A Discrete Process Modelling Study of ARGESIM Benchmark 'C2 – Flexible Assembly System' with Warteschlangensimulator

Alexander Herzog

Simulation Science Center Clausthal-Göttingen / TU Clausthal, Arnold-Sommerfeld-Straße 6, 38678 Clausthal-Zellerfeld, Germany; *alexander.herzog@tu-clausthal.de*

SNE 34(1), 2024, 23-28, DOI: 10.11128/sne.34.bn02.10673
Received: 2024-02-10; Revised: 2024-02-19
Accepted: 2024-02-23
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. In ARGESIM benchmark "C2 – Flexible Assembly System" a multi station conveyor based production system is introduced. The aim is to analyze the model for bottlenecks and to maximize the throughput. This article studies the benchmark model using the discrete-event, stochastic simulation tool Warteschlangensimulator.

Introduction

This article describes and studies the solution of the ARGESIM comparison model "C2 – Flexible Assembly System" using the discrete-event, stochastic simulation tool Warteschlangensimulator. A full description of the considered model can be found in the SNE model definition [1].

The model consists of a conveyor system moving a fixed number of pallets in a circle. The pallets carry the workpieces and are loaded/unloaded at one station. The complete system consists of 8 stations which are connected to the conveyor belt. Each station can be targeted or bypassed by the pallets. Depending on the load of the stations a pallet may need multiple laps in the circle for full processing. The optimization goal is to choose the number of pallets circulating in the system in a way that maximizes the throughput of the system. Lead times of the workpieces and utilization of the different process stations are considered also. Based on the utilization of the stations improvement recommendations are made to further increase throughput.

1 Warteschlangensimulator

Warteschlangensimulator (see [2]) is a free and open source, platform independent, Java-based, event-driven, stochastic simulator. The permissive Apache 2.0 license allows to use the simulator in teaching, research and industrial/commercial context without restrictions.

The simulator allows graphical modelling of queueing systems in form of flow charts. Therefore over 100 station types are available. Inter-arrival times, service times etc. can be modelled using one of the 41 built-in probability distributions (including the option to map measured values as an empirical distribution). An automatic distribution fitter to find a distribution that matches measured values best is also available. For more complex definitions a formula parser is integrated; so for example shifted or truncated probability distribution can be used, too.

Models can be executed in animation mode showing the movement of the entities through the system including the display of queues and in a fast simulation mode without graphical output using multiple CPU cores for faster executing. Since the simulation runtimes are often in the range of a few seconds up to one minute, the effects of changes to input parameters can be investigated in a very interactive way.

During simulation all relevant statistic performance indicators are recorded automatically and are available via the built-in report viewer. Filtering and exporting the results is also possible. The fact that recording the performance indicators does not have to be configured manually in the model keeps the modelling of even large systems clear.

To handle complex control strategies, stations can optionally be extended using Javascript or Java code for branching, holding or changing entities passing through the stations.

While Javascript code is interpreted by an internal Javascript engine, Java code is compiled on the fly using the Java runtime environment running the simulator itself and then executed with full machine speed.

Warteschlangensimulator comes with English and German user interface, documentation and example models. A German text book about modelling and simulation using Warteschlangensimulator is also available, see [3].

The built-in parameter study function allows to easily evaluate a model for different parameter sets. This function was used to generate the results shown in figures 3, 4 and 5. If there is a clear target value for a statistic performance indicator and there are some input parameters which are identified as control values, the built-in optimizer can be used to automatically maximize or minimize the target value.

2 Model

A full description of the considered model can be found in the SNE model definition [1]. The conveyor belts are modelled in Warteschlangensimulator using delay stations. The delay times are calculated from the given belt speeds and the belt lengths. Since both values are fixed for each segment, the delay times are also fixed and deterministic. In general, delay stations have an unlimited capacity. The conveyor belt capacity restrictions (due to the belt lengths and the pallet sizes) are implemented using a decide station in front of the delay station which will only direct pallets to corresponding B2 processing area if there is enough space. The model consists of 8 subsystems as shown in Figure 1:

- Pallets arrive on the left side of the subsystem from the predecessor station. This arrival takes some time, which is modelled by the most left delay station.
- As the next step it is checked if there is enough free capacity on the upper processing line. This is done by the decide station. Also some heuristic considerations for shifting a pallet or to keep it on the B1 bypass main line are done here.
- If there is not enough space, the pallet will stay on the lower bypass lane and be moved to the exit via the main conveyor belt (B1 area).
- If there is free capacity in the B2 processing area, the pallet is shifted to the processing line (Sx).
- The upper processing line (B2) consists of two delay station on the left and on the right modelling the conveyor belt in the B2 area.
- The processing in the B2 area is modelled using a small delay station for the conveyor right at the process station and the process station itself. A process station can in contrast to a delay station only process a limited number of pallets at time. In this model the process stations can only handle one pallet per time.
- After processing the pallets are shifted back to the main conveyor belt (Sy).

The decision to shift to B2 or to stay at the B1 lane is made at each subsystem from the available capacity at B2, the need to process a pallet at a particular station and some heuristic considerations:

- There are three identically A2 stations (A2a, A2b and A2c). Each pallet only has to be processed at one of these stations. To balance the workload between the three A2 stations, the first will only take every third pallet which needs A2 processing, the second only every second pallet that needs A2 processing and the third accepts any pallet which needs to get A2 processing. (See appendix of this article for the exemplary Javascript code used to decide if an arriving pallet is to be processed at A2a.)
- Pallets needs processing either at the stations A3, A4 and A5 or at station A6 only. Station A3 is bypassed if there are less than three pallets in the B2 lane of A6 (since processing at A6 is faster than the sum of A3, A4 and A5 which would be needed as alternative).
- A4 is bypassed if A3 was bypassed (due to the heuristic rule or due to a too high load at A3) because in this case A6 (which covers the jobs done at A3, A4 and A5) has to be visited anyway. For the same reason A5 is bypassed if A3 or A4 processing was bypassed.

See Figure 2 for a schematic overview of the entire modelled system. At A1 the workpieces are loaded on the pallets and also the finished products are unloaded. Workpieces can be processed at one of the A2 stations (in the upper half of the main circle in the figure) first and then in the A3 to A6 area or vice versa.

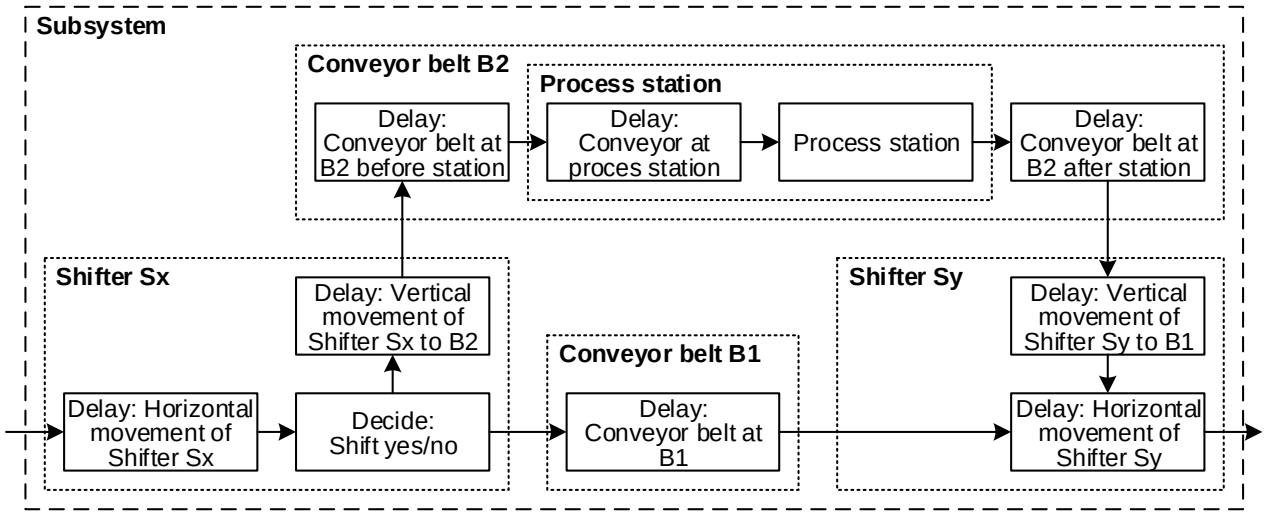

Figure 1: Schematic illustration of a single subsystem.

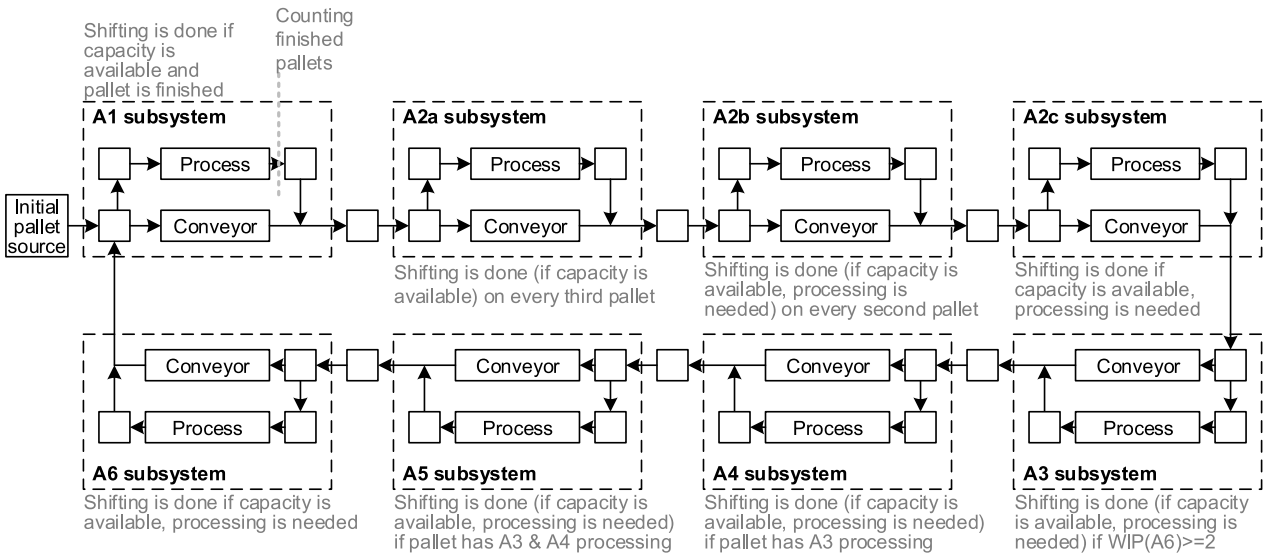

Figure 2: Schematic illustration of the entire system.

If the B2 processing areas of all possible stations in one area are blocked, a pallet may need to do several laps in the system, which will increase the lead time for the workpiece (the model files considered in this article are available for download, see URL in appendix).

3 Analytical considerations

- The minimum lead time of a pallet, which will be reached if there is only one pallet in the system, is 191.667 seconds:

Processing at one of the A2a to A2c stations (72 seconds) and bypassing the other two (8 seconds each), bypass stations A3, A4 and A5 (since A6 is empty; 8 seconds each), processing at A6 (43.333 seconds) and finally processing at A1 (28.333 seconds).

Additionally there are six short conveyor belts connecting the stations on the upper and the lower part of the model (1.333 seconds each; see the small boxes between the submodels in Figure 2).

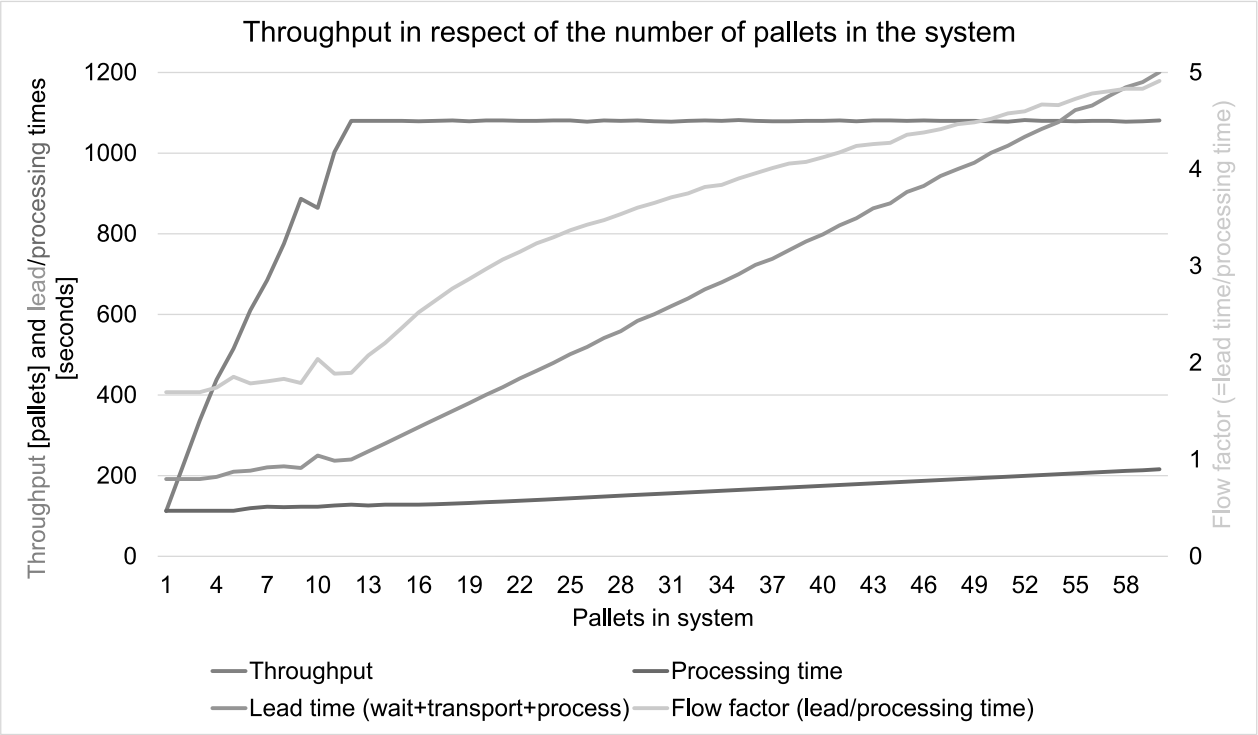

Figure 3: Throughput in respect of the number of pallets in the system.

In case of only one pallet in the system always the fastest processing options can be chosen and a workpiece will never need to do multiple laps in the systems resulting in the minimum possible lead time.
On the other side this would result in a throughput of only 112 workpieces within the considered time of 6 hours.
The simulation results (see red throughput graph on the left edge of Figure 3) confirm these considerations.

- From the simulation results (see blue graph in Figure 4) we will see that the A2 stations are the bottleneck of the system, i. e. these stations are running at 100 % utilization while there is still unused available work performance at the other stations.
  The three A2 stations can handle one pallet per minute each. Therefore the maximum system throughput within 6 hours is 1080 pallets, which is confirmed by the throughput simulation results in Figure 3 (see red graph).

4 Simulation results

The key performance indicators in respect of the total number of pallets in the system are shown in Figures 3 (throughput, lead times, processing times and flow factor) and 4 (utilization of the individual stations). The total simulated time was 8 hours for each model, but the first two hours (used as warm-up phasis) are not recorded to statistics due to the benchmark definition.

Since the simulated model is a closed queueing network and there is a time valued used as termination condition for the simulation the automatic parallelization which Warteschlangensimulator offers on open queueing models can not be applied here and so the model is simulated using only one CPU core. Typical wall clock times needed to simulate the 8 hours are in the range below 0.5 seconds. The number of simulated events in this time is between 10,000 and 650,000 (depending on the number of pallets in system), i. e. over one million simulated events per second on models with higher number of pallets.

To get the maximum theoretically possible throughput of 1080 pallets within 6 hours, only 12 pallets in the system are needed (red graph in Figure 3). Increas-

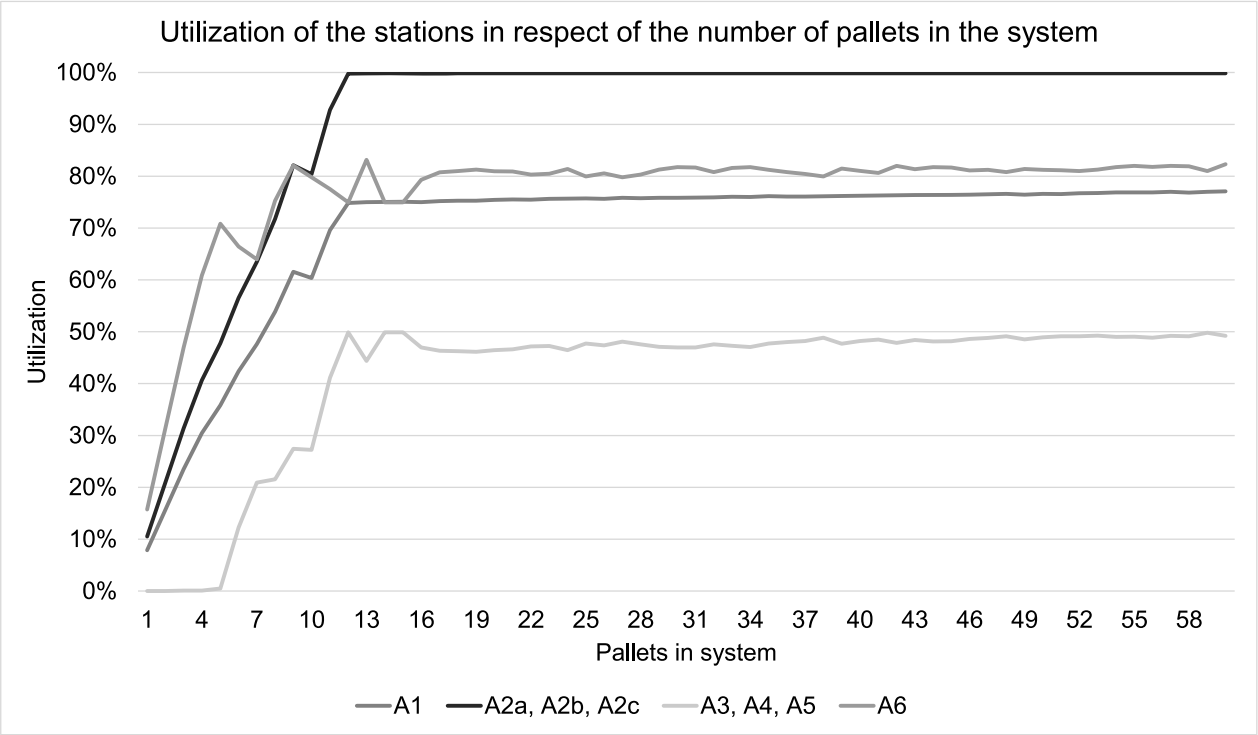

Figure 4: Utilization of the stations in respect of the number of pallets in the system.

ing the number of pallets further only increases the lead times (green graph) but not the throughput itself. The processing times (blue graph) are increasing also since on high load more pallets have to go through the longer A3, A4 and A5 path than the faster A6 only path.

5 Optimization

Figure 4 clearly shows that the A2 stations are the bottleneck which is limiting the throughput (blue graph). Decreasing the processing times at these stations from 60 to 50 seconds (i. e. by 16.6 %) would increase the throughput to 1295 pallets (by 19.9 % compared to the original model), see red graph in Figure 5. The new optimal number of pallets in the system would be 14.

Comparing Figure 3 and Figure 5 also shows that the average lead time and the flow factor is lower on identical number of pallets in the system. Since the A2 stations have been the bottleneck, increasing the performance at these stations reduces the number of pallets which have to do multiple laps in the system.

In the new model the A2 stations are still the bottleneck (and are working at 100 % utilization) but the utilization of the other stations is better (90 % at A1 and 87 % at A6 instead of 76 % and 82 % before). So increasing the available work performance at the A2 stations also leads to a better utilization of the available work performance at the other stations. When the available work performance at the A2 stations is increased in a way they are no longer the bottleneck, the next station to be considered for optimization would be A1.

References

[1] Model definition: C02 – Flexible Assembly System www.sne-journal.org/benchmarks/c02

[2] Warteschlangensimulator homepage a-herzog.github.io/Warteschlangensimulator

[3] Herzog, A. *Simulation mit dem Warteschlangensimulator*. Wiesbaden: Springer Gabler; 2021. 498 p.

Model files

The corresponding model files of the original model and the optimized model (shorter processing times at the A2 station) can be downloaded from:

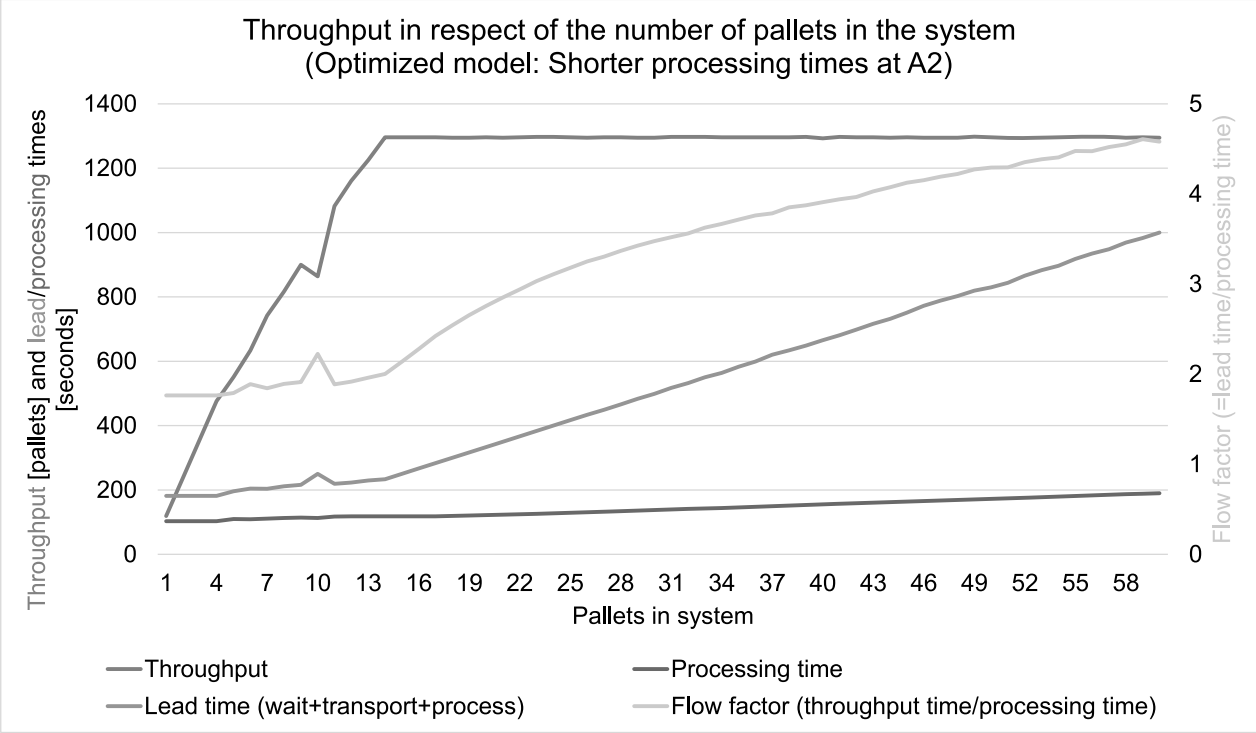

Figure 5: Throughput in respect of the number of pallets in the system – increased processing times at the A2 stations.

github.com/A-Herzog/Warteschlangensimulator/tree/master/sne-benchmarks

The model `xml` files can be loaded directly into Warteschlangensimulator and simulated or animated there. The parameter studies (`zip` files) on which the Figures 3, 4 and 5 are created are also contained in the download package available via the URL above.

Javascript code for the shift decision at A2a

The following Javascript code is used for deciding if a pallet arriving at the A2a station is to be shifted to the B2 line for processing of if it will stay on the B1 lane and bypass processing.

```
// Get current number of pallets in B2 lane.
   let wipB2part1=Simulation.getWIP("A2a B2(1)");
   let wipB2part2=Simulation.getWIP("A2a horizontal");
   let wipB2part3=Simulation.getWIP("A2a");
   let wipB2=wipB2part1+wipB2part2+wipB2part;
// Count number of arriving pallets at this station
   let counter=Simulation.calc("A2aCounter")+1;
   Simulation.set("A2aCounter",counter);
// B2 has a capacity of 3. So processing there is
// only possible, if there are less than 3 pallets.
   let canB2=(wipB2<3);
// Does the pallet require processing at A2?
   let needA2=(Simulation.getClientValue(2)==0);
// Heuristic: Only process every third pallet
// passing A2a.
   let selectPallet=(counter%3==0);
// Decide if pallet is to be shifted.
   if (canB2 && needA2 && selectPallet) {
     // Status: A2 done
     Simulation.setClientValue(2,1);
     // Shift to B2
     Output.print(2);
   } else {
     // Stay on B1 lane
     Output.print(1);
   }
```

The code at the stations A2b and A2c is very similar. Only the `selectPallet` statement differs: A2b selects every second pallet passing the station and A2c every pallet which passes the station and needs processing.