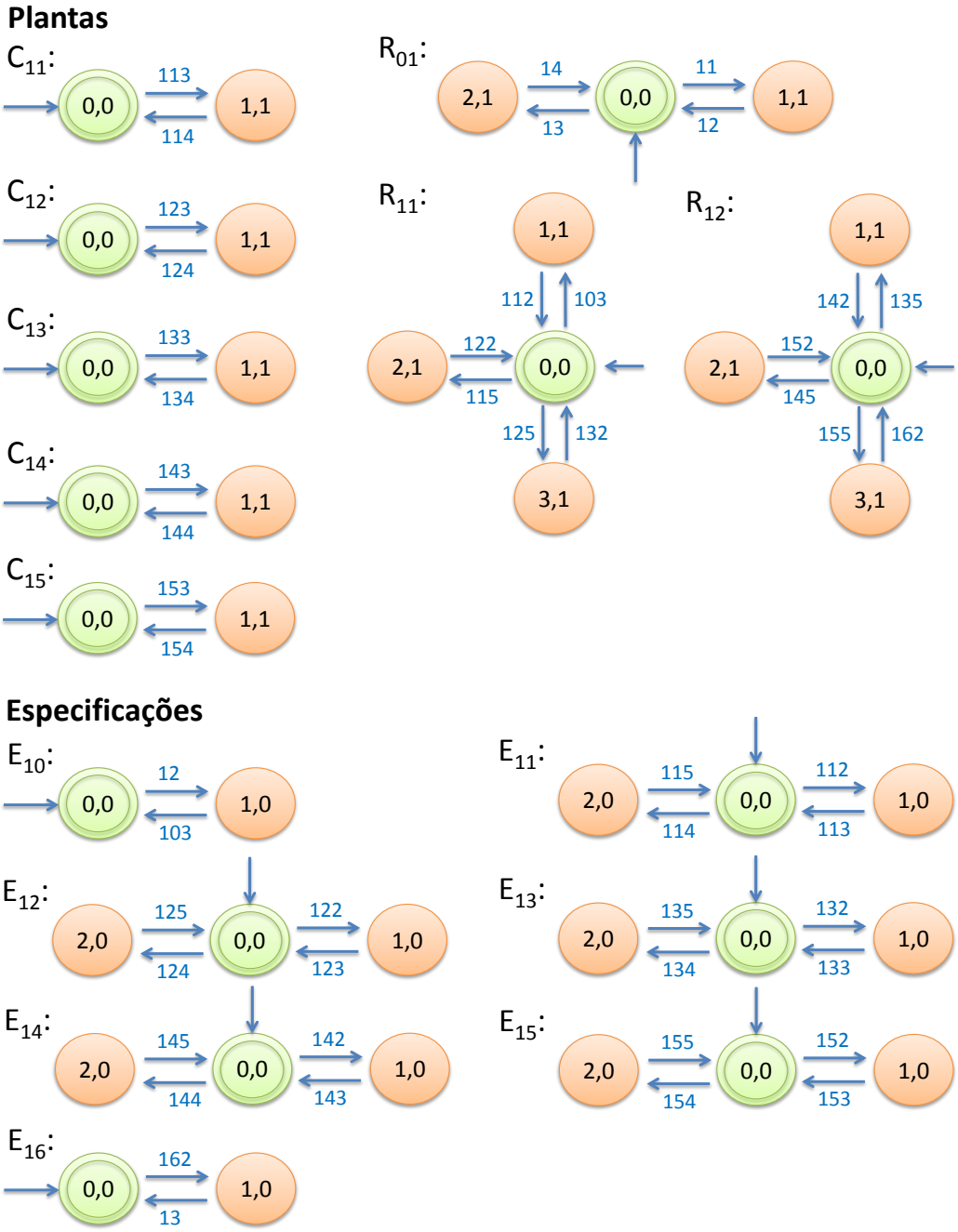


Figura 28 – Autômatos do Layout Radial com 5 PMs - 2 Robôs

A Figura 29 apresenta os autômatos das plantas e especificações do layout linear com entrada e saída única para 5 *PMs*.

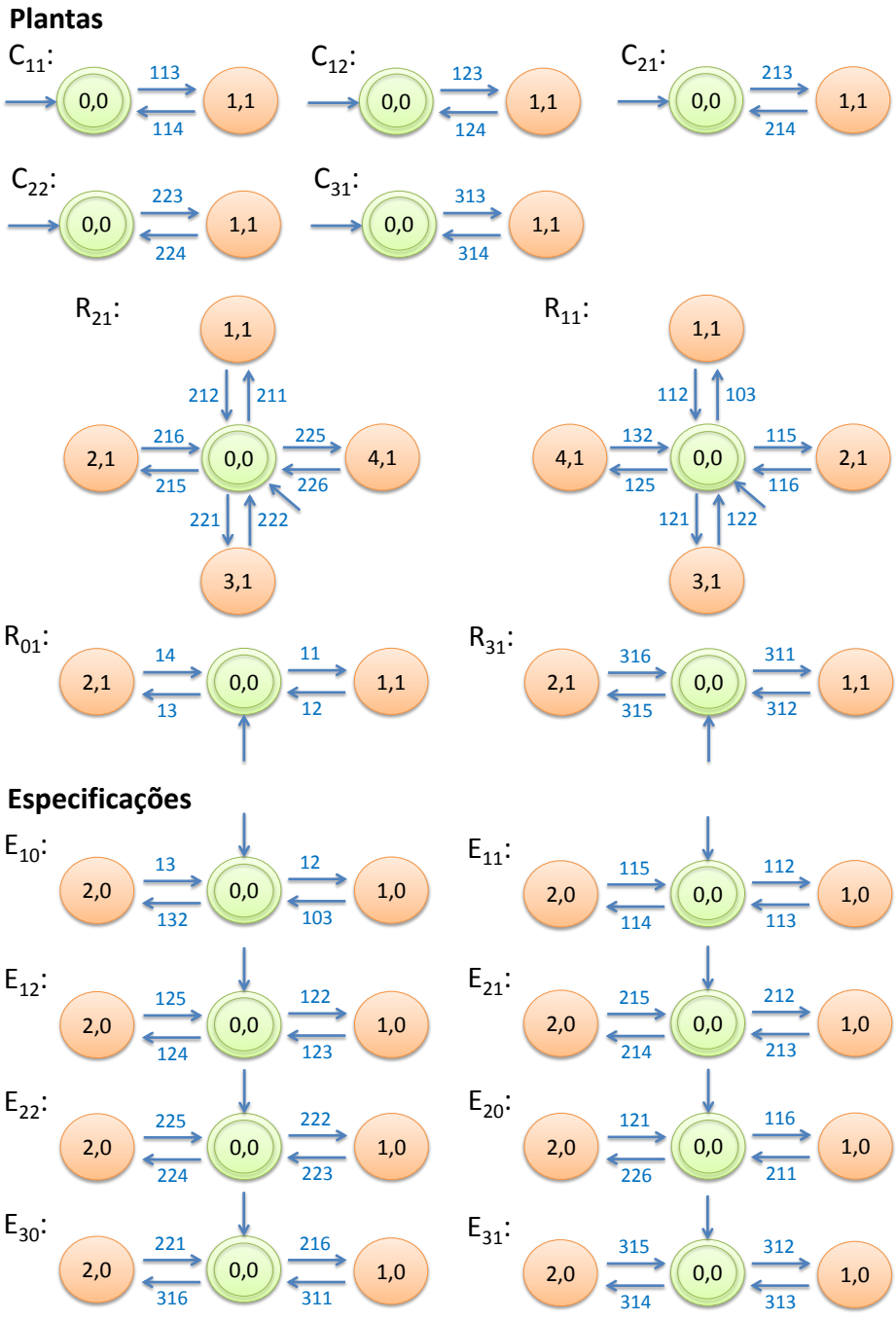


Figura 29 – Autômatos do Layout Linear com 5 PMs - Entrada e Saída Única

A Figura 30 apresenta os autômatos das plantas e especificações do layout linear com entrada e saída separada para 5 *PMs*.

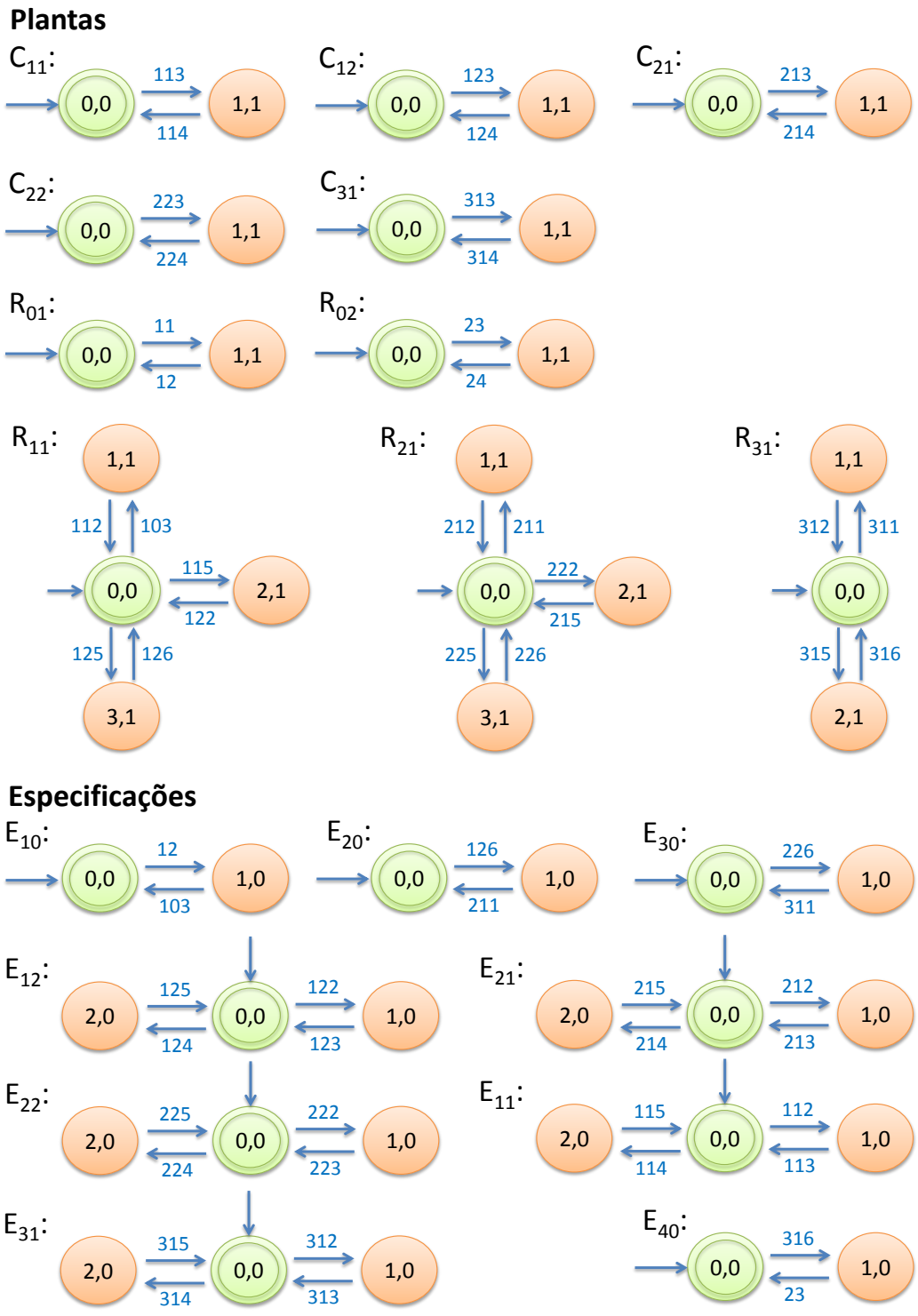


Figura 30 – Autômatos do Layout Linear com 5 PMs - Entrada e Saída Separada

A Figura 31 apresenta os autômatos das plantas e especificações do layout radial com 1 robô para 6 *PMs*.

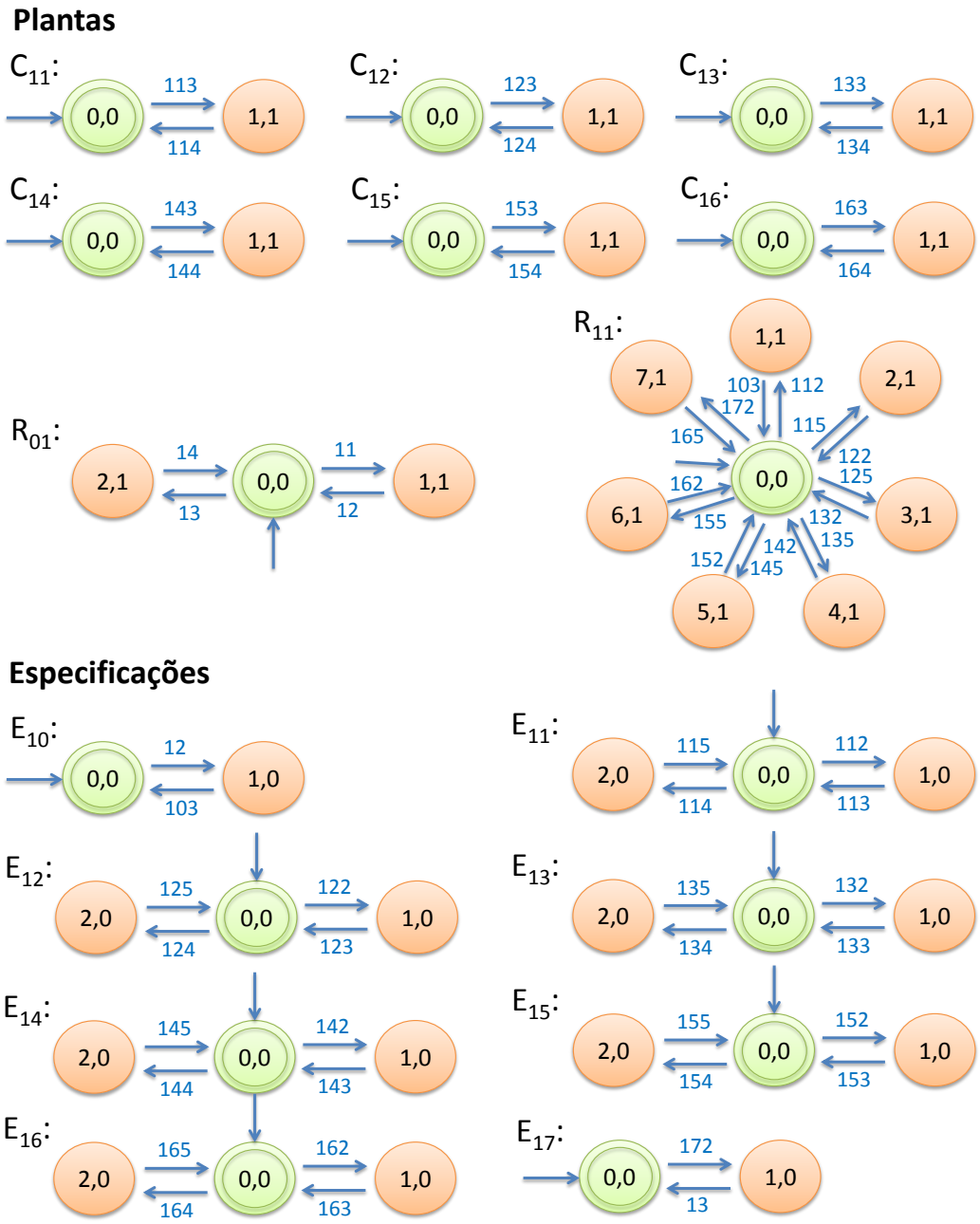


Figura 31 – Autômatos do Layout Radial com 6 PMs - 1 Robô

A Figura 32 apresenta os autômatos das plantas e especificações do layout radial com 2 robôs para 6 *PMs*.

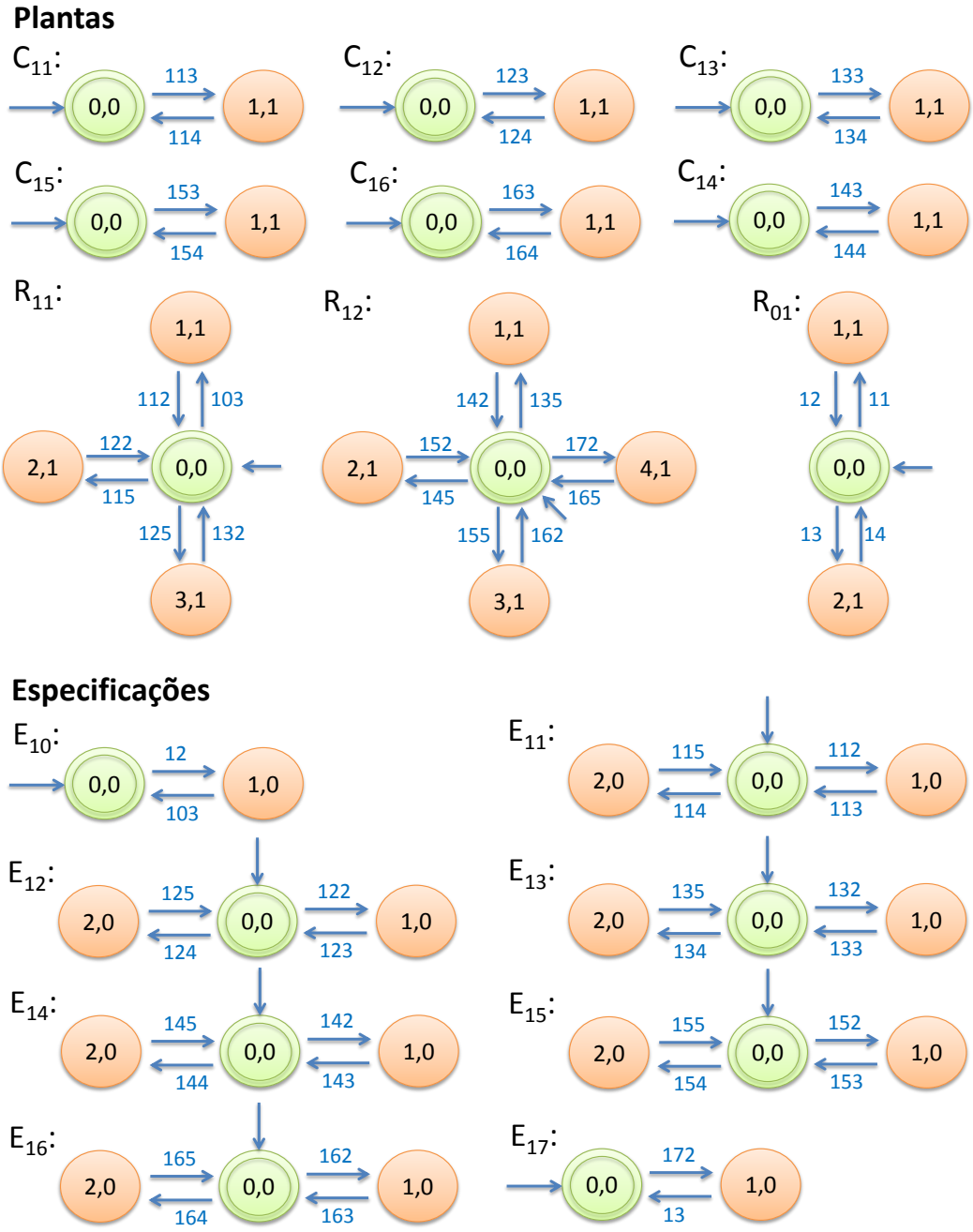


Figura 32 – Autômatos do Layout Radial com 6 PMs - 2 Robôs

A Figura 33 apresenta os autômatos das plantas e especificações do layout linear com entrada e saída única para 6 *PMs*.

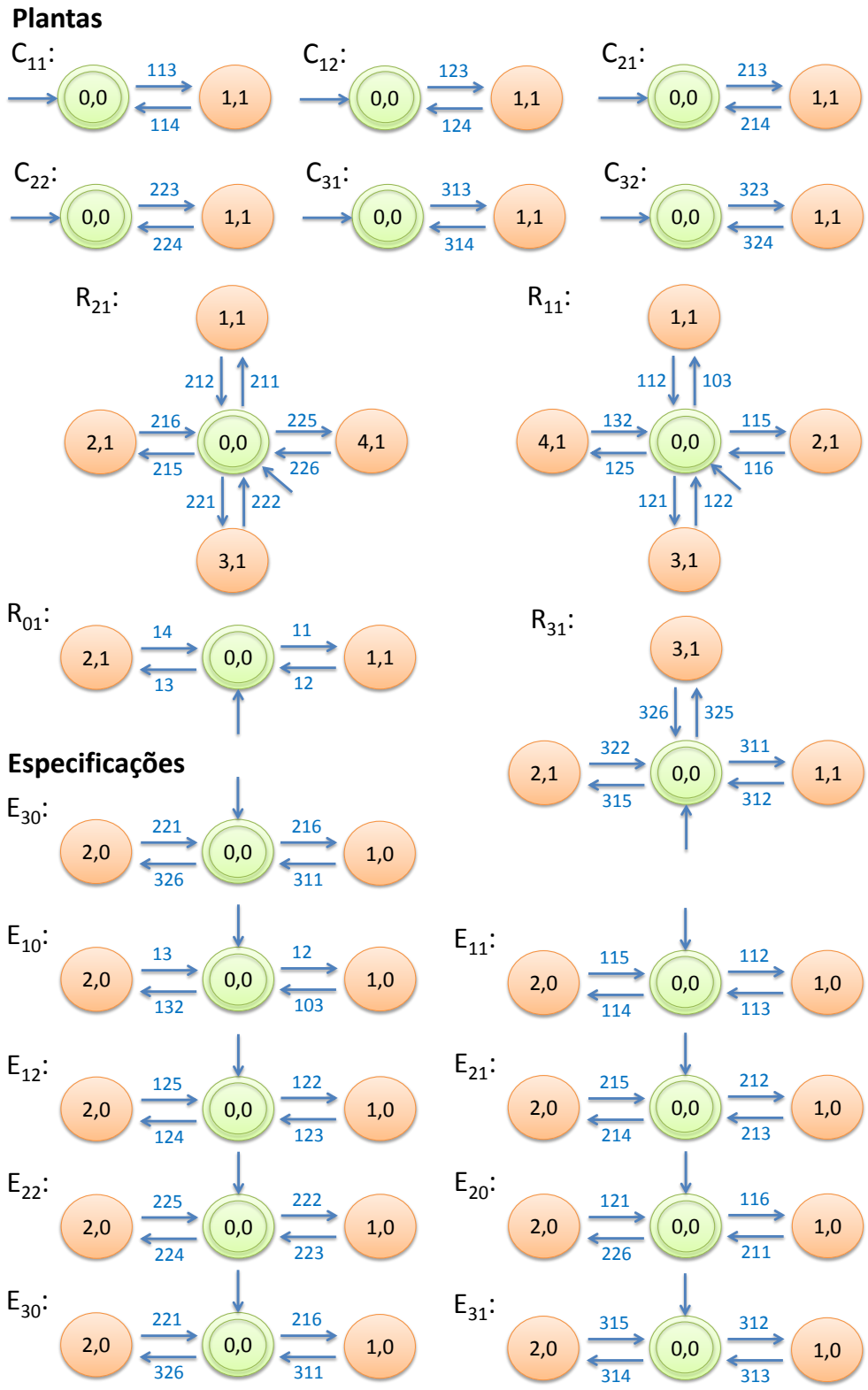


Figura 33 – Autômatos do Layout Linear com 6 PMs - Entrada e Saída Única

A Figura 34 apresenta os autômatos das plantas e especificações do layout linear com entrada e saída separada para 6 *PMs*.

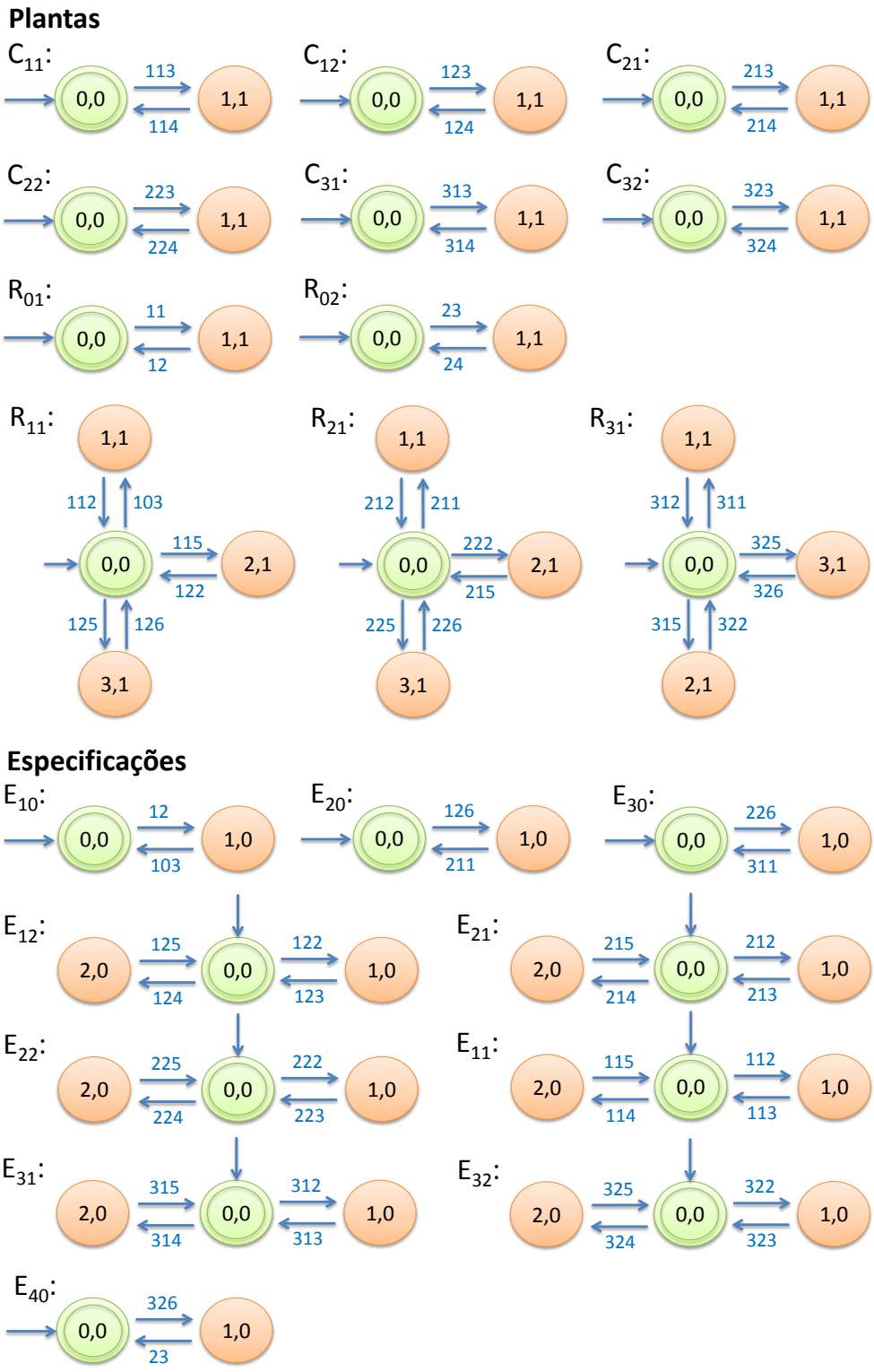


Figura 34 – Autômatos do Layout Linear com 6 PMs - Entrada e Saída Separada

ANEXO C – Gráficos de PAR

Neste anexo são apresentados os gráficos de PAR da melhor sequência obtida para os layouts com tamanho de lote igual a 12, 25 e 50 *wafers*, respectivamente. As Figuras 35, 36 e 37 apresentam os valores de PAR com lotes de 12 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

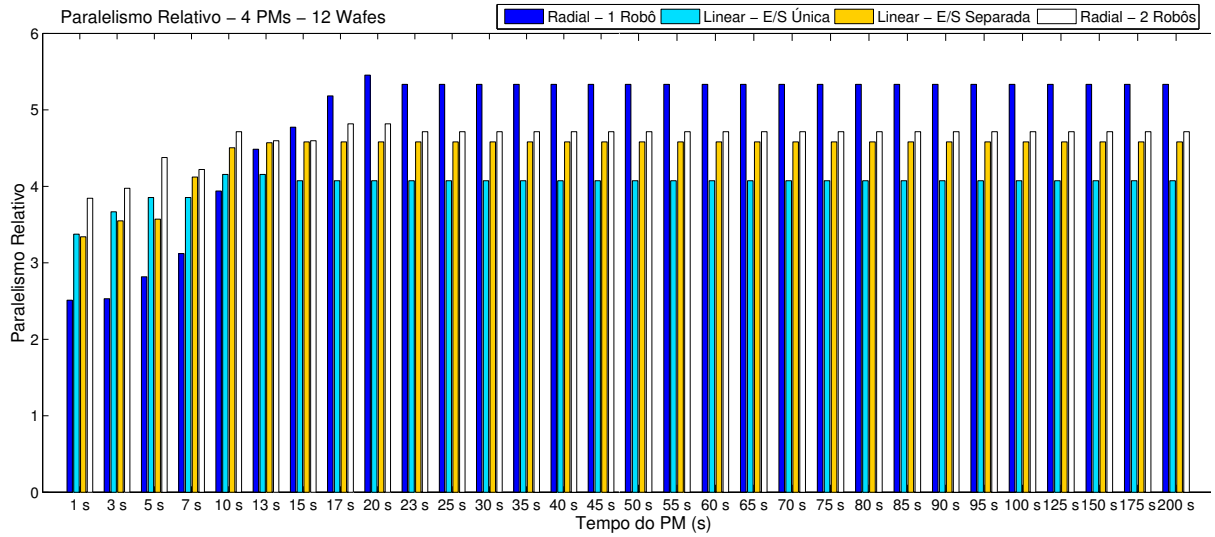


Figura 35 – Paralelismo Relativo: 4 PMs / lote de 12 *wafers*

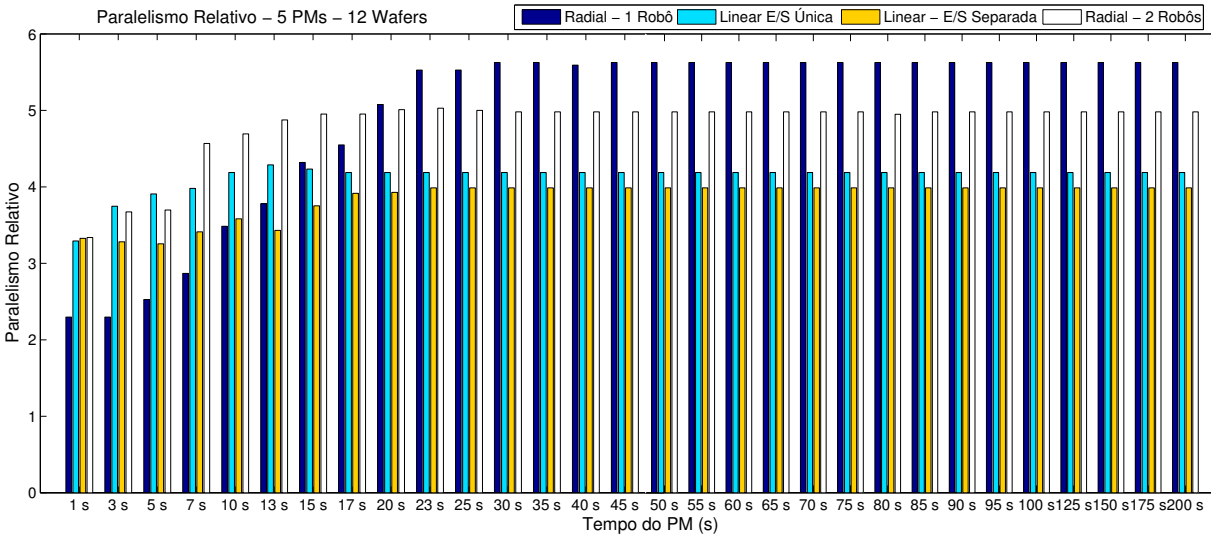


Figura 36 – Paralelismo Relativo: 5 PMs / lote de 12 *wafers*

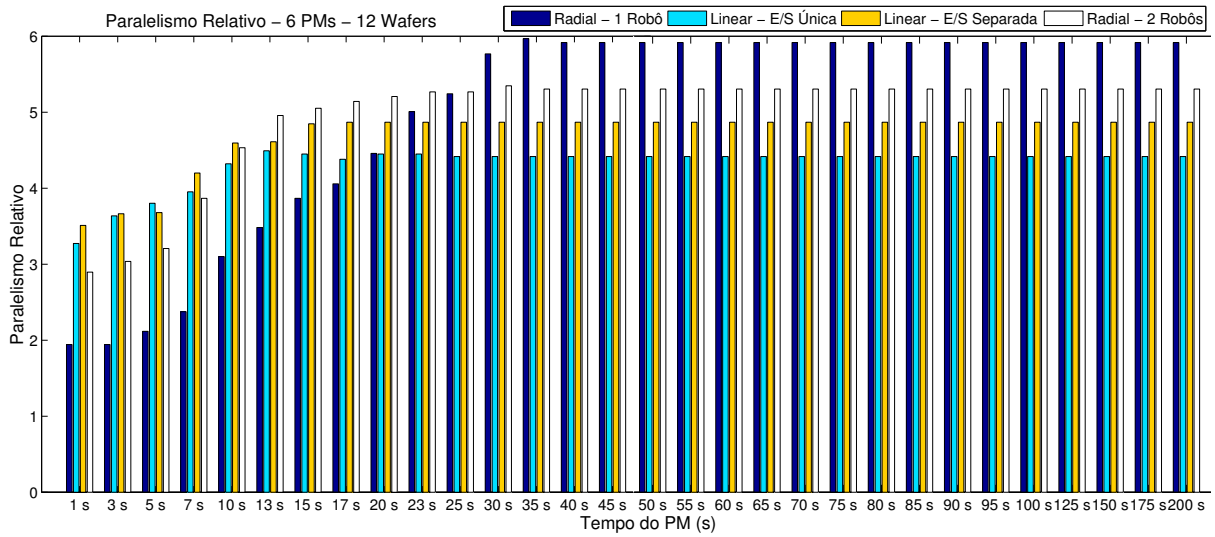


Figura 37 – Paralelismo Relativo: 6 PMs / lote de 12 *wafers*

As Figuras 38, 39 e 40 apresentam os valores de PAR com lotes de 25 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

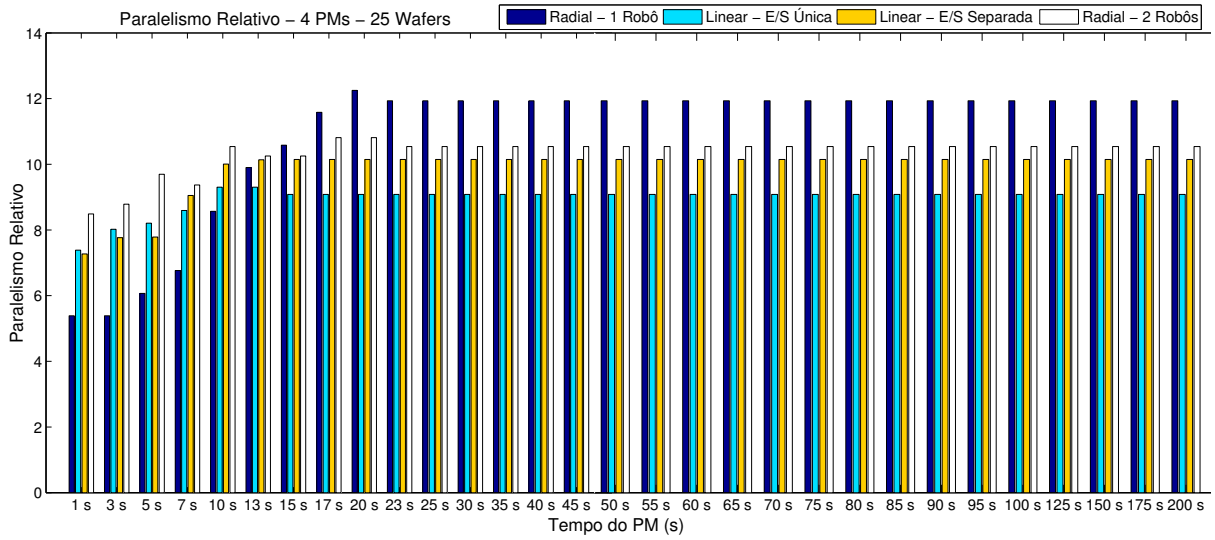


Figura 38 – Paralelismo Relativo: 4 PMs / lote de 25 *wafers*

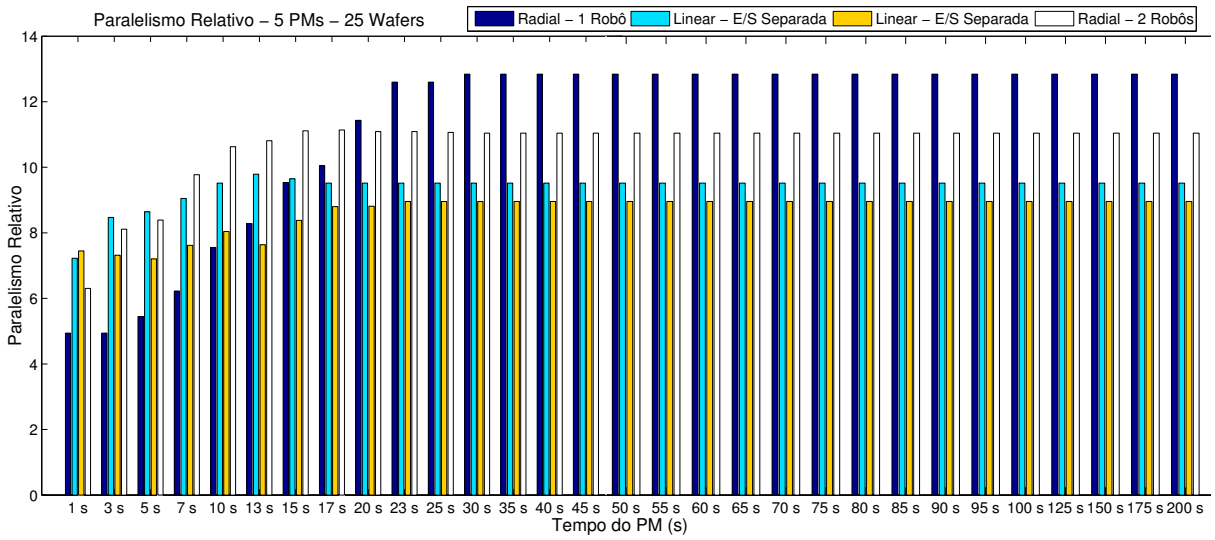


Figura 39 – Paralelismo Relativo: 5 PMs / lote de 25 *wafers*

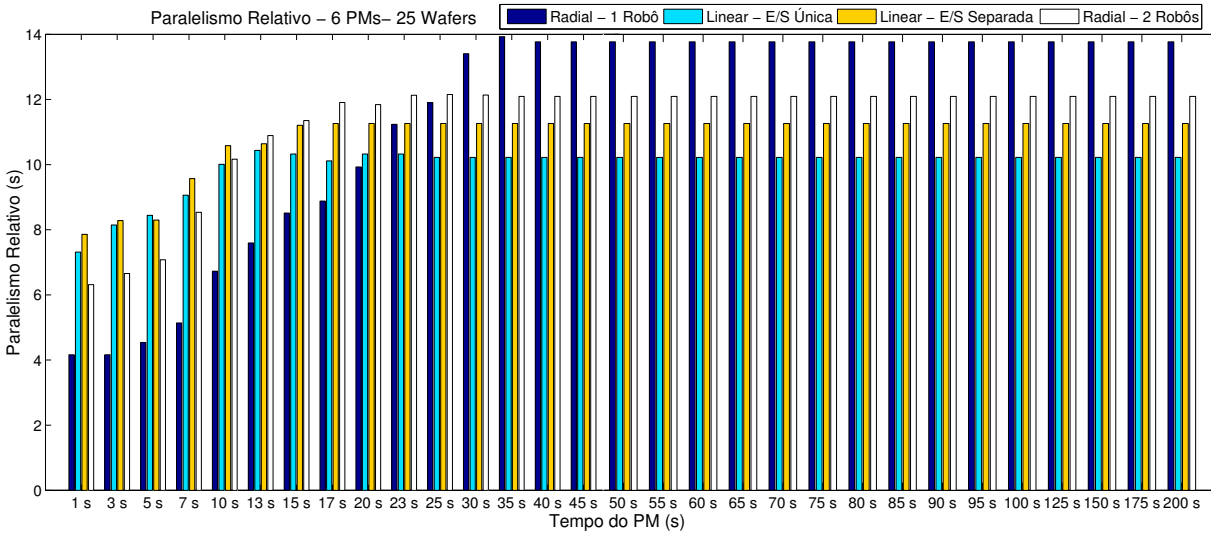


Figura 40 – Paralelismo Relativo: 6 PMs / lote de 25 *wafers*

As Figuras 41, 42 e 43 apresentam os valores de PAR com lotes de 50 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

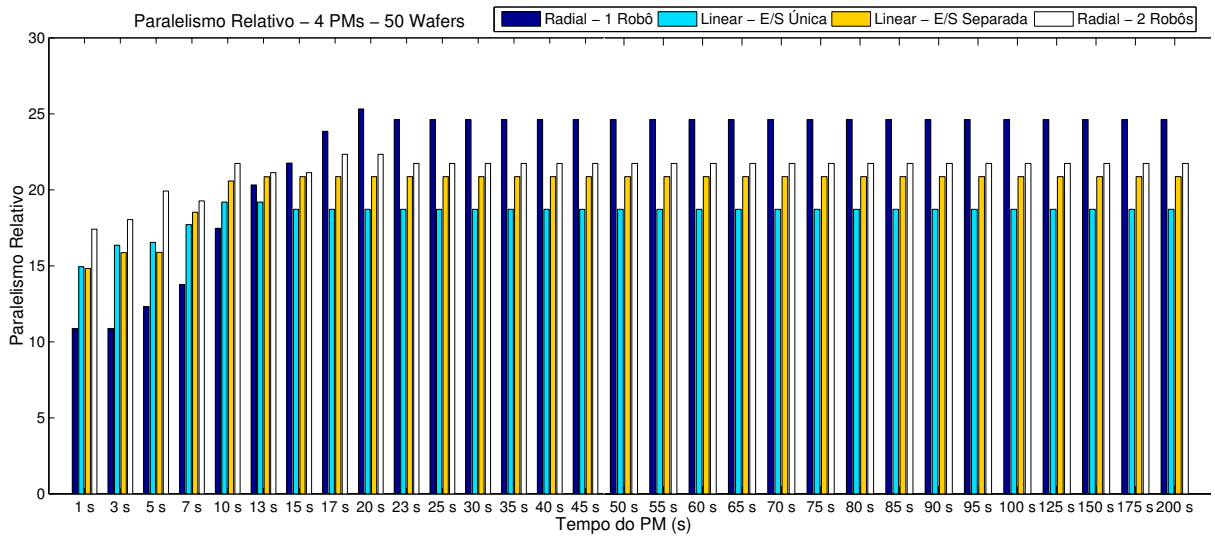


Figura 41 – Paralelismo Relativo: 4 PMs / lote de 50 *wafers*

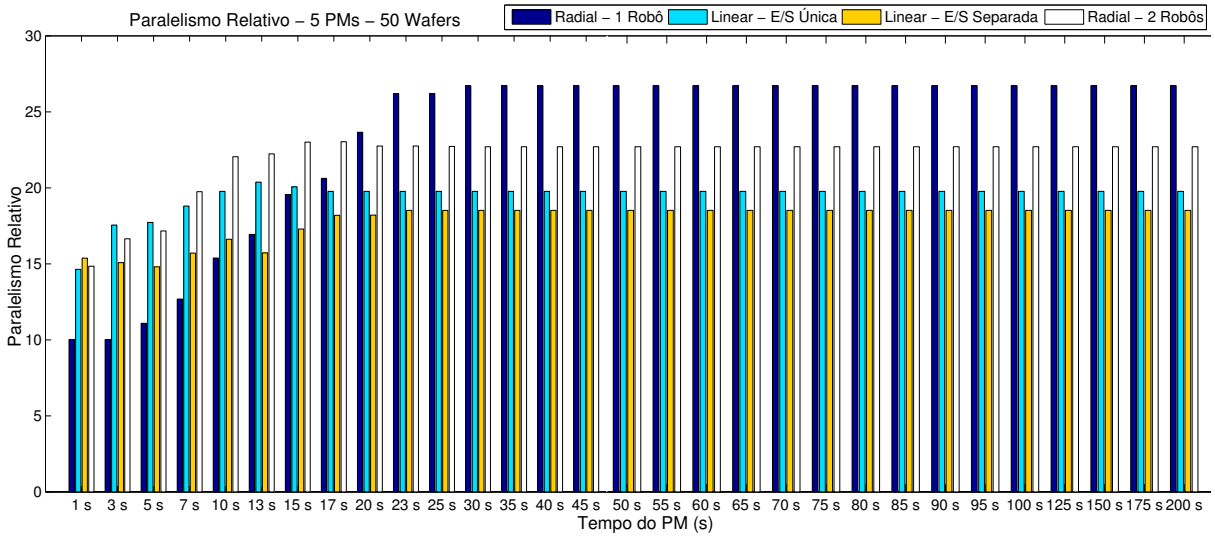


Figura 42 – Paralelismo Relativo: 5 PMs / lote de 50 *wafers*

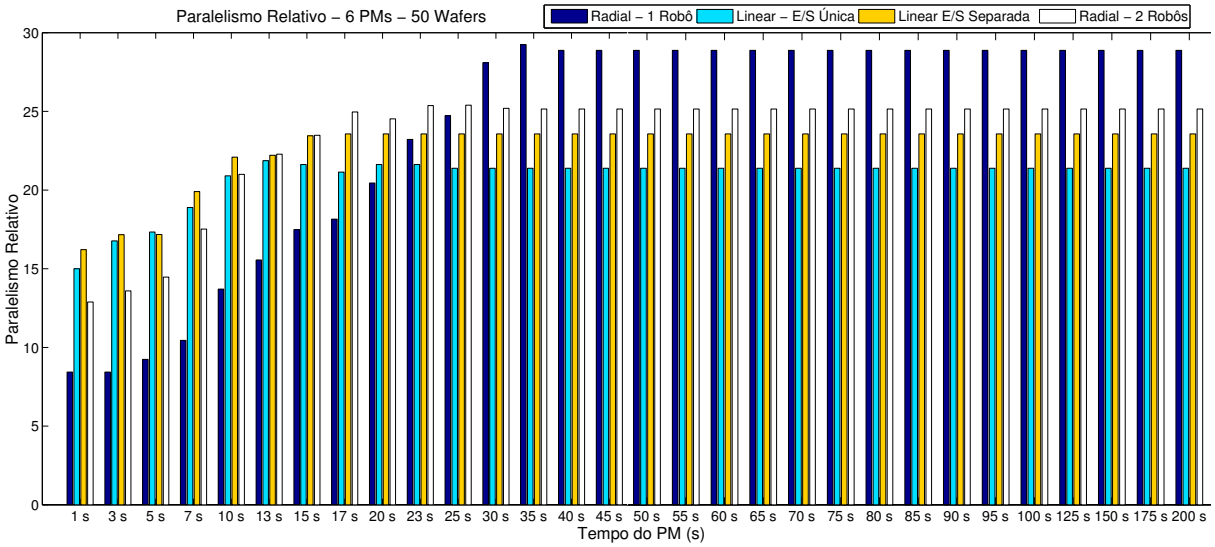


Figura 43 – Paralelismo Relativo: 6 PMs / lote de 50 *wafers*

ANEXO D – Gráficos de *makespan*

Neste anexo são apresentados os gráficos de *makespan* da melhor sequência obtida para os layouts com tamanho de lote igual a 12, 25 e 50 *wafers*, respectivamente. As Figuras 44, 45 e 46 apresentam os valores de *makespan* com lotes de 12 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

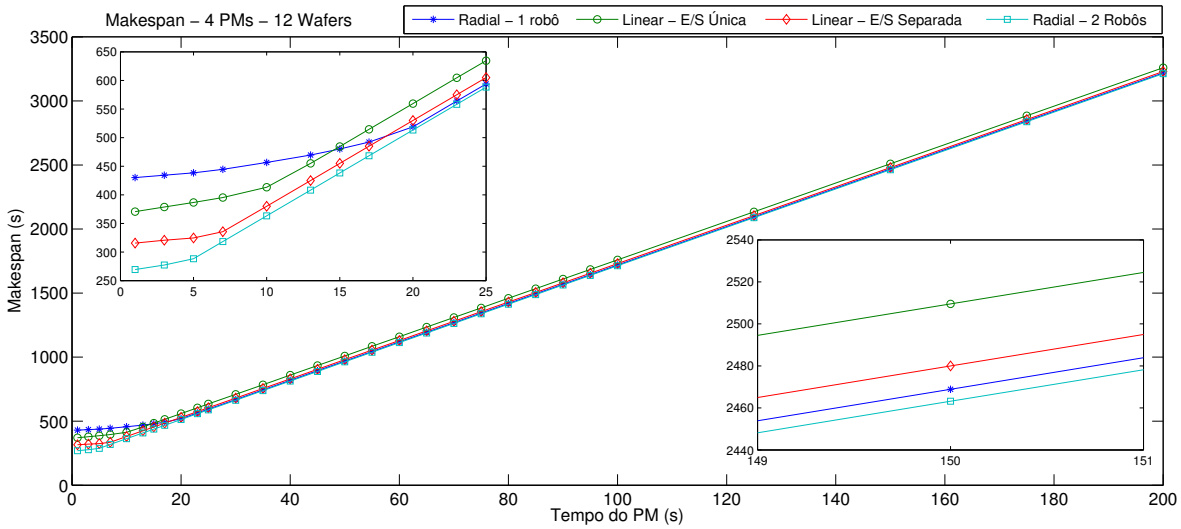


Figura 44 – *Makespan*: 4 PMs / lote de 12 *wafers*

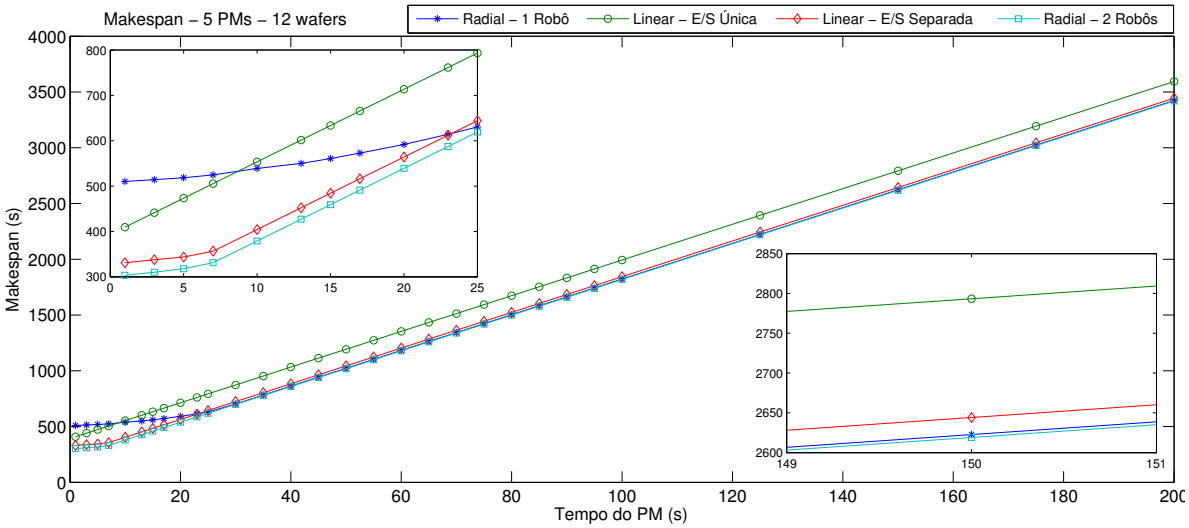


Figura 45 – *Makespan*: 5 PMs / lote de 12 *wafers*

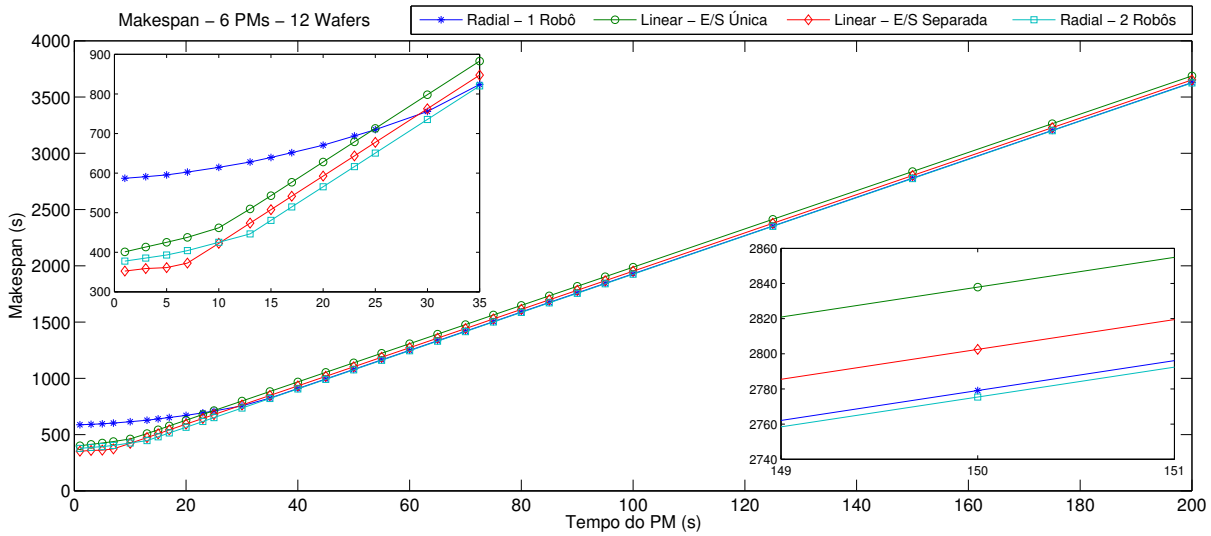


Figura 46 – *Makespan*: 6 PMs / lote de 12 *wafers*

As Figuras 47, 48 e 49 apresentam os valores de *makespan* com lotes de 25 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

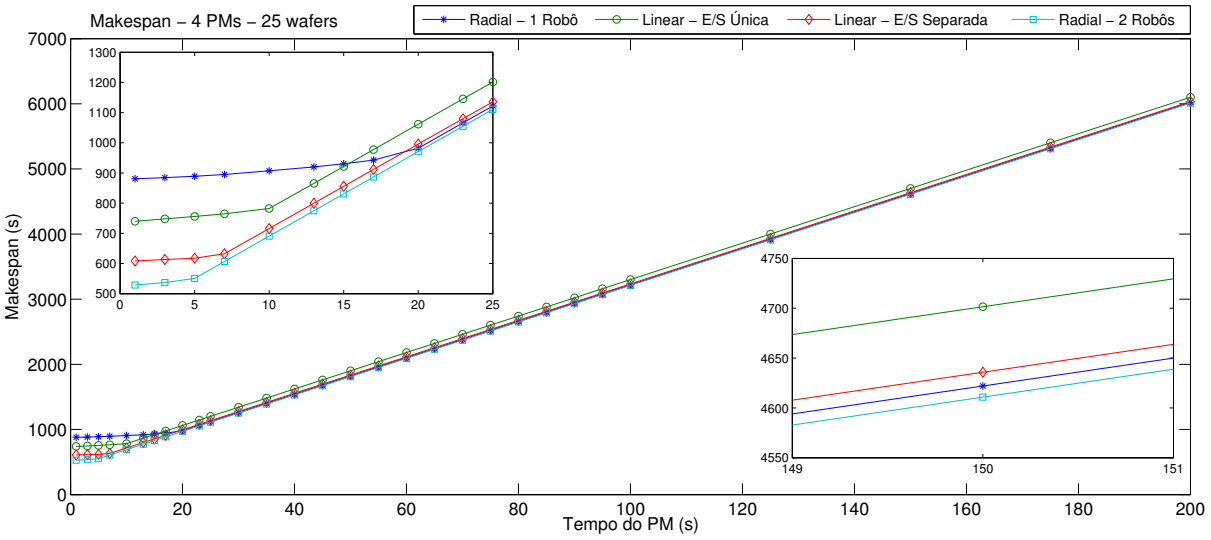


Figura 47 – *Makespan*: 4 PMs / lote de 25 *wafers*

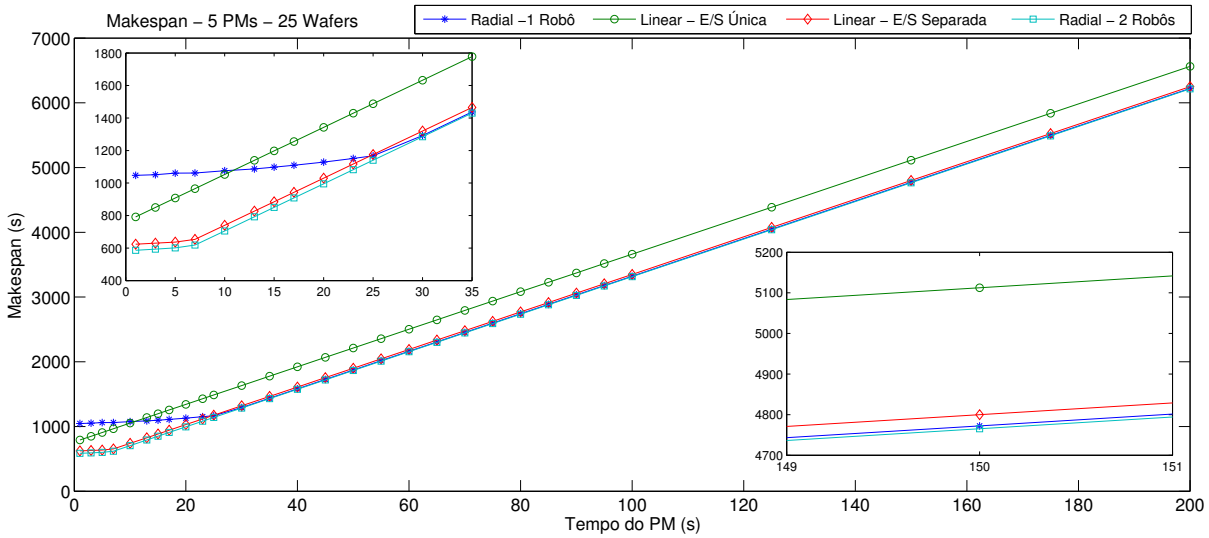


Figura 48 – *Makespan*: 5 PMs / lote de 25 *wafers*

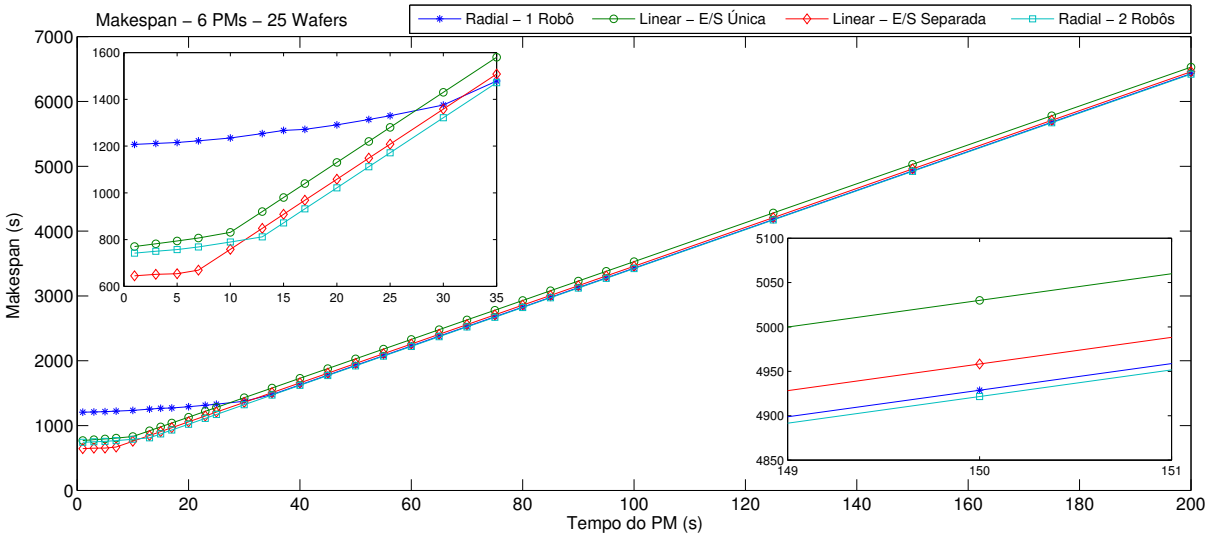


Figura 49 – *Makespan*: 6 PMs / lote de 25 *wafers*

As Figuras 50, 51 e 52 apresentam os valores de *makespan* com lotes de 50 *wafers* nos layouts considerados para 4, 5 e 6 *PMs*, respectivamente.

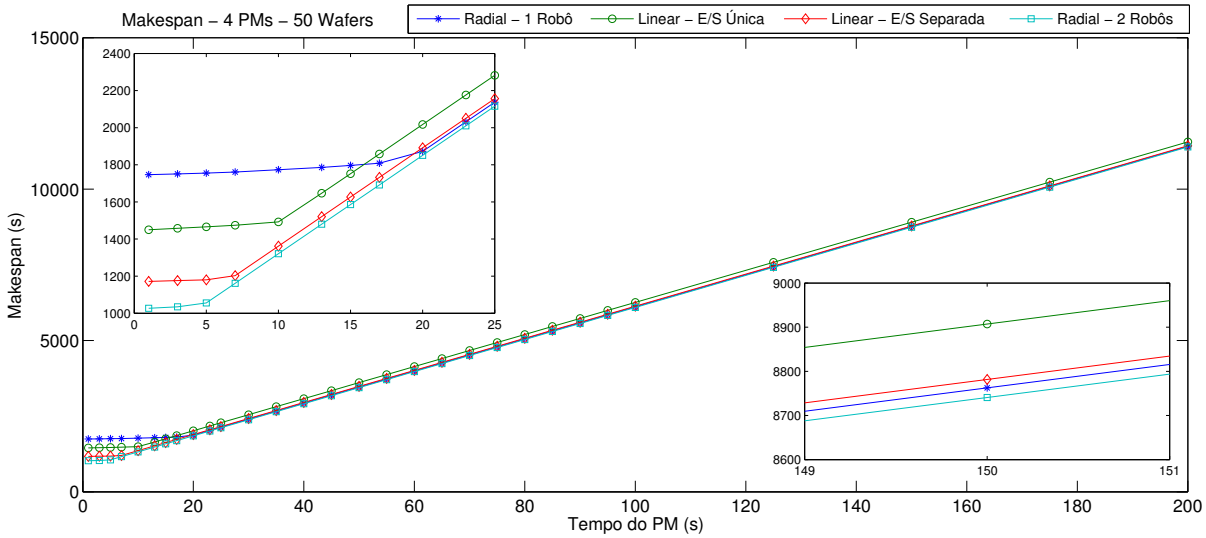


Figura 50 – *Makespan*: 4 PMs / lote de 50 *wafers*

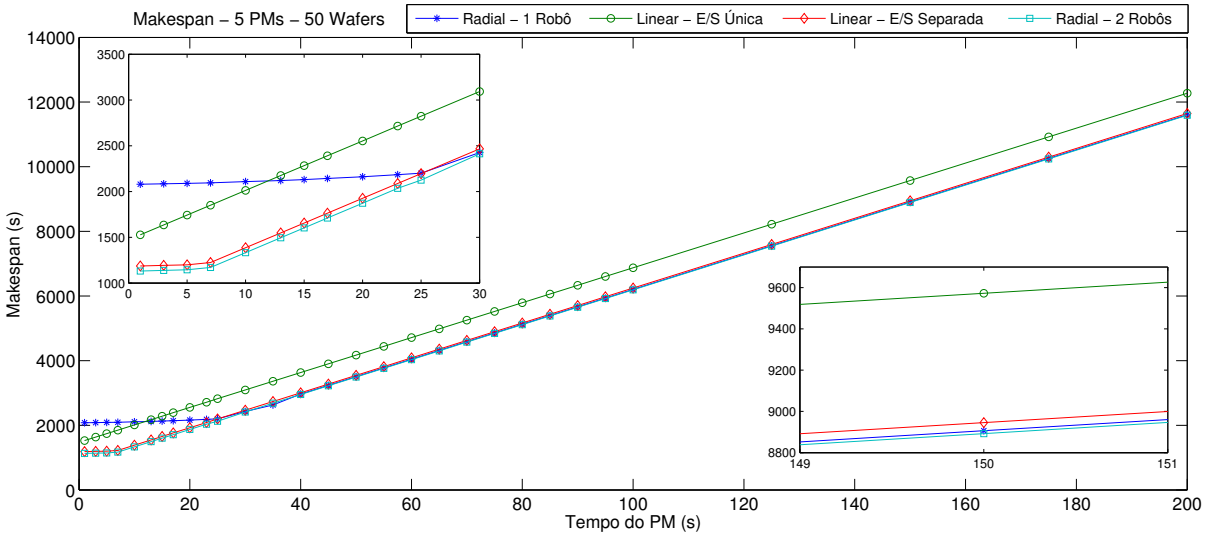


Figura 51 – *Makespan*: 5 PMs / lote de 50 *wafers*

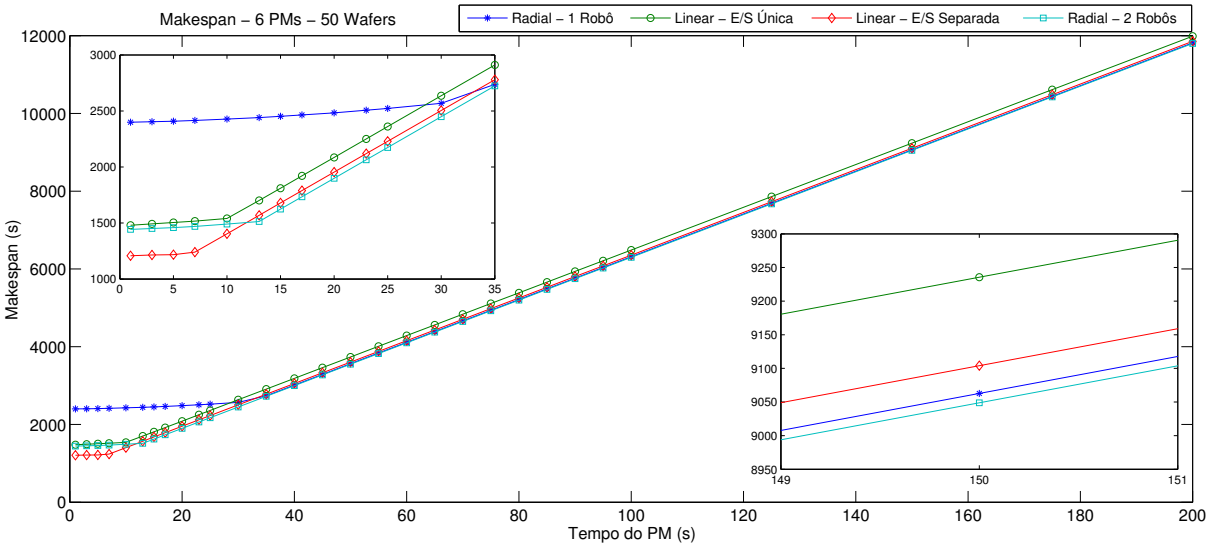


Figura 52 – *Makespan*: 6 PMs / lote de 50 *wafers*