半導体入門講座・第 4 回

# ニーズ、仕様、半導体設計

チップは半導体工場に入る前に、コンピューターで全部設計し、何度もチェックします。大きなチップには 100 億個以上のトランジスタがあり、手では描ききれず、チェックもしきれないので、どの段階も設計ソフトに頼り、買ってくる区画もあります。「何をするチップか」から設計データを渡すまで、どの段階も、テープアウトのあとで間違いが見つかるのを防ぐためにあります。

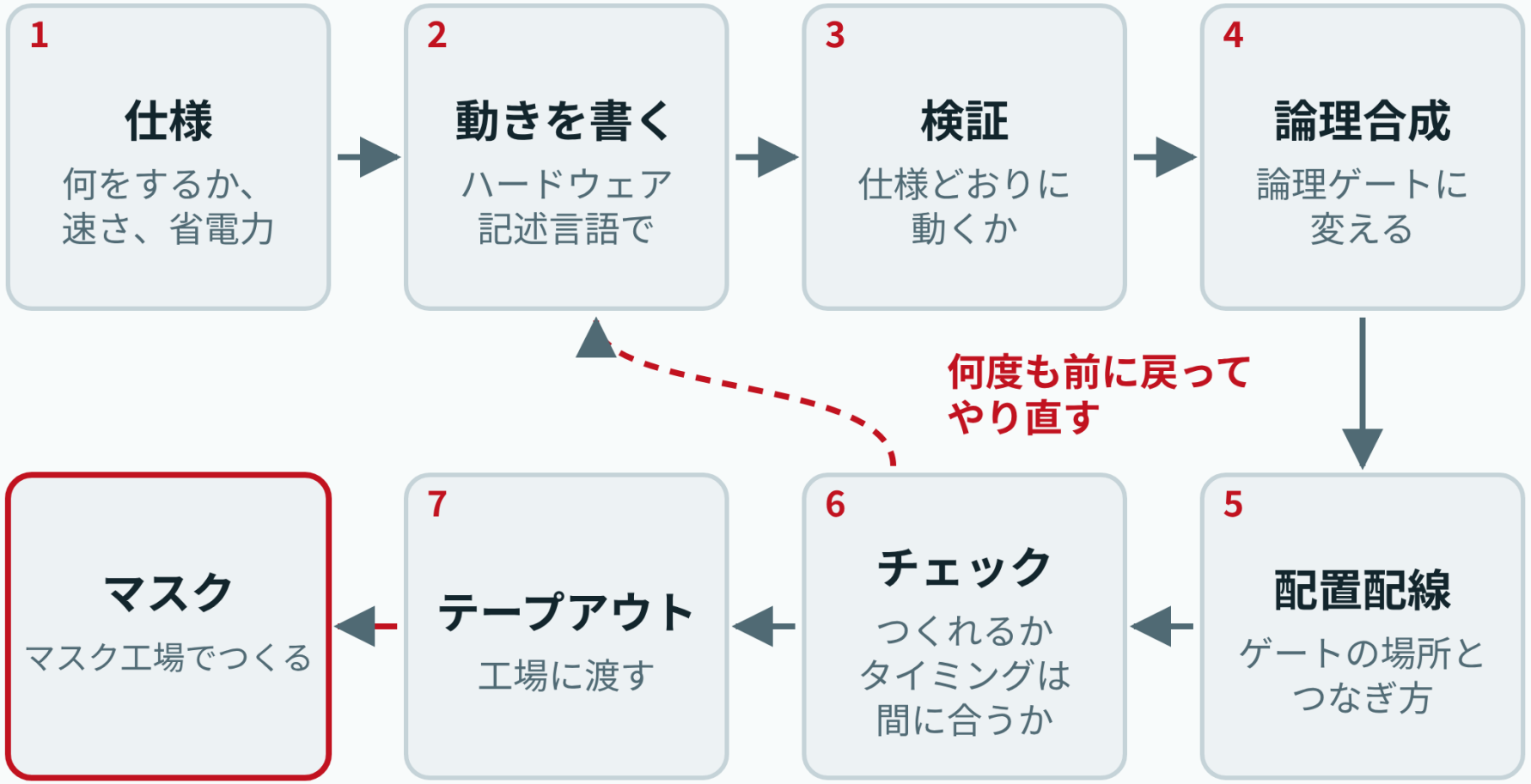

授業や非営利の目的なら無料で使えます。fab24hrと明記してください。販売することや、自分の作品として使うことはできません。

# わかること

1　デジタルのチップ設計が、仕様、動きを書く、検証、論理合成、配置配線、サインオフとテープアウトの順に進むこと、それぞれの段階でどんな設計ソフト（ EDA ）を使うかを言える。

2　検証のしかた（仕様から問題を出す、カバレッジを見る、数学で証明する）と、テープアウトのあとで見つかった間違いがなぜ特に高くつくのかを説明できる。

3　チップの中でよく買ってくる区画である半導体 IP 、ソフト IP とハード IP の違い、工場のルールが PDK を通じてどう設計に入るかを知る。




テープアウトのあとで設計を変えるには、マスクをつくり直すことになります

チップはコンピューターで設計し、何度もチェックしてからテープアウトし、マスクをつくりに出します。テープアウトまでは何度も戻ってやり直し、そのあとで変えるにはマスクのつくり直しになります。

# 一歩ずつ

## 1　仕様を決める。何をするか、どれくらい速く動くか、電力を最大どれだけ使うか、面積はどれくらいか。

なぜ：あとの段階はすべてこれに沿います。ソフトはその目標に合うやり方を探し、検証はそこから問題を出します。

しないと：仕様がなければ、設計が正しいかも、検証が終わったかも言えません。

## 2　工場とプロセスを決めたら、買ってくる IP の区画を選び、そのプロセス世代に合わせた版があるかを確かめる。

なぜ：すべての区画をゼロから描くことはまずなく、 1 つのチップは 1 つのプロセスでしかつくれません。

しないと：合わないものを選ぶと、だれかが合う版をつくるのを待つか、工場を変えることになります。

## 3　自分でつくる区画の動きを、ハードウェア記述言語（ HDL ）で文字にする。

なぜ：「 1 拍ごとにこの 2 つの数を足す」なら 1 行で済むのに、トランジスタで描くとかなりの広さになります。

しないと：書き間違えると、あとのソフトはその間違いのとおりにつくってしまいます。

# 一歩ずつ

**4　検証する。仕様から出した問題でシミュレーションし、カバレッジを見て、買ってきた IP もつないでいっしょに試す。**

なぜ：コンピューターなら 1 行直してすぐ試せますが、本当にチップをつくるには何か月もかかります。

しないと：間違いがテープアウトまで残ると、設計を直してマスクをつくり直すことになり、しかもできたチップの 1 個 1 個に同じ間違いが入ります。

**5　論理合成する。標準セルライブラリを使い、書いた動きを、論理ゲートをつないだ回路に翻訳する。**

なぜ：同じことをするつなぎ方はたくさんあり、ソフトは表を引いて速さ、面積、電力を比べ、目標に合うものを選びます。

しないと：手でつなぐのでは、つなぎきれず、どれがよいかも比べられません。

**6　配置配線をする。論理ゲートを並べ、クロックを届け、何層もの配線層に線を通して、レイアウトをつくる。**

なぜ：並べ方は大事です。線が長いほど信号は遅くなり、電力も多く使います。

しないと：うまく並べないと、線が通りきらなかったり信号が間に合わなかったりして、並べ直しになります。

# 一歩ずつ

**7　サインオフする。ルールを守っているか、レイアウトと回路が同じか、トランジスタが速め・遅めにできたとき（コーナー）でも信号が間に合うかを確かめる。**

なぜ：すべて通ってはじめてテープアウトします。マスクはこのデータからつくられるので、そのあとは気軽に変えられません。

しないと：まちがいをひとつ見逃すと、何か月も先にチップが戻ってきてから気づき、マスクをつくり直すしかなくなります。

# 仕様は要求リスト。ソフトはそれに沿ってやり方を探す

設計は仕様から始まります。このチップは何をするのか、どれくらい速く動くのか、電力を最大どれだけ使うのか、面積はどれくらいか。これらはよく引っ張り合います。速くしたいなら回路も電力も増えがちで、チップが大きいほど 1 枚のウェーハに並ぶ数が減り、 1 個あたりが高くなります。小さな電池 1 個で長くもたせたいものなら、省電力が最優先です。スマートフォンのチップには 100 億個以上のトランジスタがあり、仮に 100 億個として 1 秒に 1 個、寝ず食べずで描いても 300 年以上かかります。だからどの段階も、 EDA（電子設計自動化）と呼ばれる設計ソフト一式に頼ります。ただし、ソフトが自分で設計するわけではありません。エンジニアが目標を伝えます。たとえば 1 拍にかけてよい時間はここまで（チップの回路は、クロックという一定のリズムに合わせて動きます）。ソフトはその制約の中でやり方を探してレポートを返し、通らないところがあれば、人が理由を読み取って直し、もう一度流します。

# 書いた動きから、１層ずつの図形へ

デジタル回路はまず、ハードウェア記述言語（HDL）で回路の動きを文字で書きます。「１拍ごとにこの２つの数を足す」なら１行で済むのに、トランジスタで描くとかなりの広さになるからです。次に論理合成ツールが、標準セルライブラリ（そのプロセス用にあらかじめつくった小さな部品。論理ゲートのほか、０か１を１つ覚えておく小さな回路もあり、部品ごとの速さや消費電力は前もって計算してある）を使って、論理ゲートをつないだ回路に翻訳します。同じことをするつなぎ方はたくさんあり、ソフトは表を引いて、目標に合うものを選びます。続く配置配線で、論理ゲートをチップの上に並べ、クロックをすみずみまで届け、何層もの配線層の中に線を通して、層ごとの図形をつくります。これをレイアウトと呼びます。線が長いほど信号は遅くなり、電力も多く使います。こうしたソフトはどれも、工場が渡すファイル一式、PDK（プロセス・デザイン・キット）に沿って計算します。設計ルール、トランジスタのモデル、各層のデータが入っていて、モデルが不正確なら、ソフトがどれだけ細かく計算しても、実物は合いません。

# 検証：仕様から問題を出し、まだ試していないところを探す

検証は、チップをつくる前に、設計が仕様どおりに動くかを確かめることで、一般には設計の仕事の大きな部分を占めることが多いと言われます。問題は仕様から出します。設計者が書いた回路から出すと、書き間違えたところまで「正しい」と扱われてしまうからです。シミュレーションでは、コンピューターにそのチップのふりをさせ、入力を送り込んで出力を正解とくらべ、コンピューターにランダムな問題もたくさんつくらせます。設計を 1 回直すたびに、それまでの問題をすべて流し直します。答えが全部合っていても、間違っているところをまだ試していないだけかもしれないので、カバレッジで足りない問題を探します。それでも問題は試しきれません。 32 ビットの数を 2 つ足すだけで入力は 2 の 64 乗通りあり、 1 秒に 10 億通り試せても 500 年以上かかります。だから特に大事な小さめの部分は、形式検証で、たとえば「この 2 つの信号が同時にオンになることはない」といったことが必ず成り立つと、数学で証明します。

# 多くの区画は買ってくる：半導体 IP

大きなチップにはトランジスタが 100 億個以上あり、すべての区画をゼロから描くことはまずありません。プロセッサコア、メモリインターフェース、音のような信号を数字に変えるアナログ回路などは、半導体 IP として買ってくることがよくあります。前もって設計して検証しておき、ライセンスを受けたほかの人が自分のチップに組み込める回路の設計で、売っているのは設計図であって、チップではありません。ソフト IP は文字で書いた回路の記述で、別の工場でもつくり直せます。ハード IP はもう配置ずみの図形で、ある工場のある世代のプロセスと結びついています。アナログ回路はたいていハード IP です。トランジスタ 1 つ 1 つの大きさや置き方に敏感で、ほかの人が並べ直すと正しく動くとはかぎらないからです。アナログ回路は小さくしてもあまり得をせず、かえってつくりにくくなることもあるので、多くのアナログのチップは、何世代も前のつくり方でつくられます。買ってきた IP は単独で検証されたもので、つないでみると、クロック、電源、再起動のしかたが両側で同じとはかぎりません。だからチップ全体でもう一度検証します。

# よくある誤解

### よく聞く：チップの設計とは、プログラムを書くことだ。

実は：ハードウェア記述言語は書き方がプログラムに少し似ていますが、プログラムは 1 行ずつ順番に実行されるのに対し、 HDL が書くのは回路で、たくさんの部分が同時に動き、たいていは決まったリズム（クロック）に合わせて動きます。書き終えても終わりではありません。検証、論理合成、配置配線をして、サインオフを通ってはじめて渡せます。

### よく聞く：検証も、工場の計測も、チップのテストも、どれもチェックなので、 1 つやれば十分だ。

実は：調べているものが違います。検証はテープアウトの前に、ほとんどコンピューター上で、設計そのものが正しいかを確かめます。間違っていれば、どのチップも同じように間違います。計測と検査は工場で加工の途中に、膜の厚さや線幅など、重要な工程がうまくいったかを確かめます。チップのテストは、ふつうウェーハができたあととパッケージのあとに 1 回ずつ、全数について、できたその 1 個が壊れていないかを確かめます。家を建てるのにたとえると、着工前に設計図を確かめ、建てている途中で壁がまっすぐかを測り、引き渡し前に 1 軒ずつ水道と電気を試すようなものです。

# 遊ぶ

## 第 1 章「 12 枚をどう分ける」から：設計でモデルを使って計算したコーナーを本当につくり、設計が持ちこたえるかを見る

スマホで読み取って、ここから遊ぶ

fab24hr.com/ja/?story=1&utm_source=slides

# 考える

1　補聴器をつくる小さな会社が、初めて自分たちでチップを設計します。マイクの音を数字に変えるアナログ回路や、チップ全体に拍子を送るクロックの回路は、自分で描くこともできれば、できあいの IP を買うこともできます。どれを自分でつくり、どれを買うか、どう決めますか？ 買う前に、売り手に何を聞きますか？

2　検証には多くの人と時間がかかりますが、減らせばテープアウトのあとで間違いが見つかるおそれがあります。スケジュールが厳しいとき、「チェックはここまでで十分」をどう決めますか？

# 必読

## チップはどうやって設計する？ 半導体設計（ LSI 設計）の流れ

fab24hr.com/ja/blog/chip-design/

## 半導体 IP （ IP コア）とは？ チップに「買ってきた回路」が入っている理由

fab24hr.com/ja/blog/silicon-ip/

## EDA とは？ チップを設計するソフトは、各段階で何をしている？

fab24hr.com/ja/blog/what-is-eda/

## チップの検証とは？ テープアウトの前に、設計が正しいとどう確かめる？

fab24hr.com/ja/blog/chip-verification/

### 関連

SoC とは？ スマートフォンの中の「システム・オン・チップ」をやさしく解説

アナログ半導体とデジタル半導体は何が違う？

VLSI 設計と半導体工場の仕事はどう違う？

# 半導体入門講座：チップの一生をたどる

## https://fab24hr.com/ja/learn/design/

公開されている教科書、業界の標準、大学の講義資料、論文をもとにまとめました。ゲームに出てくる会社、人物、装置の番号、数字は架空のものです。

授業や非営利の目的なら無料で使えます。 fab24hr と明記してください。販売することや、自分の作品として使うことはできません。

公開資料をもとに AI の助けで作った一般的な模式図。誤りがあるかもしれません。特定の会社の装置ではありません