Computing Exact Fourier Series Coefficients of IC Rectilinear Polygons from Low-Resolution Fast Fourier Coefficients

Robin Scheibler[a], Paul Hurley[b*]

[ab]IBM Research, Zurich

ABSTRACT

We present a novel, accurate and fast algorithm to obtain Fourier series coefficients from an IC layer whose description consists of rectilinear polygons on a plane, and how to implement it using off-the-shelf hardware components.

Based on properties of Fourier calculus, we derive a relationship between the Discrete Fourier Transforms of the sampled mask transmission function and its continuous Fourier series coefficients. The relationship leads to a straightforward algorithm for computing the continuous Fourier series coefficients where one samples the mask transmission function, compute its discrete Fourier transform and applies a frequency-dependent multiplicative factor.

The algorithm is guaranteed to yield the exact continuous Fourier series coefficients for any sampling representing the mask function exactly. Computationally, this leads to significant saving by allowing to choose the maximal such pixel size and reducing the fast Fourier transform size by as much, without compromising accuracy.

In addition, the continuous Fourier series is free from aliasing and follows closely the physical model of Fourier optics. We show that in some cases this can make a significant difference, especially in modern very low pitch technology nodes.

Keywords: Fourier transform, optical lithography, aliasing, FFT, DFT, continuous Fourier Series

1. INTRODUCTION

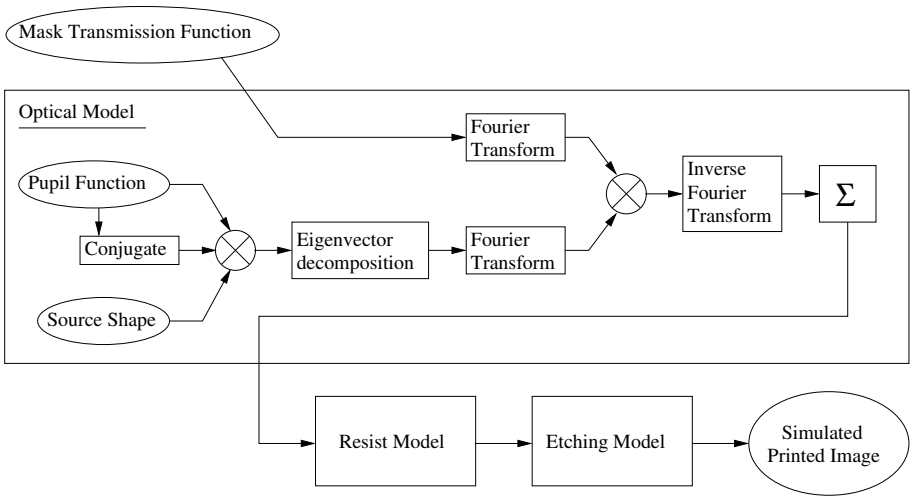

Figure 1. Details of the print simulation of a given IC mask layout.

The calculation of the Fourier representation of IC layers is a crucial part in print simulation of modern optical lithography (as shown in Figure 1), and in mask optimization procedures such as model-based optical

*pah@zurich.ibm.com

proximity correction (OPC) or source-mask optimization (SMO). Other uses of the FFT include the computation of a pre-compensation filter to reduce proximity effects,[1] and approximating the diffraction orders of the mask.[2]

There are presently essentially two approaches to computing the Fourier representation of the mask transmission function.

The first is to compute the exact Fourier series directly from the polygon vertices of the polygons. Done naively, the complexity grows as $O(MN)$ where M is the number of vertices and N the number of frequency points to compute – no longer tractable for modern IC layouts. In addition, the computation usually requires a custom implementation and considerable time and effort to make it computationally efficient.

The second, and most common way, is to sample the mask transmission and compute its discrete Fourier transform (DFT) coefficients. The advantage over the first method is that efficient FFT algorithms can be deployed, for which extremely optimized libraries and hardware are readily available. The main drawback is sampling the mask introduces aliasing, with the coefficients computed no longer corresponding to the original polygons. Circumventing this requires oversampling, thus increasing the size of the DFT to compute.

An alternative is the method as outlined in,[3] which calculates the Fourier series accurately using a fast algorithm, combining direct computation together with FFT methods. That work is also a good source of details previous work in continuous Fourier series for polygons.

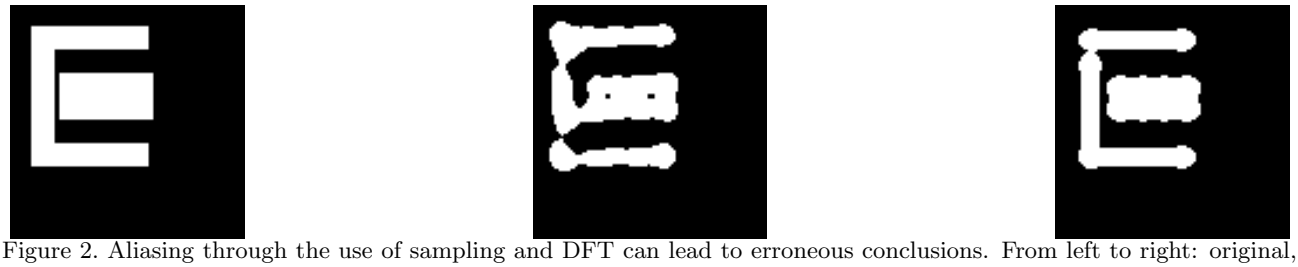

Figure 2. Aliasing through the use of sampling and DFT can lead to erroneous conclusions. From left to right: original, estimated image using DFT, estimated image using exact methods. (light wavelength $193nm$, NA=1.44, pitch $22nm$, simple thresholding used to simulate photoresist, sampling rate for FFT at pitch)

Using the DFT directly as the representation of an image can have unfortunate consequences. In the simple simulation shown in Figure 2, the DFT result would lead one to draw to the wrong conclusion.

In many circumstances, the true Fourier series coefficients, when presented with the FFT coefficients, would be desired. It turns out this is not necessarily a lossy process. The primary contribution of this paper is to (a) show that this is possible under certain circumstances, and, (b) provide the algorithm to do this.

Section 2 provides the context in which the algorithm is performed. Then, in Section 3 we describe how the algorithm works. mathematical justification for the algorithm is provided. Finally Section 4 concludes.

2. SETUP

We consider the transformation over a rectangular subset $\boldsymbol{T} = [0, T_x) \times [0, T_y), \quad T_x, T_y \in \mathbb{N}$, which we call a tile. In the context of semiconductor manufacturing process, this tile may consist of the entire layer or a (small) subsection thereof. In many practical situations, the layer will be divided up into individual tiles.

A rectilinear polygon $\mathcal{P}$ contained in the tile is a subset of the tile whose boundaries are parallel either to the x-axis or the y-axis. A polygons is described by the ordered list of its K vertices:

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{K-1}, y_{K-1})\} \quad , \quad (x_i, y_i) \in \mathbb{Z}^2.$$

We assume the vertices of the polygon are ordered clockwise as illustrated in Fig. 3.

In IC design, the polygons contained in the tile are non-overlapping, and thus the mask transmission function can be defined as the sum of the indicator function of the different polygons $\mathcal{P}_0, \ldots, \mathcal{P}_{M-1} \subseteq \boldsymbol{T}$:

$$f_T(x, y) = \sum_{i=0}^{M-1} \mathbb{1}_{\{(x,y) \in \mathcal{P}_i\}}.$$

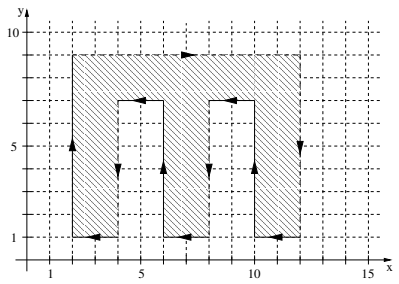

Figure 3. Example of a rectilinear polygon.

2.1 Sampling

The discrete image is done essentially by sampling $f(x, y)$ on a regular grid. We choose the grid in order to minimize the size of the discrete image created and such that every polygon can be represented as an integer number of pixels. Let a pixel be of size $p_x \times p_y$ where p_x is the greatest common divider (GCD) of T_x and the x-coordinates of the polygons' vertices and respectively p_y is the GCD of T_y and the y coordinates of the polygons' vertices. Then, the discrete image is given by:

$$\hat{f}_T[m, n] = f_T(p_x m, p_y n) \qquad \begin{array}{rcl} m & = & 0, \ldots, N_x - 1 \\ n & = & 0, \ldots, N_y - 1 \end{array}$$

where $N_x = \frac{T_x}{p_x}$ and $N_y = \frac{T_y}{p_y}$.

3. ALGORITHM

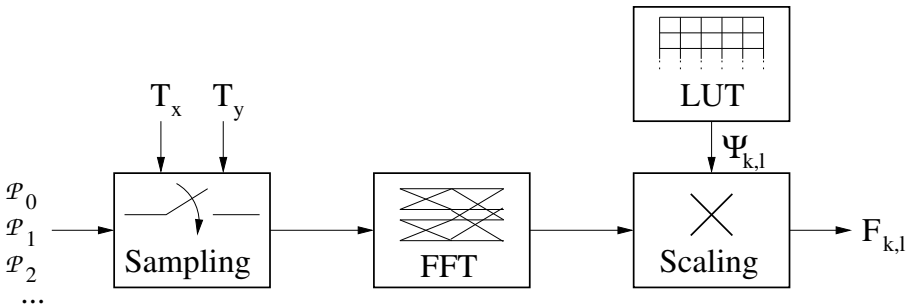

Figure 4. A block diagram of the algorithm. LUT stands for lookup table.

An outline of the three steps of the algorithm, as shown in Fig. 4, is as follows:

1. **Sampling:** Choose p_x and p_y and sample the polygons as described to obtain $\hat{f}_T[m, n]$.

2. **FFT:** Compute the 2D FFT,$\hat{F}_{k,l} = \mathrm{FFT}_{k,l}\left\{\hat{f}_T[m, n]\right\}$.

3. **Scaling:** Obtain the needed Fourier series coefficients $F_{k,l}$ of $f(x, y)$ using $F_{k,l} = \Psi_{k,l}\hat{F}_{k,l}$, where the scaling factor is given by $\Psi_{k,l} = \beta e^{-j\theta}$ and

$$\beta = p_x p_y \,\mathrm{sinc}\left(\frac{k}{N_x}\right)\mathrm{sinc}\left(\frac{l}{N_y}\right) \text{ and } \theta = \pi\left(\frac{k}{N_x} + \frac{l}{N_y}\right).$$

where $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$.

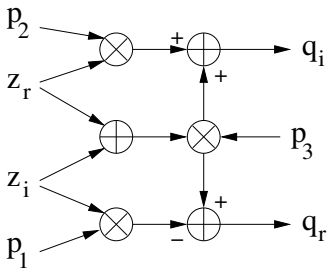

Figure 5. A possible implementation of the scaling step. The indices r and i denote respectively real and imaginary parts. The output $q = \beta e^{-j\theta} z$. It is assumed that p_1, p_2 and p_3 are stored in a lookup table.

The sampling and FFT have standard, well-known hardware implementations. The scaling factor can be written as:

$$\Psi_{k,l} = \beta e^{-j\theta} \ , \ \beta = p_x p_y \operatorname{sinc}\left(\frac{k}{N_x}\right) \operatorname{sinc}\left(\frac{l}{N_y}\right) \text{ and } \theta = \pi\left(\frac{k}{N_x} + \frac{l}{N_y}\right).$$

Thus in practice, the multiplication of a complex number z by $\Psi_{k,l}$ can be implemented using only three additions and three multiplication, as described in.[4]

$$\begin{aligned}
p_1 &= \beta(\cos\theta - \sin\theta) & p_2 &= -\beta(\cos\theta + \sin\theta) & p_3 &= \beta\cos\theta \\
s_1 &= \Re e\{z\} + \Im m\{z\} & & & & \\
m_1 &= s_1 p_3 & m_2 &= p_1 \Im m\{z\} & m_3 &= p_2 \Re e\{z\} \\
\Re e\left\{\beta e^{-j\theta} z\right\} &= m_1 - m_2 & \Im e\left\{\beta e^{-j\theta} z\right\} &= m_1 + m_3 & &
\end{aligned}$$

where the values p_1,p_2 and p_3 are precomputed and stored in a lookup table. In hardware, this can be implemented using simple logic or a dedicated circuit. such as the one from Fig. 5. Alternatively, one can use a CORDIC unit performing multiplication by a scalar and vector rotation in a single step or only vector rotation and use two multiplications in addition to the multiplication by β.[5] In that case, only β and θ need to be stored in the lookup table.

Fig. 6 shows an example for a simple mask transmission function together with a comparison to known solutions.

4. CONCLUSIONS

We demonstrated a new algorithm to compute the continuous Fourier series of mask transmission function in optical lithography. Our approach is based on the standard discrete Fourier transform followed by a frequency-dependent multiplication step.

The proposed algorithm allows significant computational saving in many cases while the use of the continuous Fourier series circumvents the accuracy limitations of the traditionally used discrete Fourier transform.

In addition, the simple structure of the algorithm make it suitable for software or hardware implementation using off-the-shelf highly optimized existing libraries or architectures.

APPENDIX A. MATHEMATICAL DERIVATION

Let $f(x, y)$ be the indicator function of M non-overlapping rectilinear polygons living in a tile $\boldsymbol{T} = [0, T_x) \times [0, T_y)$. The coordinates of the polygons are:

$$\bigcup_{i=0}^{M-1} \{(x_{i,j}, y_{i,j})\}_{j=0}^{K_i - 1}, \qquad x_{i,j} \in \{0, \ldots, T_x\}, \, y_{i,j} \in \{0, \ldots, T_y\},$$

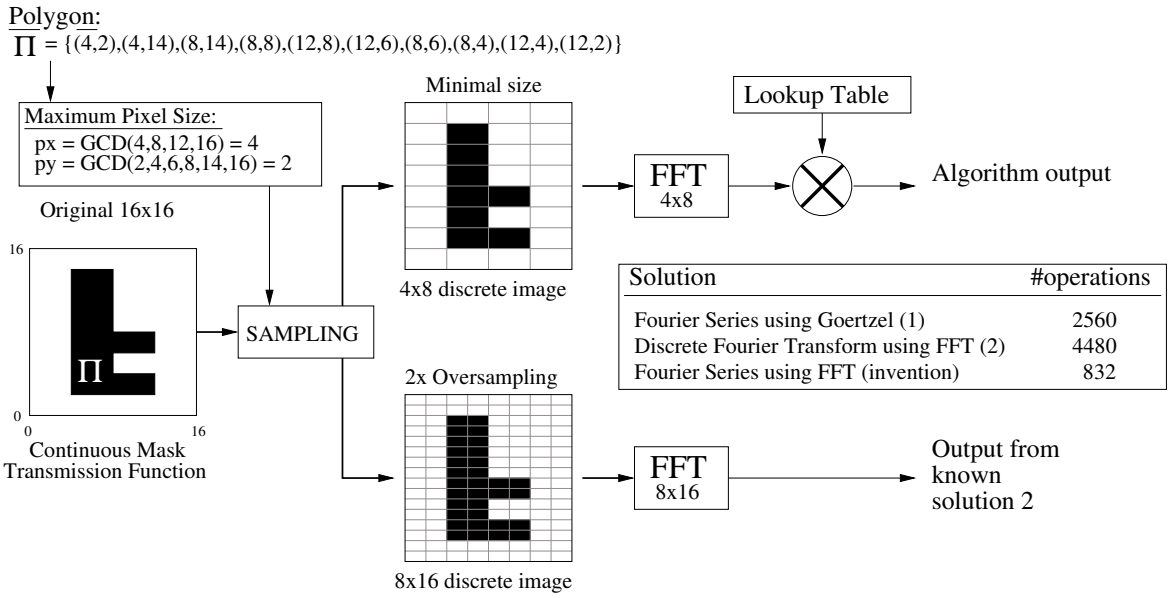

Figure 6. An example of the algorithm along with the known solution. The complexity is given for the example in number of operations.

where K_i is the number of vertices of the i^{th} polygon. We now choose $p_x, p_y \in \mathbb{N}$ such that:

$$\begin{aligned} x_{i,j} = p_x \tilde{x}_{i,j}, \quad \tilde{x}_{i,j} \in \mathbb{N}, \quad \forall i,j \qquad \text{and} \qquad y_{i,j} = p_y \tilde{y}_{i,j}, \quad \tilde{y}_{i,j} \in \mathbb{N}, \quad \forall i,j \\ T_x = p_x N_x, \quad N_x \in \mathbb{N} \qquad \text{and} \qquad T_y = p_y N_y, \quad N_y \in \mathbb{N} \end{aligned} \tag{1}$$

It is easy to see that we can always find such numbers as $p_x = 1$ and $p_y = 1$ always satisfy these relationships. However, in the practical case we are interested in choosing p_x and p_y as large as possible, hence:

$$p_x = \text{GCD}\left(T_x, \bigcup_{i=0}^{M-1} \{x_{i,j}\}_{j=0}^{K_i-1}\right) \text{ and } p_y = \text{GCD}\left(T_y, \bigcup_{i=0}^{M-1} \{y_{i,j}\}_{j=0}^{K_i-1}\right)$$

where $\text{GCD}(a, b, \dots)$ gives the greatest common divider of all the number in argument. Let us now define the pixel function:

$$\psi(x,y) = \begin{cases} 1 & \text{if } 0 \le x < p_x \text{ and } 0 \le y < p_y \\ 0 & \text{otherwise} \end{cases} .$$

Finally it is possible to define the sampled version of $f(x,y)$:

$$\hat{f}[m,n] = f\left(mp_x, np_y\right), \quad \begin{matrix} m & = & 0, \dots, N_x - 1 \\ n & = & 0, \dots, N_y - 1 \end{matrix}$$

If the relations of Eq. (1) are respected, then we have the following relationship between $f(x,y)$ and $\hat{f}[m,n]$:

$$f(x,y) = \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \hat{f}[m,n] \psi\left(x - mp_x, y - np_y\right) \tag{2}$$

In order to compute the Fourier series coefficients of this expression, we need to make f and ψ periodic of period T_x and T_y in the x and y direction respectively. Thus, the 2D Fourier series coefficients can be computed as:

$$\mathcal{F}\{f\} = \int_0^{T_x} \int_0^{T_y} f(x,y) e^{-j(w_x k x + w_y l y)} \, dx \, dy$$

where $w_x = \frac{2\pi}{T_x}$ and $w_y = \frac{2\pi}{T_y}$. We can now plug Eq. (2) into this:

$$\mathcal{F}\{f\} = \int_0^{T_x}\int_0^{T_y} \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} \hat{f}[m,n]\psi\left(x-mp_x, y-np_y\right) e^{-j(w_x kx + w_y ly)}\, dx\, dy$$

We will now use the following relation:

$$e^{-j(w_x kx+w_y ly)} = e^{-j(w_x k(x-mp_x)+w_y l(y-np_y))} e^{-j(w_x kmp_x + w_y lnp_y)} = e^{-j(w_x ku+w_y lv)} e^{-j2\pi\left(\frac{km}{N_x}+\frac{ln}{N_y}\right)} \tag{3}$$

where we used the substitution $u = x - mp_x$ and $v = y - np_y$. Hence it becomes:

$$\begin{aligned}\mathcal{F}\{f\} &= \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} \hat{f}[m,n] e^{-j2\pi\left(\frac{km}{N_x}+\frac{ln}{N_y}\right)} \int_{-mp_x}^{T_x-mp_x}\int_{-np_y}^{T_y-np_y} \psi(u,v) e^{-j(w_x ku+w_y lv)}\, du\, dv \\ &\overset{(a)}{=} \left(\sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} \hat{f}[m,n] e^{-j2\pi\left(\frac{km}{N_x}+\frac{ln}{N_y}\right)}\right)\left(\int_0^{T_x}\int_0^{T_y} \psi(u,v) e^{-j(w_x ku+w_y lv)}\, du\, dv\right) = \mathcal{DFT}\left\{\hat{f}\right\}\mathcal{F}\{\psi\}\end{aligned} \tag{4}$$

where we used in (a) the periodicity of $\psi(x,y)$ to remove the dependency of the integral on m and n.

The only thing left to do now is compute the value of $\Psi_{k,l} = \mathcal{F}_{k,l}\{\psi\}$. But first, note that ψ can be written

$$\psi(x,y) = \mathrm{rect}_{p_x}\left(x - \frac{p_x}{2}\right)\mathrm{rect}_{p_y}\left(y-\frac{p_y}{2}\right) \quad \text{where} \quad \mathrm{rect}_B(t) = \begin{cases} 1 & \text{if } -\frac{B}{2} \le t < \frac{B}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and if we periodize it with period T, the corresponding Fourier series is given by:

$$\mathcal{F}_k\{\mathrm{rect}_B\} = B\,\mathrm{sinc}\left(\frac{k}{T}B\right) = \frac{\sin\left(\pi\frac{k}{T}B\right)}{\pi\frac{k}{T}}$$

If in addition we use the delay property and the separability of the Fourier series, we obtain

$$\Psi_{k,l} = p_x\,\mathrm{sinc}\left(\frac{k}{T_x}p_x\right)e^{-j2\pi\frac{k}{T_x}\frac{p_x}{2}}\, p_y\,\mathrm{sinc}\left(\frac{l}{T_y}p_y\right)e^{-j2\pi\frac{l}{T_y}\frac{p_y}{2}} = p_x\,\mathrm{sinc}\left(\frac{k}{N_x}\right)e^{-j\pi\frac{k}{N_x}}\, p_y\,\mathrm{sinc}\left(\frac{l}{N_y}\right)e^{-j\pi\frac{l}{N_y}}$$

In conclusion we showed that the Fourier series of $f(x,y)$ can be expressed in terms of the DFT of $\hat{f}[m,n]$:

$$F_{k,l} = \Psi_{k,l}\hat{F}_{k,l}$$

where $F_{k,l} = \mathcal{F}_{k,l}\{f\}$ and $\hat{F}_{k,l} = \mathcal{DFT}\left\{\hat{f}\right\} = \sum_{m=0}^{N_x-1}\sum_{n=0}^{N_y-1} \hat{f}[m,n] e^{-j2\pi\left(\frac{km}{N_x}+\frac{ln}{N_y}\right)}$.

REFERENCES

1. D. G. L. Chow, J. F. McDonald, D. C. King, W. Smith, K. Molnar, and A. J. Steckl, "An image processing approach to fast, efficient proximity correction for electron beam lithography," *J. Vac. Sci. Technol., B: Microelectronics and Nanometer Structures* **1**, pp. 1383–1390, Oct. 1983.
2. A. E. Rosenbluth, S. J. Bukofsky, M. S. Hibbs, K. Lai, A. F. Molless, R. N. Singh, A. K. K. Wong, and C. J. Progler, "Optimum mask and source patterns to print a given shape," in *Proc. SPIE*, **4346**, pp. 486–502, Sept. 2001.
3. R. Scheibler, P. Hurley, and A. Chebira, "Fast continuous Haar and Fourier transforms of rectilinear polygons from VLSI layouts," *arXiv:1010.5562v1* , 2010.
4. H. Nussbaumer, *Fast Fourier transform and convolution algorithms*, Springer series in information sciences, Springer-Verlag, 1981.
5. P. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on* **56**, pp. 1893 –1907, Sept. 2009.