2002 IC/CAD Contest

Problem 7 : Chip Placement for Multiple Project Wafer

Source: Global UniChip Corp., Taiwan

December 31, 2001

1. Introduction

A wafer that contains different chip designs for multiple projects is called a Multiple Project Wafer (MPW). The purpose of an MPW is to let customers share the expensive cost of a common mask tooling set for an engineering-run and obtain their samples quickly for fast prototyping.

A brief procedure of an MPW from chip tape-out to sample back is presented as follows.

(1). Customers submit chip layouts to an MPW vendor.

(2). An MPW vendor arranges the chip layouts from different projects on a reticle. A reticle is a block that contains several chip layouts, and is used for making a mask-tooling set. Fig. 1 shows an example of reticles.

(3). A mask house makes a set of mask tooling according to the reticle arranged by an MPW vendor.

(4). A manufacture foundry fabricates wafers by exposing the masks repeatedly.

(5). A die saw house cuts the fabricated wafers to obtain bare dice. A bare die is a chip that is cut from a wafer but has not been packaged. Fig. 2 shows an example of a bare die.

(6). An MPW vendor ships the obtained bare dice to its customers.

This procedure differs mainly from that of a normal engineering-run in steps 2 and 5. In step 2, an MPW vendor must make more effort to arrange multiple chip layouts from different projects in a reticle. However, for a normal engineering-run, only one chip layout for a specific project is duplicated in a reticle. In step 5, it is much easier to perform die saw for an engineering-run wafer than an MPW. Due to complex chip placement in a reticle, an MPW must consider many restrictions during die saw. These two steps will be discussed later in details.

Figure 1 demonstrates an example of an MPW. Suppose that there are 6 projects, from CHIP_A to CHIP_F, are submitted in step 1 to be fabricated in an MPW. To achieve this goal, we must properly arrange the chip layouts of those projects on a reticle in step 2 and then use the reticle for making a mask-tooling set in step 3. The sizes of the generated masks and the reticle are identical. As the masks are ready, a manufacture foundry starts fabrication in step 4 by exposing the masks repeatedly on a wafer like that shown in Fig. 1. A wafer is only exposed under a set of mask-tooling (a reticle); however, a set of mask tooling (a reticle) can be reused for the exposure of multiple wafers. Basically, the more

compact the reticle is, the more bare dice for each project will be obtained. The chip layout cannot overlap with each other on the reticle and must leave enough cut line space for later die saw. Die saw is an important process to obtain bare dice, which will be discussed later. The dimension of a reticle cannot exceed a certain size, for example, 20mm x 20mm.

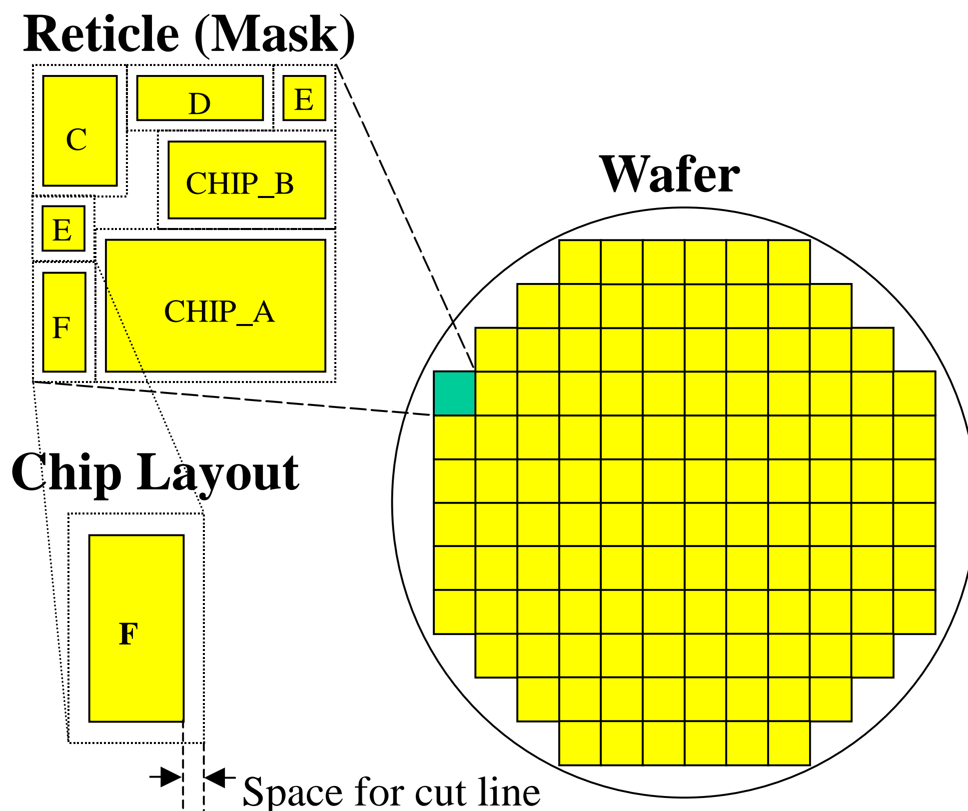

Fig. 1. A Multi-Project Wafer

In Fig. 1, there are 120 duplications of the reticles in the wafer. In other words, there are 120 chip layouts for CHIP_A, CHIP_B, CHIP_C, CHIP_D, and CHIP_F, and 240 chip layouts for CHIP_E in the wafer. However, it is impossible to obtain the above number of bare dice without loss of a project in a wafer, because some dice, e.g., CHIP_C and CHIP_E, will be destroyed when die saw is performed to obtain one die, e.g., CHIP_A.

Figure 2 shows a part of the wafer shown in Fig. 1. In Fig. 2, we can apply cut lines *1* & *2* and cut lines *b* & *c* to get a bare die “CHIP_A.1”. And we can apply cut lines *2* & *3* and cut lines *e* & *f* to get a bare die “CHIP_B.2”. Please notice that CHIP_A.2 will be *destroyed* and CHIP_B.1 will be *discarded* while we apply the above cut lines to obtain CHIP_A.1 and CHIP_B.2. The way to cut the wafer is called die saw. A cut line must start and end at wafer edge. That means we must cut the whole wafer along the specified cut lines and cannot start or stop any cut line inside the wafer. Because the wafer will not be totally broken after applying cut lines, the final bare dice are only dependent on the positions of planned cut lines, but are independent of the sequence of applying cut lines.

For the example shown in Fig. 2, we finally obtain one die of CHIP_A (CHIP_A.1), two dice of CHIP_B (CHIP_B.2 and CHIP_B.4), and two dice of CHIP_F (CHIP_F.3 and CHIP_F.4) from the partial wafer.

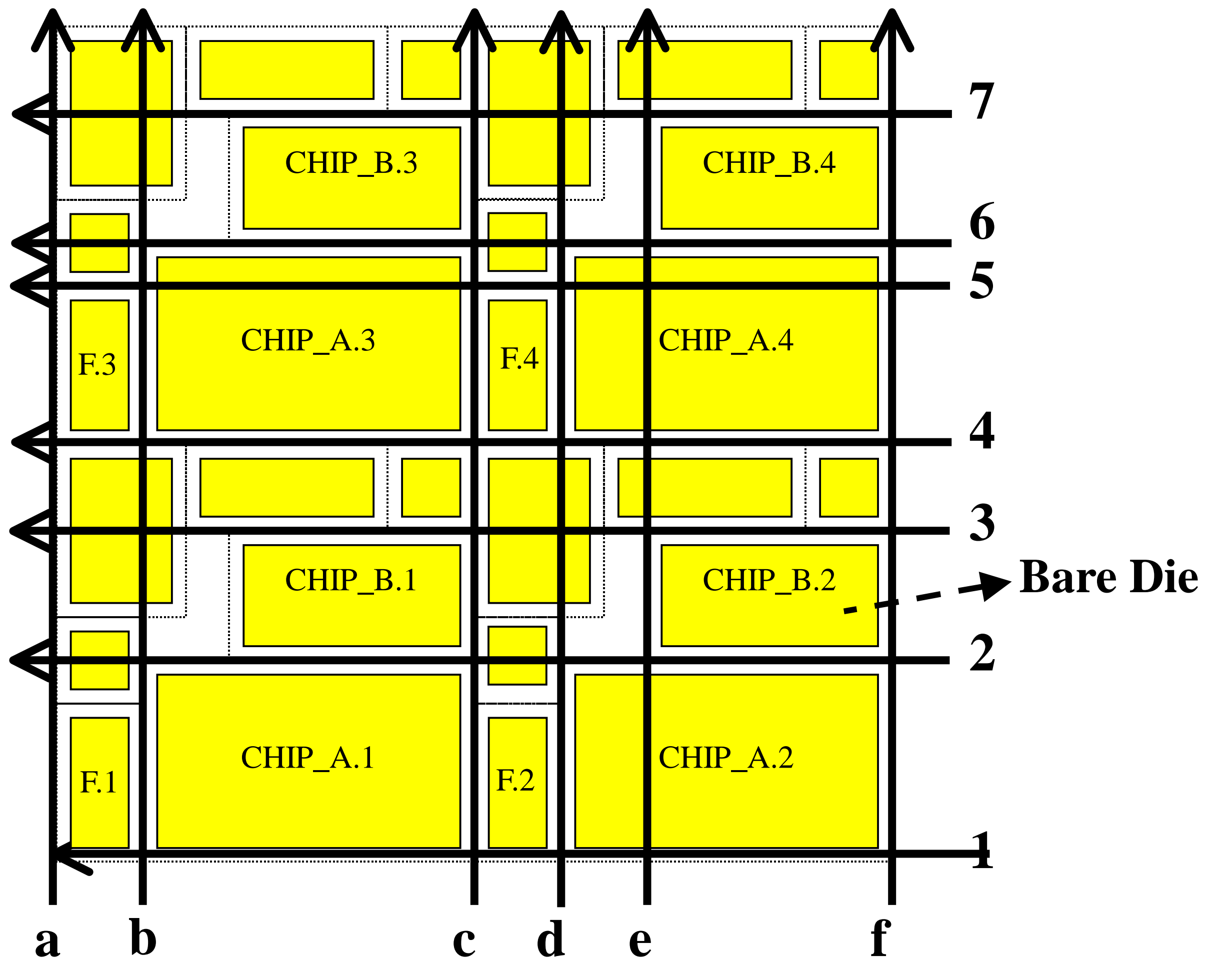

Fig. 2. An example of Die Saw

2. Problem Description

Given (1) a wafer size, (2) the maximum dimension of a reticle, (3) the dimension of chip layout of each project, and (4) the requested number of bare dice for each project, the developed software must first satisfy the above criteria, and then minimize the total cost of masks and wafers, and maximize the number of bare dice. We assume that the cost of a set of mask and a wafer are 100 and 1, respectively. For example, if we need two reticles (two sets of masks) to fill in all projects and six wafers to satisfy the requested number of bare dice for each project, the total cost is 206. A chip layout can be rotated when it is being placed on a reticle.

3. Input

(1). Configuration file (mpw.cfg)

The configuration file contains the following information.

WAFER_SIZE *wafer_size*
RETICLE_SIZE *width height*
NO_BARE_DICE *project_ID #_of_bare_dice*
…

wafer_size: the diameter of a wafer (unit: mm)
width: the maximum width of a reticle (unit: mm)
height: the maximum height of a reticle (unit: mm)
project_ID: a unique name of a project
#_of_bare_dice: the requested number of bare dice

Example 1: An example of **mpw.cfg**

WAFER_SIZE *200*
RETICLE_SIZE *20 20*
NO_BARE_DICE *CHIP_A 120*
NO_BARE_DICE *CHIP_B 40*
NO_BARE_DICE *CHIP_C 80*
NO_BARE_DICE *CHIP_D 40*
NO_BARE_DICE *CHIP_E 120*
NO_BARE_DICE *CHIP_F 200*

(2). Chip size data file (chip_size.dat)

This file contains the dimension of each chip layout. To simplify the problem, the specified chip sizes have taken cut line space into account. The file format is defined as follows.

NO_OF_PROJECT *#_of_project*
project_ID1 width1 height1
project_ID2 width2 height2
. . . . .

#_of_project: the number of projects
project_ID: a unique name of a project
width: the width of a chip layout including cut line space (unit: mm)
height: the height of a chip layout including cut line space (unit: mm)

Example 2: An example of **chip_size.dat**

```
NO_OF_PROJECT   6
CHIP_A  9.140    5.150
CHIP_B  3.410    6.125
CHIP_C  4.098    2.734
CHIP_D  5.826    1.820
CHIP_E  2.560    2.560
CHIP_F  1.980    4.462
```

4. Output

Our problem allows the use of multiple reticles. There are three output files associated with each reticle. The names of the three output files are in the form of **placement_*reticle_id*.dat, diesaw_*reticle_id*.dat, and baredie_*reticle_id*.dat, respectively,** where *reticle_id* in an output file name denotes the identification number of a reticle. For example, if three different reticles are used, one should hand in the following files:

a). placement_1.dat, diesaw_1.dat, baredie_1.dat

b). placement_2.dat, diesaw_2.dat, baredie_2.dat

c). placement_3.dat, diesaw_3.dat, baredie_3.dat

(1). Chip placement of a reticle (placement_*reticle_id*.dat):

This file contains information about the coordinate and rotation of each chip layout in the reticle. The position of each chip layout is identified by its coordinate at the lower left corner. The lower left corner of the reticle is the reference point (0, 0). The format of this file is defined as follows.

PROJECT	**X-COOR**	**Y-COOR**	**ROTATION**
project_ID1	*x-coordinate1*	*y-coordinate1*	*rotation1*
project_ID2	*x-coordinate2*	*y-coordinate2*	*rotation2*
…….			

project_ID: a unique name of a project. It is allowed to duplicate the chip layout of a project in a reticle.

x-coordinate: the x-axis coordinate (unit: mm)

y-coordinate: the y-axis coordinate (unit: mm)

rotation: “N” or “R”. If a chip layout is rotated 90 degrees as CHIP_N shown in Fig. 3 (b), it is denoted by “R”, otherwise it is denoted by “N”.

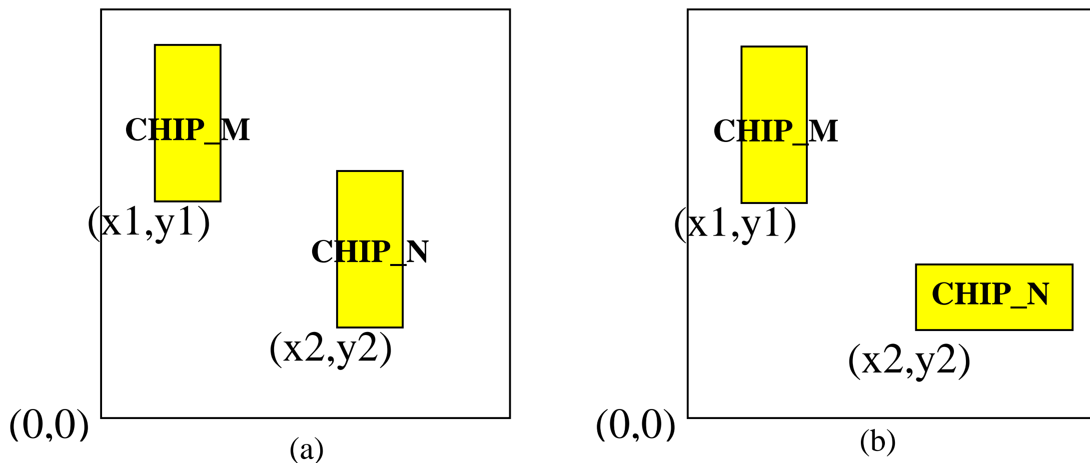

Fig. 3. (a) Chip placement without rotation, (b) CHIP_N with 90-degree rotation

Example 3: An example of placement_1.dat (associated with the first reticle)

```
PROJECT   X-COOR      Y-COOR      ROTATION
CHIP_A    1.980       0.0         N
CHIP_B    4.995       5.15        R
CHIP_C    0.0         7.022       R
CHIP_D    2.408       8.56        N
CHIP_E    0.0         4.462       N
CHIP_E    8.56        8.56        N
CHIP_F    0.0         0.0         N
```

(2). Die saw data of each wafer (diesaw_*reticle_id*.dat):

This file contains the coordinates and directions of cut lines for die saw. The format is defined as follows.

WAFER *wafer_id1*
HORIZONTAL_LINE
y_coordinate1
y_coordinate2
. . . . .
VERTICAL_LINE
x_coordinate1
. . . . .
WAFER *wafer_id2*
HORIZONTAL_LINE
y_coordinate1
. . . . .
VERTICAL_LINE
x_coordinate1
. . . . .

wafer_id: a unique ID of a used wafer. The ID is a sequence number and starts from 1.
y_coordinate: the y axis coordinate of a horizontal cut line.
x_coordinate: the x axis coordinate of a vertical cut line.
The reference point (0,0) is at the center of the wafer. The unit of coordinate is in *mm*.

The *wafer_id* of a wafer must be unique across all **diesaw_*reticle_id*.dat** files**,** i.e., all *wafer_ids* in diesaw_1.dat, diesaw_2.dat, …, diesaw_n.dat are different from each other. The same rule is also applied to **baredie_*reticle_id*.dat**. For the example shown in the beginning of this section, the *wafer_id* of a wafer in diesaw_1.dat (baredie_1.dat) must be unique and different from those in diesaw_2.dat (baredie_2.dat) and diesaw_3.dat (baredie_3.dat).

Example 4: An example of **diesaw_1.dat** (associated with the first reticle)

```
WAFER 1
HORIZONTAL_LINE
-100.25
-97.415
. . . . .
0.0
98.682
VERTICAL_LINE
-100.0
-95.234
. . . . .
100.128
WAFER 2
HORIZONTAL_LINE
. . . . .
VERTICAL_LINE
. . . . .
```

(3). The number of bare dice (baredie_*reticle_id*.dat):

This file contains the obtained quantity of bare dice for each project in each wafer. The format is defined as follows.

```
WAFER wafer_id1
project_ID      #_of_ bare_dice
project_ID      #_of_ bare_dice
. . . . .
WAFER wafer_id2
project_ID      #_of_ bare_dice
……
```

wafer_id: a unique ID of a used wafer. The ID is a sequence number and starts from 1.
project_ID: a unique name of a project
#_of_bare_dice: the number of obtained bare dice

Example 5: An example of **baredie_1.dat** (associated the first reticle)

```
WAFER 1
CHIP_A  80
CHIP_B  20
CHIP_C  60
CHIP_D  30
CHIP_E  40
CHIP_F   120
WAFER 2
CHIP_A  60
CHIP_B  40
CHIP_C  75
CHIP_D  40
CHIP_E  80
CHIP_F   100
```

5. Language/Platform

- Language: C or C++
- Platform: SUN OS/Solaris

6. Evaluation

- Cost of mask and wafer
- The number of obtained bare dice
- CPU time
- Memory usage

7. Questions

Please report any question regarding to this problem to cad@cs.nthu.edu.tw with the email subject "CAD Contest: Problem 7." Your question(s) will be answered in two weeks, and the Q&A's will be posted at the contest Web site.