ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

A customizable open-source simulator for semiconductor fab scheduling research

Advanced Semiconductor Manufacturing Conference 2022

Benjamin Kovács[1], Pierre Tassel[1], Ramsha Ali[1],
Mohammed El-Kholany[1], Martin Gebser[1], Georg Seidel[2]

This work was partially funded by KWF project 28472, cms electronics GmbH, FunderMax GmbH, Hirsch Armbänder GmbH, incubed IT GmbH, Infineon Technologies Austria AG, Isovolta AG, Kostwein Holding GmbH, *and* Privatstiftung Kärntner Sparkasse.

Introduction

- Cooperation with industry

University of Klagenfurt

Infineon Technologies Austria GmbH

- Infineon Technologies Austria GmbH
- Collaborative project, goals:
  - Explore, analyze, develop high-potential methods to improve factory throughput, resource utilization and reduce tardiness

Motivation

Current state-of-art

- Large-scale instances simulated with commercial software
- Several research papers use toy problems simulated in problem-specific self-developed environments

Problem

- Difficult to measure scientific progress, evaluate & compare methods

Our simulator

- Scalable: supports toy to large-scale instances
- Multiple interfaces: easy to integrate and evaluate against concurrent methods
  - Reinforcement learning (gym) interface
  - Priority-based dispatching rules
  - General API
- Open source: no licensing or confidentiality issues

Dataset selection

Requirements

OPEN SOURCE

REAL-WORLD SCALE

DOCUMENTED, IMPLEMENTATION VERIFIABLE

Selected dataset: SMT2020

- Large-scale datasets aiming to model real-world fabs
- 4 problem instances
  - High volume – low mix
  - Low volume – high mix
  - Also with development lots

Our tool

EVENT-BASED SIMULATOR

DEVELOPED IN PYTHON

HIGH-PERFORMANCE

EXTENDABLE

OPEN SOURCE

Target application
End-to-end development of novel methods

Build, train, evaluate prototypes on small instances

1 →

Investigate scalability of algorithms

2 →

Evaluate best candidates on large-scale instances

3 →

Explore transfer learning opportunities

All in one single tool

Performance

- Simulation SMT2020 datasets 1 and 2 for a two-year period
  - 2021 Notebook CPU: Python 3.9 Interpreter (+ experiment tracking): about 30 minutes
  - 2021 Notebook CPU: PyPy3 Interpreter: about 18 minutes
- 2 year-period
  - 40 000 lots completed
  - About 10 million working steps simulated, dispatching decisions executed

Experimental results

Compared to the dataset's reference results

- Similar throughput
- Higher timeliness for hot lots
- Lower timeliness for normal lots
- Slightly lower utilization, same availability

Reasons for slight differences

- Parametrization of dispatching strategies
- Undocumented features -> different implementation

TABLE II: Lot metrics for LV/HM dataset

Lot Type	Average cycle time	Throughput	Percentage on-time
Hot Lot 1	29 (30)	106 (100)	98 (73)
Hot Lot 2	31 (24)	105 (100)	99 (76)
Hot Lot 3	33 (35)	105 (100)	99 (76)
Hot Lot 4	20 (20)	105 (102)	97 (81)
Hot Lot 5	14 (14)	105 (103)	96 (83)
Hot Lot 6	17 (18)	105 (102)	98 (86)
Hot Lot 7	21 (21)	105 (102)	96 (79)
Hot Lot 8	22 (23)	105 (102)	97 (79)
Hot Lot 9	23 (24)	106 (102)	98 (77)
Hot Lot 10	24 (24)	106 (101)	98 (77)
Lot 1	53 (47)	4003 (3800)	30 (98)
Lot 2	58 (50)	3963 (3795)	30 (98)
Lot 3	62 (53)	3967 (3765)	30 (98)
Lot 4	36 (30)	4003 (3900)	32 (98)
Lot 5	26 (22)	4023 (3944)	30 (97)
Lot 6	34 (29)	3992 (3904)	30 (69)
Lot 7	37 (32)	4013 (3882)	32 (100)
Lot 8	43 (37)	3991 (3866)	29 (98)
Lot 9	42 (39)	4006 (3844)	31 (68)
Lot 10	43 (38)	4001 (3861)	32 (98)
Super Hot Lot	33 (N/A)	38 (N/A)	98 (N/A)

TABLE III: Tool metrics for LV/HM dataset

Class	Availability	Utilization	Prev. maint.	Breakdown	Setups	Wait
DE	88.87 (88.54)	83.23 (85.10)	8.88 (9.21)	2.25 (2.25)	0.09 (0.27)	0.20
DefMEt	96.66 (96.50)	44.93 (45.48)	3.00 (3.14)	0.34 (0.36)	0.00 (0.00)	0.02
Dielectric	80.82 (80.37)	77.28 (80.71)	13.54 (14.02)	5.64 (5.61)	0.00 (0.00)	0.13
Diffusion	92.98 (92.50)	71.82 (86.15)	5.53 (6.05)	1.48 (1.46)	0.00 (0.00)	0.13
EPI	80.52 (80.02)	29.49 (32.35)	13.63 (14.02)	5.86 (5.96)	0.00 (0.00)	0.10
Implant	80.76 (80.27)	52.77 (62.95)	13.62 (14.01)	5.62 (5.72)	7.84 (8.06)	0.07
LithoMet	96.75 (96.49)	89.30 (90.84)	2.91 (3.16)	0.35 (0.35)	0.00 (0.00)	0.05
LithoTrack	86.82 (86.46)	80.31 (86.82)	6.64 (5.26)	6.54 (4.88)	3.82 (2.94)	0.14
Litho	89.07 (88.62)	83.15 (84.71)	5.80 (6.17)	5.13 (5.22)	0.00 (0.00)	0.12
Planar	80.53 (80.03)	58.24 (69.69)	17.49 (18.04)	1.98 (1.94)	0.00 (0.00)	0.10
TF	87.27 (86.95)	68.15 (73.97)	9.03 (9.39)	3.70 (3.66)	0.00 (0.00)	0.06
WE	89.36 (89.07)	72.56 (77.27)	8.50 (8.80)	2.14 (2.13)	0.00 (0.00)	0.04

Integration & extensions

General interface

Python-interface for integration with arbitrary methods.

Usage

1. Create simulator instance with desired parameters & dataset
2. Set decision point
    1. Machine available
    2. Lot available
    3. Time-based
3. Get available lot, machines and their properties
4. Dispatch lot(s) on machine(s)

Gym interface for RL development

- Easy-to-use interface with declarative environment definition
  - Select action type
  - Build observation space
    - from a list of features
    - implement own features with plugins
  - Result: gym environment
  - Train / evaluate existing agents on the environment

```
DEMO_ENV_1 = {
    'action': E.A.CHOOSE_LOT_FOR_FREE_MACHINE,
    'state_components': (
        E.A.L4M.S.MACHINE.SETUP_PROCESSING_RATIO,
        E.A.L4M.S.MACHINE.IDLE_RATIO,
        E.A.L4M.S.MACHINE.MAINTENANCE.NEXT,
        E.A.L4M.S.OPERATION_TYPE.NO_LOTS_PER_BATCH,
        E.A.L4M.S.OPERATION_TYPE.CR.MAX,
        E.A.L4M.S.OPERATION_TYPE.FREE_SINCE.MAX,
        E.A.L4M.S.OPERATION_TYPE.SETUP.MIN_RUNS_OK,
        E.A.L4M.S.OPERATION_TYPE.SETUP.NEEDED,
        E.A.L4M.S.OPERATION_TYPE.SETUP.LAST_SETUP_TIME,
    )
}
```

```
args = dict(seed=0, num_actions=config['action_count'], active_station_group=config['station_group'],
            days=365 * 2, dataset='SMT2020_' + config['dataset'],
            dispatcher=config['dispatcher'], reward_type=config['reward'])

env = DynamicSCFabSimulationEnvironment(**DEMO_ENV_1, **args, max_steps=1000000000)
```

Plugins

- Custom functionality can be implemented using plugins
  - New cost function
  - Experiment tracking
  - Monitoring agent behavior
  - Data collection
  - Modifying simulator state

Usage

1. Implement (methods of) *IPlugin* interface in a custom class
   a. on_lot_done
   b. on_sim_init
   c. on_sim_done
2. Install plugin when constructing simulator instance

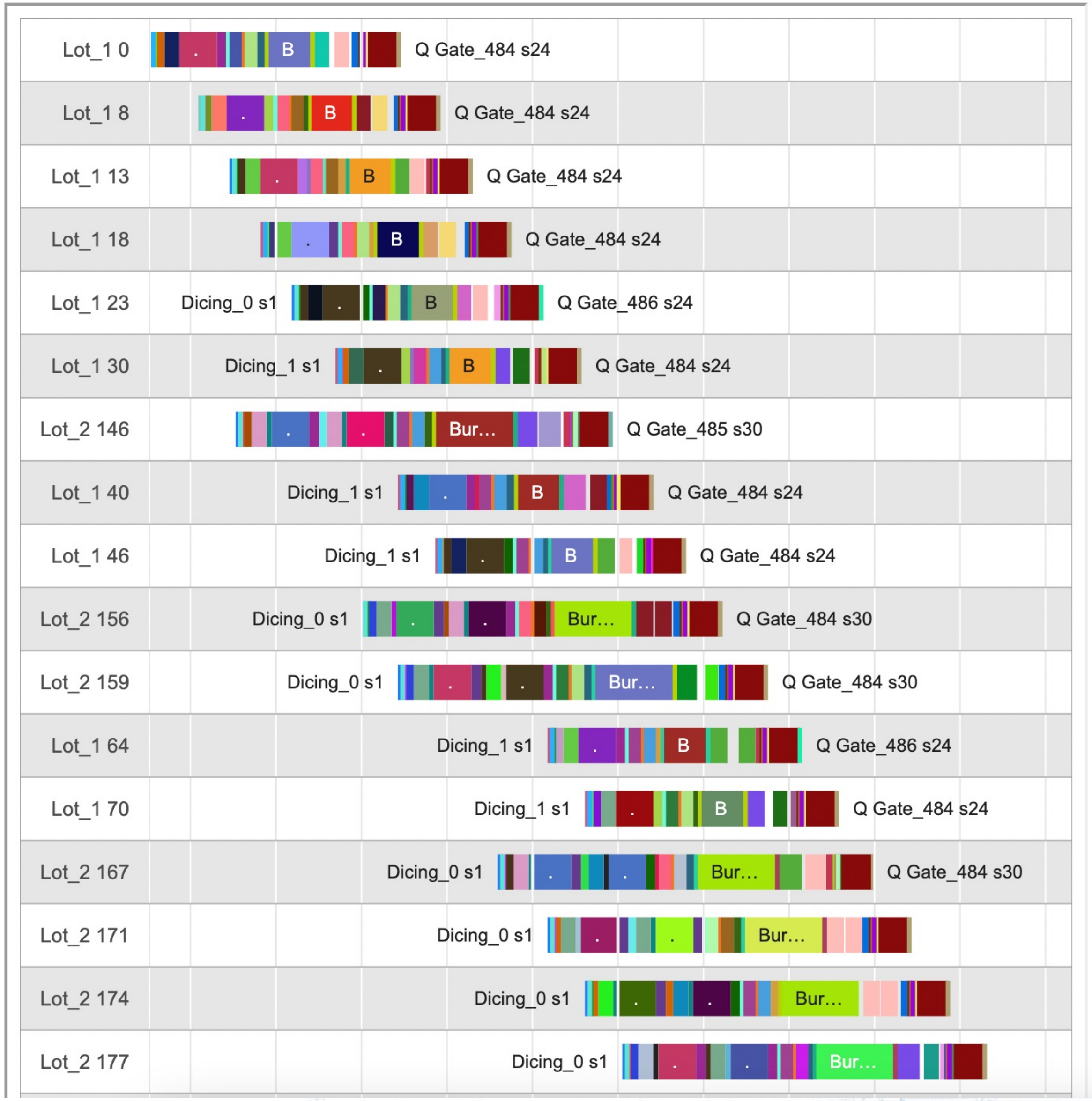
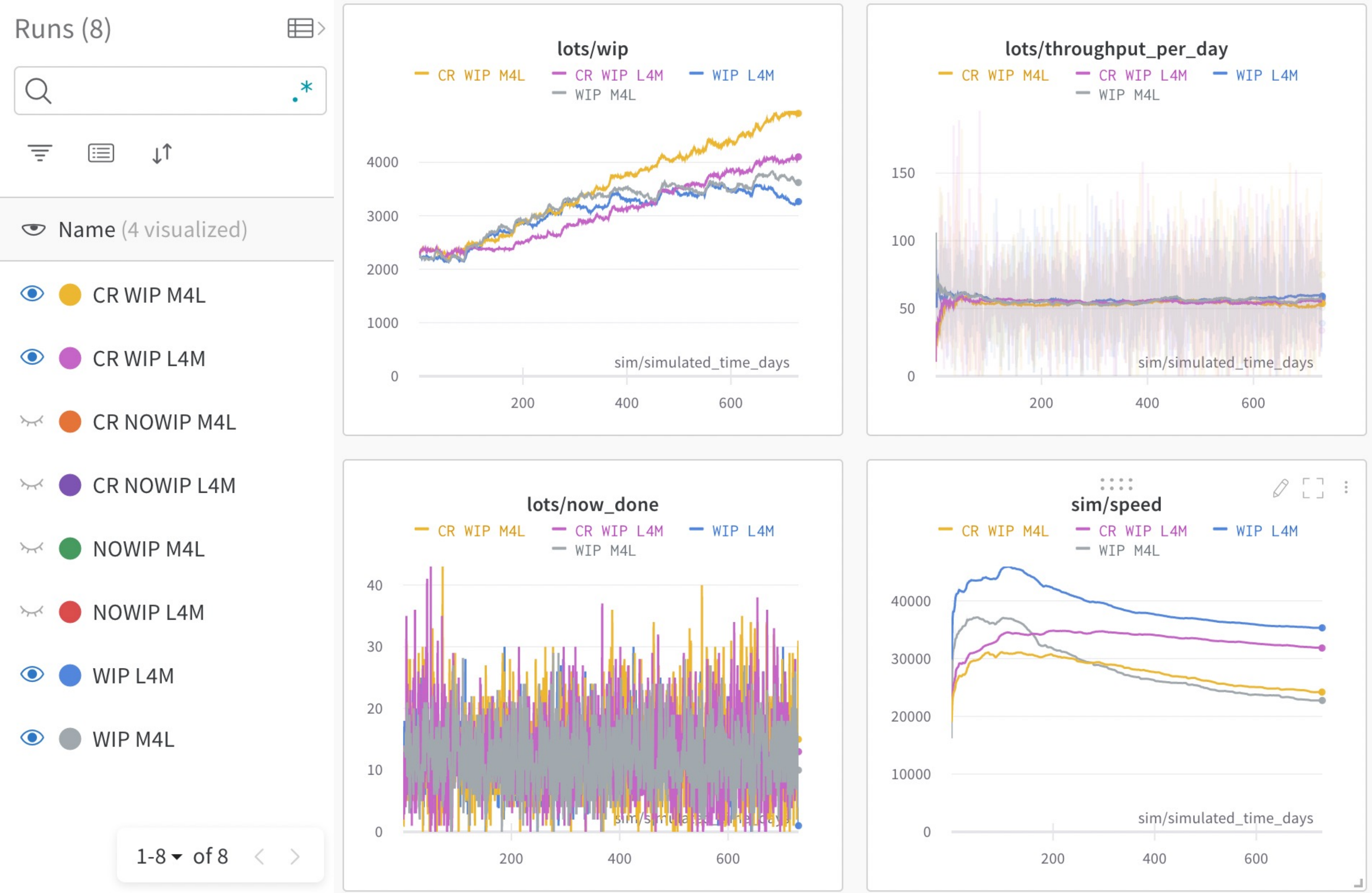


Chart plugin

- Visualize schedules for small-scale instances
- Help understanding the behavior of newly developed agents

Weights & Biases Plugin

- Track & analyze large-scale experiments
- Metrics
  - Completed lots
  - On-time lots
  - Batch utilization
  - Machine utilization
  - Speed



Conclusion

- New tool to support development, evaluation and comparison of semiconductor fab scheduling methods
- Open source, extensible
- Available on GitHub

ALPEN-ADRIA
UNIVERSITÄT
KLAGENFURT

Thank you for your attention.