Recommendation Tool for 2D/3D Integration

1st Joseph Miller
joseph.miller@ucla.edu

2nd Sander Zuckerman
sazuckerman@g.ucla.edu

***Abstract*—We developed a tool to floorplan a chip in three different ways: entirely on one chip (system on chip) integrated 3D face-to-face and face-to-back. The tool also calculates values to analyze yield and verifies that 3D integration closes timing. Based off of these results a flow is recommended for the design.**

I. Introduction

In this class, 3D integrated circuitry (3DIC) was discussed as a potential means of evolution from the current standard of 2D "system on chip" (SoC) design. To compare these design philosophies, we developed a tool to compare a given system if it were designed as a single SoC, or as a 3DIC using either face-to-face (F2F) or face-to-back (F2B) implementation. The tool is capable of floorplanning the system, generating a yield model, and performing a timing analysis on all inter-die signals for the 3D integrations.

II. Floorplanning

A. System on Chip

We designed our floorplanning tool using bash and used a combination of regular expressions to gather design and technical information from the lef, netlist, sv, rep and tech files in the project and equations to compute our floorplanning information. The equation which is used to compute how many bumps/tsvs are needed in the design is as follows:

$$num\_bumps = \left\lceil \frac{\frac{P_{design}}{V_{dd}}}{J * A_{bump}} \right\rceil * 2 + iocount \tag{1}$$

The ceiling of the square root of this quantity is then taken as the number of bumps which will go in our square grid. We also compute the core area length as follows:

$$sqrt\_core\_area = \sqrt{\frac{A\_design}{target\_util}} \tag{2}$$

With these values computed we are able to place our bumps and size our floorplan, taking into account margins for power rings/(TSVs in the following flows).

B. Face to Back

The big challenge associated with F2B is the placement of the TSVs. Our methodology here was to create 4 rectangular regions of TSVs with row_num rows and tsv_per_side_base + row_num columns. To get how many rows we need, we increment row_num in the following equation until max_tsv_num > num_tsvs.

$$\begin{aligned} &max\_tsv\_num(row\_num) = \\ &((tsv\_per\_side + row\_num) * 4) * row\_num \end{aligned} \tag{3}$$

We determine num_tsvs by using equation 1 with top die power, Area of TSV and tsvcount. We can then find how many tsvs can fit on a single side by using the following equation:

$$tsv\_per\_side = \left\lfloor \frac{f2b\_core\_area\_side}{tsv\_pitch} \right\rfloor + 1 \tag{4}$$

With all of this information we use a regex to parse the .sv file for the design and testcase to get a list of all TSV names. Using this list and start coordinates for our left, right, top and bottom sections we generate a tcl script that uses for each loops to place all of the TSVs. For the bottom and top sections of the we increment our x position by tsv pitch for the columns and y position by tsv pitch for the rows. The opposite of this is done for the sides.

The only major differences left in this flow are:

1) Bumps to the substrate carry both top and bottom die power, so we use that as the power in equation.
2) There are two dies, so the max of the two synthesized areas is taken for purposes of computing core area.

C. Face to Face

For the F2F flow we now have the TSVs going to the substrate and the bumps going to the top die. This means our power bump calculations use top die power for the bumps and both dies power for the TSVs. We use the same placement script as before for the TSVs in this flow, but source the names through a different regex as we are now interested in die-to-substrate IOs.

The major consideration for this flow is the need to match IOs between the two dies. The chip is placed bump to bump so there needs to be a mirroring about the x or y axis; in this instance we use the x-axis. To place bumps correctly we create a grid with the bottom left bump located distance x from the left side of the die for the bottom die. In the top die we create the same grid, but place it such that the bottom right bump will be located at distance x from the right side of the die. We then assign the signal IOs from left to right per row in the bottom die and right to left per row on the top die to match our signals between the dies.

III. I/O Constrained Design

In the event we do not have enough area to support our bumps in the design, we compute the difference between length needed for Bumps and length needed for TSV, Power Rings, IO and core. This difference is then applied to the top and right margins of the die for both bottom and top to keep them sized the same.

IV. Yield Analysis

Yield analysis consisted mostly of the documentation that was given in the project specifications, namely the lecture by professor Leachman at UC Berkeley and the paper by Xu et al.

The yield DY for SoC was simple, consisting of an equation dependent only on die area A and defect density D_0:

$$DY = e^{-D_0 A} \tag{5}$$

which is a product of taking the probability of die failure as a Poisson distribution. The defect density was given in in the constraints, and the die area is generated by floorplanning.

The yield for a 3D integration was dependent on many more factors, and is broken into three categories: manufacturing yield, stack yield, and assembly yield.

Manufacturing yield Y_{die} is dependent on die area A, defect density D, and a clustering parameter α:

$$Y_{die} = \left(1 + \frac{DA}{\alpha}\right)^{-\alpha} \tag{6}$$

where α is given as 2, and the area and defect densities are found similarly as in SoC. From the manufacturing yield, the stack yield is simply the minimum value between all stacked chips (in this case, only two):

$$Y_{stack} = \min\left[Y_{die_i}\right], 1 \le i \le N \tag{7}$$

The assembly yield is itself a product of bonding, thinning, and TSV yields. Bonding and thinning yields are both taken to be constants, and sample values can be found in literature. TSV yield is dependent on the TSV failure rate and the number of TSVs; sample TSV failure rates can be found in literature, and the number of TSVs is generated by floorplanning.

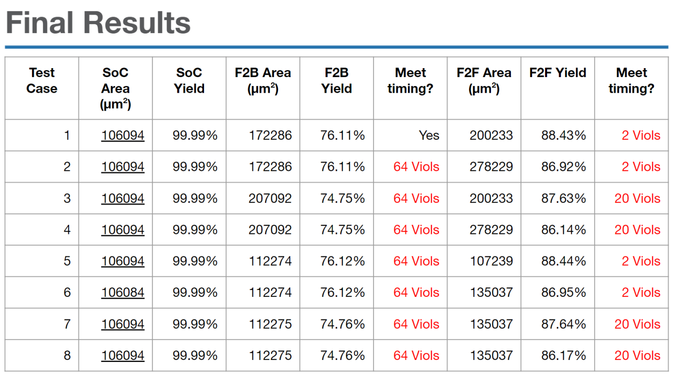
$$\begin{aligned} Y_{assembly} &= Y_{bonding} \times Y_{thinning} \times Y_{TSV} \\ &= Y_{bonding} \times Y_{thinning} \times (1 - f_{TSV})^{\text{\# of TSVs}} \end{aligned} \tag{8}$$

The final yield Y is simply a product of assembly and stack yields:

$$Y = Y_{stack} \times Y_{assembly} \tag{9}$$

V. Timing Analysis

Timing analysis was performed by generating timing reports on all the signals that cross the die boundary from Innovus. A list of all inter-die signals is generated by the floorplanning script through regular expressions and markers are given for input and output signals. We then loop through all inter-die signals for the bottom and top die and run report_timing -to on signals leaving the bottom die and going to top die, and report_timing -from on signals from top die to bottom die.


Final Results

Test Case	SoC Area (μm²)	SoC Yield	F2B Area (μm²)	F2B Yield	Meet timing?	F2F Area (μm²)	F2F Yield	Meet timing?
1	106094	99.99%	172286	76.11%	Yes	200233	88.43%	2 Viols
2	106094	99.99%	172286	76.11%	64 Viols	278229	86.92%	2 Viols
3	106094	99.99%	207092	74.75%	64 Viols	200233	87.63%	20 Viols
4	106094	99.99%	207092	74.75%	64 Viols	278229	86.14%	20 Viols
5	106094	99.99%	112274	76.12%	64 Viols	107239	88.44%	2 Viols
6	106084	99.99%	112274	76.12%	64 Viols	135037	86.95%	2 Viols
7	106094	99.99%	112275	74.76%	64 Viols	135037	87.64%	20 Viols
8	106094	99.99%	112275	74.76%	64 Viols	135037	86.17%	20 Viols

Fig. 1. Our final results tabulated.

For the top die, signals entering are treated as *from*, while signals leaving are treated as *to*. We check the results of these commands for the timing value and, if the prior command failed, we run with -unconstrained. In the event both fail, the signal is marked as unconnected. Arrival time is given by Innovus for timing reports using the unconstrained flag and an A is prepended to the timing value; Slack is given by timing reports using the constrained flag and an S is prepended; in the event of an unconnected signal, an X is prepended.

Determining whether a signal closes timing, then, is simple. By looping through every net on both the top and bottom dies, we compute the arrival time by taking the value if an A is present, doubling the other signal's timing if an X is present and subtracting the signal's time from the clock period if an S is present. We then sum both the top and bottom die's arrival times together. When this is subtracted from the clock period, the total worst-case slack time is found, and if this value is negative, then that net does not pass timing.

VI. Recommendation parameters

Given the constraints of considering solely floorplanning area, manufacturing yield, and timing analysis (for 3D integrations), our tool recommends an SoC implementation every single time, as seen in figure 1.

As the industry currently moves towards 3DIC, it is obvious that our tool does not accurately reflect the capabilities of 3D integrations as compared to the previous (and current) standard of 2D systems on chips. This is, again, a result of the constraints given by the project specifications; were we to consider the usage of multiple SoCs in a larger system compared to multiple 3DICs in a single system, power usage/efficiencies between the two, actual manufacturing costs, computational efficiency, etc, then 3DICs would likely be the recommended implementation in a lot of cases.

References

[1] Leachman, Robert C. "Yield Modeling and Analysis." IEOR 130, Fall 2020, UC Berkeley.

[2] Xu, Qiang, et al. "Yield Enhancement for 3D-Stacked ICS: Recent Advances and Challenges." 17th Asia and South Pacific Design Automation Conference, 2012, https://doi.org/10.1109/aspdac.2012.6165052.