INFICON

WHITE PAPER

Unlocking Smart Manufacturing Capability with Automatic Throughput Calculation and Quality Assurance

1. Introduction

The maintenance of accurate equipment throughput information is critical to the business of semiconductor factories. Throughput data are used in capacity modeling, planning, process engineering, cycle time management, as well as many other types of analyses related to the operations of a fab. The value of a “perfect” throughput data set, to accurately forecast the time required for any lot to run on any tool with any recipe, is self-evident. However, many practical challenges make the creation and curation of such a data set difficult.

Historically, throughput studies have been done using stopwatches and manual data collection. While accurate, this approach takes a large amount of time and labor due to the volume of throughput information needed. This approach also suffers from the problem that actual tool throughput changes over time as the recipes are iterated. This can lead to information becoming out of date. More recently, automatic means of calculating throughput have been developed using data present in the MES, as well as data from other systems (FDC, EI, tool logs, wafer tracking systems, etc.) with relevant information. Automatically-calculated throughput data is the best way to guarantee a comprehensive and current data set. The main challenge with this approach is ensuring that the values are accurate. Automatic calculations use time-stamped events to determine throughput. The accuracy of the results depends on the accuracy of the input data used.

Until recently, throughput data has been used mostly in areas of the fab outside of operations. As fabs move toward Smart Manufacturing standards, however, accurate and appropriately granular throughput data is a fundamental requirement. Full factory scheduling and advanced WIP management, for example, are not possible without accurate and comprehensive throughput data. Therefore, throughput data should be viewed not only as essential for planning, but also as a requirement for improved and smarter factory operations. In this paper we review the challenges associated with calculating a throughput data set automatically and describe solutions to overcome them, as well as to ensure the overall quality of the resulting information.

2. The Scope of the Challenge

In this section, we will review some of the challenges and complexities of calculation a throughput data set.

2.1 Data Size

How big is a perfect throughput data set? How many entries should be in it? Let’s make an order-of-magnitude estimate.

In a typical semiconductor fab, it takes many hundreds of processing steps on a route to turn a bare wafer into a finished product ready for dicing and assembly. Each product a fab makes has its own route with a unique order and subset of steps. Each processing

step can be performed by one or many tools. An ideal throughput data set should store a record for any possible processing step combination. Therefore, at the most granular level, throughput data should include a numerical time estimate for every eligible tool to run each step on every specific route.

As a simple example, a fab that makes ten products on ten routes that have 1000 steps each, where each step has an average of three eligible tools that can run it, would have roughly:

10 products/routes x 1000 steps x 3 tools/step = 30,000 rows in a throughput lookup table

Foundries with many diverse products can have a much larger potential data set, whereas smaller or less diversified fabs would have a smaller set. Nevertheless, a number on the order of $O(10^4)$ entries or greater is a good starting point to estimate the size of a perfect throughput data set.

In practice, many simplifications can be made that can reduce the size of the data set. For example, it is often the case that many (route, step) combinations are not unique physical processing steps. An ALD tool that deposits a 100 ångström metal film might be step 183 on route A, step 141 on route B, and step 246 on route C, but the time it takes to run the recipe is the same in all three of these cases because the physical process is the same. Thus, some compression can be achieved by grouping (route, step) combinations by the physical process performed. Additionally, it might be decided that unique rows are not needed for each tool, but simply for each equipment type (groups of identical tools). There are other complications, however, that increase the data set size. For example, a multi-chamber tool with identical chamber types may require several rows for a single process, one for each number of chambers used. Differences between parallel and alternate operational modes may need to be considered, as well as different recipe versions for the same process. Additional complexities are discussed below.

Ultimately though, one advantage of automatic throughput calculation is that working with tens or even hundreds of thousands of rows of data is possible. Manually creating and maintaining a similar set is practically prohibitive. Given the expected size and need for fast lookup, the right tool must be employed to manage a throughput data set. Flat files and spreadsheets are capable of the task, but not ideal. They are not meant to provide easy automatic updates or integration with applications and reports. A much better solution is to store throughput data in a Relational Database Management System (RDBMS). A modern database can easily store and perform operations with tables with millions of rows and allow for the definition of functions and procedures that can be useful when interacting with throughput data.

2.2 Complexity Introduced by Tool Properties

Throughput data are affected by several properties of tools that must be accounted for to ensure accurate calculation and retrieval. Here we describe some of the most important ones:

2.2.1 Processing Modes - Single Wafer versus Batch

Tools can be grouped into two general categories - single-wafer, and batch processing tools. Single wafer tools process one wafer at a time, whereas batch tools process many wafers at once. From the perspective of throughput, this leads to a fundamental difference in the wafter dependence of their timing behavior:

- **single wafer tools**: The total job time scales with the number of wafers in the lot.
- **batch tools**: The total job time depends only on the process time, not on the number of wafers in the batch.

These two large groups contain several further subtleties. As an example, disc implanters are like batch tools in that they process many wafers at once, but it is often the case that the disc size is less than a typical lot size. Therefore, the number of discs used affects the way throughput needs to be recorded and retrieved for such tools. The details of distinct processing modes must be properly modeled and accounted for when calculating and retrieving estimated throughput values.

2.2.2 Operation Modes

For tools with multiple chambers:

- **parallel versus alternate**: Tools with multiple load ports and identical chambers can run in either parallel mode (where each chamber runs a different job) or alternate mode (where all chambers process the same job).
- **different chamber paths**: For tools with chambers of different types, the number of chambers of each type and the path wafers take for each recipe influence the ultimate throughput values. Similar challenges exist for wet bench tools with different baths.

2.2.3 Cascading versus Non-Cascading

The timing information from tools that cascade must be interpreted differently from those that do not. Cascading tools can have overlapping processing time events that must be disentangled to determine actual throughput.

These and many more tool details must be accounted for in order to store accurate throughput information.

2.3 Properly Indexing Throughput Values Based on Applications

An automatically calculated throughput data set needs to be indexed properly in order to be useful for different applications. The data must be stored in a way that anticipates how the values will be “looked up.”

In capacity modeling, for example, the main goal is to calculate the required utilization of equipment types (sets of identical tools) based on a plan of intended wafer starts. The plan does not consider which specific tools within an equipment type wafers will visit at each step of their route. If a fab has three identical furnaces at a given step that 50 wafers must run on, the capacity modeler needs to know how much furnace time will be required

at the step, independent of whether the wafers actually run on furnace 1, furnace 2, or furnace 3. Thus, in this case it is useful to store throughput values by equipment type rather than by individual tool.

For advanced factory scheduling, it is useful to know throughput by specific tool. A scheduling algorithm arrives at an optimized schedule by considering hypothetical scenarios like:

1) How long will this lot take if placed on tool A? What about on tool B?

2) Lot XYZ555 is running on tool F and started fifteen minutes ago. When will it finish and arrive at the next step?

In these cases, storing throughput by individual tool is a better approach. The lesson here is that careful consideration of the design of throughput data tables, both in terms of storage and retrieval, is required to ensure maximum utility and applicability.

2.4 Calculation Slope

How much computational power is need to calculate a throughput data set?

In order to calculate throughput automatically, time-stamped job and/or equipment data needs to be used. Ideally, this data is provided directly from the tools via the GEM interface and stored in a factory-wide data system such as the MES. The minimum information requirement is the *process start* and *process end* event for all jobs each tool runs, although more detail can be learned if additional information such as individual wafer processing times are also collected. In the case of a wafer processing tool, each lot is one job and has unique processing start and processing end events. In the case of a batch tool, several lots in a batch may have the same methods by which these events can be used to calculate throughput values. As for the question of computational power, however, an algorithm needs to be able to read historical lot processing data, apply appropriate groupings, calculate averages within the groups, and store the results.

Returning to the example fab in section 2.1, Data Size, on page 2-1, we see that our final result set will have approximately 30,000 rows. The amount of data used to generate these rows can vary based on the window of factory history used to calculate the averages. Let's suppose the same fab has 10,000 active WIP wafers, and that each product has a cycle time of eight weeks. In this case:

$$\frac{100{,}000 \text{ WIP wafers} \times 1000 \text{ steps on each route}}{8 \text{ weeks of product cycle time}} = 12.5 \text{ million processes per week}$$

In this example, an algorithm must process 12.5 million rows of data in order to calculate throughput averages from one week of historical factory data. Including the logic needed to account for the equipment operation and configuration complexities described above, a modern enterprise-level relational database on a server with sixteen cores can perform a calculation of this size in a few minutes. As the time window gets larger, however, the calculation time can greatly increase. This is due mainly to the increased retrieval time of older records from the MES, where history tables are very large and can be slow, rather than the increased number of records used for the computation.

Overall, the computational time and resources required are not insignificant. However, with properly optimized logic they are not a practical barrier to automatic calculation of throughput for a semiconductor fab.

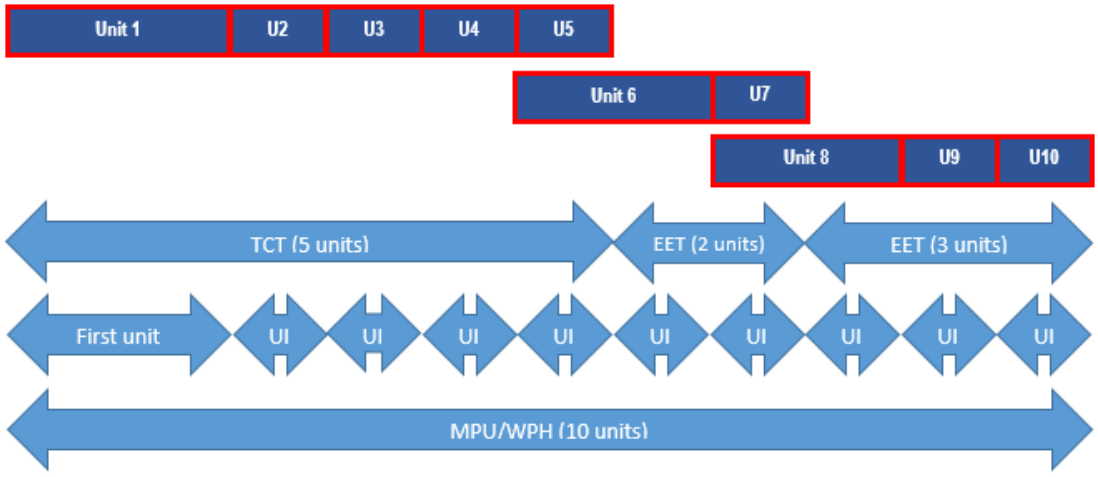
3. Description of a General Approach to Automatic Throughput Calculation

In the previous section we reviewed some of the challenges to calculating throughput data. These challenges include the hardware and software resources required, as well as some of the conceptual and configurational challenges. In this section, we will introduce details of a method to do the calculation itself.

3.1 Definitions

First, we will introduce some useful terminology when talking about throughput. Consider the Gantt chart, shown in Figure 1, of three lots processing in succession.

Figure 1 Gantt chart



- **First Unit**: The first unit is the time required to process the first unit for a tool that is starting from standby. For nearly all tools, the first wafer takes longer than subsequent wafers due to factors like robot handling times. For a single wafer, a unit is a wafer. For a batch tool, a unit is a batch.
- **Unit Interval (UI)**: The unit interval is the time required to process one unit that is not a first unit. For a wafer tool, this would be any wafer other than the first for a non-cascaded lot, or any wafer of a cascaded lot. For a batch tool, this would be a batch that is cascading on a previous one.
- **Theoretical Cycle Time (TCT)**: The TCT is the duration of one non-cascaded job. For a lot with N wafers on a wafer tool: TCT = first unit time + (N-1) x unit interval time. For a batch tool: TCT = first unit time.
- **End to End Time (EET)**: The EET is the duration of a cascaded job after the end of the previous job. For a non-batch tool and a lot with N wafers: EET = unit interval * N. For a batch tool: EET = unit interval.

- **Minutes per Unit (MPU)**: The MPU is the time required per unit. Unlike first unit and UI, which are intrinsic properties of the process and the tool, MPU is an extrinsic property that depends on the cascade length. However, if the three lots did not cascade, there would be additional first unit penalties as well as the missed time between cascades added to the total processing time. This would increase the resulting MPU. MPU is defined as:

  $$\frac{\text{total processing time in minutes}}{\text{total number of units processed}} = \text{MPU}$$

  For many purposes, it is desired to estimate an average MPU for a tool. This is the MPU that results based on running an average cascade length, and if a batch tool, with average batch sizes. For a wafer tool:

  $$\frac{1 \text{ x first unit} + (N_{cascade}\text{-}1) \text{ x unit interval}}{\text{total number of units processed}} = MPU_{average}$$

  Here, $N_{cascade}$ denotes the average number of wafers in a cascade. For a batch tool:

  $$\frac{1 \text{ x first unit} + (N\text{-}1) \text{ x unit interval}}{N_{cascaded\ batches} \text{ x average batch size}} = MPU_{average}$$

  In this case, $N_{cascaded\ batches}$ denotes the average number of cascade batches.

- **Wafers per Hour (WPH)**: WPH is another common quantity used in throughput applications and can be calculate from MPU:

  $$\frac{1}{\text{MPH}} \text{ x 60 minutes} = \text{WPH}$$

3.2 Calculating First Unit and Unit Interval Times from Tool Events

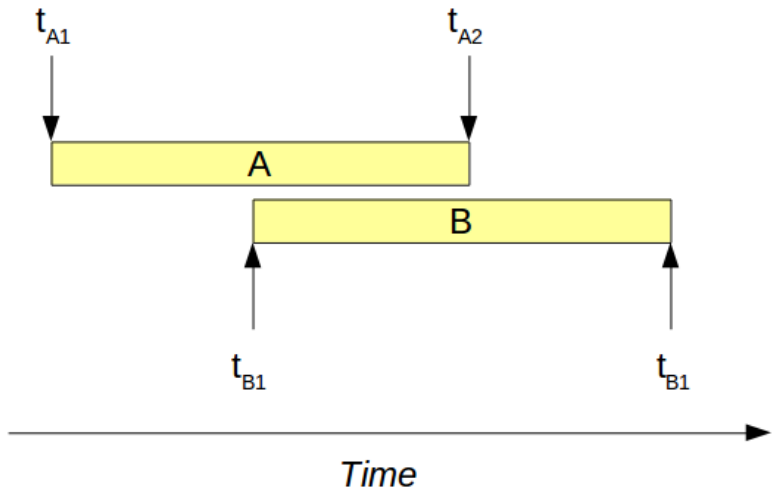
An important question to consider is, “What exact throughput information should be stored?” As explained above, the first unit and unit interval times to run a given process on a tool are intrinsic properties. One the other hand, MPU and WPH are extrinsic, composite quantities that can only be computed using first unit and unit interval times, along with additional information. It is therefore optimal to calculate and store first unit and unit interval times as the base level of throughput data. From these fundamental building blocks, all other relevant information can be calculated.

With the question of what to store resolved, we can now ask, “How can first unit and unit interval times be calculated using tool events?” To do this, we will consider specific examples.

3.2.1 Example 1: Single Chamber, Cascading Wafer-Processing Tool

When a lot is processed on a tool, two triggers need to be recognized and captured from the tool data - the time the processing started and the time the processing ended. We will assume in this discussion that a fab has invested in the hardware and software necessary to capture and store tool events in the MES. Figure 2 shows the case of two cascaded lots that undergo an identical process on a single chamber wafer-processing tool.

Figure 2 Two cascaded lots



The variables are defined as:

t_{A1} = the time that lot A begins processing

t_{A2} = the time that lot A ends processing

t_{B1} = the time that lot B begins processing

t_{B2} = the time that lot B ends processing

With these time stamps, and knowing that the tool is a single chamber wafer-processing tool, it is possible to calculate the first unit and the unit interval times. Let N_A be the number of wafers in lot A and N_B be the number of waters in lot B.

$$\text{unit interval} = \frac{t_{B2} - t_{A2}}{N_B}$$

$$\text{first unit} = (t_{A2} - t_{A1}) - ((N_A - 1) \times \text{unit interval})$$

These definitions are exact for this simple case. Let us also suppose that we have decided to use one week of fab history to calculate our throughput dataset. We can then look for every instance where the same process cascaded on the same tool and average the results to increase our confidence in the calculated unit interval and first unit values.

3.2.2 Example 2: Single Chamber Cascading Batch Tool

The logic for a single chamber cascading batch tool is nearly identical to that in Example 1. The difference is that instead of considering single lots, the logic needs to define jobs which consist of multiple lots. Jobs can be identified by looking at identical start and end times for lots on the same tool, or by other means depending on the data available in the MES and elsewhere. In this case:

t_{A1} = the time that job A begins processing

t_{A2} = the time that job A ends processing

t_{B1} = the time that job B begins processing

t_{B2} = the time that job B ends processing

The results and the interpretation are slightly different now.

unit interval = $t_{B2} - t_{A2}$

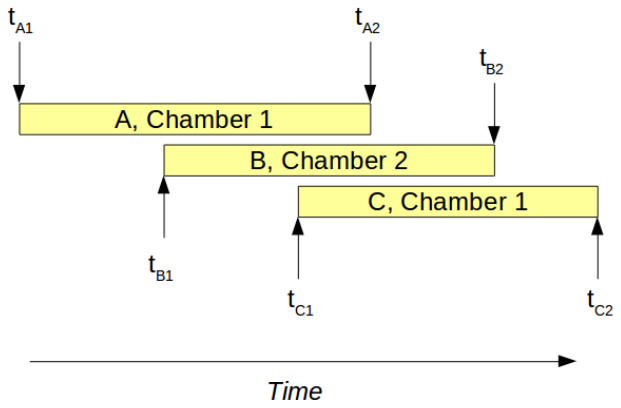
first unit = $(t_{A2} - t_{A1})$ - unit interval

Here the first unit is the first batch that runs after the tool is idle and the unit interval is the time for each subsequent cascade batch.

3.2.3 Example 3: Two Chamber, Cascading Wafer-Processing Tool Running in Parallel

Let's look at an example with two chambers running in parallel. Suppose three lots run in succession. The first lot processes in Chamber 1, the second lot in Chamber 2, and the third in Chamber 1. See Figure 3.

Figure 3 Two chambers running in parallel



t_{A1} = the time that lot A begins processing in Chamber 1

t_{A2} = the time that lot A ends processing in Chamber 1

t_{B1} = the time that lot B begins processing in Chamber 2

t_{B2} = the time that lot B ends processing in Chamber 2

t_{C1} = the time that lot C begins processing in Chamber 1

t_{C2} = the time that lot C ends processing in Chamber 1

The equations from Example 1 need to be modified slightly to take into account which chambers the lots are running on.

$$\text{unit interval} = \frac{t_{C2} - t_{A2}}{N_C}$$

$$\text{first unit} = (t_{A2} - t_{A1}) - ((N_A - 1) \times \text{unit interval})$$

3.2.4 Non-Exact Cases

The previous examples are meant to illustrate simple cases where exact results can be determined. It is not always the case that the first unit and the unit interval values can be calculated using only lot start and end times. For lots that do no cascade or for a non-cascading tool, for example, no definite information about the differences between the first unit and the unit interval can be determined. The examples are not intended to be comprehensive, but rather an introduction to ways in which throughput quantities can be calculated from lot event information.

3.2.5 Other Data Sources

Many fabs store additional information useful for calculating throughput quantities. As an example, wafer-level events make all calculations easier and more exact. A throughput calculating algorithm should incorporate these sources, when the exist, to improve accuracy.

3.3 Averaging Data to Improve Quality

The window of history over which to average throughput is an interesting design choice. If the window is too long:

- the time to retrieve records may be impractically long
- if a tool's performance is degrading over time, this information is lost in the averaging
- a process shift may go unnoticed
- improvements may go unnoticed

If the window chosen is too short:

- there can be little or no data for key processes
- little data or lack or representational jobs can lead to noisy averages

In our experience, for most fabs, one week at a time is a good practical amount of data to analyze. The INFICON FPS Data Warehouse, for example, stores weekly summaries of throughput data. Each week, the previous seven days of lot history is analyzed and the throughput summary statistics are added to a throughput history table. With this architecture, it is possible to store years of data with high compression and utility.

3.4 Lookup and Fill-in Logic - the Final Piece

Let's take a moment here to review the requirements we have discussed so far:

- a database
- proper table architecture to store the throughput data
- additional tables to store equipment configuration information (batch versus wafer, number of chambers, etc.)
- an algorithm to comb through tool event data potentially stored in the MES or any other sources and compute throughput averages from events with methods like the ones above, and store them in the database

A fab with these components in place has a complete throughput storage system. The last piece to consider is the retrieval logic.

The critical property of throughput retrieval logic is that is must return an answer. Applications such as an automated factory scheduler or a capacity model need to be able to get a throughput value for any possible (lot, process, tool) combination. It is always possible that no historical data exists for a certain (lot, process, tool). In that case, the mechanism for returning throughput vales from the tables needs to be able to provide a best guess based on fill-in logic.

A detailed discussion of fill-in logic is beyond the scope of this paper, but we want to call out its necessity. This is yet another reason to use a database rather than a spreadsheet to store throughput information. With a database, a throughput retrieval function can be defined that can access the stored throughput data, combine any other data depending on which value is being retrieved (first unit, MPU, UPH, etc.) as well as do any fill-in logic required to return the best possible answer, all algorithmically.

4. Throughput Quality Assurance

We will now shift our focus from discussing the practical and logical requirements of calculating a throughput data set to what is in some ways a more difficult problem, ensuring that the numbers are meaningful. As with all programmatic pursuits, the principle of “garbage in, garbage out” applies to automatic throughput calculation. In this section we will review some of the contributing causes to meaningless or noisy throughput data, how to identify them, and how to mitigate them for successful applications.

4.1 Real-World versus Perfect-Event Data

When discussing how to calculate throughput quantities from tool events, we have assumed that:

- *process start* and *process end* events exist and are recorded in a factory-wide data system (usually the MES) for every tool set
- these events accurately record the times of the actual lot events

In the real world, however, it is often the case that these assumptions are not true. Very old tools without a GEM interface are not able to broadcast events to a factory-wide data system in real time. It is also possible that a fab has not invested in the hardware and software required to retrieve events from GEM-compliant tools, in which case the event data lives locally on the tool computer only and cannot be easily retrieved. Tools that are set up to broadcast events into a factory data system may still record *process start* and

process end at inconsistent times that do not reflect the actual activities happening on the tool for a variety of reasons. In our experience, the latter problem does not apply only to old tools. We have seen state-of-the-art wafer-event tracking systems that incorrectly record start and end processing events simply due to incorrect capture set up and event identification (for example, assigning the wrong tool signal trigger as a *process start* or *process end* event). Thus, to ensure throughput quality, it is critical to identify which toolsets have potential event recording problems.

4.2 Input Data Variation

With the technique of averaging throughput data over a week of fab history, it is possible not only to calculate means for first unit and unit interval times, but also standard deviations. Ideally, for a given process on a tool, the standard deviation of calculated values should be much lower than the mean. In practice, this is often not necessarily the case. A large standard deviation relative to the mean can indicate a number of issues that should be understood before proceeding to trust and use the calculated means. Reasons for high variation of calculated throughput data are:

- **Real Tool Processing Time Variation**: It is possible for a tool to have inherent job-to-job variation in the time it takes to run the same process. Some examples of this include:
  - an implant tool that many or may not stop to automatically adjust the beam intensity for a varying amount of time during processing (that is, between *process start* and *process end* events)
  - an etch tool that relies on endpoint detection to determine when to stop the recipe
  - a wet bench tool running in parallel where the order of the baths visited by each respective batch can influence the time for the other batch to run
  - a lithography tool with a dimming light source, causing each successive run to be slightly slower than the previous run

  These real sources of variation can lead to large standard deviations relative to the calculated means.
- **Low Numbers of Jobs to Average**: Suppose that tool A only ran process XYZ123 twice in the last week, and that one of the jobs aborted, stranding the lot on the tool for an hour before an operator recovered it. It is not possible to use only these two jobs to calculate throughput for process XYZ123 with high confidence.
- **Inaccurate Tool Events**: As mentioned before, it is possible that the events recorded for a tool are not accurate.
- **Manual or Absent Event Logging**: For some tools, the triggers to record start and end events may need to be manually initiated by an operator. For other tools, *process start* and *process end* events may not be recorded in a factory data system at all. In this case, *track in* and *track out* events may be used as substitutes. In both these cases, as expected, job-to-job variation can be quite high.

- **Fast Recipes**: All time event information recorded in MES is aliased to some degree by the frequency of the various systems recording events. This, along with inherent variation even for a tool with very tight time repeatability, can add up to differences between jobs on the order of seconds. For a short recipe (a ten second metal sputtering, for example), this variation relative to the mean is much greater than a longer recipe (an eight hour diffusion job, for example).

As can be seen from these complications, once a system has been set up to automatically calculate a complete throughput data set, the work is far from done. While the calculation itself is nearly instantaneous relative to the years of labor-hours it would take to generate the same data set manually, the amount of quality assurance work that must be done is significant.

4.3 How to Tell "Good" Throughput Data from "Bad"

The standard deviation of the throughput averages is often a good indicator for the overall quality of the throughput data. Ultimately, automatically calculated throughput needs to be accurate, that is, the numbers should accurately predict what will happen on a tool. However, the accuracy of automatically calculated throughput data is very often closely related to the statistical quality. Thus, there are two important questions to answer:

- Is the data accurate?
- Does it have good statistical quality?

In our experience, it is most useful to start with the second question when beginning to asses the quality of an automatically calculated throughput data set.

Statistical analysis of throughput data can give excellent insight into the types of issues introduced previously. It helps to differentiate tools that have the potential to deliver good throughput data from those that do not. As an example, suppose a particular process on a tool only ran twice in one week. The same tool ran another process that has 53 data points, with a very low standard deviation relative to the mean. The fact that a high running process has good statistical quality gives confidence that the results for the other process with low data can still be trusted. However, a tool that has poor statistical quality, even for high running processes, likely suffers from one or many issues like those explained previously. The issues need to be addressed or the data overridden.

In order to perform statistical analysis, it is useful to look at the coefficient of variation (C_V) of the calculated first unit and unit interval value.

$$C_V = \frac{\sigma}{\text{mean}}$$

A low C_V indicates good statistical data quality, whereas a high C_V indicates a poor statistical quality (high variation).

Let's consider a concrete example. Suppose that we have the following data for process XYZ123 running on tool A:

Week	Unit Interval Mean [s]	Unit Interval Std. Dev. [s]	Number of Jobs
1	110.6	7.2	11
2	114.2	6.8	3
3	113.1	7.3	8
4	109.8	5.9	14

It is straightforward to calculate the C_V for each week by simply dividing the standard deviation by the mean value. This provides the following data:

Week	Unit Interval Mean [s]	Unit Interval Std. Dev. [s]	Coefficient of Variation	Number of Jobs
1	110.6	7.2	0.065	11
2	114.2	6.8	0.060	3
3	113.1	7.3	0.065	8
4	109.8	5.9	0.054	4

For a more robust unit interval estimate for this particular process and tool, we may want to use all four weeks of data. In that case, we would take a weighted average of the unit interval, as well as a properly weighted pooled standard deviation, to calculate the C_V of the unit interval for the last four weeks.

While the math to generate these statistics is precise, the assessment of throughput quality is a more subjective judgment. As a purely pragmatic figure of merit, we consider $C_V = 0.1$ to be useful threshold to sort the quality of throughput data from a tool. More specifically, if the top ten processes run by a tool over four weeks of factory history have a $C_V < 0.1$, then the tool's throughput data has sufficiently good statistical quality to make it unlikely that there are issues with the accuracy. This standard can be different depending on the application as well, since some use cases have wider tolerances than others. An application like capacity modeling has the most stringent requirements for throughput accuracy. In this case, values that are off by 20% can have large effects on predictions of the model. A rapidly updating factory scheduler is not affected nearly as much by a 20% throughput estimate delta.

In our experience across many fabs, with careful analysis and proper equipment configuration, it is possible to get approximately 90% of the tools in a factory to meet the criterion of $C_V < 0.1$ for top processes across four weeks of operation.

4.4 Using SPC Charts for Throughput Data

In addition to knowing the current stability of recent throughput data, it is useful to know if the data starts to become unstable and change. Like many other indicators in a fab, automatically calculated throughput values can be monitored by an SPC system to identify potentially problematic processes.

In this section, we will present several example SPC charts of throughput data automatically calculated in the INFICON FPS data warehouse. The limits shown here are simple 3-sigma limits, though these can easily be configured by process. The charts give an idea of what “good” and “bad” statistical stability look like week to week. See Figure 4 – Figure 7.

Figure 4 Batch tool, long process, good C_V

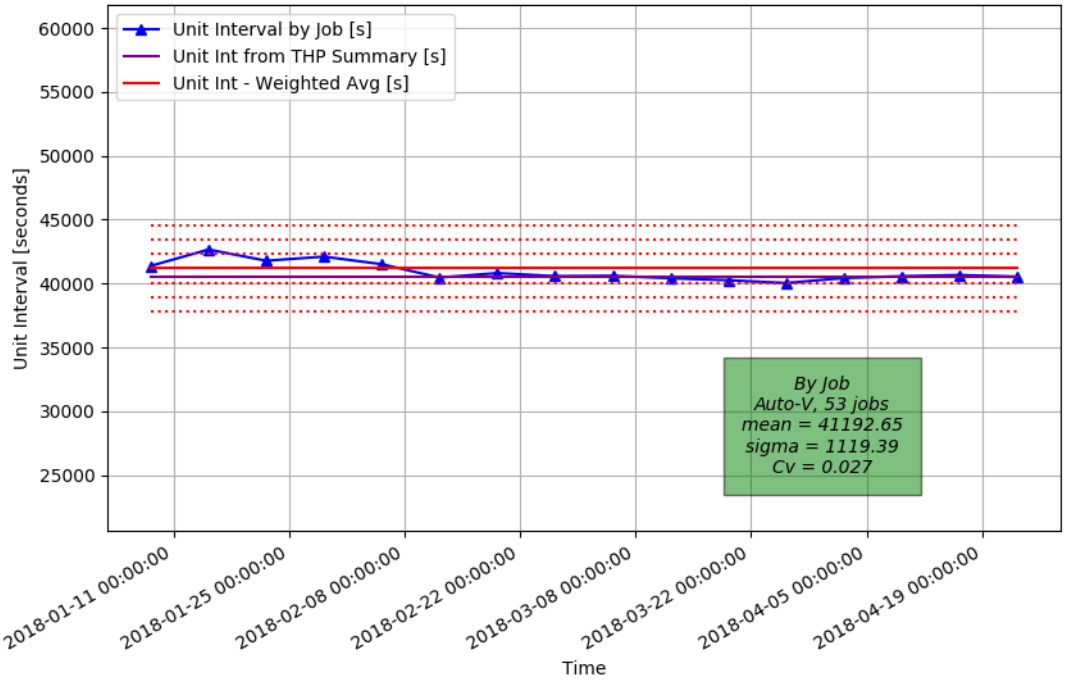


Figure 5 Shorter process, good C_V

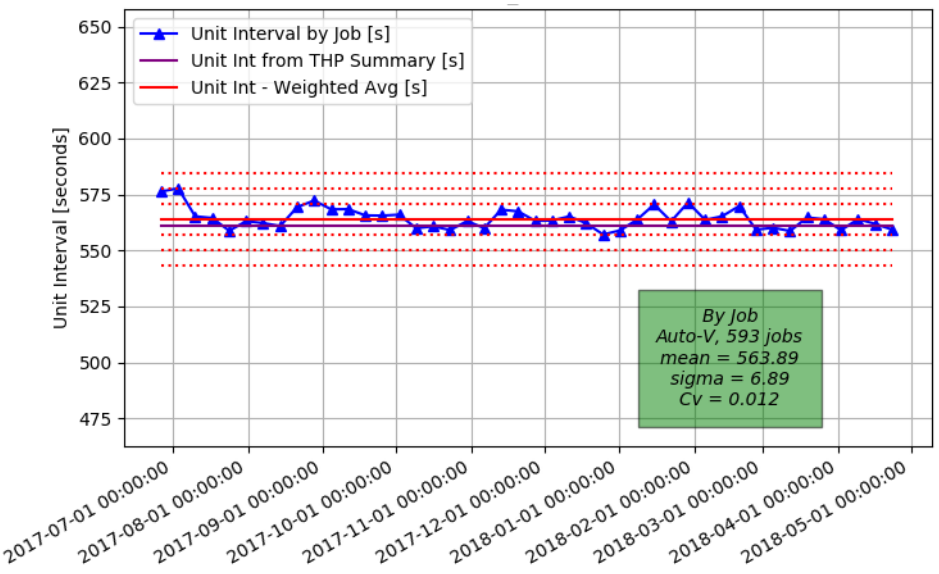


Figure 6 Implant tool, shorter process, borderline acceptable C_V

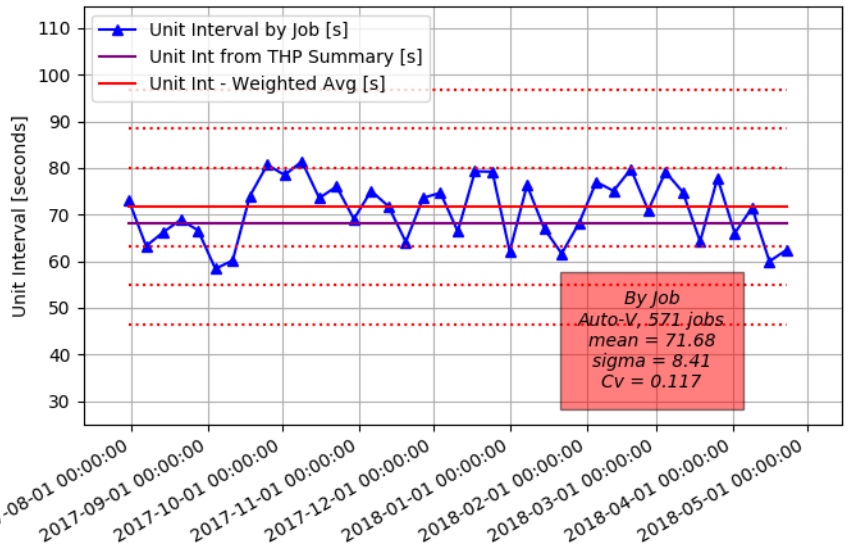
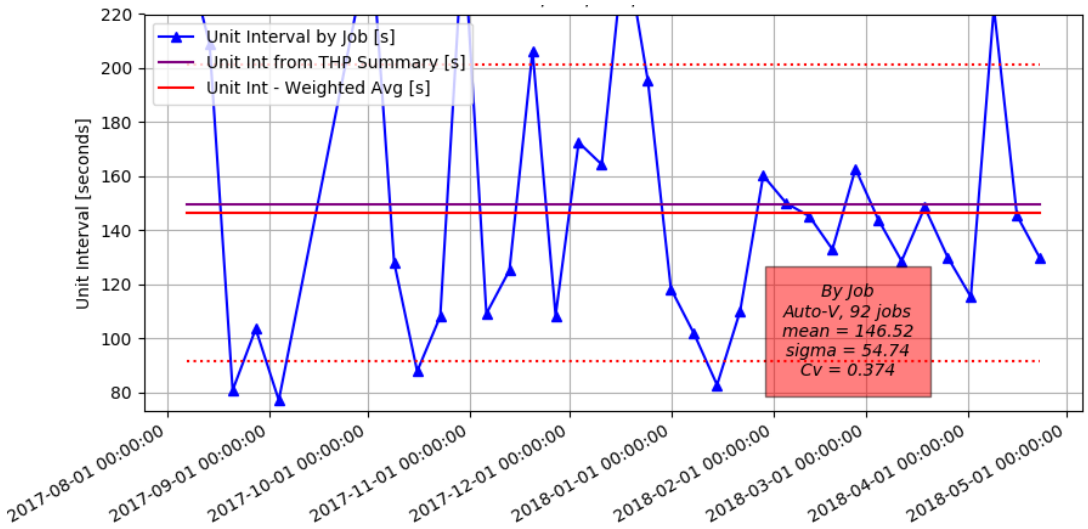


Figure 7 Etch tool, unacceptably variable



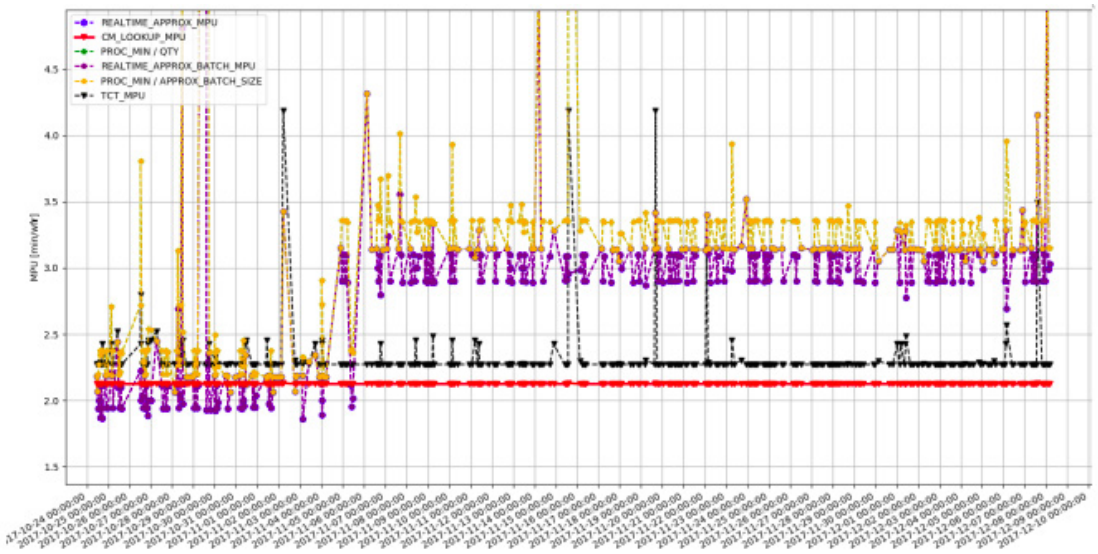
4.5 Investigating Poor Statistical Quality

With a database that stores throughput values and summary statistics, it is possible to create a report that ranks all of the tools or equipment types in the fab by the statistical quality of their throughput. Such a report provides a starting point to investigate the most important tools with the worst quality data. To do this, it is necessary to analyze the raw event data that is being used to automatically calculate throughput. Many issues can be identified simply by looking at a historical trend chart of the job time data for a given process.

In this section, we will review a selection of examples of real issues that can easily be seen by these trend charts.

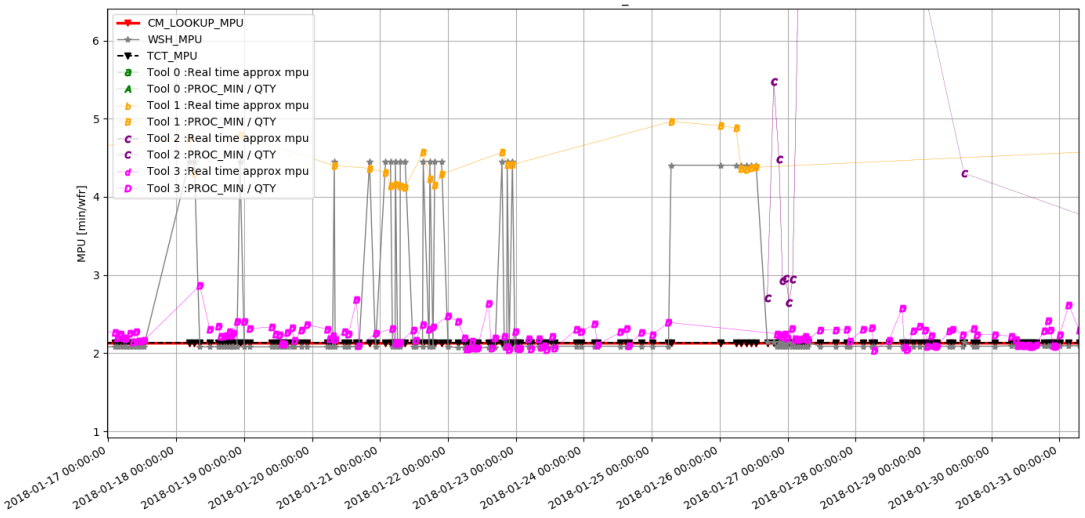
In Figure 8, the orange and purple data points represent two variations of an instantaneous MPU calculation based on a single job. The details of these calculations are not important for this discussion. The red and black lines show a lookup value, retrieved from an automatically calculated throughput table based on a four week average, for the same job. The important observation illustrated in this data is that there is a distinct shift in the job times due to a real process shift. The time for this process increased sharply from approximately 2.25 MPU to approximately 3.1 MPU. When this shift occurred in the window over which the throughput values were calculated, it led to a high coefficient of variation for the process that week.

Figure 8 Process shift



In Figure 9, a group of identical tools (equipment type) with a high C_V for many processes was examined. It was found that the timing of the tools was highly mismatched for several processes. In Figure 9 it is seen that Tool 1, Tool 2 and Tool3 have different mean processing times.

Figure 9 Mismatch tools



In general, with a throughput database indexed by tool, it is possible to run rigorous tool matching analyses. The timing of identical tools can be compared using an independent samples t-test and statistically significantly mismatched tools can be identified automatically. This approach can help detect early equipment failures, performance drifts, and even yield issues.

Figure 10 is an example of a metrology tool that samples a different quantity of wafers each time it runs a lot. The number of wafers sampled comes from an external system and was not incorporated into the recipe information, meaning that all job data for this tool is attributed to the same process. The result is noisy data, with highly varying job times depending on the actual number of wafers measured. As can be seen, the automatically calculated throughput data for this tool would be very accurate if the averages were split up by the number of wafers sampled.

Figure 10 Missing recipe information

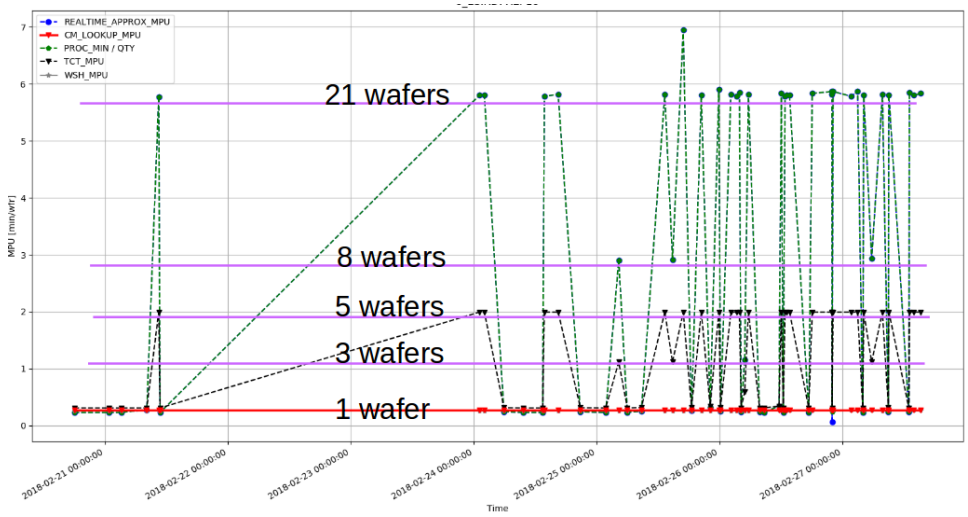
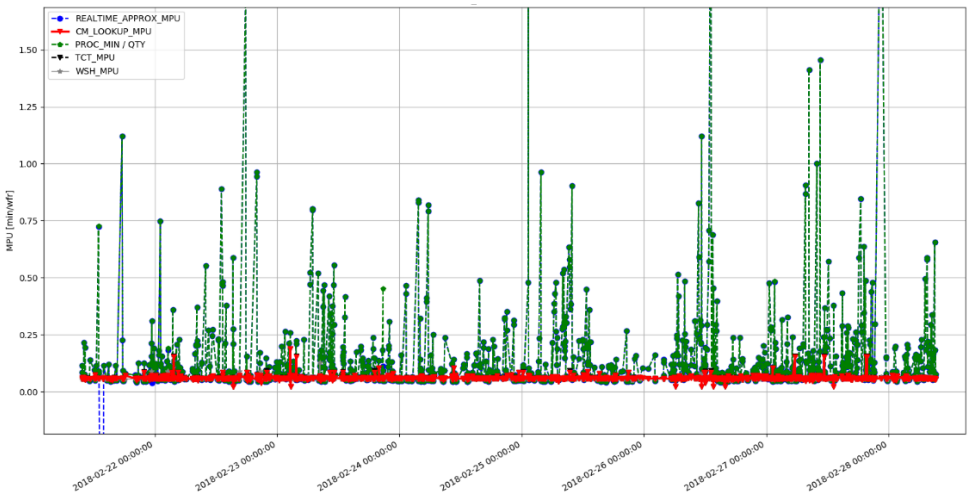


Figure 11 shows an example of a tool where the process start and process end events are recorded as the result of manual action by an operator. There is large, unsystematic variability in the data, and several values hover near zero. Without additional data, a toolset like this is not amenable to high-confidence automatic throughput calculation.

Figure 11 Inherent noise



4.6 Mitigating Poor Quality

In some cases, it can be found that a toolset has statistically unsatisfactory throughput data, the cause is understood, and no reasonable action exists to address the cause. In these cases, it is useful to provide an ability to override throughput values in the lookup function. As an example, in the INFICON FPS Data Warehouse, looked up throughput can be overridden either by incorporating a manually maintained table, or by importing values from an external throughput system.

5. Cloud Computing and Throughput Calculation: A Speculative Look at the Future

We will now take a break from describing the minutiae of throughput quality review for a moment to look at the bigger picture of where these types of calculations may go in the future. The semiconductor manufacturing industry, where data security and integrity is of paramount importance, has been slow to adopt and has even been skeptical of cloud-based solutions for storing core factory data. At the same time, there is a growing industry awareness that cloud computing can enable powerful and rapid analysis of enormous amounts of data at a potentially cheaper cost than a traditional data warehouse approach.

If a factory were to put all of its data, structured and unstructured (MES data, FDC, SPC, tool logs, production management databases, etc.) into a data lake in the cloud, this would have exciting implications for more sophisticated automatic throughput analysis. Rather than being limited to the power of a single data warehouse server (with potentially high database license costs per core), throughput calculations could be performed cheaply and more flexibly by the cloud. It would be easy to experiment with different averaging windows, different off-the-shelf analysis techniques, and even AI-based approaches to calculate and assess the quality of throughput data. There would be no need to store summary statistics, as all of the raw history data could be preserved and statistics recalculated over any window needed.

While the cloud may be the future home of such analysis, it is important to highlight that all of the work of developing the logic to handle the various operational and processing modes of tools (ensuring that equipment types are configured correctly, validating that events are correct, etc.), is still required in order to do any analysis correctly. Thus, starting with an appropriate model and database-based approach is still useful and necessary preparation, even if a cloud-based approach is ultimately adopted.

6. Conclusion

In this paper, we have reviewed the challenges of automatically calculating a throughput data set, explained a method to do it, and introduced checks that can be used to ensure maximum usefulness of the results. To recap the most important points:

- Automatically calculating throughput values is easier than manual calculation and curation, but a substantial amount of work is required to ensure meaningfulness.
- There are many details and intricacies to handle when developing calculation logic. Proper tool modeling is essential.

- Assessing the statistical quality of the results is a great start to investigate the overall quality of the data. Many useful insights, including on subjects like tool matching and inherent equipment timing variation, can be learned in this process.
- With continuous SPC monitoring of throughput values, it is possible to detect process shifts, tool degradation, and even potential yield issues early on.
- Not all tools are amenable to automatic calculation with good statistical quality. The key is to be able to identify these tools, and to override their values in the throughput retrieval system.
- While the hardware and techniques to store and calculate throughput values may change in the future, most of the logical and practical challenges will not.
- Fabs must invest in the hardware and software required to store tool events (via the GEM interface) in a factory-wide data system (such as the MES) to enable automatic throughput analysis, but this investment is usually small and inexpensive relative to the vast gains in smart manufacturing capability enabled by this data.

A robust and trusted throughput calculation, storage, and retrieval system enables powerful next generation manufacturing capabilities. Automated and optimized factory scheduling, for example, requires such a system. The need for comprehensive throughput data is increasingly migrating from pure planning to execution and operations applications in the fab. By installing or developing such a system, factory personnel will not only pave the way for advanced manufacturing applications like scheduling, but also learn valuable and actionable information about equipment that ultimately enables greater operation excellence.

