Increasing Throughput for Robotic Cells With Parallel Machines and Multiple Robots

H. Neil Geismar, Chelliah Sriskandarajah, and Natarajan Ramanan

***Abstract*—We address the problem of scheduling robots' moves in a robotic cell that is used by a Dallas-area semiconductor equipment manufacturer. The cell has parallel machines, multiple robots, and Euclidean travel times. We describe a plan of operation that allows the robots to operate concurrently, efficiently, and with no risk of colliding. We propose a set of sequences of robot moves, analytically determine this scheme's throughput, and determine problem instances for which it is optimal. Through simulation, we demonstrate that our scheme is superior to the heuristic dispatching rule currently in use by the manufacturer.**

***Note to Practitioners*—Efficient scheduling of a robotic cell can greatly increase productivity and revenue for manufacturers in many different industries. This increase becomes more pronounced for larger cells that employ multiple robots and parallel machines at various production stages. This paper describes a schedule of robotic actions that is optimal under a common set of conditions for such large cells, in addition to many other types of cells. When this set of conditions does not hold, even though optimality could not be proven, this schedule is shown to be superior to one currently in use by some semiconductor manufacturers. We also present a scheme that allows the robots to operate concurrently, efficiently, and with no risk of colliding. Additionally, an approximation to the improvement in revenues realized by using this schedule is provided.**

***Index Terms*—Least common multiple (LCM) cycles, manufacturing, multiple robots, parallel machines, robotic cells.**

I. Introduction

Robotic cells are used in many different industries, including the manufacture of semiconductors, glass products, cosmetics, fiber-optics, printed circuit boards, and building products. Their efficient use increases the rate of production and thus provides a competitive advantage.

Previous studies have addressed cells with one robot. We address the problem of scheduling a robotic cell with parallel machines and three robots working concurrently. FSI International, Inc., a Dallas-area robotic cell manufacturer, uses a heuristic dispatching method to schedule its three-robot cell that is used in semiconductor wafer fabrication. We propose a predefined schedule, determine its throughput analytically, and compare the two schedules by using software simulation and simulated data.

A robotic cell is, in essence, a flow shop with a fixed number of servers. The cell consists of m processing stages, each with one or more machines, an input device, an output device, and robots that perform all material-handling functions. Parts flow through the cell, visiting each stage in a fixed order.

There have been many studies on the type of cell that we discuss, though most of them have been for cells with a single robot. Sethi *et al.* [9] performed the first purely analytic study that determined optimal cycles of robot moves for cells with either two or three machines. Crama and van de Klundert [2] studied cells with additive travel-times, and Dawande *et al.* [4] studied cells with constant travel times; each found optimal cycles for cells with any number of machines. Geismar *et al.* [6] found an optimal cycle for a cell with parallel machines in each processing stage and constant travel times for a case that is very common in practice. This paper extends [6] and applies it to practice by considering an industrial problem with parallel machines, three robots, and Euclidean travel times. A version of this study with more background on robotic cell scheduling research can be found in [5].

Manuscript received October 28, 2003.

H. N. Geismar and C. Sriskandarajah are with the University of Texas at Dallas, Richardson TX 75083-0688 USA (e-mail: ngeismar@yahoo.com; chelliah@ utdallas.edu).

N. Ramanan is with FSI International, Inc., Allen, TX 75013 USA.

Digital Object Identifier 10.1109/TASE.2004.829430

II. Physical Description of Robotic Cells

Let $M = \{1, 2, \ldots, m\}$ be the set of indexes for the m processing stages of a robotic cell. Each stage $i \in M$ has $m_i \geq 1$ identical parallel machines. The machines in stage i are denoted $M_{ia}, M_{ib}, \ldots, M_{i,\alpha(m_i)}$, where the function $\alpha(j)$ assigns to any positive integer $j \leq 26$ the jth letter of the alphabet, e.g., $\alpha(3) = c$. The input device is denoted by M_0 and the output device by M_{m+1}. Each part is picked up by a robot at the input device, then carried for successive processing to a machine in stage 1, a machine in stage $2, \ldots$, a machine in stage m, and finally stored at the output device M_{m+1}. Each transfer is performed by one of r robots. After loading a machine, the robot may wait at that machine for the entirety of its processing, or it may travel to another machine (or the input device) to unload another part. The robots are denoted $R_j, j = 1, \ldots, r$. Each robot can hold only one part at a time. Each machine $M_{i\gamma}, i \in M, \gamma = a, \ldots, \alpha(m_i)$, can hold only one part at a time. There are no buffers for intermediate storage of parts between the machines. This is equivalent to a blocking condition in a classical flowshop: a part that has completed processing on $M_{i\gamma}$ cannot leave $M_{i\gamma}$ unless some machine in stage $i+1$ is unoccupied, $i = 0, \ldots, m-1; \gamma = a, \ldots, \alpha(m_i)$ [8].

FSI International's cell currently under production has 18 stages, some of which have parallel machines. The machines are housed in modules that resemble racks used for computer hardware. Each module contains the machines of one stage. The machines of a particular stage are stacked atop one another [see Fig. 1(a)]. This simplifies programming a robot's movements, because the horizontal (x, y) coordinates of the machines of a particular stage are all the same. Only the vertical (z) coordinate changes for different machines of the same stage. Some modules contain only one machine $(m_i = 1)$, but the one-stage-per-module design allows machines to be added to stages with minimal disruption to the layout or to the programming of the robots. These modules are arranged in a horseshoe configuration [see Fig. 1(b)].

This cell also contains three robots (R_1, R_2, R_3) that transfer the silicon wafers between the various stages. Each robot is assigned specific stages whose machines it loads or unloads. These assignments are determined by the cell's configuration: its long, narrow shape limits the modules (hence the stages) that each robot can reach. Therefore, the robots' assignments are fixed input parameters to this study. The set of stages whose machines robot R_j unloads is denoted $\mathcal{R}_j$. There are also specifically designated shared stages whose machines perform a processing function just like any other stage, but each shared stage is loaded by one robot and unloaded by another. The set of indexes for shared stages is denoted by Q.

A wafer begins processing when robot R_1 unloads it from the input device and loads it onto some machine in stage 1. After stage 1 processing, R_1 unloads the wafer and loads it onto a machine in stage 2. Robot R_2 unloads it (stage 2 is a shared stage) and successively loads it onto machines in stages 3, 4, and 5. Robot R_3 unloads the wafer (stage 5 is shared) and successively loads it onto stages 6, 7, 8, 9, 10, and 11. R_2 unloads stage 11 (stage 11 is shared) and transfers the wafer through stages 12, 13, and 14, before loading it onto some machine of the shared stage 15. Robot R_1 unloads stage 15, carries the wafer through stages 16, 17, and 18, then places the completed wafer into the output device. Hence, for this cell, $\mathcal{R}_1 = \{0, 1, 15, 16, 17, 18\}, \mathcal{R}_2 = \{2, 3, 4, 11, 12, 13, 14\}, \mathcal{R}_3 = \{5, 6, 7, 8, 9, 10\}$, and $Q = \{2, 5, 11, 15\}$.