UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Modelagem e Validação de Componentes de um Sistema Fabril por Teoria de Filas

Autor: Guilherme Augusto During

Orientador: Marcelo Götz

Porto Alegre, 8 de dezembro de 17

Sumário

Agradecimentos

Agradeço, primeiramente, à minha mãe Carla Düring, ao meu pai Frederico Düring e à minha avó Sônia Marnatti pelo amor, apoio e educação que me deram durante toda a minha vida.

Agradeço ao meu Irmão Frederico Düring Filho pela força e pelo companheirismo e aos meus grandes amigos adquiridos na universidade sem os quais eu jamais completaria este trabalho e a graduação.

Agradeço ao professor Marcelo Götz pelo suporte e auxílio na execução deste trabalho e a todos os bons professores que possuem o real desejo de ensinar os alunos.

Agradecimento aos familiares e amigos que possuo os quais

Resumo

A utilização de sistemas à eventos discretos se faz presente em diversos ambientes, especialmente fabris. Para a análise destes, utilizam-se diferentes técnicas de representação bem como a possibilidade de técnicas de controle de tais sistemas. Neste trabalho é apresentada uma forma de modelagem de um sistema à eventos discretos através do emprego de técnicas de filas. Ainda, utiliza-se também a representação por autômatos a fim de descrever o comportamento de máquinas modeladas a associar este a restrições para a elaboração de um controle supervisório, utilizado para validar a modelagem criada por filas. Tais tarefas são executadas através de *softwares* de simulação de sistemas, sendo estes: FlexFact para componentes fabris modelados; a plataforma de autômatos DESTool; o *software* de simulação de sistemas e sinais MATLAB/Simulink.

Abstract

The use of discrete event systems is present at different environments, especially in the industry. To analyze it, different techniques are used for its representation and implementation of control methods. This work presents a type of modeling for a discrete event system using queuing theory. In addition, automata representation is also used as a way to describe the behavior of the modelled machines and to associate them to restrictions in order to elaborate a supervisory control system, using this control to validate the created models. These tasks are executed via simulation softwares, them being: FlexFact for the modelled components; the automata platform DESTool; the signals and systems simulation software MATLAB/Simulink.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

ε - Conjunto vazio de eventos.

Σ - Conjunto de todos os possíveis eventos.

Σ_c - Conjunto de eventos controláveis.

Σ_u - Conjunto de eventos não controláveis.

Σ* - Conjunto de todas as sequências formadas pelos eventos de Σ.

δ - Função de transição δ: Σ x Q → Q.

q0 - Estado inicial.

Q - Conjunto de todos os possíveis estados.

Qm - Subconjunto de estados marcados onde Qm ϵ Q.

L_a - Linguagem admissível.

L_r - Linguagem restritiva.

Lista de Abreviaturas e Siglas

UFRGS - Universidade Federal do Rio Grande do Sul

MATLAB - MATrix LABoratory

SED - Sistema a Eventos Discretos

FIFO - First In, First Out

1 Introdução

Sistemas que executam tarefas de forma discreta podem ser chamados de Sistemas a Eventos Discretos, ou SED. Tal adequação pode proporcionar a estes o uso de ferramentas para modelagem e implementação de técnicas de controle, de forma que se possa obter maior eficiência de operações realizadas. O aumento de rendimento em uma linha de produção pode ser tomado como exemplo da utilização correta de controle.

Para este tipo de sistema, o qual trata da ocorrência de eventos discretos no tempo, pode-se utilizar algumas técnicas de modelagem, como Redes de Petri, Cadeias de Markov, Teoria de Filas, Teoria de Linguagem e Autômatos, entre outros, sendo que se emprega esta última para a utilização da Teoria de Controle Supervisório, amplamente aplicada, a nível acadêmico e em pesquisa, no controle de processos discretos.

1.1 Motivação

Considerando a ampla utilização e importância de SED no âmbito industrial, o estudo aprofundado neste tema é considerado relevante para a Engenharia de Controle e Automação. Ainda, o emprego de *softwares* de simulação nesta área e de sistemas de controle, como MATLAB, proporcionam melhor compreensão da utilização dos conceitos teóricos estudados quando aplicados a modelos práticos.

Toma-se ainda como incentivo o estudo em Teoria de Filas, área a qual aborda, além de outros conteúdos, a análise e modelagem de sistemas, podendo ser também aplicada ao estudo de SED.

1.2 Objetivos

A ideia proposta para o trabalho consiste na modelagem de componentes de um *software* de simulação fabril utilizando Teoria de Filas. A validação destes modelos será por meio da aplicação de técnicas de controle supervisório.

Seguindo este raciocínio, propõe-se para a resolução da proposta uma análise dos componentes do simulador FlexFact seguida da modelagem destes através do *software* MATLAB, utilizando o Simulink onde pode-se dispor de sua plataforma gráfica de blocos para a construção dos modelos de interesse. Com esta etapa executada, passa-se para uma verificação dos itens montados aplicando, nestes, rotinas de controle supervisório a fim de analisar se o comportamento das estruturas criadas junto à biblioteca SimEvents se assemelham ao equivalente simulado no próprio *software* de origem FlexFact.

1.3 Estrutura do Trabalho

O estudo apresenta seu desenvolvimento dividido em 7 principais capítulos. No capítulo seguinte é abordada uma fundamentação teórica básica necessária para a compreensão do trabalho. No capítulo Materiais e Métodos são listados os *softwares* empregados na parte prática. O desenvolvimento referente a proposta e aos métodos de resolução da mesma são apresentadas no Capítulo 4, onde se apresentam os componentes a serem modelados e a forma como estes serão criados via *software*, passando-se então para o Capítulo 5, onde são apresentados os modelos obtidos, sendo estes avaliados por meio da aplicação de técnicas controle, no Capítulo 6, onde também se mostram os resultados desta verificação. Por fim, em Conclusões e Trabalhos Futuros, Capítulo 7, mostra-se uma síntese dos resultados e aprendizado obtidos neste estudo, bem como ideias para trabalhos futuros envolvendo os temos abordados neste trabalho.

2 Revisão Bibliográfica

De forma a explicitar os temas constituintes do trabalho em questão, nas subseções seguintes será apresentada uma breve explanação sobre sistemas a eventos discretos bem como uma descrição sobre teoria de controle supervisório, utilizada para tais sistemas. Ainda, uma abordagem sobre teoria de filas se faz presente, visto que é um dos alicerces para o objetivo deste trabalho como descrito anteriormente.

2.1 Sistemas a Eventos Discretos

Se o espaço de estados de um sistema é descrito por um conjunto discreto como (0, 1, 2, ...) e as transições entre estes estados se dão em pontos distintos no tempo, pode-se associar tais transições a eventos e se referir a um sistema a eventos discretos (Cassandras, 1993). Para tal definição, pode-se tomar como exemplo de sistemas a eventos discretos um conjunto de semáforos em uma avenida, onde estes realizam transições discretas entre suas fases a fim de controlar o transito.

Este tipo de sistema pode ser descrito por alguns conceitos básicos e também definido segundo uma linguagem e modelo gráfico de autômatos, utilizados na teoria de controle supervisório.

2.1.1 Conceitos Básicos

Cassandras e Lafortune, 2008, descrevem um sistema a eventos discretos como um sistema que possui seu espaço de estados na forma discreta, diferente de um no qual compreende variáveis contínuas, onde geralmente estas estão atreladas à mudança no tempo. Para este conceito, tem-se a mudança de estados ligada a eventos assíncronos, onde a evolução dos estados do sistema depende diretamente da ocorrência destes. Nota-se que a mudança de estados está sempre associada a um evento, mas nem todo evento altera os estados de um sistema, como descrito por Götz, 2016.

Götz, 2016, ainda esclarece que para a modelagem e representação de sistemas a eventos discretos não são aplicadas técnicas utilizadas para sistemas de variáveis contínuas, uma vez que não é possível utilizar equações diferenciais ou de diferenças para descrever seu comportamento, onde, para este caso, se procuram equações que regem o comportamento de um sistema conforme a ocorrência de eventos.

Dentre alguns modelos utilizados para representar estes sistemas estão a teoria de linguagem e autômatos, abordados na seção seguinte, e a teoria de filas.

2.1.2 Linguagem e Autômatos

Em um modelo de SED, são de interesse conjuntos ou sequências de eventos que este pode gerar. Desta forma, pode-se denotar Σ como o conjunto finito de eventos deste sistema, seu *alfabeto*, e Σ* como o conjunto de todas as sequências possíveis dos elementos de Σ, incluindo o conjunto vazio ε. Se tomarmos uma sequência $u = \sigma_1 \sigma_2 ... \sigma_k$ pertencente a Σ*, esta representará um caminho parcial possível, julgando-se parcial pelo fato de após σ_k ainda haver a possibilidade de outros eventos. Ainda, um conjunto de todos os caminhos admissíveis e fisicamente possíveis pode ser denotado como L e pertencer a Σ*, também podendo ser considerado como uma *linguagem* gerada pelo *alfabeto* Σ (Ramadge e Wonham, 1989).

Schlick, 2013, especifica algumas operações possíveis com linguagens e sequências, bem como características destas. Considerando o conjunto de eventos E, pode-se considerar o conjunto E* como o fechamento Kleene do conjunto E, sendo que este constitui todas as combinações possíveis entre os elementos de E. As equações 1 e 2 mostram o conjunto E e seu fechamento Kleene. Para um sistema, temos que:

$$\mathrm{E} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\} \tag{1}$$

$$\mathrm{E}^* = \{\varepsilon, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{aa}, \mathrm{ab}, \mathrm{ac}, \mathrm{ba}, \mathrm{bb}, \mathrm{bc}, \mathrm{ca}, \mathrm{cb}, \mathrm{cc}, \mathrm{aaa}, \mathrm{aab}, \dots\} \tag{2}$$

Tratando-se de uma sequência, ou palavra, s = tuv, onde t, u e v ϵ E*, pode-se considerar t como o prefixo, u como subsequência e v como o sufixo desta palavra. Deve-se ainda considerar que ε é também um sufixo de s.

Outra importante relação matemática descrita pelos mesmos autores é o prefixo fechamento, o qual indica que, para uma linguagem L, $\bar{\mathrm{L}}$ irá possuir todos os prefixos possíveis, como mostrado nas equações 3 e 4.

$$\mathrm{L} = \{\mathrm{abc}\} \tag{3}$$

$$\bar{L} = \{\varepsilon, \mathrm{a}, \mathrm{ab}, \mathrm{abc}\} \tag{4}$$

Para uma linguagem L a qual representa todas as trajetórias possíveis de um sistema e caracterizam completamente este sistema, esta linguagem é necessariamente uma linguagem prefixo fechada (Košecká, 1992).

Košecká, 1992, ainda explica que tal linguagem característica pode ser representada por uma máquina que gera todas as sequências de L. Desta forma, o gerador, G, pode ser representado como mostrado a seguir.

$$\mathrm{G} = (\mathrm{Q}, \Sigma, \delta, \mathrm{q}_0, \mathrm{Qm}) \tag{5}$$

Onde:

- Q é o conjunto de todos os possíveis estados;
- Σ o conjunto de todos os possíveis eventos;
- δ representa a função de transição δ: Σ x Q → Q;
- q_0 indica o estado inicial;
- Q_m constitui o subconjunto de estados marcados onde Q_m ϵ Q.

Outros autores, como Hopcroft, Motwani e Ullman, 2001, utilizam a mesma representação para descrever um autômato de estados finitos, podendo-se assim caracterizar suas informações básicas em uma equação considerando os conjuntos indicados.

Para representar uma linguagem, utiliza-se um autômato onde este pode ser apresentado na forma de um grafo. Neste, sendo um modelo de diagrama de transição de estados, tem-se que cada nó representa um estado do sistema e cada arco originado de cada nó representa uma transição rotulada pelos eventos contidos no alfabeto. Dos

estados representados, destacam-se o estado inicial, apontado por uma seta sem origem especificada, e os estados marcados, utilizados de forma adequada ao desejado conforme a modelagem feita do sistema (Götz, 2016). A Figura 2.1 exemplifica um autômato.

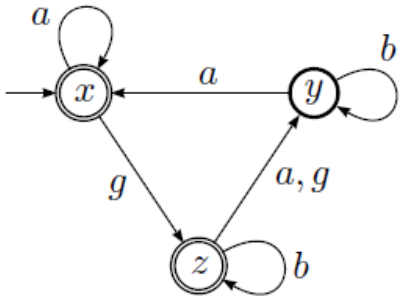


Figura 2.1 - Diagrama de transição de estados.

Fonte: Götz, 2016.

2.2 Teoria de Filas

Pode-se descrever um sistema de filas por clientes chegando para serem atendidos por um serviço, esperando por tal serviço em caso do atendimento não ser imediato e deixando o sistema após serem atendidos. O termo “clientes”, na teoria de filas, não se atem somente a pessoas, mas sim a um uso geral. A fim de exemplificar, toma-se um avião esperando em uma fila para decolar ou um programa de computador esperando para ser executado (Gross et al, 2008). Um exemplo pode ser visto na Figura 2.2 a seguir.

Cassandras, 1993, lista algumas propriedades que podem definir sistemas de filas. Estas são os padrões de chegada de clientes e atendimento destes nos serviços, parâmetros estruturais como capacidade da fila e número de servidores, e o modo de operação, tratando-se do modo de atendimento, prioridades, entre outros fatores.

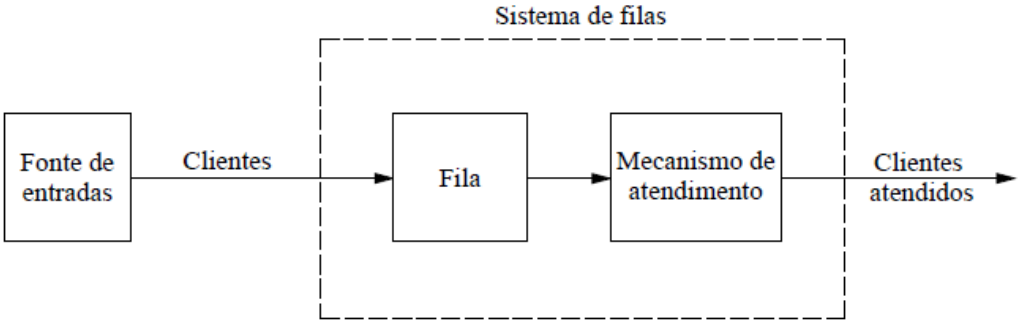


Figura 2.2 - Exemplo de sistema de fila.

Fonte: Hillier e Lieberman, 2006.

Das características citadas, Zukerman, 2017, descreve que os intervalos de tempo entre chegadas consecutivas são geralmente relacionados ao processo de Poisson, podendo também se adequar a um modo determinístico. Ainda, são considerados os modos: Geral, onde não se conhece o processo de distribuição de tempo; Geral independente, no qual o processo de distribuição é de forma independente; Identicamente distribuído em um processo de renovação, em que as variáveis randômicas têm a mesma probabilidade de ocorrência.

Quanto ao número de servidores e armazenamento, Cassandras, 1993, indica que estes são fatores a serem representados numericamente, sendo que para alguns sistemas, pode-se admitir infinitos servidores e capacidade das filas. Ainda, admite-se que a capacidade de armazenamento do sistema também inclua o espaço disponível para clientes quando em atendimento por servidores.

Em relação à disciplina de atendimento dos servidores, Adan e Resing, 2015, explicam que o modo de atendimento em um sistema de filas uma pode ser do tipo onde o primeiro a chegar é o primeiro a ser atendido, ordem randômica de atendimento, último a chegar é o primeiro a ser atendido e sistema de prioridades.

2.3 Teoria de Controle Supervisório

Para um dado modelo de sistema a eventos discretos, este consiste apenas em um gerador de sequências de eventos, sem que haja relação com controle externo. Desta forma, para iniciar a construção de um controle supervisório, pode-se estabelecer que alguns eventos do sistema sejam desabilitados, evitando que ocorram. Seguindo este raciocínio, se estabelece, a fim de modelar o controle de um SED, a divisão do conjunto de eventos Σ em eventos controláveis e não-controláveis, como $\Sigma = \Sigma_u \cup \Sigma_c$, onde os eventos controláveis Σ_c podem ser desabilitados a qualquer momento, enquanto os eventos não-controláveis Σ_u não podem sofrer influência alguma do controle implementado (Ramadge e Wonham, 1989).

Assim, essas restrições a certos eventos podem ser utilizadas para formular uma especificação de comportamento desejado e então modelada como um autômato, o qual, sincronizado com o modelo do sistema, pode vir a fazer este a atuar somente dentro de um comportamento desejado. Desta forma, tem-se uma linguagem admissível de um sistema controlado, L_a, e a linguagem restritiva, L_r, ambas sub linguagens de G, sendo este o sistema a ser controlado (Götz, 2016). Desta forma, pode-se considerar

$$Lr \subseteq La \subseteq L(G) \tag{6}$$

onde $L(G)$ consiste na linguagem gerada pelo sistema G.

A Figura 2.3 representa o modo como o controle supervisório atua sobre um SED. O supervisório S atua observando o progresso do sistema G, fazendo isto por meio da monitoração de, se possível, todos os eventos que o sistema executa, podendo estimar em qual estado o sistema se encontra. O controle em si é realizado vetando-se certos eventos, a fim de evitar que o sistema percorra uma sequência de eventos indesejada (Götz, 2016).

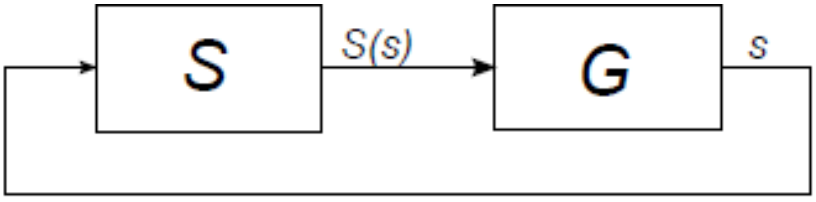


Figura 2.3 - Sistema G sendo controlado por um supervisório S.

Fonte: Götz (2016)

3 Materiais e Métodos

O trabalho em questão se constitui em boa parte na utilização de *softwares* para a resolução da problemática proposta. Desta forma, foram utilizados três programas para fins de modelagem e simulação de modelos construídos, sendo estes o FlexFact, DESTool e MATLAB, onde este último é utilizado na modelagem dos componentes de interesse e implementação de controle supervisório.

3.1 *Software* de Simulação Fabril FlexFact

O simulador FlexFact apresenta um ambiente onde é possível montar e simular um sistema fabril com componentes comuns em uma indústria como esteiras, máquinas, distribuidores e outros, vistos mais especificamente na Seção 4.2. Uma vez que este é direcionado a sistemas a eventos discretos, o programa informa os eventos que provém da atuação dos componentes utilizados, também dispondo de comunicação com meio externo a fim da aplicação de controle sobre o sistema criado. Na Figura 3.1 é possível verificar o ambiente o qual o simulador dispõe.

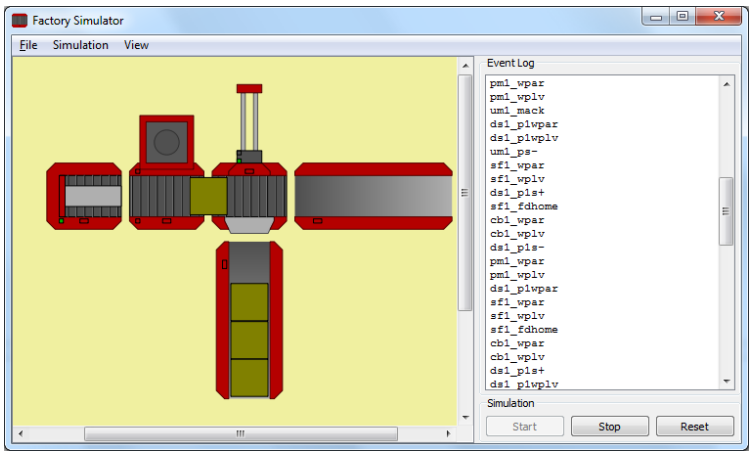


Figura 3.1 - *Software* FlexFact.

Fonte: Autor

Pode-se visualizar no painel à direita da Figura 5 uma lista a qual indica, de cima para baixo, a ocorrência dos eventos do sistema enquanto este opera em tempo real, exibindo a movimentação de peças e atuação de seus componentes.

3.2 DESTool

Para modelar o comportamento de SED pela técnica de autômatos e ainda construir sistemas de controle supervisório, utiliza-se o *software* DESTool onde pode-se interagir diretamente com autômatos, criando-se linguagens e modelos para os comportamentos de sistemas. Este programa possibilita o uso de comunicação pelo protocolo próprio Simplenet junto ao FlexFact, a fim de estabelecer o controle de plantas fabris criadas no simulador. Desta forma, pode-se fazer um paralelo entre um controle em funcionamento entre estes dois programas e um similar gerado no Simulink, a fim de verificar a fidelidade dos modelos criados com suas bibliotecas. Na Figura 3.2 é exibida a interface de trabalho encontrada no DESTool, mostrando um exemplo genérico de sistema.

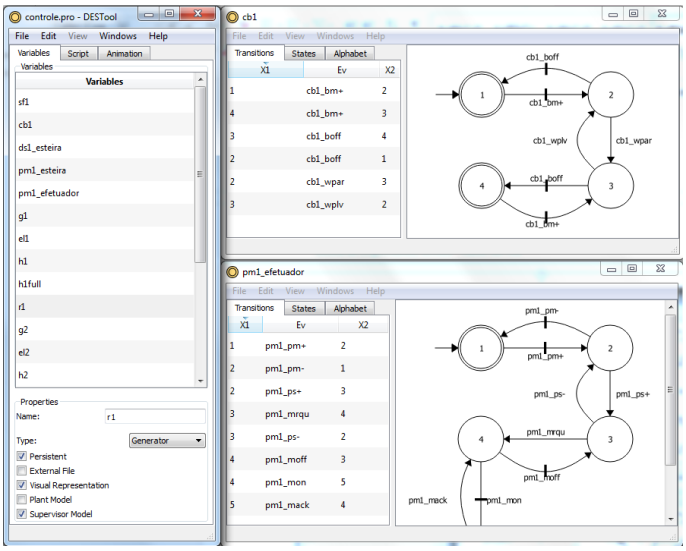


Figura 3.2 - Interface de trabalho do *software* DESTool.

Fonte: Autor

3.3 MATLAB R2017a

Utilizado para o modelamento dos componentes do FlexFact, MATLAB é amplamente empregado nos campos de cálculos matemáticos, sistemas e sinais, controle, entre outros temas relacionados. Inserido no *software*, está a ferramenta Simulink a qual dispõe de diversas bibliotecas a fim de modelar e simular sistemas dinâmicos com a utilização de uma interface gráfica de blocos, sendo SED uma possibilidade.

Tratando-se especificamente do tema deste trabalho, a biblioteca SimEvents, inclusa no Simulink, possui blocos relacionados à teoria de filas, os quais são utilizados na modelagem dos componentes fabris do simulador FlexFact. Na Figura 3.3 pode-se verificar alguns desses blocos. Cabe ressaltar que a modelagem a ser realizada também envolve outras bibliotecas, a fim de lidar com sinais de atuação dos componentes modelados.

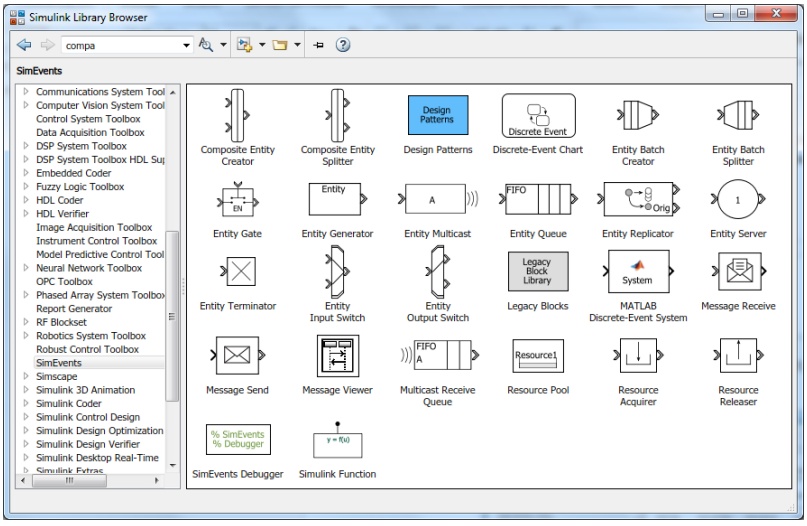


Figura 3.3 - Biblioteca SimEvents, presente no Simulink.

Fonte: Autor

4 Metodologia e Desenvolvimento

Nesta seção será mostrada a metodologia empregada na modelagem dos componentes, abordando os blocos utilizados do Simulink, o funcionamento do FlexFact, e o método utilizado para validar o comportamento dos modelos criados em relação ao obtido no *software* original.

4.1 Blocos da Biblioteca SimEvents

A proposta principal do trabalho consiste na utilização de teoria de filas para a implementação dos componentes do FlexFact no Simulink, fazendo uso dos blocos presentes na biblioteca SimEvents. Os blocos em questão operam com a passagem de entidades entre si, como clientes de uma fila, sendo possível o acúmulo destas entidades em blocos análogos a uma fila ou fazer com que estes esperem em um bloco por um tempo até serem novamente deslocados, simulando uma operação ou serviço. Na Figura 4.1 pode-se visualizar os dois exemplos citados, onde “Entity Queue” é a representação de uma fila, e “Entity Server” uma alocação de serviço.

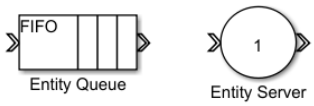


Figura 4.1 – Blocos de fila e serviço.

Fonte: Autor

Cabe ressaltar que a capacidade dos blocos pode ser alterada, modificando-se o tamanho de uma fila. Para o bloco de servidor, seu tempo de serviço pode ser também modificado fazendo com que operações temporizadas possam ser modeladas com esta biblioteca. Desta forma, pode-se comparar uma peça do simulador FlexFact como uma entidade que percorre pelos blocos, aguardando por um processamento ou para passar de uma esteira de uma máquina a outra.

A fim de inserir entidades no sistema, dois blocos podem ser utilizados, sendo estes o bloco gerador de entidades e o de envio de mensagem, ou entidade. O primeiro, “Entity Generator”, gera entidades a partir de períodos de tempo, enquanto o segundo, “Message Send”, as gera a partir de sinais externos. Nota-se que o bloco de envio de mensagem pode controlar, a partir de um sinal de entrada, o momento em que se deseja gerar uma entidade. A Figura 4.2 mostra os blocos em questão.

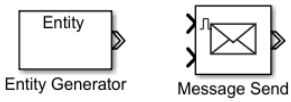


Figura 4.2 – Bloco gerador de entidades e de envio de mensagem.

Fonte: Autor

Ainda, estes blocos principais contam com opções de verificação de valores internos como, por exemplo, número de entidades que saíram e chegaram em um bloco ou número de entidades presentes atualmente no bloco. Com estas informações é possível caracterizar um comportamento e gerar um evento associado a um componente do simulador FlexFact, como a chegada de uma peça à posição central de uma esteira, o que dispara um evento.

A Figura 4.3 mostra um exemplo básico da passagem de entidades entre blocos, gerando sinais onde alguns podem ser caracterizados como eventos. Neste, um gerador de pulsos, Gráfico 1, estimula a geração de entidades pelo bloco “Message Send”. Este envia instantaneamente as entidades ao bloco “Entity Queue” caso haja disponibilidade de espaço, repetindo o mesmo processo para a transição deste ao bloco “Entity Server”. Nota-se que o primeiro pulso, em 0 s, não gera mudança de valor no Gráfico 2, já que a entidade não permanece no bloco de fila pelo fato de haver disponibilidade de serviço. Já aos 4 s o servidor continua ocupado, fazendo com que uma entidade gerada tenha que permanecer armazenada no bloco de fila, gerando uma borda de subida e consequentemente um sinal no Gráfico 3, que pode ser caracterizado como um evento de chegada. De forma semelhante, quando o servidor está disponível, uma entidade deixa o bloco, gerando uma borda de descida e disparando também um sinal, podendo ser identificado como outro evento, Gráfico 4.

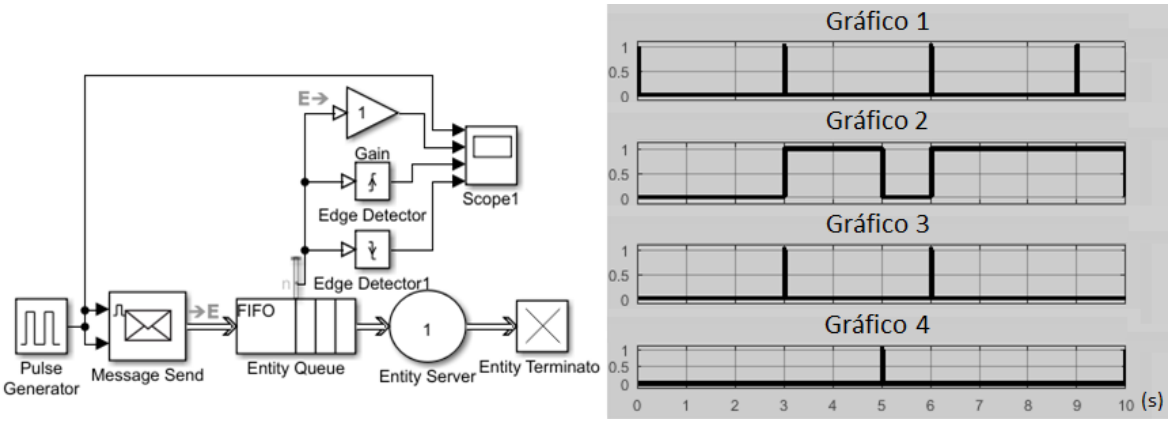


Figura 4.3 – Exemplo de sistema de filas e geração de sinais.

Fonte: Autor

Comparando, ainda na Figura 4.3, os gráficos 1 e 2, nota-se que aos 9 s é disparado um novo sinal para geração de entidade, mas, pelo fato de a fila estar completamente ocupada, esta não é adicionada, fazendo com que uma nova entidade só possa ser inserida após os 10 s, instante o qual há disponibilidade de espaço na fila.

Outros blocos da biblioteca foram utilizados a fim de completar o modo de operação desejado, sendo estes “Entity Gate”, “Entity Input Switch” e “Entity Output Switch”, definidos a seguir:

“Entity Gate” – Permite a passagem de entidades quando uma outra entidade é recebida. Para esta forma de operação, o mesmo deve estar no modo “Release Gate”, indicado por “REL” no bloco;

“Entity Input Switch” – Permite que entidades de diferentes fontes sigam um mesmo caminho;

“Entity Output Switch” – Utilizado para que uma entidade possa seguir mais de um caminho ao sair de um bloco.

Os três blocos podem ser verificados na Figura 4.4.

Para lidar com manipulação e geração de sinais, blocos comumente utilizados no Simulink também foram empregados, para que fosse possível trabalhar com lógica de sinais e geração de eventos. A Figura 4.4 também mostra estes blocos.

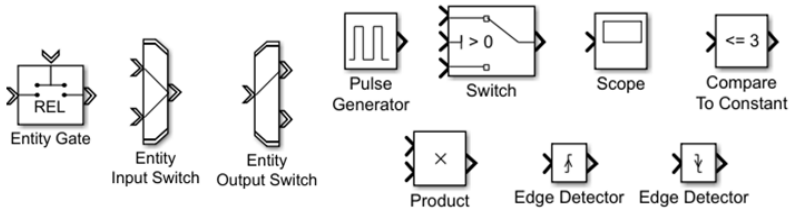


Figura 4.4 – Blocos de trânsito de entidades e para operações com sinais.

Fonte: Autor

4.2 Componentes do Simulador FlexFact

A fim de simular de forma fiel uma planta fabril, o *software* FlexFact dispõe de componentes que representam diferentes máquinas, como esteiras, máquinas de processos, distribuidores e um alimentador de itens os quais se movimentam pelos componentes.

Cada componente possui um conjunto de eventos para as ações ocorridas durante sua operação, bem como um conjunto de sinais que representam a condição de um processo como, por exemplo, indicar se uma máquina está processando ou se uma esteira está em movimento. Estes sinais são ativados e desativados por eventos controláveis, fazendo possível o controle destes sinais de operação desabilitando tais eventos e habilitando-os quando desejado. Na Figura 4.5 é mostrado o componente Conveyor Belt, o qual consiste em uma esteira com dois sentidos de operação.

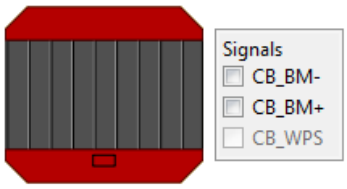


Figura 4.5 – Componente Conveyor Belt.

Fonte: Autor

Ainda, na mesma figura, é mostrado um quadro indicativo dos sinais de operação da esteira, sendo os dois primeiros de acionamento do seu movimento e o terceiro para indicar a presença de peças na posição central. A relação entre um evento e um sinal pode ser exemplificada, para este componente, pela ocorrência do evento cb_bm+ para ativar o sinal CB_BM+, ligando a esteira, e a ocorrência do evento cb_boff para desativar o sinal CB_BM+, cessando o movimento. Pode-se notar que os sinais associados a eventos controláveis são acessíveis ao usuário, enquanto os sinais relacionados a eventos não controláveis não podem ser ativados manualmente. Na Todos os componentes do simulador possuem a mesma condição de operação para o sensor de presença e, com exceção dos componentes Exit Slide e Stack Feeder, todos também possuem a mesma mecânica de operação de suas esteiras de movimento, diferenciando-se apenas pela nomenclatura de eventos e sinais.

Tabela 4.1 são listados, para o componente ConveyorBelt, todos os sinais e seus respectivos eventos associados.

Todos os componentes do simulador possuem a mesma condição de operação para o sensor de presença e, com exceção dos componentes Exit Slide e Stack Feeder, todos também possuem a mesma mecânica de operação de suas esteiras de movimento, diferenciando-se apenas pela nomenclatura de eventos e sinais.

Tabela 4.1 – Sinais e eventos do componente Conveyor Belt.

Fonte: Autor

Sinal	Evento Associado	Descrição do Evento
CB_BM-	cb_bm-	Ativa o sinal CB_BM-, ligando a esteira para retorno.
	cb_boff	Desativa o sinal CB_BM-, deligando a esteira.
CB_BM+		Desativa o sinal CB_BM+, deligando a esteira.
	cb_bm+	Ativa o sinal CB_BM+, ligando a esteira para avanço.
CB_WPS	cb_wpar	Ocorre quando o sinal CB_WPS é ativado, indicando a chegada de uma peça na posição central.
	cb_wplv	Ocorre quando o sinal CB_WPS é desativado, indicando a saída de uma peça da posição central.

A Figura 4.6 mostra o componente Processing Machine, utilizado para processar peças que passam pela planta. Como dito, este segue a mesma ideia do componente Conveyor Belt, diferenciando-se pela presença de uma ferramenta de operação, a qual também possui sinais e eventos associados. A Figura 4.6 ainda indica a presença de dois sensores à esquerda, em verde, os quais indicam o posicionamento da ferramenta na posição inicial, fora dela e na posição de operação. Estes sensores geram os sinais PM_PS+ e PM_PS-, descritos na Tabela 2, presente no Apêndice.

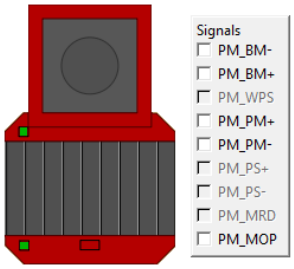


Figura 4.6 – Processing Machine e seu quadro de sinais.

Fonte: Autor

Os componentes que consistem no início e final do ciclo de uma peça são o Stack Feeder (Alimentador) e o Exit Slide, componente de saída de peças. Ambos destoam da maioria dos componentes, pois a movimentação de peças nestes se dá de forma específica, como explicado a seguir.

O Stack Feeder, Figura 4.7, é o único componente capaz de introduzir peças na planta simulada, podendo armazenar até 9 unidades de uma só vez, liberando-as uma a uma

dependendo da disponibilidade do componente posicionado adjacentemente. A esteira de movimentação possui uma divisão onde só é possível movimentar uma peça por vez até que esta esteja completamente fora de seu espaço. Os sinais e eventos deste componente constam no Apêndice, mostrados na Tabela 3, que descreve o modo de funcionamento.

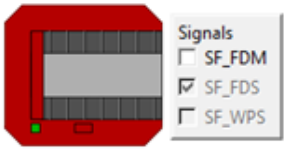


Figura 4.7 – Componente Stack Feeder.

Fonte: Autor

O componente o qual constitui o final da linha de produção, sendo o mais simples, seria o Exit Slide, onde as peças são retiradas por meio de comandos manuais, eliminando peças uma por vez ou de forma total. Este é visto na Figura 4.8, junto ao seu indicador de sinais, o qual conta somente o sinal do sensor de presença, associado somente a eventos não controláveis. A Tabela 4 mostrando a relação de eventos e sinais deste componente pode ser vista no Apêndice.

Figura 4.8 – Componente Exit Slide.

Fonte: Autor

5 Modelos Elaborados

A modelagem executada em relação aos componentes corresponde ao comportamento dos mesmos quando se considera a geração e execução de eventos e condição dos seus sinais. Desta forma, pode-se obter uma representação próxima a original, onde se obedece ao mesmo comportamento do simulador FlexFact, e assim avaliar os modelos analisando o comportamento destes quando operando de maneira controlada.

A fim de obter um desempenho fiel ao original, quanto à operação das máquinas, foram considerados dois níveis de modelagem, onde um representa a mecânica de movimentação de peças pelas máquinas, enquanto que o outro procura desempenhar as funções de sinais e eventos gerados e/ou recebidos.

5.1 Conveyor Belt

Como já abordado, o comportamento das esteiras presentes na maioria dos componentes obedece à mesma estrutura encontrada na máquina Conveyor Belt. Deste modo, executando a modelagem de forma detalhada, outros modelos teriam de ser acrescidos somente de suas operações específicas. Um panorama do diagrama de blocos montado para o componente Conveyor Belt é mostrado na Figura 5.1, onde este é explicado em seguida, por partes. O mesmo encontra-se também no Apêndice do trabalho, para melhor visualização.

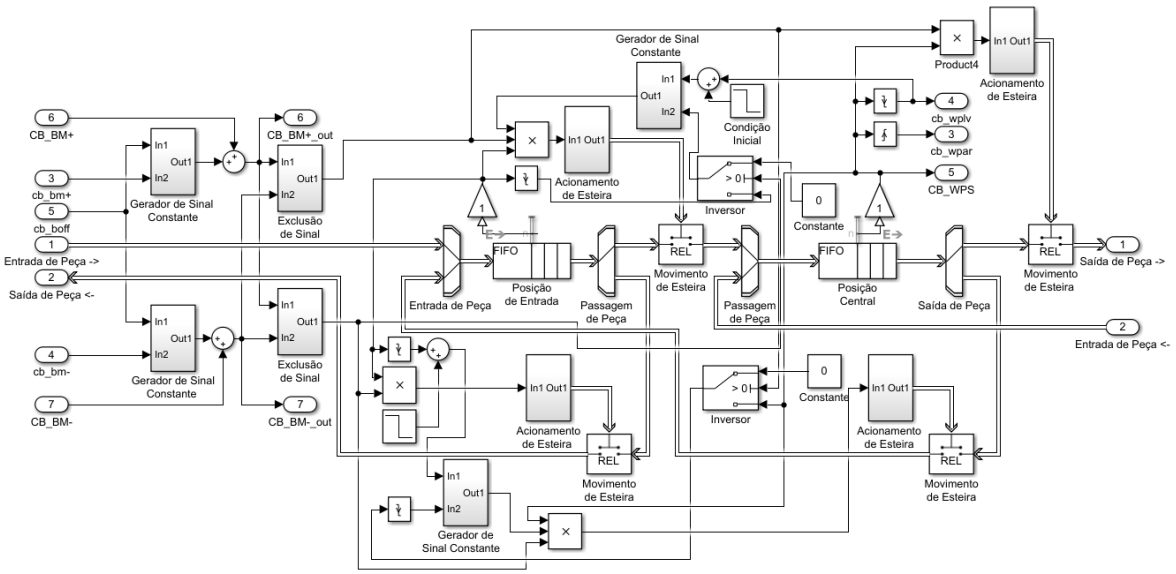


Figura 5.1 – Panorama geral da modelagem por blocos de filas do Conveyor Belt.

Fonte: Autor

No simulador FlexFact, as esteiras dos componentes abrigam, individualmente, duas peças sob sua extensão. Portanto, um bloco modelado deve abrigar duas entidades, de forma que os sinais e eventos de presença de peça sejam gerados quando há tal ocorrência. Na Figura 5.2, pode-se verificar a posição das peças na esteira de forma que uma, na posição central, ativa o sinal de presença e outra se encontra posicionada na entrada do bloco.

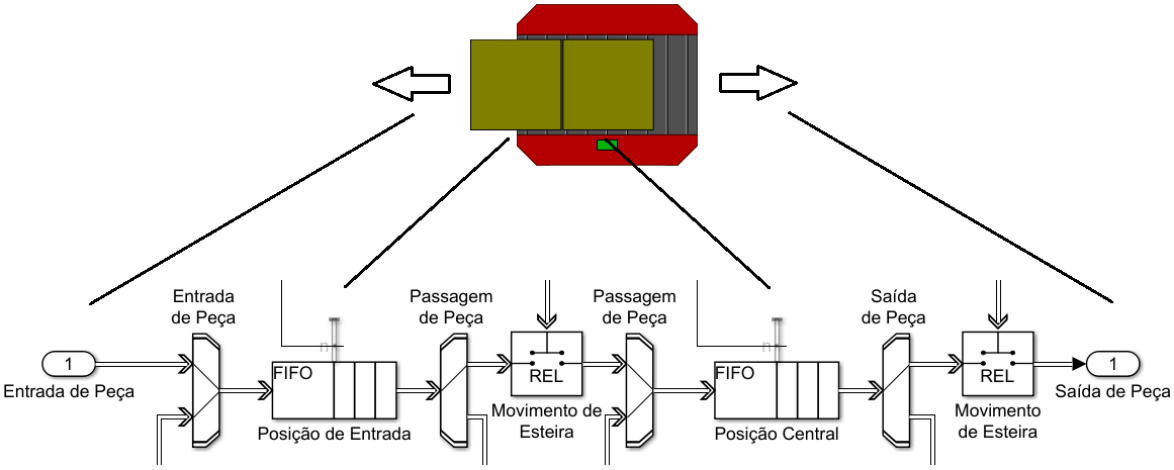


Figura 5.2 – Posição de peças no Conveyor Belt e seu modelo por blocos de filas.

Fonte: Autor

Como as posições extremas de saída e entrada de peças nas máquinas são compartilhadas entre máquinas adjacentes, o modelo construído assume que uma peça na entrada já pertença à máquina, enquanto uma peça na posição de saída faça parte da posição de entrada da próxima, fazendo esta então parte de outro modelo. O movimento das peças na esteira passagem entre modelos se dá pela operação dos blocos Entity Gate, nomeados Movimento da Esteira, ainda na Figura 5.3. Assim que recebem uma mensagem, liberam a passagem de uma entidade ao bloco seguinte.

Nota-se ainda a presença de blocos para a divisão ou união de caminhos, usados para distinguir os movimentos de avanço ou retorno da esteira. A Figura 5.3 mostra a disposição de outros Entity Gates dispostos a fim de permitir o movimento de retorno da esteira.

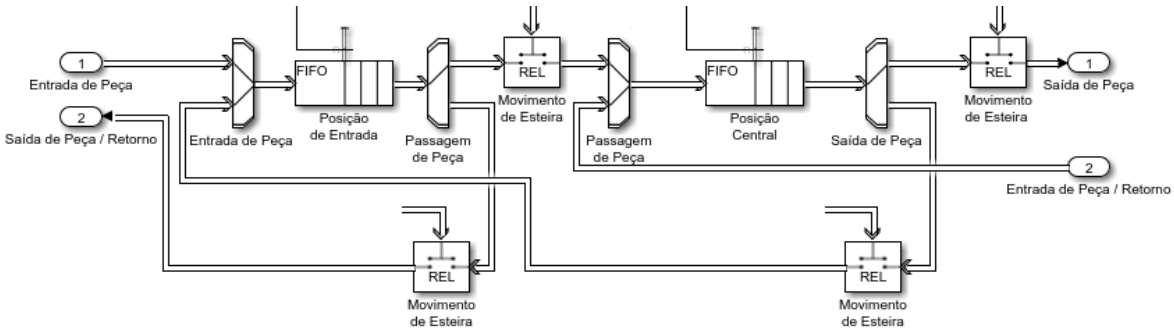


Figura 5.3 – Entity Gates posicionados a fim de permitir dois sentidos de movimento.

Fonte: Autor

Nesta montagem, quando acionado o par de avanço de peças, estas se deslocam à direita, enquanto se acionado o par de retorno, as entidades retrocedem aos blocos anteriores. A operação da esteira para avanço, no simulador, é associada ao sinal CM_BM+ e este acionado pelo evento cb_bm+ e desativado por cb_boff, onde esta estrutura de operação deve se manter fiel na estrutura de blocos. Para o funcionamento deste sinal em conjunto aos seus eventos, foi criado um bloco o qual gera um sinal constante a partir do recebimento de eventos. Este bloco é mostrado na Figura 5.4.

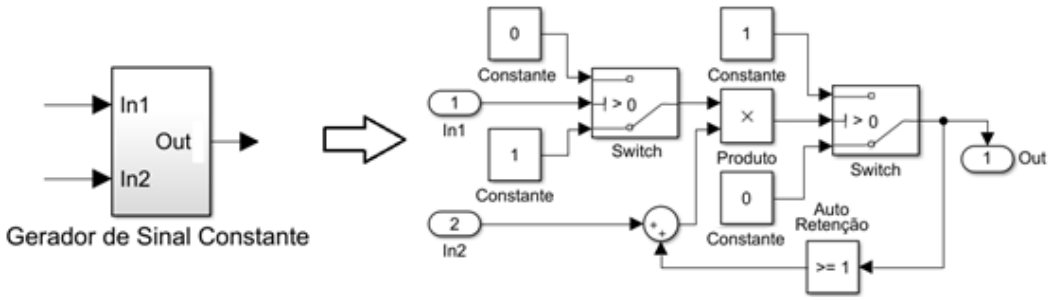


Figura 5.4 – Gerador de sinal constante.

Fonte: Autor

O funcionamento deste ocorre pela seguinte forma: Se um evento é recebido pela entrada 2, este é multiplicado pela saída do bloco Switch à esquerda, inicialmente em 1, já que este somente comuta sua saída para a porta superior quando sua condição central é satisfeita. O resultado deste produto satisfaz a condição central do Switch à direita, de forma que sua saída, agora igual a 1, confirme o bloco Auto Retenção, onde este emite um sinal unitário constante, mantendo uma lógica de memorização do sinal. Para interromper esta saída constante, um sinal impulsivo na entrada 1 comuta o bloco Switch à esquerda para 0, resultando em um produto 0, comutando novamente o Switch à direita para 0 e derrubando a auto retenção antes estabelecida.

Este bloco que gera um sinal constante a partir de sinais impulsivos, ou eventos, é utilizado não somente na relação entre eventos e sinais de máquinas, mas também empregado na operação de outras partes da modelagem, como mostrado na Figura 5.5, onde se apresenta o esquema da operação do movimento da esteira.

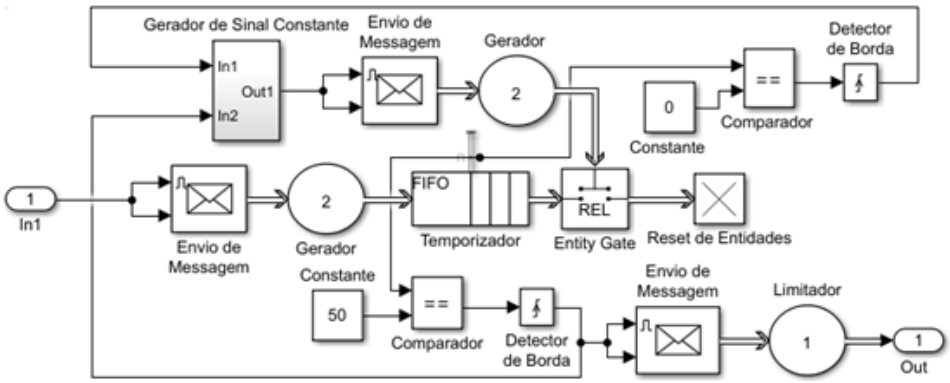


Figura 5.5 – Esquema de acionamento de esteiras.

Fonte: Autor

Esta montagem simula uma temporização do avanço de uma peça pela esteira, fazendo isso pelo uso em conjunto de serviços e seus tempos escolhidos. A entrada do bloco, In1, consiste no sinal CB_BM+ (ou CB_BM-, para retorno), fazendo com que entidades sejam geradas em intervalos constantes de tempo pelos blocos Envio de Mensagem e Gerador, sendo estas recebidas pelo bloco de fila Temporizador. Quando este atinge um limite pré-estabelecido, indicando o tempo necessário para o avanço de uma peça na esteira, uma mensagem é enviada ao diagrama principal permitindo que uma entidade se mova adiante. Neste mesmo momento, o mesmo sinal garante que um processo semelhante reinicie o contador, esvaziando o bloco de fila Temporizador.

Além da execução do movimento, a importância deste bloco se baseia na necessidade de separar por tempo a ocorrência de eventos, já que a passagem contínua de entidades pelos blocos de fila não gera alterações de sinais de saída. Fato semelhante é mostrado na Figura 4.3, onde a passagem de uma entidade por um bloco não gera sinais, a menos que essa permaneça neste por um período.

Ainda, tratando dos sinais de avanço e retorno da esteira, é necessário o cancelamento de quaisquer movimentos quando ambos os sinais estão ativos. Para o Conveyor Belt, se os eventos cb_bm+ e cb_bm- acontecem em sequência, a esteira deve ligar e travar em seguida, visto que os dois sinais, CB_BM+ e CB_BM-, estão ativos. A Figura 5.6 mostra a ligação feita para tal cancelamento.

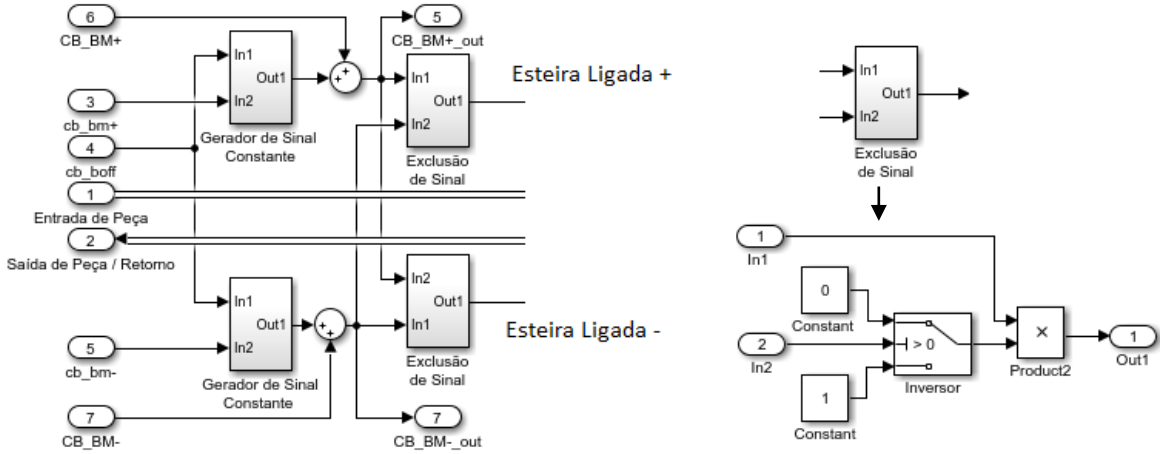


Figura 5.6 – À esquerda, ligação dos sinais e, à direita, bloco de cancelamento de sinais simultâneos.

Fonte: Autor

Por fim, o acionamento da esteira depende de outros sinais além do principal, sendo estes: Sinal de presença na posição atual; Confirmação de que uma entidade deixou o bloco seguinte. Os três sinais são indicados na Figura 5.7 pelas letras a), b), e c). Nota-se a necessidade de uma condição inicial para o sinal b), uma vez que nenhuma peça tenha passado antes da primeira.

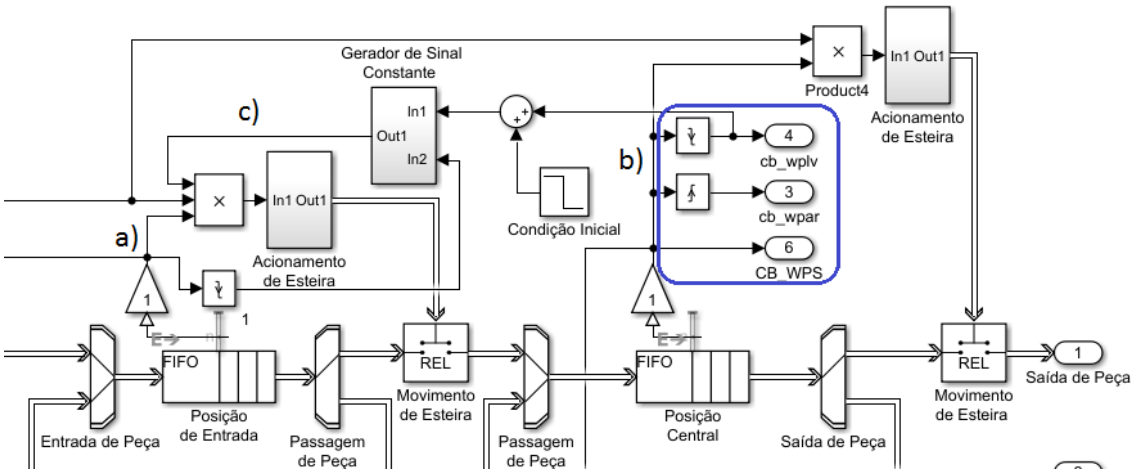


Figura 5.7 – Sinais usados no o funcionamento do avanço de entidades.

Fonte: Autor

5.2 Processing Machine

O segundo componente modelado pela biblioteca SimEvents, a máquina Processing Machine, utiliza o mesmo mecanismo de esteira do Conveyor Belt, diferindo somente pelos nomes de sinais e eventos associados à movimentação em si. Portanto, a modelagem abordada a seguir compreende somente a operação de processamento da máquina. Cabe ressaltar que as operações de movimento da esteira e processamento de peças são desacopladas, podendo ocorrer ambas ao mesmo tempo.

A primeira etapa da operação na máquina se dá pelo avanço do atuador sobre a máquina, sendo este procedimento, como já explicado, associado a sinais e eventos. A Figura 5.8 mostra a montagem de blocos construída a fim de representar este mecanismo.

Neste modelo, o evento pm_pm+ aciona o sinal constante PM_PM+, o qual gera entidades de forma temporizada, preenchendo a fila do bloco Temporizador. Quando este valor chega ao limite, sendo este estabelecido de forma a representar o tempo em que esta ação leva no simulador FlexFact, os eventos pm_ps+ e pm_mrqu são gerados, indicando a chegada na posição e operação e que o processamento de peça está pronto para começar.

O evento pm_pm- e seu sinal associado trabalham de forma semelhante, porém comutando o bloco de passagem Reset e esvaziando a fila do Temporizador, representando o recuo da ferramenta até aposição inicial, disparando o evento pm_ps- no fim de curso.

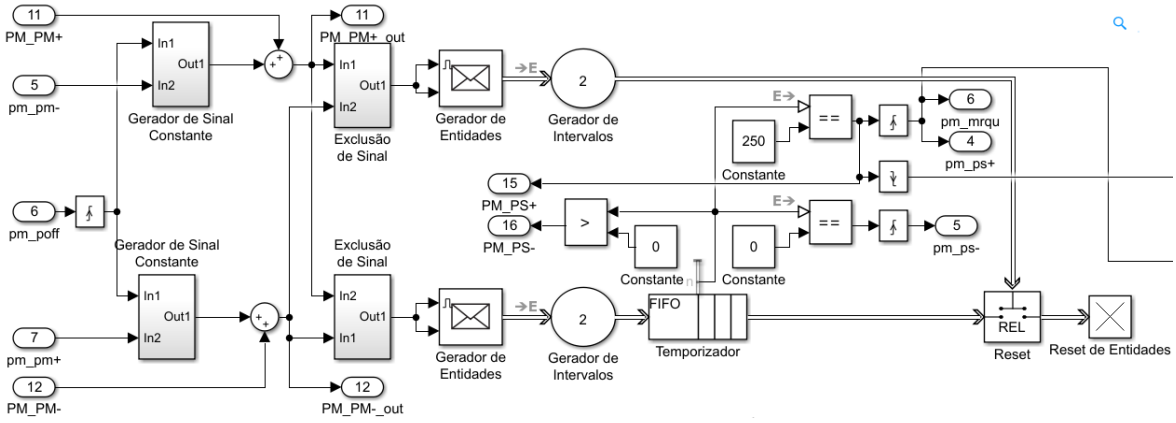


Figura 5.8 – Montagem de blocos para o avanço e recuo do operador do componente Processing Machine.

Fonte: Autor

A operação de processamento segue uma montagem semelhante ao esquema do avanço do atuador, diferenciando-se em alguns pontos. Na Figura 5.9 pode-se verificar a presença de três geradores de sinal constante, onde operam da seguinte forma:

Nº 1 – Recebe os eventos pm_mon e pm_moff os quais ligam e desligam o operador da máquina através do sinal PM_MOP.

Nº 2 – Gera uma confirmação a partir do evento pm_mrqu para que a operação a máquina possa acontecer. Essa confirmação é desativada no momento em que o atuador está fora da posição de operação.

Nº 3 – Gera um sinal constante para reiniciar a operação a máquina, uma vez que a operação é reiniciada se já foi finalizada ou se o atuador se deslocou antes que a operação tenha acabado.

O evento pm_mack ocorre quando o Temporizador atinge o número de entidades estabelecido, o que representa o tempo gasto para uma operação do atuador.

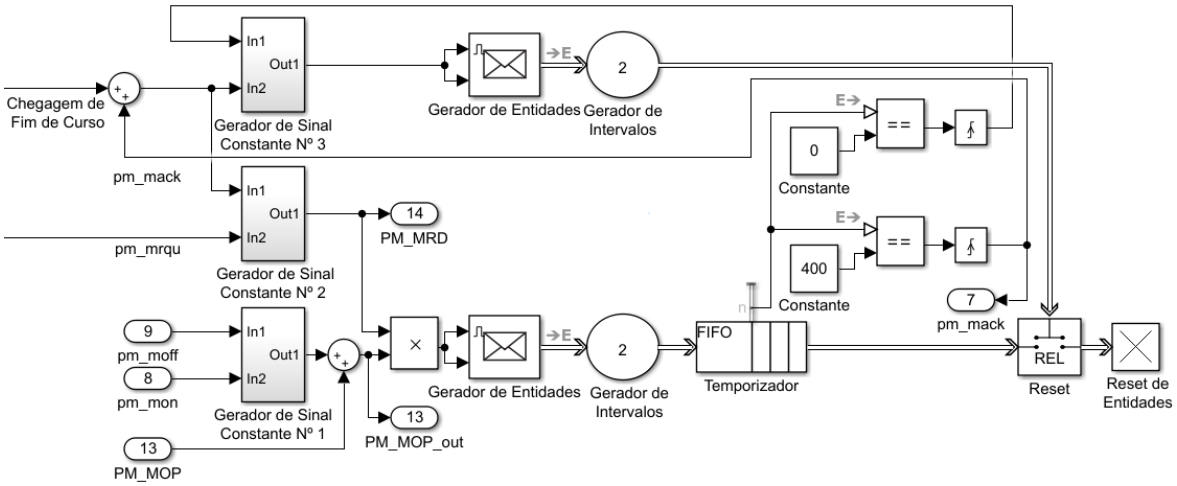


Figura 5.9 – Esquema de blocos para o processamento de peça.

Fonte: Autor

5.3 Stack Feeder

A máquina responsável por fornecer peças ao sistema consiste em duas operações básicas, sendo estas as de adição de peças e movimentação destas adiante. Na Figura 5.10, diagrama de blocos para o alimentador, pode-se verificar a presença de dois pares de geradores de entidades em paralelo. Esta associação se faz necessária para que seja possível gerar uma entidade por vez de forma exata. Assim, a dupla de cima, em azul, gera entidades enquanto a inferior, em vermelho, limita sua passagem uma a uma.

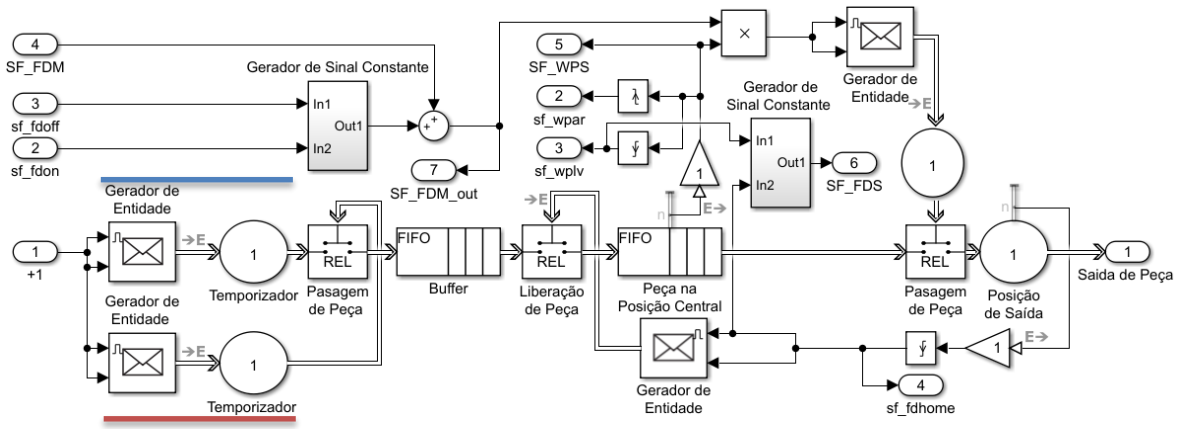


Figura 5.10 – Diagrama de blocos do componente Stack Feeder.

Fonte: Autor

Para o avanço das peças de forma isolada, foram posicionados blocos de forma que entidades fossem liberadas de forma singular, fazendo com que uma nova entidade ocupe a posição central da máquina somente quando uma peça já tenha saído totalmente de sua extensão. Este procedimento é realizado a partir de uma realimentação da confirmação da saída total de uma peça no processo de liberação para uma nova peça na posição central.

Uma vez que este posicionamento depende de tal liberação, o bloco de passagem após o Buffer possui uma condição inicial de liberação, já que este depende da saída completa de uma peça para liberar a passagem e para a primeira ocorrência isto não seria possível.

5.4 Exit Slide

A modelagem realizada para o componente de saída de peças da planta se ateve a uma montagem relativamente simples, uma vez que este não possui esteiras ou atuadores, fazendo que qualquer acionamento externo, não associado a sinais ou eventos, seria necessário somente para eliminar entidades individualmente ou todas em um só comando.

A Figura 5.11 mostra o esquema onde se tem uma posição de entrada, local onde se encontra o sensor de posição, o buffer de peças e a eliminação das mesmas por comandos. Para o comando de eliminar todas as entidades, nota-se a presença do uso de um bloco de auto retenção, o qual depende de uma confirmação do buffer estar vazio para que este sinal seja desligado. O bloco em si obedece ao mesmo esquema encontrado nos Geradores de Sinal Constante, utilizados nos modelos anteriores.

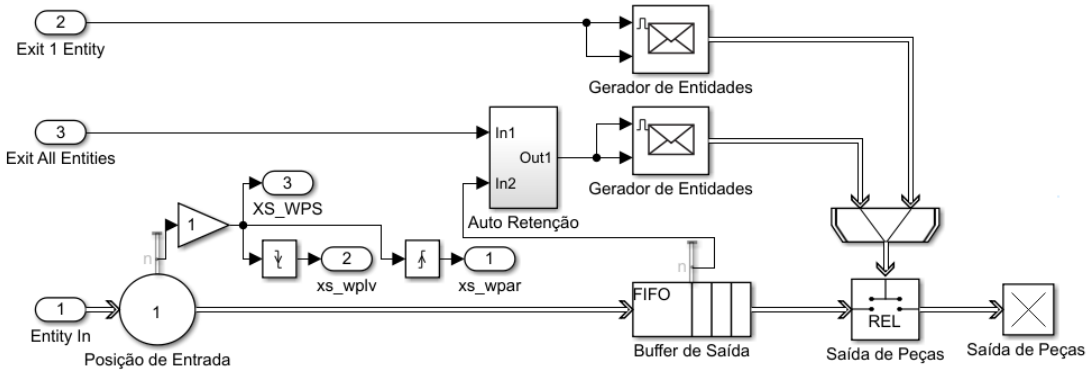


Figura 5.11 – Modelo do componente Exit Slide.

Fonte: Autor

6 Validação e Resultados

Com a modelagem completa, é possível realizar uma avaliação destes modelos a fim de garantir que seu comportamento obedece ao encontrado no simulador FlexFact. Desta forma, foram consideradas dois meios de checagem. No primeiro, os blocos criados foram verificados individualmente, com sinais artificialmente inseridos no modelo. Na segunda verificação, montou-se uma planta com os componentes criados e aplicado nestes um sistema de controle supervisório, o qual gerencia o conjunto.

6.1 Validação Individual

Para a checagem individual, cada sistema individualmente foi colocado em uma montagem junto a geradores de sinal, os quais representam eventos controláveis, e geradores e/ou eliminadores de entidades, simulando entrada e saída de peças. Ainda, o uso de blocos Scope se fez necessário para a checagem de sinais. A Figura 6.1 mostra a representação de um bloco criado, o qual contém seu respectivo esquema, dispondo de entradas e saídas para comunicação.

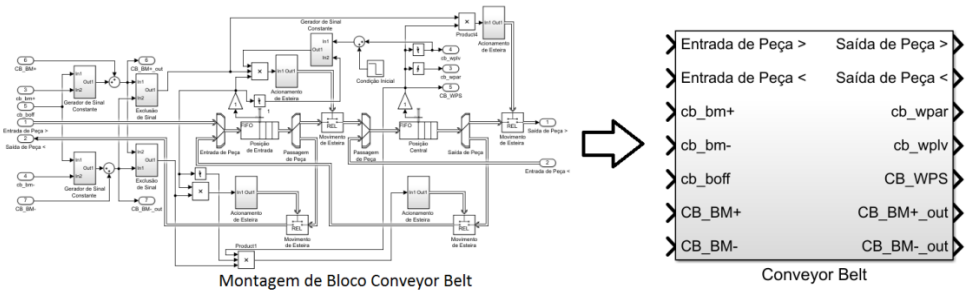


Figura 6.1 – Subsistema representado por um só bloco.

Fonte: Autor

O Conveyor Belt, primeiro bloco testado, foi colocado junto a um gerador de entidades, o qual coloca peças na esteira, e um eliminador de entidades, representando somente a saída destas da máquina. Os geradores de pulsos foram programados para simular eventos em determinados momentos, ligando e desligando os sinais da esteira e, consequentemente, seu movimento. A Figura 6.2 mostra a montagem feita.

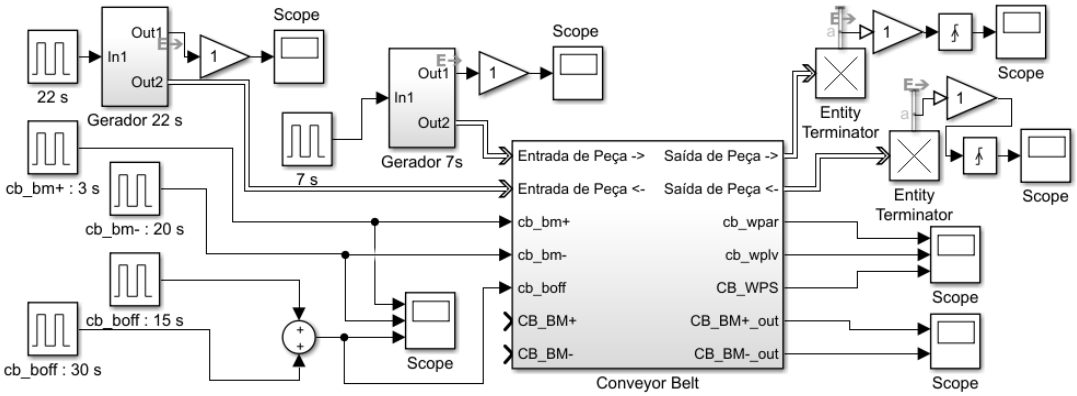


Figura 6.2 – Conveyor Belt com montagem para análise.

Fonte: Autor

Neste, foram testados os dois sentidos de movimento, onde uma entidade seria adicionada à esquerda e a esteira ligada por CB_BM+. Após o desligamento, realizado pelo evento controlável cb_boff, a esteira é religada por CB_BM- e após a passagem de uma entidade no sentido de retorno, desligada novamente. As Figura 6.3 mostra o momento dos eventos controláveis e seu reflexo no acionamento dos sinais de operação.

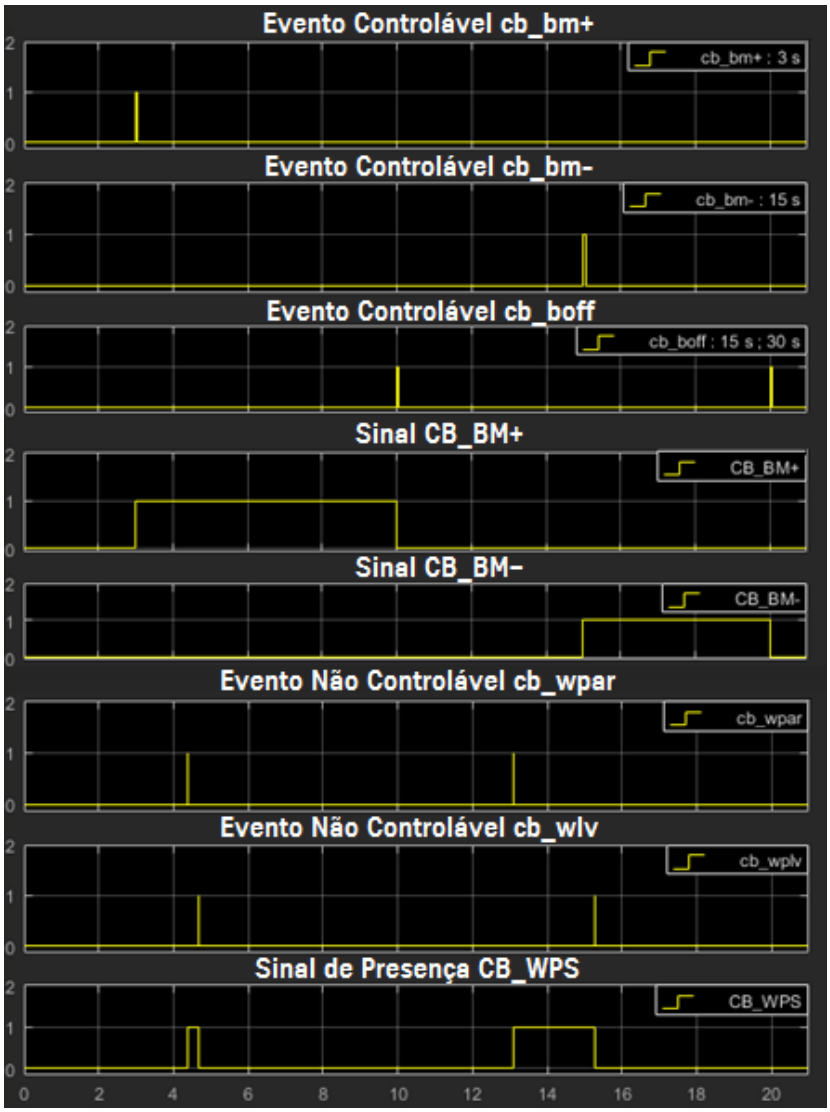


Figura 6.3 – Sinais CB_WPS, CB_BM+ e CB_BM- associados aos seus eventos.

Fonte: Autor

Na Figura 6.3 nota-se relação direta dos eventos controláveis com o acionamento dos sinais, acontecendo comutação dos sinais no exato momento dos eventos. Para os gráficos do sinal e eventos do sensor de presença, pode-se separar a explicação em duas partes:

1º - Como a esteira já está ligada quando a entidade chega, esta leva somente o tempo de movimento para sair da esteira.

2º - No sentido de retorno, nota-se que a esteira liga somente instantes após a entidade já estar presente. Desta forma, a mesma leva um tempo maior para sair da esteira, resultando em um tempo maior da resposta do sensor CM_WPS ligada.

Para o bloco Procesing Machine, foi montada uma montagem semelhante, porém visando a checagem da operação do atuador, já que a movimentação da esteira se assemelha ao apresentado para o Conveyor Belt. Nesta montagem, próprios sinais não controláveis de posicionamento e fim de processo foram usados como eventos controláveis de forma sequencial da operação do atuador, como mostrado na Figura 6.4. Nas figuras 6.9, 6.10 e 6.11 são registrados os eventos e sinais do processo.

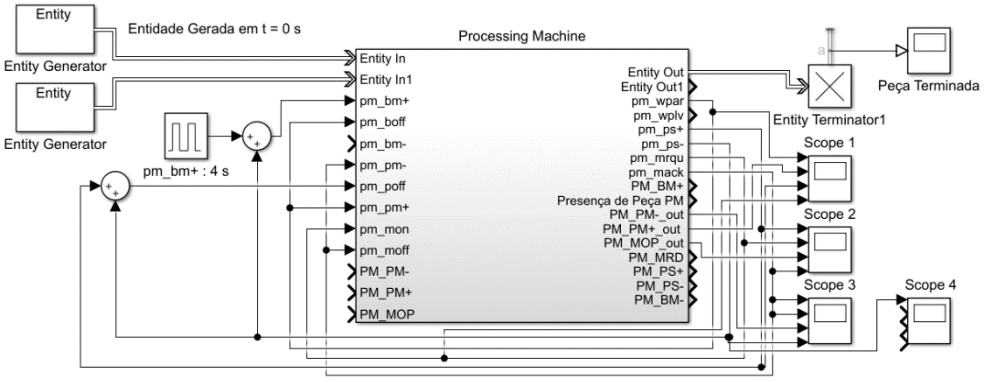


Figura 6.4 – Processing Machine para esquema de simulação.

Fonte: Autor

Figura 6.5 – Evento controlável de avanço pm_pm+ e seu sinal, PM_PM+, junto aos eventos não controláveis pm_ps+, de fim de curso e pm_mrqu.

Fonte: Autor

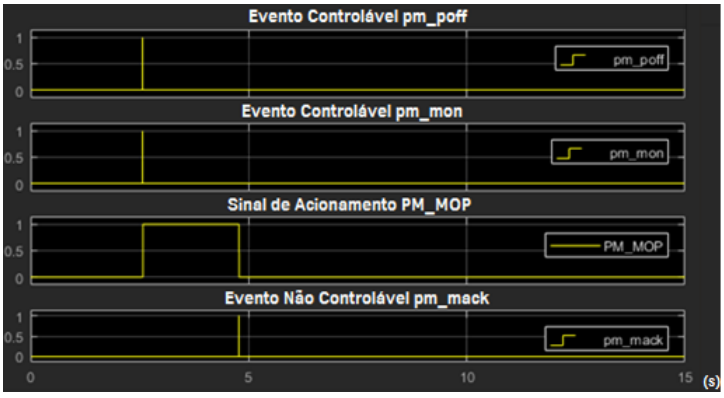


Figura 6.6 – Eventos controláveis pm_poff e pm_mon, sinal de operação PM_MOP e eventos de fim de processo pm_mack.

Fonte: Autor

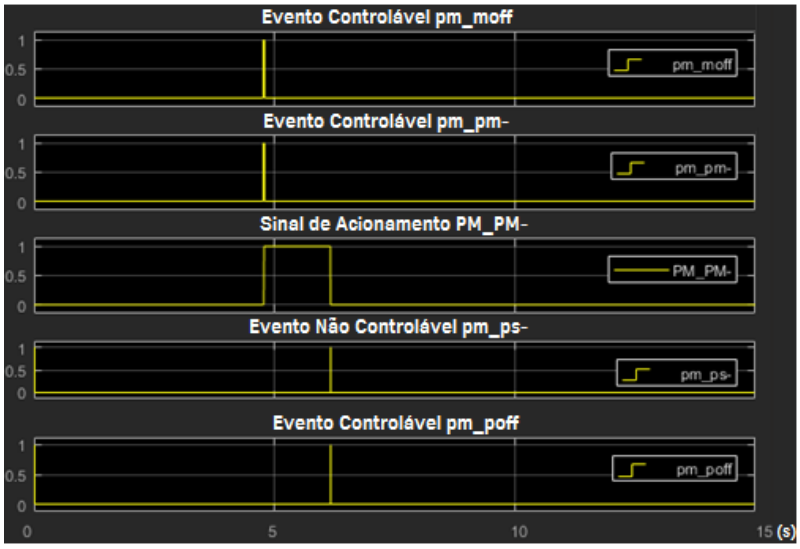


Figura 6.7 – Eventos controláveis pm_moff e pm_pm-, sinal de operação PM_PM-, evento de fim de curso pm_ps- e evento controlável pm_poff.

Fonte: Autor

Para os outros dois componentes, simulações semelhantes foram montadas, como mostram as figuras a seguir. Em relação ao Stack Feeder, esquema da Figura 6.8, sinais de adição de peça e eventos controláveis de movimento da esteira foram inseridos no bloco, resultando nos gráficos da Figura 6.9.

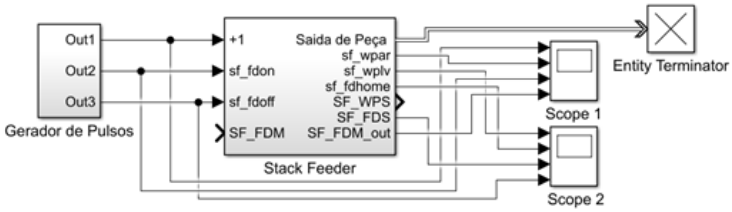


Figura 6.8 – Esquema para simulação com Stack Feeder.

Fonte: Autor

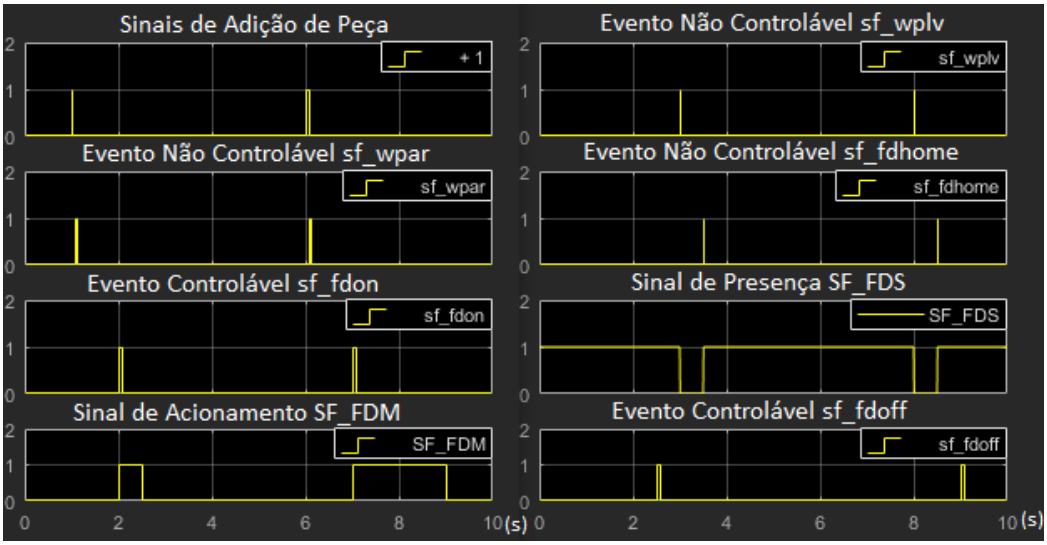


Figura 6.9 – Eventos e sinais de simulação com Stack Feeder.

Fonte: Autor

A Figura 6.10, mostra o esquema de blocos para a simulação do Exit Slide, onde pulsos foram gerados para simular a retirada de entidades, mostrados no gráfico da mesma figura. Para este, uma entidade entra no bloco e logo é retirada, aos 0,5 s. Depois, 4 entidades são adicionadas periodicamente e retiradas pelo comando “Retirar Todas Entidades”, a qual esvazia completamente o buffer de saída.

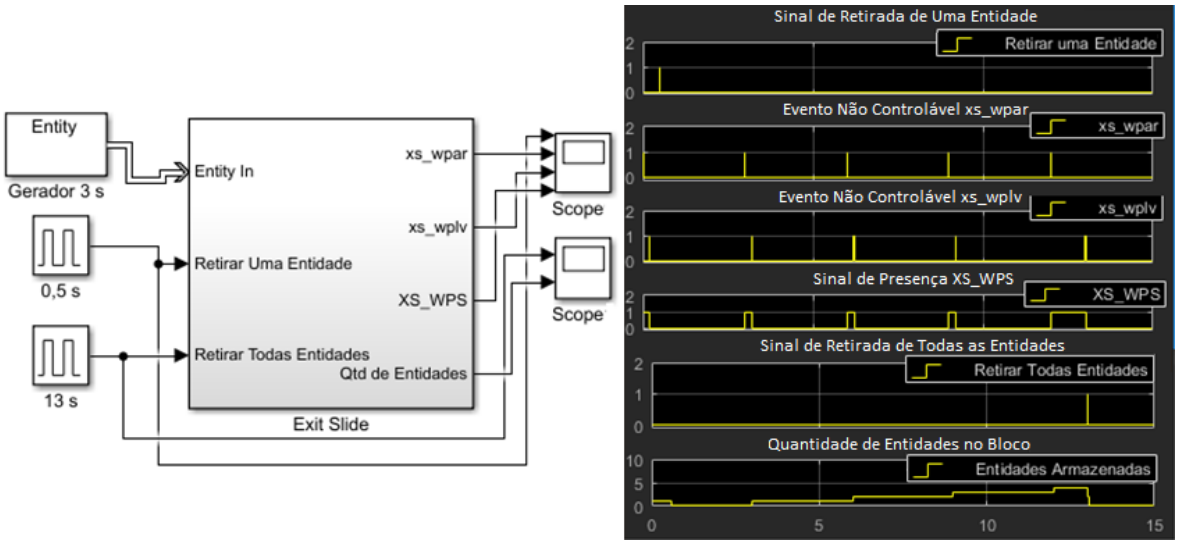


Figura 6.10 – Esquema de simulação para o Exit Slide e gráfico de resultados.

Fonte: Autor

6.2 Validação Utilizando Controle Supervisório

Para a utilização de controle supervisório na validação, primeiro é necessária a construção de uma planta no simulador FlexFact de forma que, com a implementação conjunta de um controle supervisório provido pelo *software* DESTool, a planta se comporte como o esperado. Com este conjunto em operação, pode-se construir um controle semelhante no Simulink através de diagramas da biblioteca Stateflow, onde se permite monitorar eventos ocorridos e gerenciar eventos controláveis. A Figura 6.11 mostra o esquema para simulação original e sua forma semelhante implementada no Simulink.

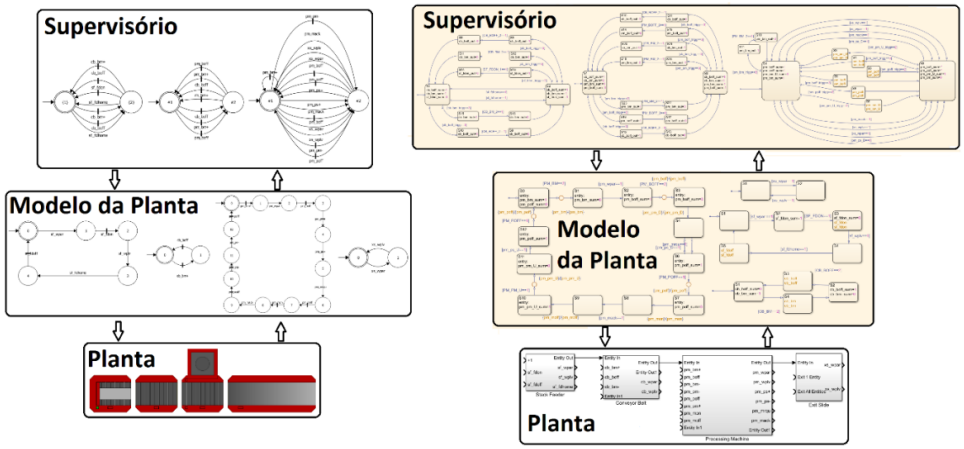


Figura 6.11 – Esquema de montagem e controle de planta no Flexfact e DESTool (a esquerda) e o modelo análogo montado no Simulink (a direita)

Fonte: Autor

Assim, a planta criada no FlexFact foi disposta como mostrado na Figura 6.12. Nesta, peças inseridas pelo Stack Feeder são movimentadas pelas esteiras das máquinas, trabalhadas na Processing Machine e retiradas do sistemas pelo Exit Slide.

Figura 6.12 – Planta utilizada para validação dos modelos.

Fonte: Autor

Para cada máquina, foi elaborado um autômato o qual representa seu comportamento onde, a partir destes, posteriormente se combinaria com a restrição desejada para construir o controle supervisório. Na Figura 6.13 pode-se verificar os modelos das máquinas.

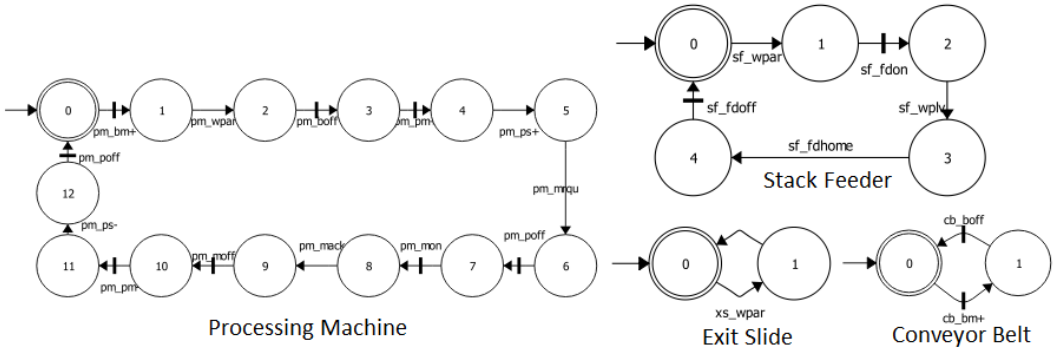


Figura 6.13 – Autômatos para o modelo de cada componente da planta.

Fonte: Autor

Com os modelos obtidos, foram elaboradas restrições de operação para que a planta obedecesse a uma sequência de procedimentos pré-estabelecida, como mostrado em autômatos na Figura 6.14. Estas restrições estabelecem que:

1 – O acionamento da esteira do Stack Feeder só pode ocorre se a esteira do componente Conveyor Belt estiver ligada.

2 – A esteira da máquina Conveyor Belt só pode ser acionada se a esteira da máquina Processing Machine estiver ligada.

3 – A esteira da máquina Processing Machine só pode ser acionada se não há peças na posição do sensor do componente Exit Slide, o qual indicaria que sua capacidade total está ocupada. A ocorrência do evento xs_wpar sem o evento xs_wplv, de saída do sensor, indica este fato.

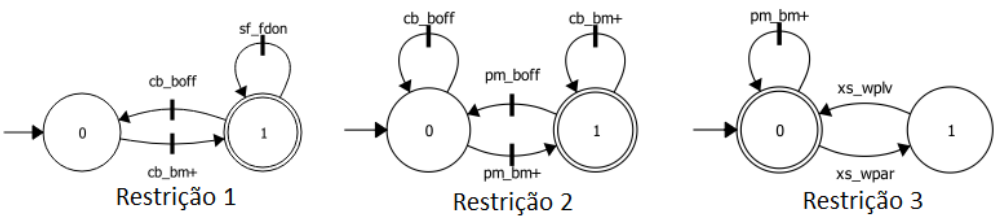


Figura 6.14 – Restrições para a operação da Planta.

Fonte: Autor

A ferramenta DESTool, em conjunto com a biblioteca libFAUDES, permite realizar todas as operações necessárias para a criação do supervisório, seguindo a metodologia descrita na Seção 2.3. Após a sequência de operações, obteve-se os autômatos a seguir (Figura 6.15), os quais foram implementados no Simulink a fim de validar os modelos construídos com a biblioteca SimEvents.

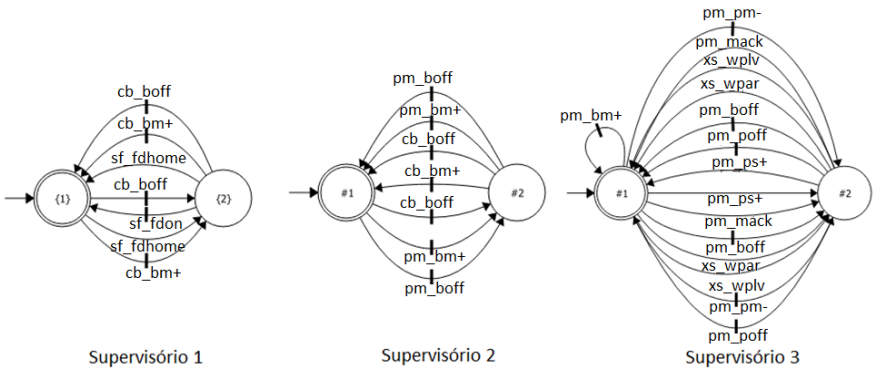


Figura 6.15 – Supervisórios obtidos a partir do *software* DESTool.

Fonte: Autor

Inicialmente, este supervisório foi testado na planta montada no simulador FlexFact, obtendo-se operação satisfatória da planta. Nesta, ao se adicionar peças pelo Stack Feeder, as mesmas se movimentam até a máquina Processing Machine onde são operadas, uma a uma. Sempre que este procedimento está ocorrendo, as demais esteiras param sempre que suas capacidades estão completas, para que peças não sejam empurradas adiante sem que antes tenham sido processadas.

Para utilizar este controle, o mesmo deve ser implementado no Simulink, juntamente aos modelos em autômatos das máquinas, de forma a se comunicar com os modelos criados por blocos de filas, onde pode-se adquirir resultados a fim de avaliá-los e constatar a correta operação dos modelos em comparação aos originais do FlexFact.

Utilizando a biblioteca Stateflow, presente também no Simulink, é possível a confecção de máquinas de estado para que estas trabalhem de forma semelhante aos autômatos presentes no DESTool. Na Figura 6.16 pode-se verificar a construção de uma máquina de estados a qual representa o comportamento do Stack Feeder. A transformação de um autômato em máquina de estados foi feita considerando os eventos controláveis e não-controláveis como saídas e entradas, respectivamente, da máquina de estados.

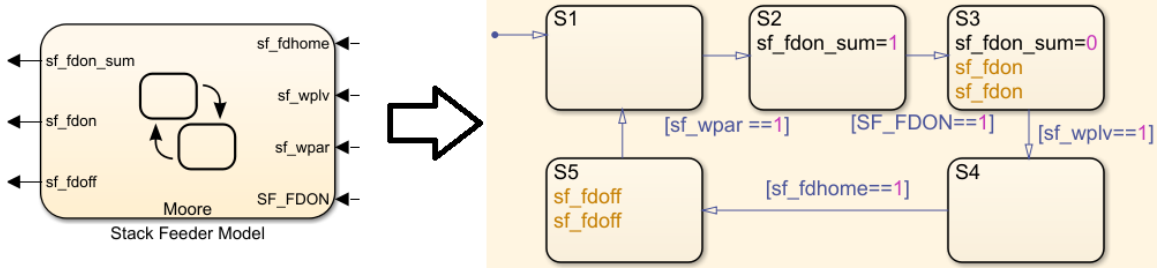


Figura 6.16 – Máquina de estados para o autômato do comportamento na máquina Stack Feeder.

Fonte: Autor

Nesta, podem-se notar diferentes notações utilizadas na execução de diferentes tarefas do sistema. As condições de igualdade presentes nas transições permitem a passagem de um estado a outro somente quando tal condição é satisfeita. Os comandos internos aos estados consistem em dois argumentos diferentes onde as notações terminadas em "_sum" indicam que o evento de mesmo nome está habilitado e pode ocorrer. No caso da Figura 6.16, o evento "sf_fdon" está habilitado pelo modelo da planta, fazendo com que esta informação de possibilidade de ocorrência seja enviada a outras máquinas de estado. Pode-se verificar que após a ocorrência do evento "sf_fdon" no estado S3, em laranja, o comando de evento habilitado "sf_fdon_sum" é anulado, indicando que tal evento não está mais habilitado a ocorrer.

Seguindo o mesmo raciocínio, pode-se verificar, na Figura 6.17, uma máquina de estados que representa o supervisório o qual opera gerenciando o componente Stack Feeder e a esteira Conveyor Belt.

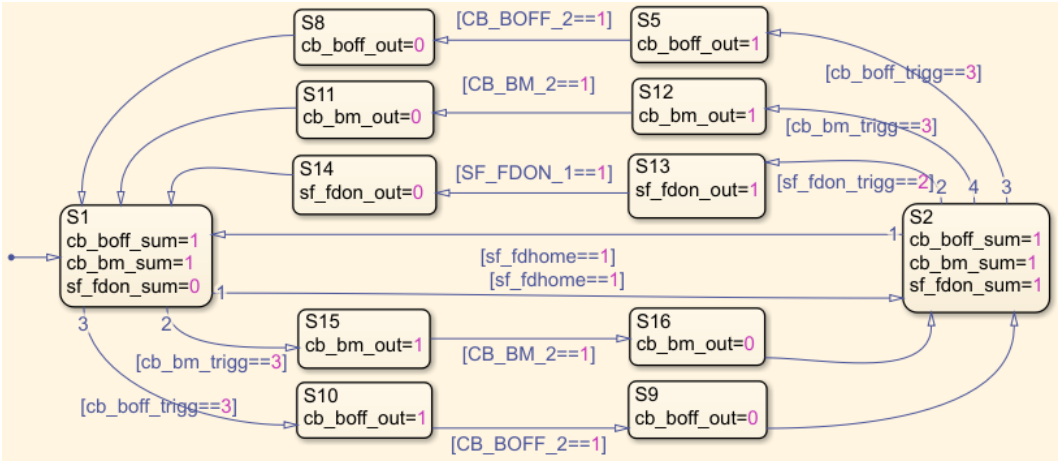


Figura 6.17 – Máquina de estados para o autômato do controle supervisório entre a máquina Stack Feeder e a esteira Conveyor Belt.

Fonte: Autor

Neste supervisório, verifica-se a presença de condições de transição possuindo nomes de eventos controláveis seguidos de "_trigg". Estas condições verificam se todos os supervisórios e modelo da planta em específico estão com o evento em questão habilitado. Se houver concordância de todos os autômatos, a condição é satisfeita e ocorre a passagem para outro estado, o qual habilita o evento controlável no modelo da planta e este repasse ao modelo criado via SimEvents. Destaca-se que o estado de execução do evento possui a atribuição de uma variável com o nome do evento em questão seguido de "_out", o qual representa a saída para o modelo associado a tal evento.

Para a execução correta dos autômatos criados via Stateflow assemelhando-se ao *software* DESTool, é necessária a sincronização entre todos os autômatos quando uma ocorrência de evento acontece e, consequentemente, uma transição de estado. A separação da ocorrência de um evento em dois estados condicionada por uma verificação de ocorrência de evento no modelo é necessária, pois uma vez que as máquinas de estado no Simulink saem de sincronia, o modelo passa a não mais funcionar.

Com estas confirmações montadas juntamente com a verificação de eventos habilitados, foi possível montar um diagrama de blocos completo contando os modelos

criados, os modelos das máquinas por máquinas de estado e, também por máquinas de estado, implementar o controle supervisório, resultando no esquema mostrado na Figura 6.18, no qual consta os blocos e ligações.

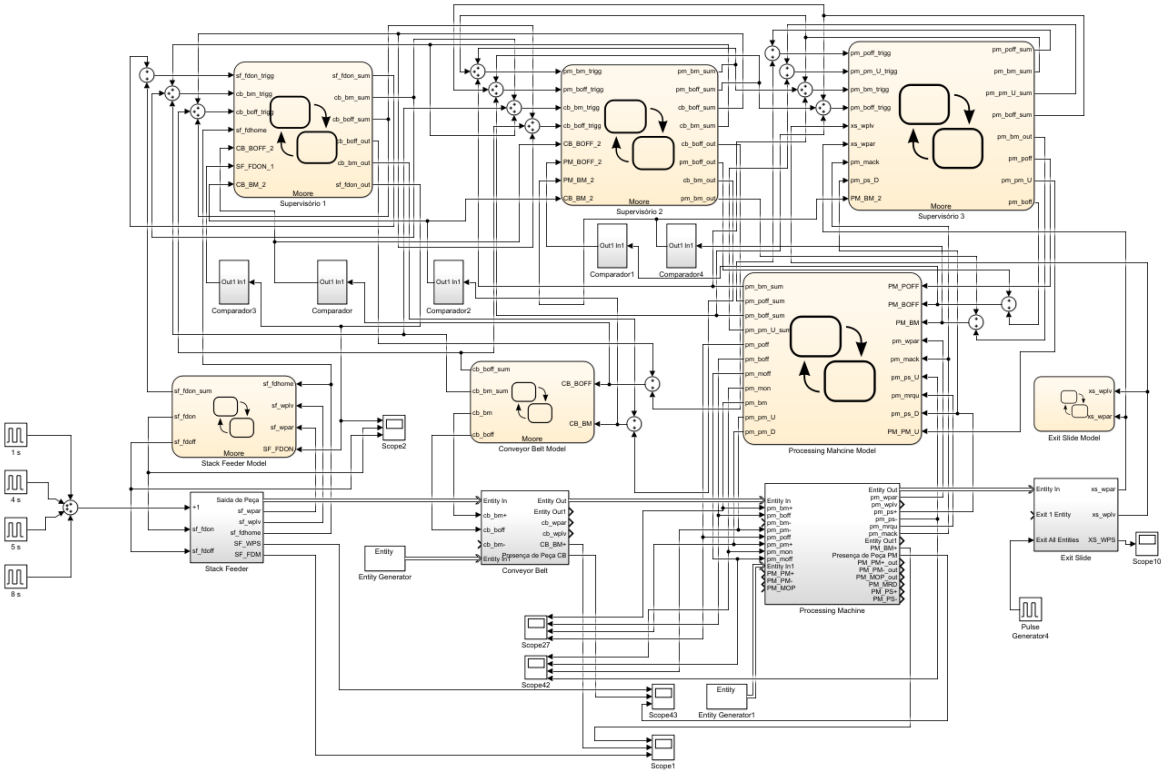


Figura 6.18 – Diagrama da planta montada no Simulink junto aos modelos e supervisórios em máquinas de estado.

Fonte: Autor

Neste modelo, são adicionadas 4 entidades em tempos diferentes, onde a planta opera processando individualmente as peças, ligando e desligando a esteira a fim de evitar movimentação inadequada de peças.

Para verificar a correta operação da planta, nota-se, primeiramente, a passagem da primeira entidade desde o alimentador, sinal SF_WPS, até a máquina de processamento, sinal PM_WPS. Neste momento, o qual se inicia um processamento de peça, pode-se notar o desligamento sucessivo de todas as esteiras nos três últimos gráficos, fazendo com que o sistema fique em espera até que a peça seja liberada pela máquina, reiniciando a movimentação. Pode-se perceber que esta sequência se repete, já que outra peça chega na posição central da máquina para processamento.

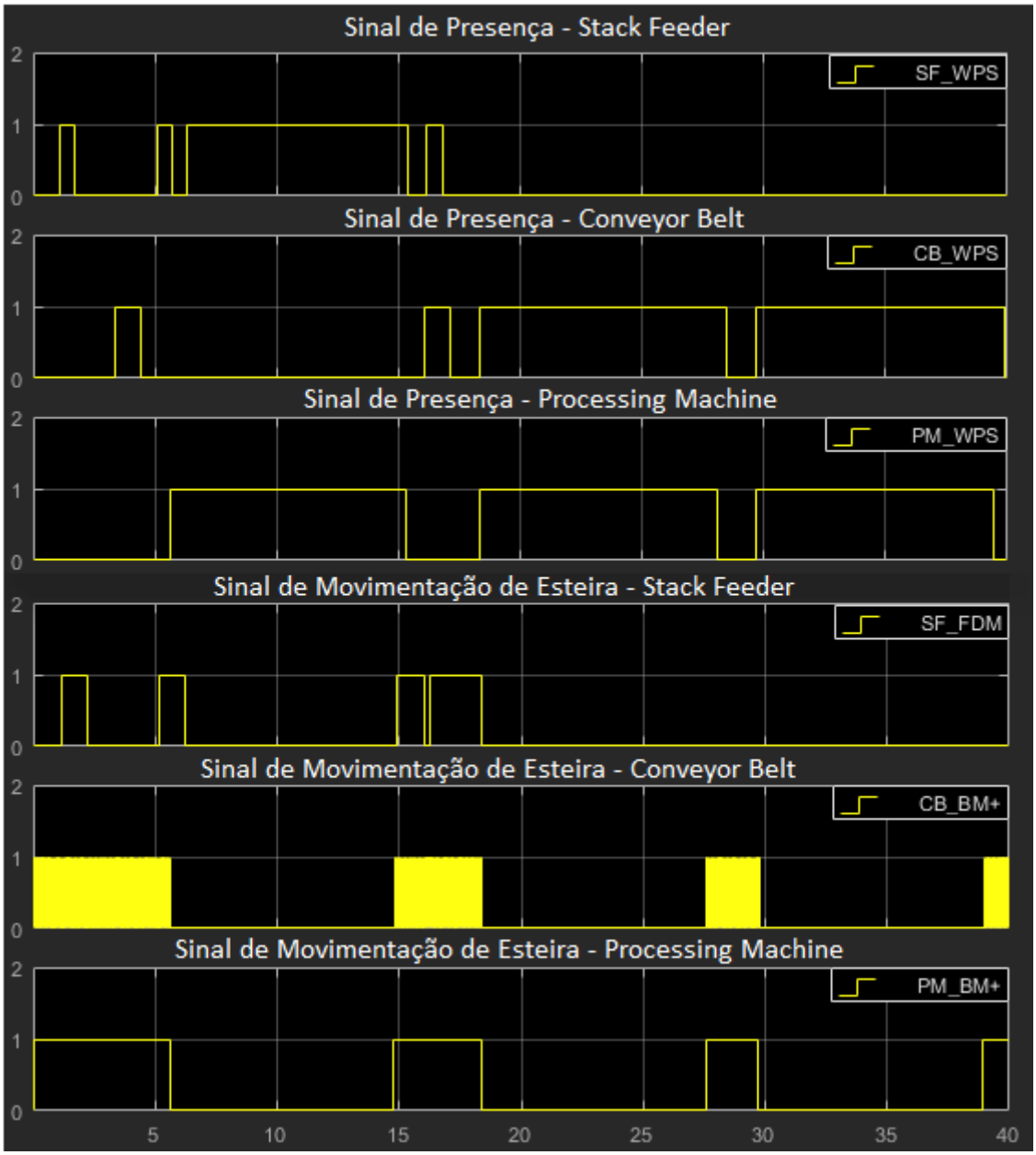


Figura 6.19 – Gráficos dos sinais do sensor de presença e do acionamento das esteiras das máquinas.

Fonte: Autor

Cabe ressaltar que entre as posições centrais entre dois componentes adjacentes existe a posição de entrada, não registrada por sensor, fazendo com que nos gráficos da Figura 6.19 estejam presentes pequenos intervalos sem peças presentes nas maquinas.

Este comportamento da planta controlada é semelhante ao obtido quando se associando estes mesmos supervisórios através do *software* DESTool ao simulador FlexFact.

7 Conclusões e Trabalhos Futuros

Ao se utilizar técnicas de filas para a modelagem de um sistema a eventos discretos, nota-se a possibilidade de diferentes níveis de detalhamento, podendo-se modelar desde os comportamentos principais, como o movimento de peças em uma máquina, e a mecânica de operação específica, como execução de processos e gerenciamento de eventos.

Tratando-se da relação de filas com a biblioteca empregada do Simulink, os blocos do conjunto SimEvents apresentam características fiéis ao comportamento de sistemas de filas, podendo-se associar os blocos desta biblioteca com outros a fim de se manipular eventos e sinais, como realizado neste trabalho. Ainda, a associação entre bibliotecas permitiu implementar comportamento das máquinas semelhante ao simulador através de diagramas de blocos.

Em relação à validação dos modelos criados, a utilização de controle supervisório cumpre um papel indispensável, uma vez que a correta execução deste controle somente viria a apresentar resultados satisfatórios no caso de modelos fiéis aos componentes originais. O emprego deste tipo de controle possibilita o gerenciamento de sistemas a partir de ocorrências de eventos, fato modelado na criação nos blocos no Simulink.

Como forma primordial de modelagem e comportamento de sistemas, o uso de autômatos é fundamental, já que toda a construção do controle supervisório utilizado provém de uma plataforma que opera somente nesta linguagem. Ainda, propriedades como operação de paralelo entre autômatos pode gerar um modelo que corresponde à operação conjunta de dois subsistemas, fato mostrado quando se construiu o controle supervisório deste trabalho através do *software* DESTool.

Nota-se então a ampla gama de possibilidades quando se trata da modelagem de sistemas utilizando o *software* MATLAB, onde a confecção de modelos de sistemas a eventos discretos pode ser executada em um nível alto de detalhamento, já que são encontrados diversos meios na ferramenta Simulink para realizar tal tipo de tarefa, utilizando-se como exemplos o uso das bibliotecas SimEvents e Stateflow para implementar modelos em filas e máquinas de estado.

As realizações deste trabalho podem ser expandidas desde a melhoria dos modelos criados, trazendo ainda mais detalhes para suas construções, até a confecção de novos modelos a partir de outros componentes do simulador FlexFact. Também, pode-se associar os modelos criados ao simulador em si, fazendo-se isso por comunicação externa dos sinais de operação dos componentes através de protocolo Modbus.

Por fim, a implementação de controle para SEDs no Simulink necessita de ferramentas mais direcionadas a este fim, para que não haja a necessidade da transferência de sistemas de controle criados em outros softwares para o MATLAB. Desta forma, a criação de uma biblioteca que trabalhasse diretamente com a criação de comandos de sistemas a eventos discretos ampliaria o estudo nessa área.

8 Referências

Adan I. and Resing J. (2015) Queueing Systems, Apostila, Department of Mathematics and Computing Science, Eindhoven University of Technology.

Carvalho, A. (2015), Estudo da Utilização do MATLAB/Simulink Aplicado ao Controle de Sistemas A Eventos Discretos, Trabalho de Conclusão do Curso de Engenharia de Controle e Automação, UFRGS.

Cassandras, C. G. (1993) Discrete Event Systems: Modeling And Performance Analysis, Irwin, 0256112126.

Cassandras, C. G. e Lafortune, S. (2008) Introduction To Discrete Event Systems, Springer, 2ª Edição, ISBN 978-0-387-33332-8.

Coffen D. D. e Garg V. K (1999) Supervisory Control of Real-time Discrete Event Sustems using Lattice Theory, Artigo, Vol. XX, No Y, IEEE Transactions on Automatic Control.

Götz, M. (2016) Apostila De Sistemas A Eventos Discretos, Autômatos & Teoria De Controle Supervisório, Apostila, UFRGS, Departamento De Sistemas Elétricos De Automação E Energia.

Gross, D., Shortie, J. F., Thompson, J. M., Harris, C. M., (2008) Fundamentals of Queueing Theory, 4ª Edição, ISBN 9780471791270.

Hillier, F. S., Lieberman, G. J., (2006) Introdução à Pesquisa Operacional, 8ª Edição, McGraw-Hill.

Hopcroft, J. E., Motwani, R. e Ullman, J. D. (2001) Introduction to Automata Theor, Languages and Computation, 2ª Edição.

Kendall, D. G. (1952) Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of the Imbedded Markov Chain, Artigo, Oxford University and Princeton University.

Košecká, J. (1992) Control of Discrete Event Systems, Artigo, Department of Computer and Information Science, University of Pensilvania.

MATLAB & Simulink, (2017) Getting Started with Stateflow. Disponível em: https://www.mathworks.com. Acesso em 15 de outubro de 2017.

MATLAB & Simulink, (2017) Getting Started with SimEvents. Disponível em: https://www.mathworks.com. Acesso em 05 de setembro de 2017.

Ramadge, P. J. G. e Wonham, W. M. (1989) The Control of Discrete Event Systems, Artigo, IEEE.

Schlick, C. M. (2013) Simulation of Discrete Event Systems, Apresentação, Lehrstuhl und Institut für Arbeitswissenschaft, RWTH Aachen University.

Zukerman, M., (2000) Introduction to Queueing Theory and Stochastic Teletraffic Models, Artigo, City University of Hong Kong.

Apêndice

Nesta seção são mostrados conteúdos adicionais do trabalho.

Tabela 0.1 - Sinais e eventos do componente Processing Machine.

Fonte: Autor

Sinal	Evento Associado	Descrição do Evento
PM_BM-	pb_bm-	Ativa o sinal CB_BM-, ligando a esteira para retorno.
	pb_boff	Desativa o sinal CB_BM-, deligando a esteira.
PM_BM+		Desativa o sinal CB_BM+, deligando a esteira.
	pb_bm+	Ativa o sinal CB_BM+, ligando a esteira para avanço.
PM_WPS	pb_wpar	Ocorre quando o sinal CB_WPS é ativado, indicando a chegada de uma peça na posição central.
	pb_wplv	Ocorre quando o sinal CB_WPS é desativado, indicando a saída de uma peça da posição central.
PM_PM+	pm_pm+	Ativa o sinal PM_PM+, ligando o avanço da ferramenta.
	pm_poff	Desativa o sinal PM_PM+, deligando o avanço da ferramenta.
PM_PM-		Desativa o sinal PM_PM-, deligando o recuo da ferramenta.
	pm_pm-	Ativa o sinal PM_PM-, ligando o recuo da ferramenta.
PM_PS+	pm_ps+	Ocorre quando o sinal PM_PS+ é ativado, indicando a chegada da ferramenta na posição de operação.
PM_PS-	pm_ps-	Ocorre quando o sinal PM_PS- é ativado, indicando que a ferramenta está fora da posição inicial.
PM_MOP	pm_mon	Ativa o sinal PM_MOP, ligando o processamento da máquina.
	pm_moff	Desativa o sinal PM_MOP, desligando o processamento da máquina.
PM_MRD	pm_mrqu	Ocorre quando o sinal PM_MRD é ativado, indicando que o processamento de peça pode ser iniciado.
	pm_mack	Ocorre quando o sinal PM_MRD é desativado, indicando que o processamento foi finalizado.

Tabela 0.2 - Sinais e eventos do componente Exit Slide.

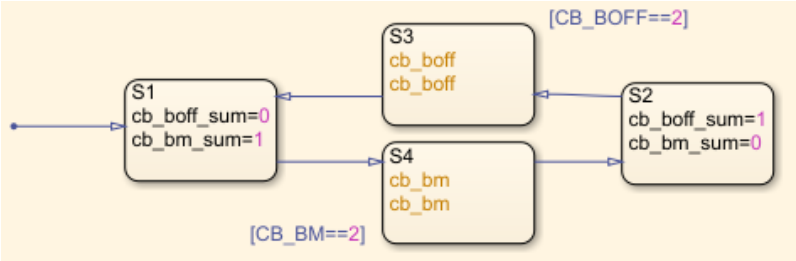
Fonte: Autor

Sinal	Evento Associado	Descrição do Evento
XS_WPS	xs_wpar	Ocorre quando o sinal XS_WPS é ativado, indicando a chegada de uma peça na posição central.
	xs_wplv	Ocorre quando o sinal XS_WPS é desativado, indicando a saída de uma peça da posição central.

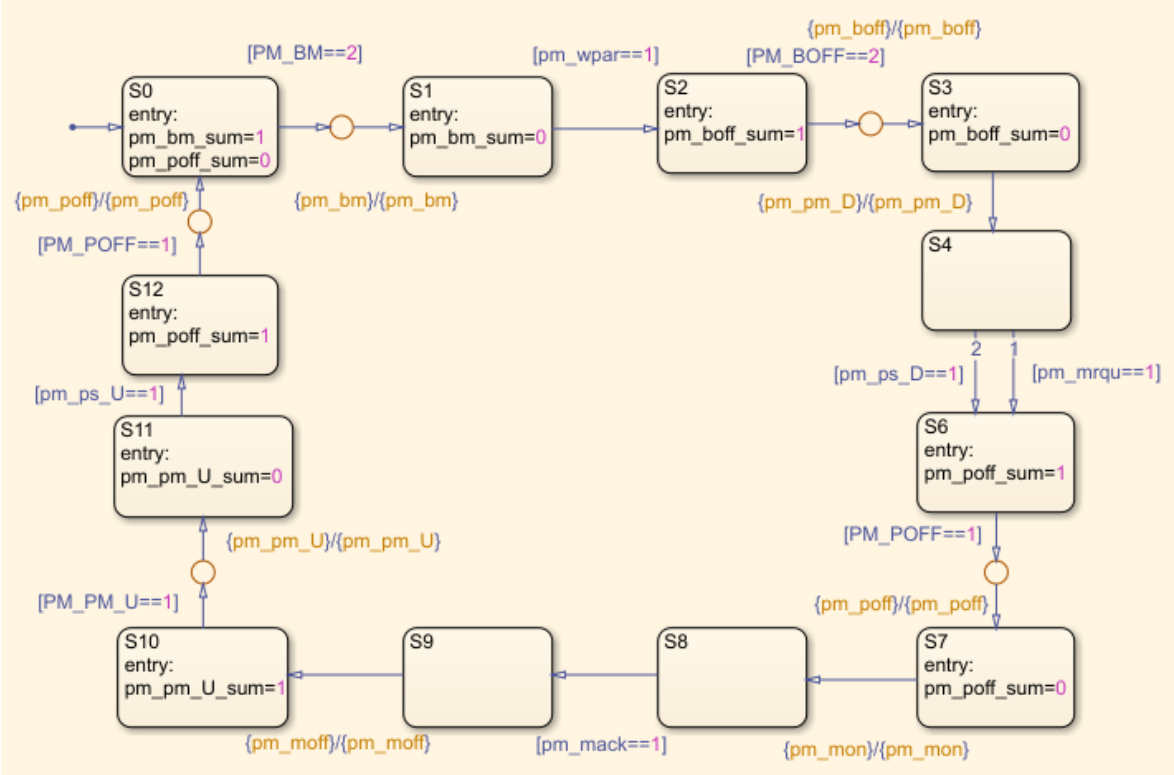
Tabela 0.3 - Sinais e eventos do componente Stack Feeder.

Fonte: Autor

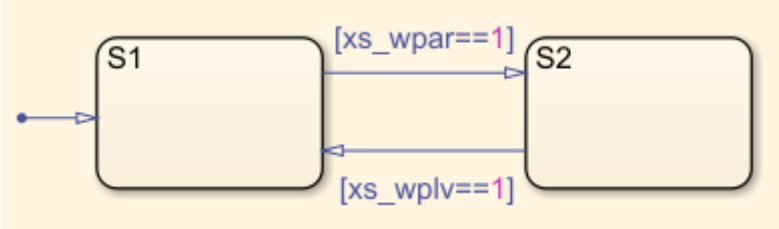
Sinal	Evento Associado	Descrição do Evento
SF_FDM	sf_fdon	Ativa o sinal SF_FDM, ligando a esteira para retorno.
	sf_fdoff	Desativa o sinal SF_FDM, deligando a esteira.
SF_FDS	sf_fdhome	Ocorre quando o alimentador está na posição inicial.
SF_WPS	sf_wpar	Ocorre quando o sinal SF_WPS é ativado, indicando a chegada de uma peça na posição central.
	sf_wplv	Ocorre quando o sinal SF_WPS é desativado, indicando a saída de uma peça da posição central.



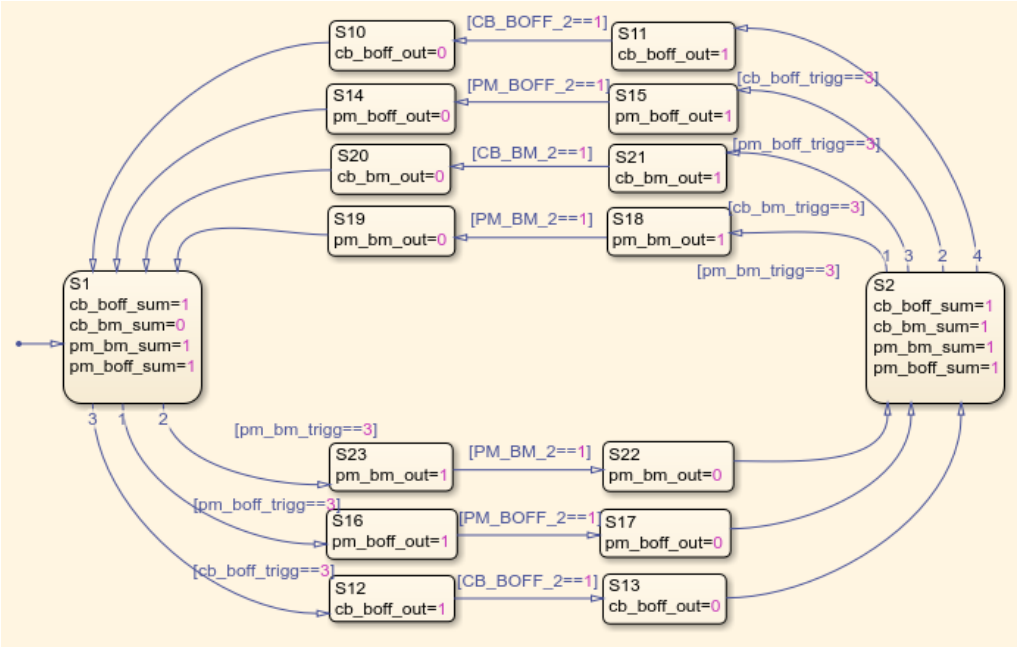
Modelo de operação na máquina Conveyor Belt.



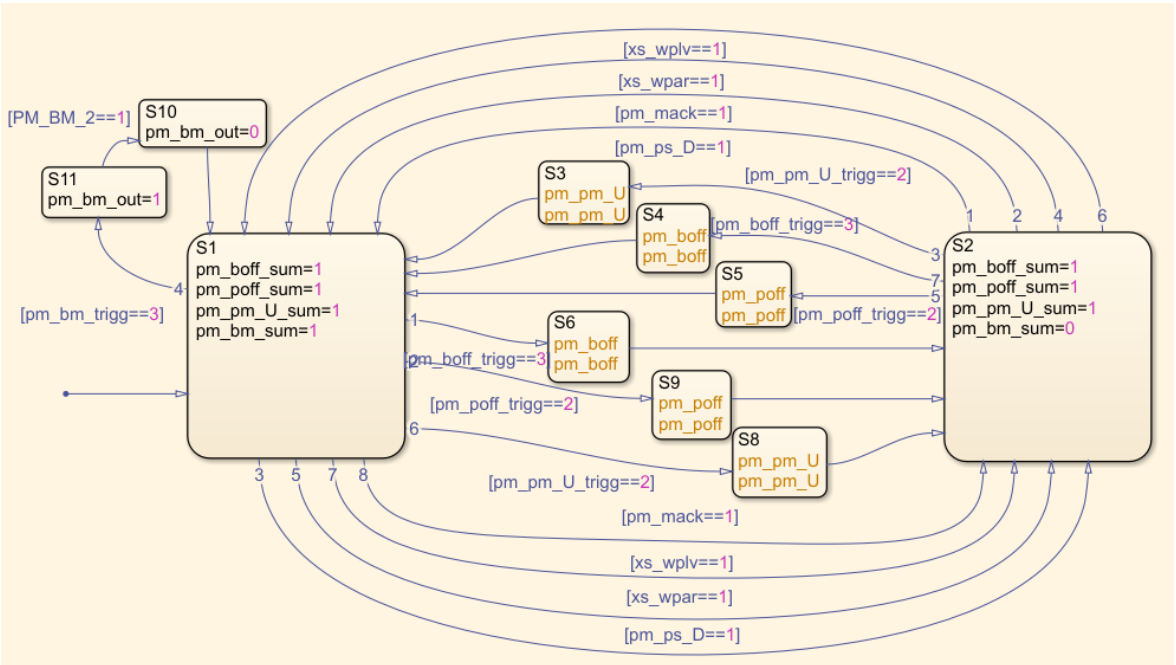
Modelo de operação na máquina Processing Machine.



Modelo de operação do armazenador Exit Slide.



Supervisório 2: Gerenciamento de eventos entre Conveor Belt e Processing Machine.



Supervisório 3: Gerenciamento de eventos entre Processing Machine e Exit Slide.

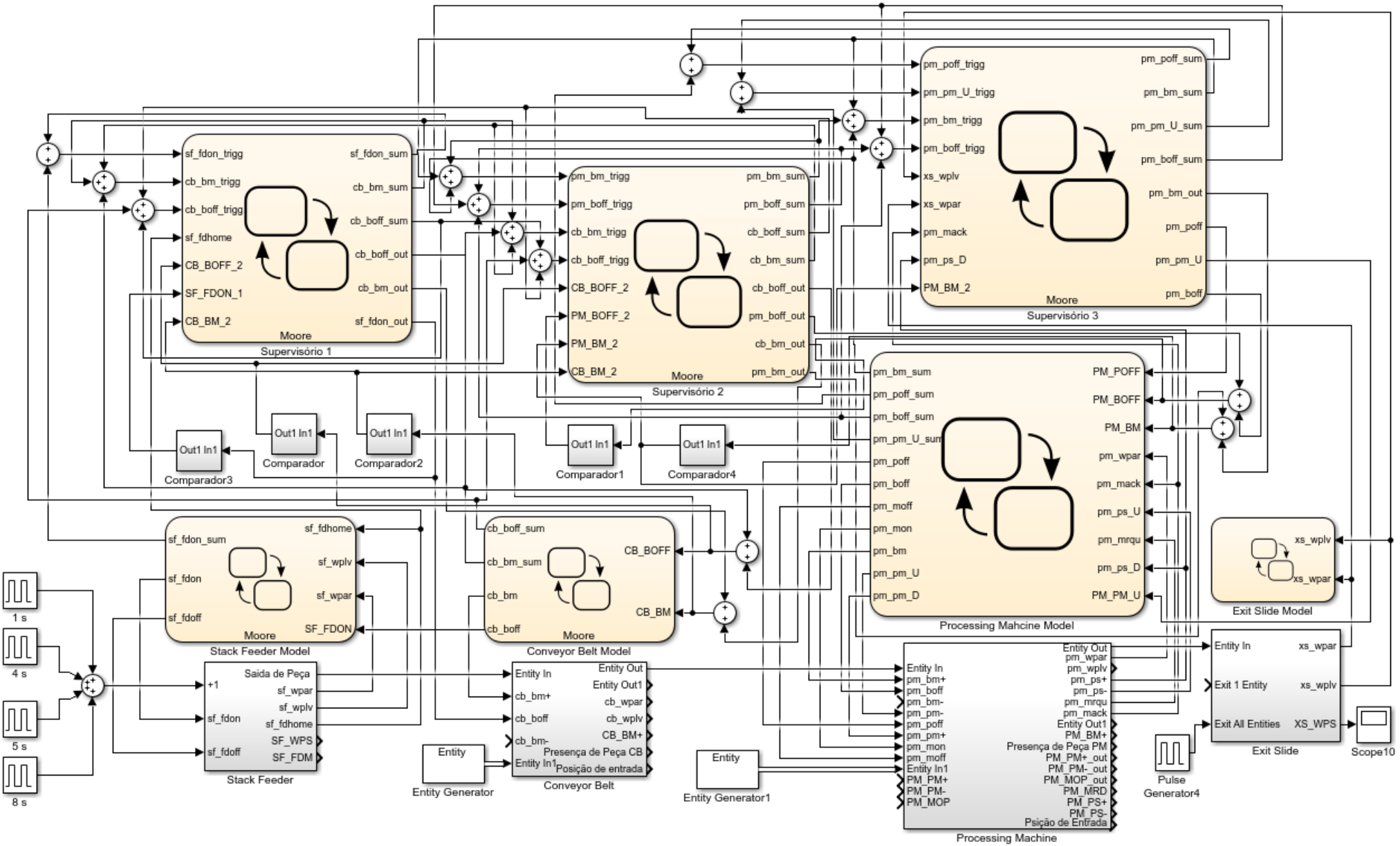
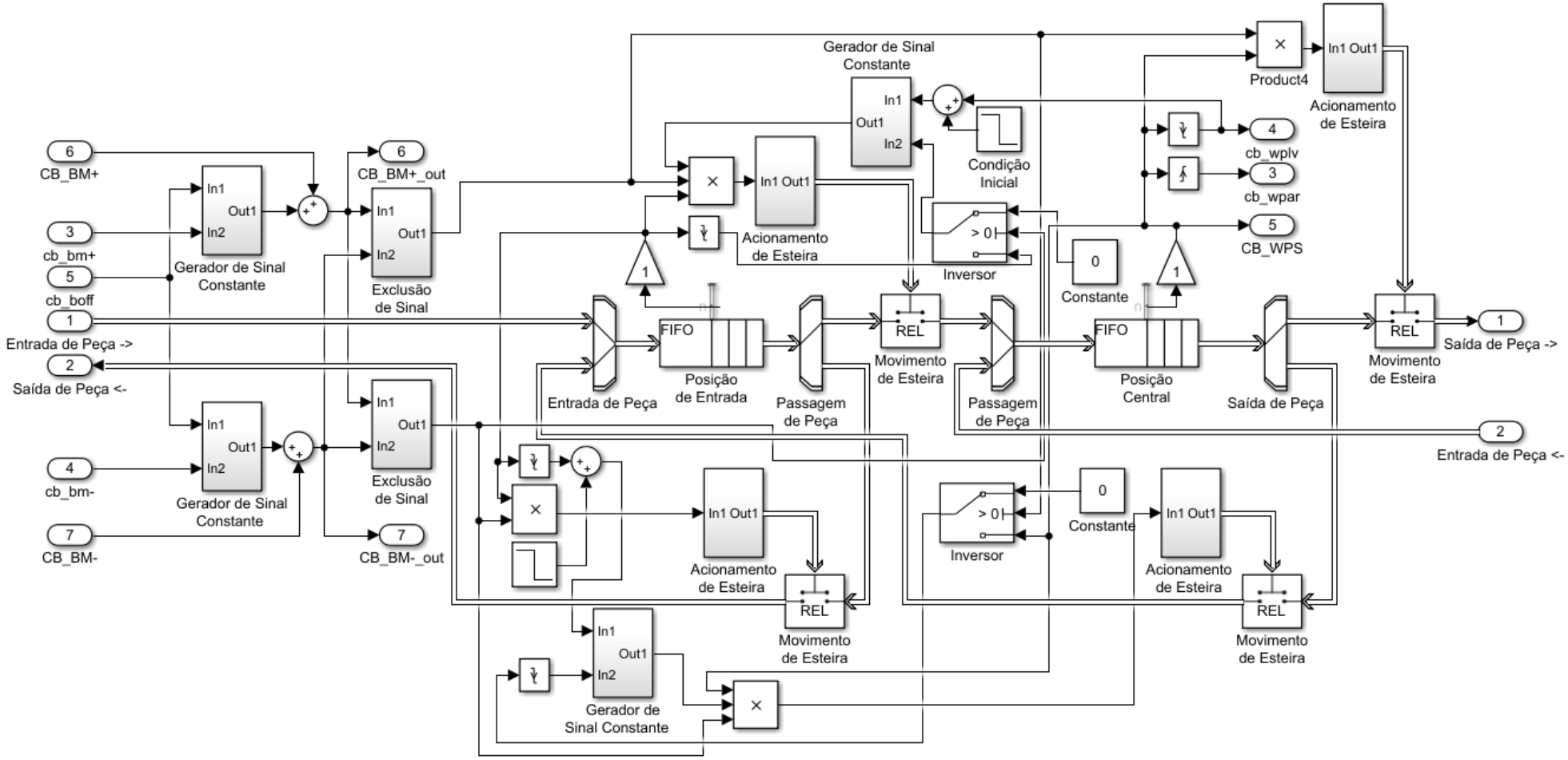


Diagrama de blocos completo para simulação de controle supervisório.



Modelo de blocos do componente Conveyor Belt.