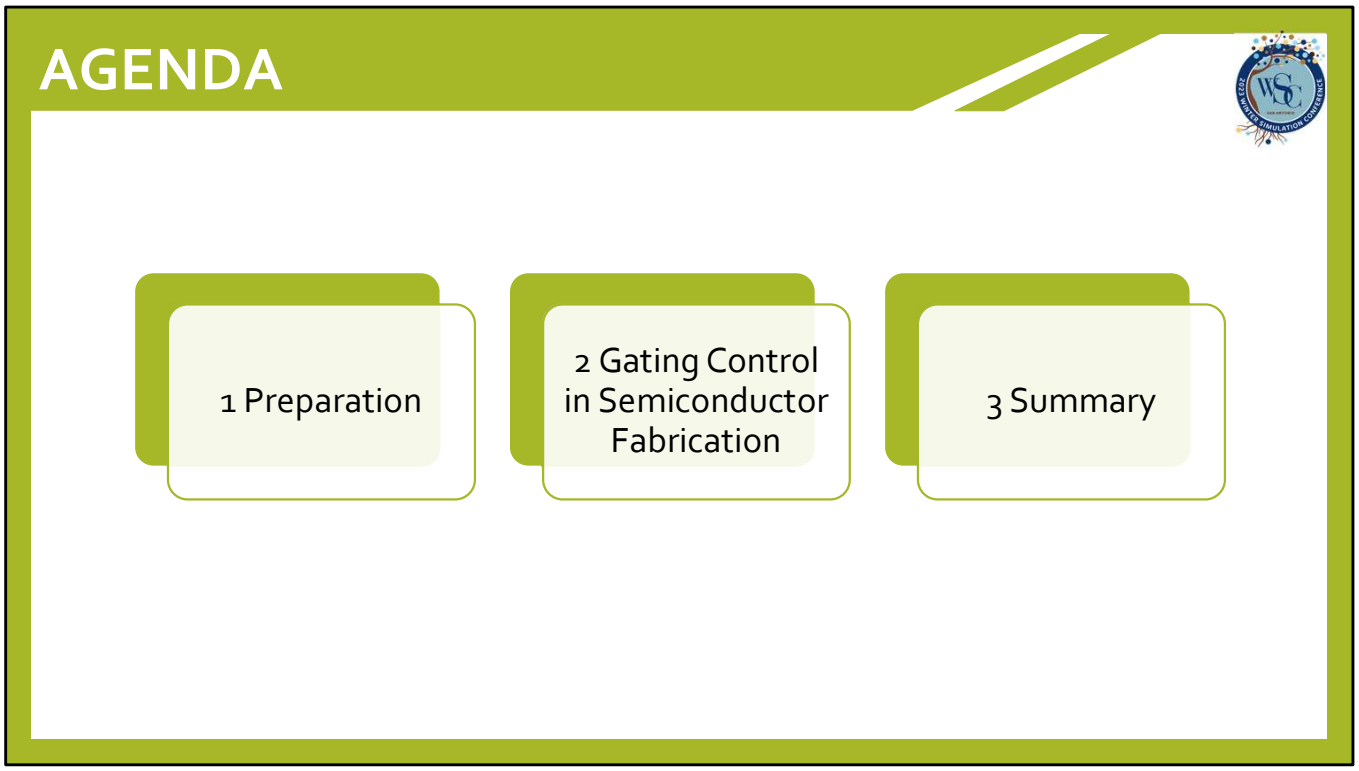
WSC 2023 Simulation Challenge

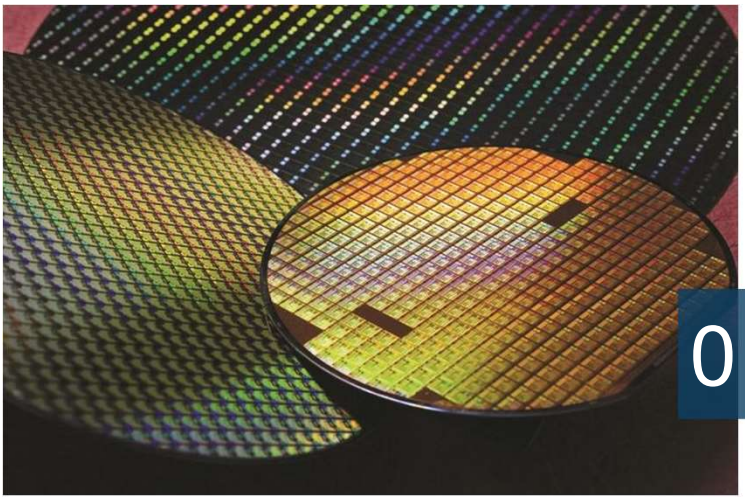
Gating Control in Semiconductor Fabrication

Welcome to the WSC 2023 simulation challenge. The case study for the competition is based on the Gating Control in Semiconductor Fabrication and this set of materials aims to help participants get a better understanding of the system as well as what they are required to do.

AGENDA



First, we will cover the preparation material offered to participants and how they can make full use of the materials; Second, we will provide a detailed illustration of the Gating Control in Semiconductor Fabrication and what is expected from participants. This will then be followed by a summary.



01 Preparation

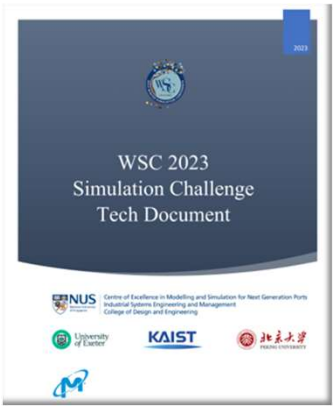
- Download Tech Document
- Download Source Code Package

Let’s start with the first part.

PREPARATION

Download Tech Document

- PDF Reader
- English Version

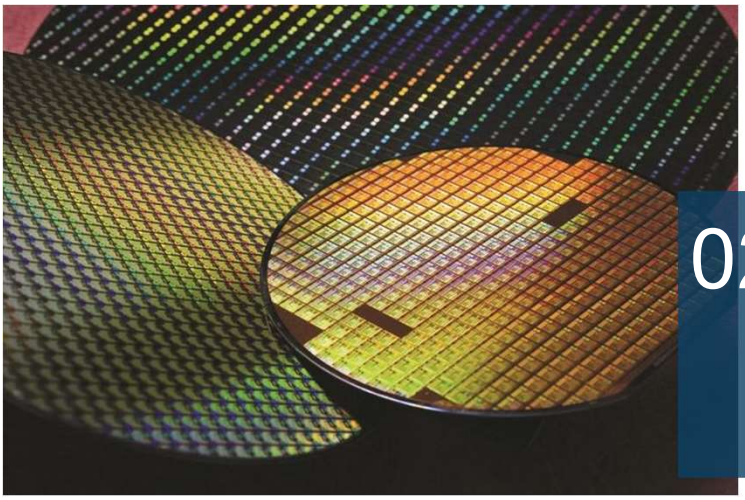


Download Source Code Package

- Decompress Package
- #C or Python

WSC Simulation Challenge 2023

Upon successful registration, participants will be provided with a technical document and a zip package containing the source code. Please refer to the document for further details regarding the case description, model structure, and instructions on file downloading and submission procedures. Once the source code is downloaded and unzipped, it can be viewed and executed via Visual Studio or PyCharm.



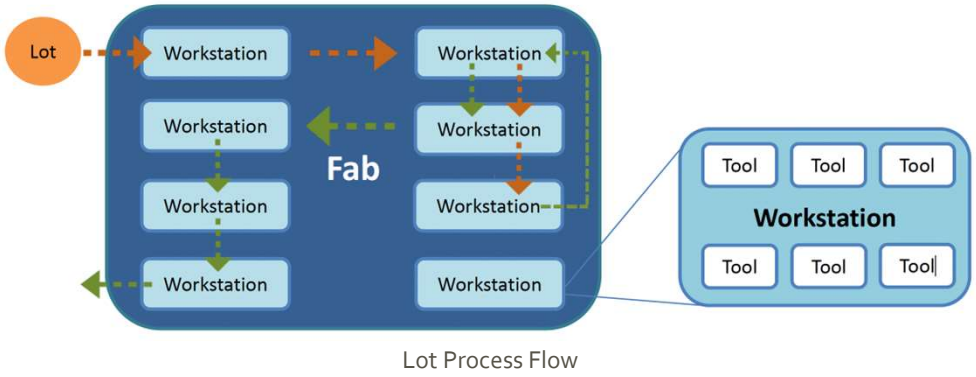
02

Gating Control System

- Background
- Formulation
- Gating Factor Logic
- Objective
- Entity Relationship Diagram
- Event Graph
- XML Input
- Editable file

Next, we have a detailed illustration of the Gating Control in Semiconductor Fabrication.

BACKGROUND



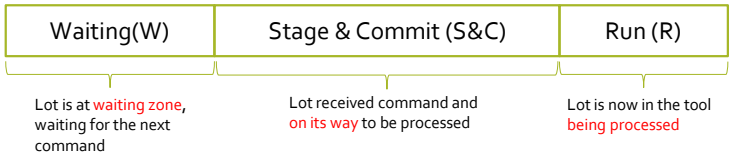
Lot Process Flow

In semiconductor manufacturing, the fabrication process involves multiple workstations. During fabrication, Lots undergo multiple steps for completion, and they are transported between workstations that contain multiple tools performing the required process steps. The Lots that are being processed in each step are referred to as the work-in-progress (WIP) of that step. Different types of Lots have different steps and Queue Time (QT) Loops. Some Lots are notably time-sensitive, so their queueing duration profoundly impacts their successful completion. Therefore, efficient time management is an absolute necessity to avert potential breaches.

Note that, given the intricacies of this procedure, re-entry of Lots to preceding workstations is possible.

FORMULATION

Each step is broken down into three distinct statuses as shown below:



Single Step Status Breakdown

Then, we start to describe our problem. Lots undergo multiple steps and each step is divided into three-step statuses: Waiting, Stage & Commit (represented by Stage in the event graph and code), and Run.

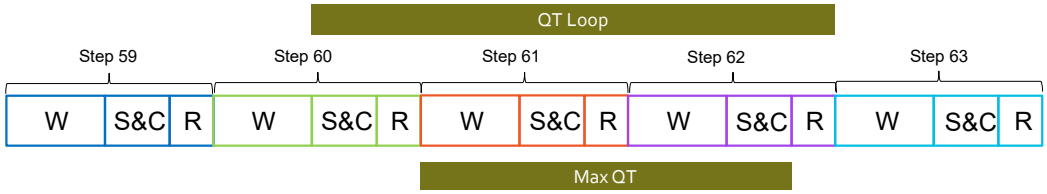
Waiting (W) - Lots that are in this step status have just left the previous workstation and are traveling to the next workstation. These Lots have not been assigned a Tool yet.

Stage and Commit (S&C) - Lots that are Staged and Committed have been assigned a Tool. There are minute differences between the Stage and Commit step statuses. However, these two-step statuses have been simplified as one-step status in the O^2DES simulation model. Note that the Lot being assigned a tool does not mean that the "stage" can be officially started. Only when the capacity of the workstation is sufficient and the Lot occupies the tool can the "stage" be officially started. More

details can be found in the event graph.

Run (R) - Lots in the Run status currently have their wafers processed by the assigned Tool.

FORMULATION



A Queue Time Loop (QT Loop) runs from the S&C step status of the source Workstation to the end of R step status in the destination Workstation.

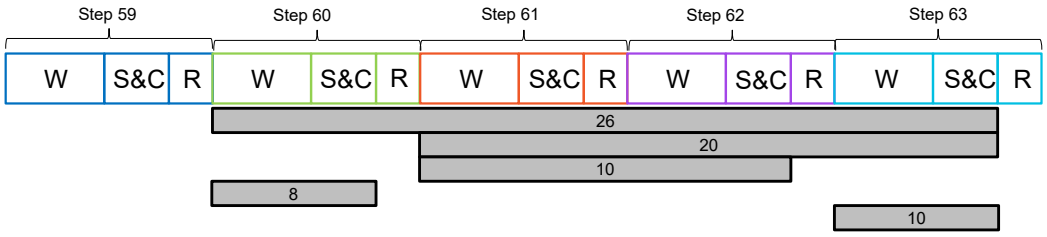
Maximum Queue Time (Max QT) represents the maximum time that Lots can spend between the R step status of the source Workstation and the R step status of the destination Workstation.

A breach occurs if a queue time exceeds its Max QT, and the Lots are considered defective. Once a breach occurs, a penalty will be incurred.

Before delving into our problem description, it is crucial to comprehend the definitions of some **key concepts:**

A Queue Time Loop (QT Loop): it runs from the S&C step status of the source Workstation to the end of R step status in the destination Workstation. This is because the Gating Factor Logic is applied before the S&C step status of the source Workstation. The Gating Factor Logic is an algorithm (elaborated in further detail later) that determines whether a Lot can be permitted into a particular Queue Time Loop. The decision is based on the WIP whose QT Loop's destination Workstation is same as this particular QT Loop's. The source Workstation's S&C and R step statuses are included as part of the Queue Time Loop because once a Lot is released into the source Workstation's S step status, it is unlikely to be removed. Therefore, its workload must be considered every time the Gating Factor Logic is applied to Lots further behind in the queue.

FORMULATION



Multiple Nested Max QTs On One Product

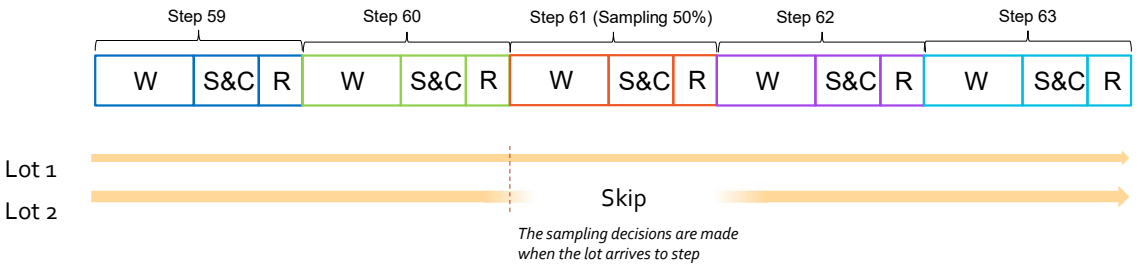
Maximum Queue Time (Max QT): amount of time a Lot can spend between the R step status of the source Workstation and the R step status of the destination Workstation. Exceeding this max QT results in a breach, then the lots are considered defective. Once a breach occurs, a penalty will be incurred.

Max QTs may not only be imposed between two steps. There may be Max QTs imposed over a series of steps, and with other Max QTs nested within. Such an example is illustrated in the figure above. Nested Max QTs and Max QTs that stretch across a few steps have their own Queue Time Loops.

FORMULATION

Sampling

Some of the Inter Steps are sampling steps, and the sampling decision of each lot can only be known when the lot has reached the sampling steps.



We include some sampling steps, which are closer to Micron's real fabrication and allow more flexibility in our problem formulation. The sampling rate shown in the figure is 50%. When the Lot does not make the sampling decision, it simply skips the sampling step.

FORMULATION

Gating Factors (GFs) is a ratio between 0 and 1 imposed on each workstation in order to control the flow of Lots downstream by comparing its capacity with the current workload. The **GF** needs to be defined at each QT Loop to reduce the workload that is allowed to enter a QT Loop such that

$$\frac{Workload}{Max\ QT} \leq \boldsymbol{GF}$$

Engineers determine the GF based on past experiences and observation of the current queuing situation at a workstation. This poses risks of:

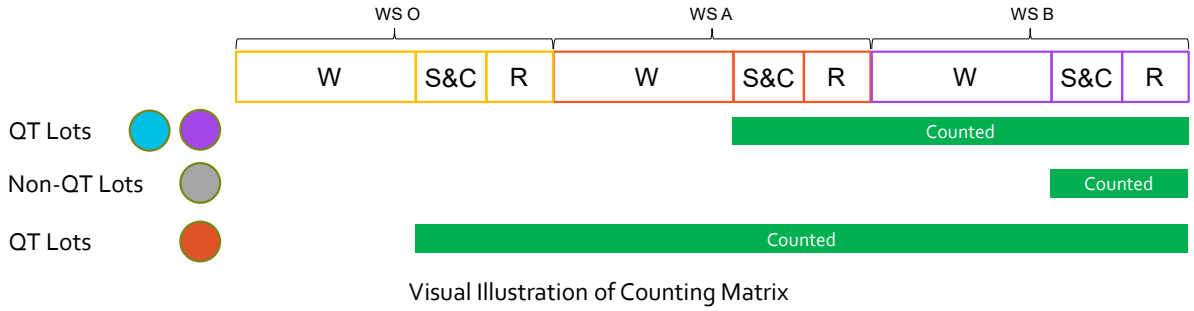
- Overestimating GF, which increases the possibility of breaches, and
- Underestimating GF, which results in idle capacity.

The Gating Factor Logic decides which Lots are released to the next Workstation and which Lots are held back based on the WIP in a particular Queue Time Loop, the Max QT imposed on that Queue Time Loop, and the Gating Factor imposed on that Workstation. Lots that are held back are referred to as being gated. The calculation of the current WIP in the Queue Time Loop depends on the lot type and its current step status.

The objective for participants is to determine the most suitable gating factor that will result in as few breaches and as little idle time as possible.

GATING FACTOR LOGIC

Gating Logic Counting Matrix			
	Lot Status	Source Workstation	Destination Workstation
Non-QT Lots	SCR	Not counted	**Counted**
	Waiting	Not counted	Not counted
QT Lots	SCR	**Counted**	**Counted**
	Waiting	Not counted	**Counted**



Visual Illustration of Counting Matrix

For QT Lots, they have at least one QT Loop whose Destination Workstation is the same as that of the particular QT Loop that the current Lot needs to enter. For Non-QT Lots, they do not have such QT Loop.

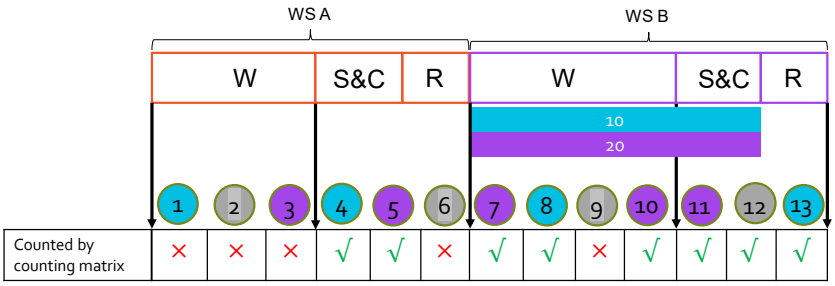
Now, the potential lots joining the workload are determined as follows:

1. Find all QT Loops with a Destination Workstation identical to that of the particular QT Loop which the current Lot needs to enter (including this particular QT Loop itself).
2. The WIP of each QT Loop is judged. If the QT Lots are in the Waiting status of the Source Workstation, they are not counted, otherwise they are counted.
3. Only when the Non-QT Lots are in the Stage & Commit and Run statuses of the Destination Workstation of this particular QT Loop, they are counted.

GATING FACTOR LOGIC

We use an example to show how Gating Decisions are made. The Gating Factor Logic steps are applied as followed:

Step 1: Make an initial list of Lots based on the counting matrix.



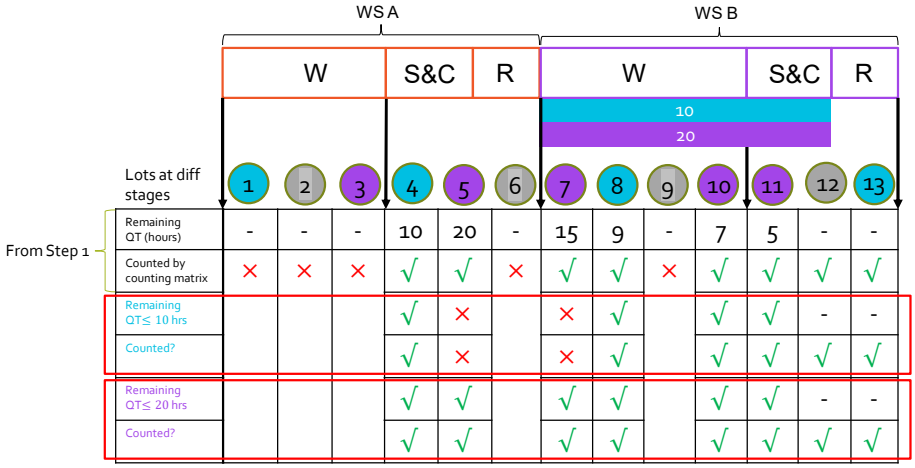
Gating Factor Logic - Step 1

Consider the flow of Lots from Workstation A (WS A) into Workstation B (WS B). At any time, 3 types of Lots flow through WS A and WS B. Amongst them, blue and purple Lots are QTLs with Max QTs of 10 and 20 hours respectively, while grey Lots are non-QT Lots.

Based on the counting matrix mentioned earlier, Lots 4, 5, 7, 8, 10, 11, 12 and 13 are included in the initial list.

GATING FACTOR LOGIC

Step 2: Get the "Counted" list based on the initial list , including QT Lots with Remaining QT until breach ≤ Max QT of the current Lot to be gated, QT Lots in the Stage & Commit and Run statuses of the Destination Workstation and Non-QT Lots. These Lots potentially cause the incoming Lot to breach.



Gating Factor Logic - Step 2

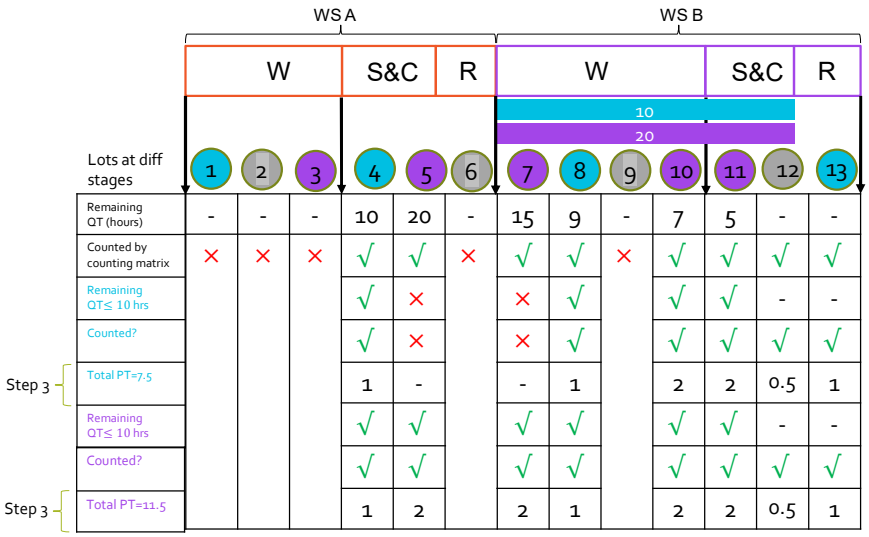
Remaining QT refers to the amount of time a QT Lot has before it breaches its Max QT.

When deciding whether to gate incoming blue Lots at the W step status of WS A, we update the “Counted” list formed in Step 1 to include Lots 4, 8, 10 and 11, because these Lots’ Remaining QT ≤ 10 hours (Max QT of blue Lots). In other words, these Lots have the highest potential to cause any incoming blue Lots to breach. Lots 5 and 7 are excluded because their Remaining QT is large enough so that their presence does not cause incoming blue Lots to be at significant risk of breaching. In addition, Lots 12 and 13 are included, because they are in the Stage & Commit and Run statuses of WS B. The same step is applied to incoming purple Lots, with the constraint Remaining QT ≤ 20 hours applied instead.

GATING FACTOR LOGIC

Step 3: Calculate the total Process Time (PT) required for the Lots in each "Counted" list.

PT	WS B
1	1
2	2
0.5	0.5



Gating Factor Logic - Step 3

The Process Time of Lots in the workstation consists of the time spent on Stage& Commit and Run. The information is given in the input file of the source code.

When deciding whether to gate the incoming blue Lots, our calculation results in a Total PT of 7.5 hours. For the purple Lots, it is 11.5 hours.

GATING FACTOR LOGIC

Step 4: If Workload / Max QT > GF, gate any incoming Lots. Else, release more Lots into the QT Loop.

Workload = PT assigned to each Tool

Suppose there are 3 tools in Workstation B, and both gating factors applied are 0.24.

1) When gating **incoming blue Lots**, each Tool is assigned a PT of 7.5/3=2.5 hours.
   PT assigned to each Tool / Max QT=2.5/10=0.25 >0.24.
   Therefore, blue Lots **are gated** at the W status of WS A.

2) When gating **incoming purple Lots**, each Tool is assigned a PT of 11.5/3=3.833 hours.
   PT assigned to each Tool / Max QT=3.833/20=0.19< 0.24.
   Therefore, purple Lots **are released** from W status of WS A into the QT Loop.

Here we examine the ratio of PT assigned to each tool to the Max QT. If this ratio is greater than the Gating Factor (GF), any incoming Lots should be held back, or "gated." Conversely, if the ratio is less than or equal to GF, additional Lots should be released into the QT Loop.

When assessing incoming blue Lots, the workload is calculated by dividing the total PT by the number of tools, which is 7.5/3=2.5 hours. Subsequently, we compare the calculated ratio (PT assigned to each Tool / Max QT) to the gating factor. In this case, 2.5/10=0.25, which is greater than 0.24. Consequently, blue Lots will be gated at the Waiting (W) of Workstation A.

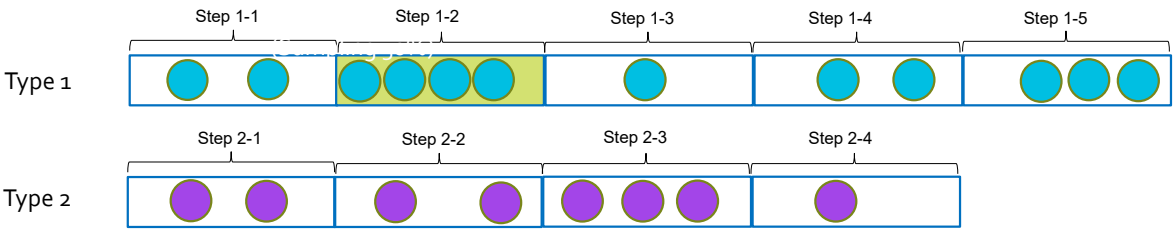
For incoming purple lots, each tool in Workstation B is assigned a workload of 11.5/3=3.833 hours. The calculated ratio (PT assigned to Tool / Max QT) is 3.833/20=0.19, which is less than the gating factor 0.24. Therefore, purple lots are released from

the Waiting of Workstation A into the Queue Time Loop.

OBJECTIVE

1. To minimize the maximum WIP among steps.

- For example, there are two types of Lots in the system. When the simulation terminates, the WIP existing in the system is as follows:



$Maximum\ value\ among\ steps = 4$

The problem is based on a terminating scenario. After a certain period, the simulation terminates, leaving the existing WIP in the system. Our objective function based on the WIP encompasses **Two** parts:

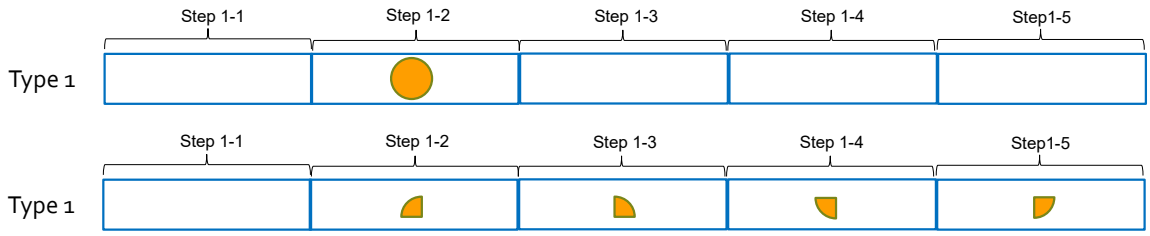
1. To minimize the maximum WIP among steps.

To help understanding, we present a basic example above. There exist two categories of Lots: Type 1 and Type 2. When the simulation terminates, the WIP existing in the system is shown above. Each circle symbolizes a Lot. Our objective is to identify the step with the maximum value across all steps and endeavor to minimize this value. In this case, it is obvious that Type 1 in step 1-2 has the maximum WIP, which is 4.

OBJECTIVE

2. To minimize the penalty value generated by the breaches.

- For example, if a breach happens in step 1-2, the penalty of one unit of WIP will be distributed between step 1-2, step 1-3 ,step 1-4 and step 1-5.

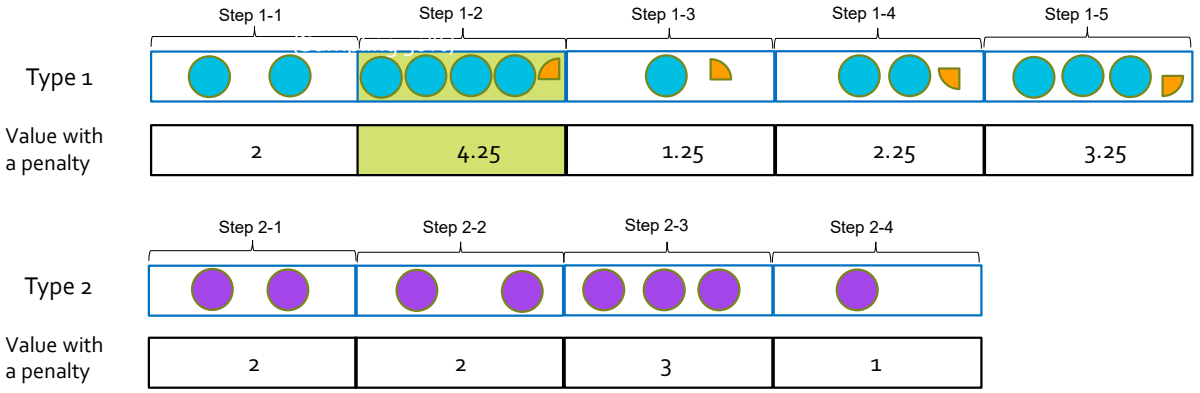


2. To minimize the penalty value generated by the breaches.

If a breach happens at a certain step, the penalty will be incurred where the one unit of WIP will be evenly distributed and anchored permanently across the current and all subsequent steps. The example above illustrates how a breach, once occurred, is distributed in the form of a penalty.

OBJECTIVE

Combing **Two** parts together:



Finally, we combine the two parts together. Here, there is a breach in one of the Lots of Type 1. The Lots of Type 2 don't experience any breach, so the result after combining the two parts is the same as the first part.

OBJECTIVE

- $Objective\ Function = Min\ Max$ { WIP at each step }

Maximum value with a penalty among steps	2	4.25	1.25	2.25	3.25
	2	2	3	1	

$Objective\ value = 4.25$

- You may use simulation, optimization, learning, or a hybrid of them to find the optimal solution to the problem.

In the given example, the value of the objective function is 4.25.

In the challenge scenarios, the types of lots and the number of processing steps involved will become more diverse and complex. We hope everyone brings your zeal and determination to tackle these challenges - remember, every intricate problem is an opportunity for a fun and captivating adventure!

You may use simulation, optimization, learning, or a hybrid of them to find the optimal solution to the problem.

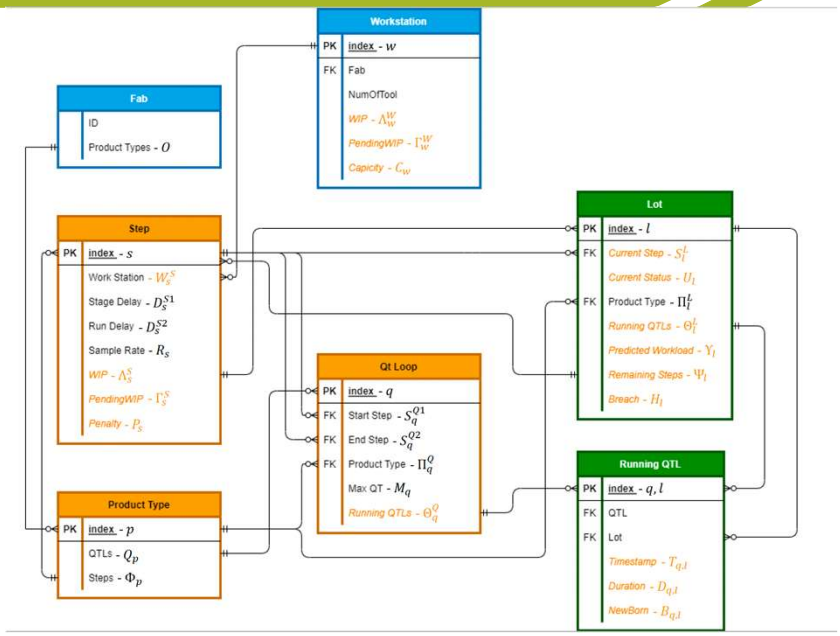
OBJECTIVE

- s: Index of Step
- p: Index of Product Type
- O: The set of product types in Fab
- Φ_p: The set of steps in Product Type p
- Λ_s^S: The set of WIP in Step s
- P_s: The penalty of Step s

$$Objective\ Function = \min \max_{s \in \Phi_p,\, p \in O} (|\ \Lambda_s^S\ | + P_s)$$

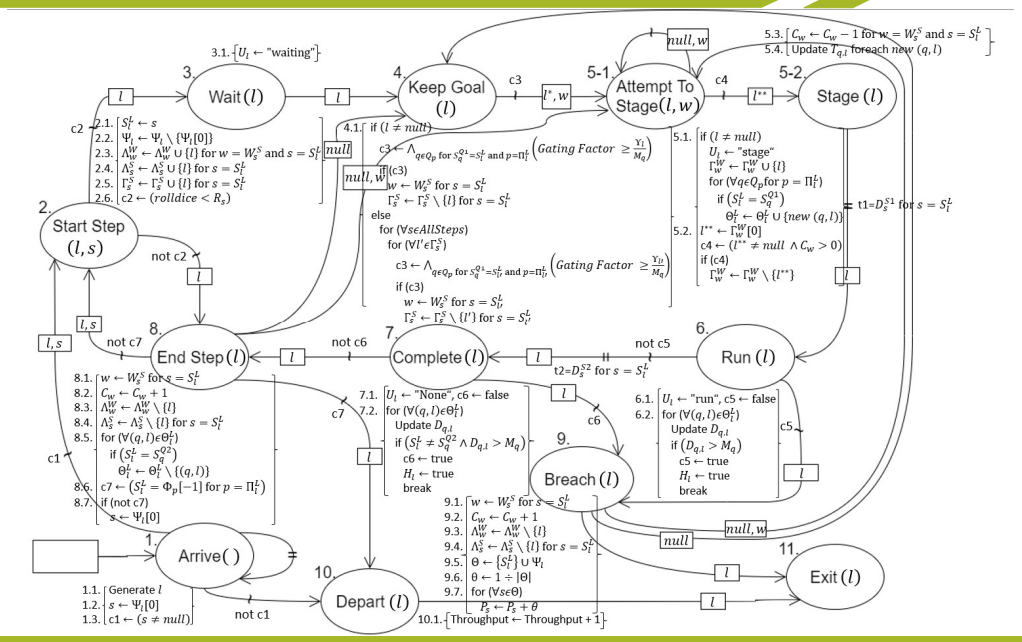
Then we describe our objective function in mathematical notations, which is to minimize the maximum of "the number of WIP in Step s plus the penalty of Step s" among all Steps of all Product Types. More information about notation can be found in the next Entity Relationship Diagram (ERD).

ENTITY RELATIONSHIP DIAGRAM (ERD)



Here is the Entity Relationship Diagram. There are seven Entities, including Fab, Workstation, Step, Produce Type, QT Loop, Lot and Running QTL. Each entity has some attributes. Attributes in black represent static attributes, which are constant. Other attributes (in orange) represent dynamic attributes, which change over time.

EVENT GRAPH

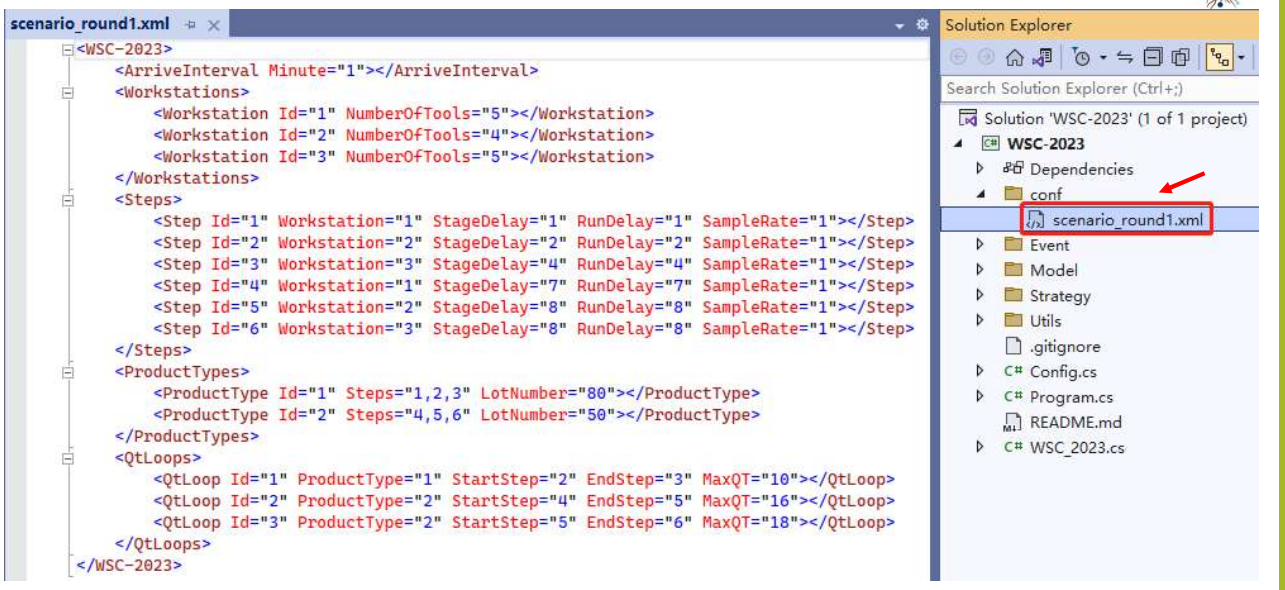


For the Event Graph, there are 12 events: Arrive, Start Step, Wait, Keep Goal, Attempt to Stage, Stage, Run, Complete, End Step, Breach, Depart, and Exit.

A Lot needs to go through Start Step, Wait, Keep Goal, Attempt to Stage, Stage, Run, Complete and End Step to complete a step. When Lot reaches the events Run and Complete, we will respectively calculate whether the current time spent of this Lot exceeds its Max QTs. If it exceeds, it will enter the event Breach.

The gating factor is involved in the event Keep Goal, and the GF determines whether Lots can go from the Waiting status to the Stage status. This is where you need to focus.

XML INPUT



The code specifies various parameters of a system with ArriveInterval, Workstations, Steps, Product Types, and Queue Time Loops (QtLoops) for an initial simulation scenario.

In this scenario, the time interval between the arrival of the Lots in the system is 1 minute on average.

There are three workstations. Workstation 1 and 3 both have 5 tools, while Workstation 2 has 4 tools.

We have six steps in this scenario, for each of which we provided the associated workstations, the time spent on Stage, the time spent on Run, and the probability of being sampled. The sample rate here is 1, which means that none of the steps can be skipped.

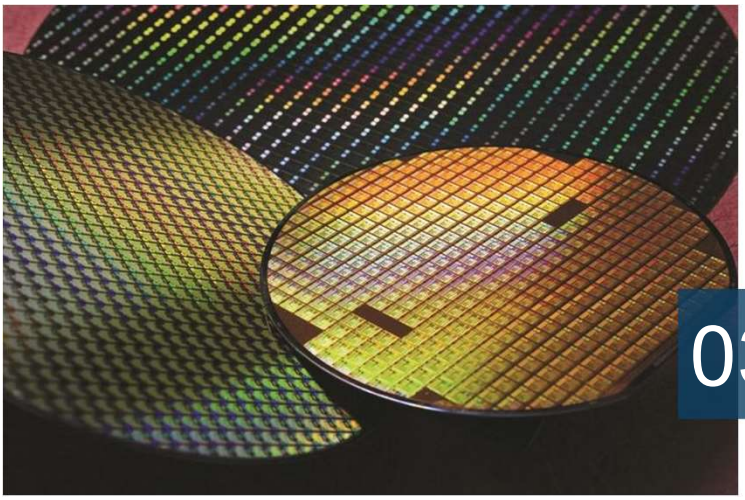
Two product types are defined in this scenario. Type 1 needs to

go through Steps 1, 2, and 3, with a total of 80 Lots. Type 2 involves Steps 4, 5, and 6, with a total of 50 Lots.

The scenario also defines three Queue Time Loops along with their associated ProductType, StartStep, EndStep, and MaxQT. For instance, the QtLoop 1, is associated with Product Type 1, starts from Step 2, ends at Step 3, and has a Maximum Queue Time of 10.

EDITABLE FILE

UserDefined.cs is edited by participants. Here, you're invited to implement your own code to define gating factors in your unique way, effectively controlling the flow of WIP in the simulation. The current value returned by UserDefinedGatingFactor() is null. The default method can be found in Default.cs, which sets the gating factors that allow all WIP to pass through.



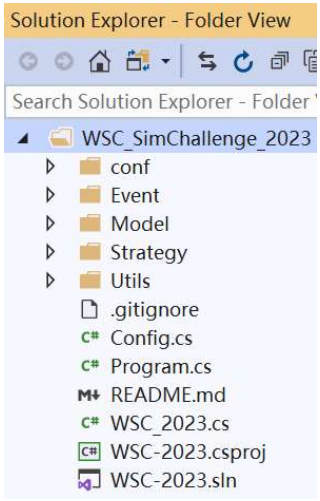
03 Summary

- Structure of Simulation System

Finally, let’s summarize the simulation system.

STRUCTURE OF SYSTEM

Gating Factor System Structure



Folder Name	Files Contained
conf	scenario_round1.xml
Event	Arrive.cs; AttemptToStage.cs; Breach.cs; Complete.cs; Depart.cs; EndStep.cs; Exit.cs; KeepGoal.cs; Run.cs;, Stage.cs; StartStep.cs; Wait.cs.
Model	Fab.cs; Lot.cs; ProductType.cs; QtLoop.cs; RunningQTL.cs; Step.cs; Workstation.cs.
Strategy	Default.cs; UserDefined.cs
Utils	Parser.cs; Statistics.cs; Workload.cs

The conf folder contains the scenario files (XML Document). When the competition progresses to Round 2 and 3, we will provide scenario_round2.xml and scenario_round3.xml respectively.

The Event folder contains 12 files corresponding to the 12 events of the simulation model.

The Model folder contains 7 files corresponding to the 7 entities.

The strategy folder contains the Default.cs and UserDefined.cs files.

The Utils folder contains the toolkits to help parse the config file, format output, and calculate the workload.

Thank you for participating in the WSC Simulation Challenge 2023. Good luck!