

EDA の見本 — デジタル回路の設計と検証

小型の電動アクチュエーター制御 FPGA 設計書

Lattice iCE40 UP5K / 48 MHz / Verilog RTL ・ 検証 ・ 論理合成 ・ 配置配線

版	1.0(2026年10月20日)
検証	39 項目 合格 / 0 項目 不合格 → 合格
資源	LUT4 1166(5280 の 22 %)、FF 936、BRAM 0、DSP 2
タイミング	目標 48 MHz に対して最大 53.79 MHz(最悪の余裕 +2.24 ns)
ファイル	lp-engineering/fpga/ (README.md に再現の手順)

目次

- 1. 仕様
- 2. ブロック図と構成
- 3. 各モジュールの動作
- 4. タイミングの図
- 5. 検証の項目と結果
- 6. 論理合成 ・ 配置配線の結果
- 7. 設計上の判断と制約
- 8. 再現の手順とツール

1. 仕様

項目	内容
対象の FPGA	Lattice iCE40 UP5K(SG48 パッケージ)。ピン配置は iCEBreaker を例にした
クロック	48 MHz(外付け 12 MHz を PLL で 4 倍)。同期設計・同期リセット
エンコーダー	2 相(A/B)、2 段同期化 + 16 クロックの安定フィルタ、4 通倍、位置 32 ビット(符号付き)、不正遷移の検出
速度の計測	1 ms ごとの位置の差分(カウント/ms、符号付き 16 ビット、飽和)
PWM	中心揃え 20 kHz、2 チャンネル(H ブリッジの A 相・B 相)、デューティ 0~2400 の 2401 段階(11.2 ビット)、周期の頭で二重化
デッドタイム	0~255 クロック(0.0~5.3 μ s、0 は 1 クロックとして働く)。初期値 48 = 1.0 μ s。UART の D で変更
速度の PI 制御	Kp・Ki は Q7.8、積分は Q.8(24 ビット)、出力 $\pm$ 1200(デューティ単位)で飽和、飽和中は積分を止める。乗算は DSP 2 個
UART	115200 bps、8N1(417 クロック/ビット、誤差 -0.08 %)。ASCII のコマンド(下の表)
過電流	oc_n(Low で有効)。同期化せずに出力のフリップフロップへ入れ、1 クロック(20.8 ns)以内に 4 本のゲート信号を OFF。故障をラッチし、C で解除
非常停止	UART の E。E の文字を受けた時点で PWM を止め、目標を 0 にする。起動直後も停止状態

UART のコマンド

1 行の ASCII で、行末は CR か LF。英字の大小はどちらでもよく、空白は読み飛ばします。応答の行末は CR LF です。応答を受け取ってから次のコマンドを送ります(E だけは、いつでも即座に効きます)。

コマンド	意味	応答
S hhhh	目標速度(カウント/ms、2 の補数の 16 進 4 桁)	OK
P	位置を読む(32 ビット)	Phhhhhhhh
V	速度を読む(16 ビット)	Vhhhh
R	状態を読む(下のビット)	Rhh
E	非常停止(行末を待たずに止める)	OK
C	非常停止・過電流の解除(過電流のピンが有効なら断る)	OK / NG
K hhhh / I hhhh	比例ゲイン Kp / 積分ゲイン Ki(Q7.8)	OK
D hh	デッドタイム(クロック数)	OK
Z	位置を 0 にする	OK
それ以外	知らないコマンド・桁数の誤り・16 進でない文字	?

R のビット	意味
0	非常停止中
1	過電流でラッチ中
2	エンコーダーの不正遷移を見た
3	操作量 u が飽和

例: 起動直後は R01 (非常停止中)。C のあとは R20 (運転中)。過電流のあと、ピンが戻っていれば R02、まだ有効なら R12。

初期値: Kp = 0x0C00(12.0)、Ki = 0x0266(約 2.4)、デッドタイム = 48 クロック(1.0 μ s)、目標 = 0。

R のビット	意味
4	過電流のピンがいま有効
5	運転中(= 非常停止でも故障でもない)

2. ブロック図と構成

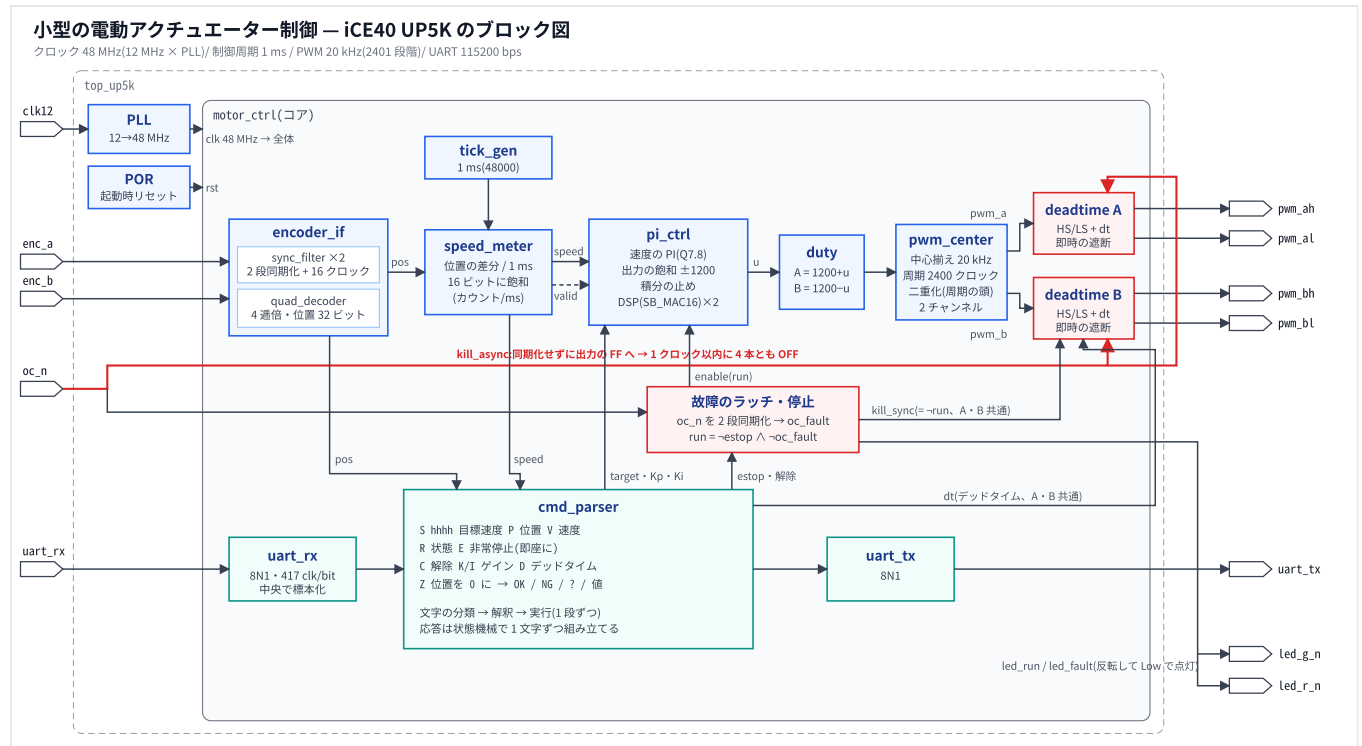


図 2-1 ブロック図。赤い線は過電流の即時の遮断(同期化しない経路)

RTL のファイル

ファイル	役割
top_up5k.v	最上位。PLL(12→48 MHz)、起動時リセット、入力ピンのプルアップ、LED の反転
motor_ctrl.v	コア。各モジュールの接続、u → デューティ、過電流の同期化とラッチ、運転の条件
encoder_if.v	A/B 相の sync_filter × 2 と quad_decoder をまとめたもの
sync_filter.v	2 段同期化 + 安定時間フィルタ(チャタリングの除去)
quad_decoder.v	4 通倍のデコード、32 ビットの位置カウンター(16 + 16 ビットに分割)
tick_gen.v	1 ms の制御周期の合図
speed_meter.v	1 ms ごとの位置の差分 = 速度(3 段のパイプライン)
pi_ctrl.v	速度の PI 制御(9 段のパイプライン、DSP 2 個)
pwm_center.v	中心揃え PWM 2 チャンネル、比較値の二重化
deadtime.v	ハイサイド/ローサイドとデッドタイム、即時の遮断(1 相分、2 個使う)
uart_rx.v / uart_tx.v	UART の受信・送信(8N1)
cmd_parser.v	コマンドの解釈、設定のレジスター、応答の組み立て

ピンの割り当て(constraints/up5k_sg48.pcf)

信号	ピン	説明
clk12	35	12 MHz 発振器
uart_rx	6	FTDI → FPGA
uart_tx	9	FPGA → FTDI

信号	ピン	説明
led_r_n	11	赤 LED(Low で点灯)
led_g_n	37	緑 LED(Low で点灯)
pwm_ah	4	PMOD1A-1 A 相 ハイサイド
pwm_al	2	PMOD1A-2 A 相 ローサイド
pwm_bh	47	PMOD1A-3 B 相 ハイサイド
pwm_bl	45	PMOD1A-4 B 相 ローサイド
enc_a	43	PMOD1B-1 エンコーダー A 相
enc_b	38	PMOD1B-2 エンコーダー B 相
oc_n	34	PMOD1B-3 過電流(Low で有効)

3. 各モジュールの動作

3.1 エンコーダー入力(sync_filter・quad_decoder)

A 相・B 相を 2 段のフリップフロップで同期化したあと、**同じ値が 16 クロック続いたときだけ出力を更新**します。16 クロック(0.33 μ s)未満のひげやバウンドは出力に現れません。数えられるエッジの間隔は 18 クロック以上(約 2.6 M エッジ/秒)です。

デコーダーは前回と今回の {A, B} の 4 ビットで遷移を分類します。正転($AB = 00 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00$)で +1、逆転で -1、A と B が同時に変わる遷移は数えずに不正遷移のパルスを出します(状態の R のビット 2 にラッチ)。

位置カウンタは、32 ビットの加算を 1 クロックで行うと UP5K では桁上げの鎖が長く 48 MHz に届かないため、下位 16 ビットの桁上げ・桁借りを 1 クロック後に上位へ足します。エッジは最短 18 クロックおきなので必ず間に合い、外に出す位置は上位の更新が終わったクロックにだけ写すため、途中の食い違った値は見えません(検証 E7・E8)。

3.2 速度の計測(tick_gen・speed_meter)

48000 クロック(1 ms)ごとの合図で位置を取り込み、前回との差(2 の補数の引き算なので 32 ビットの巻き戻りも正しい)を 16 ビットに飽和して速度とします。単位は 4 通倍後のカウント/ms で、分解能は 1 カウント/ms です。位置を 0 にしたときは前回値も 0 にするので、速度が大きく跳びません(検証 E6)。

3.3 中心揃え PWM(pwm_center)

キャリアのカウンタ cnt は 0~2399(周期 2400 クロック = 20 kHz)を回ります。デューティ d(0~2400)に対して

$$lo = 1200 - \lceil d/2 \rceil, \quad hi = 1200 + \lfloor d/2 \rfloor, \quad \text{出力} = (lo \leq cnt < hi)$$

とし、周期の中心(cnt = 1200)を挟んで幅がちょうど d クロックのパルスを出します。d が奇数のときだけ中心が 0.5 クロックずれます。48 MHz・20 kHz の三角波キャリアでは 1200 段階(10.2 ビット)しか取れませんが、この方法では **2401 段階(11.2 ビット)**になります。d は 2 段のレジスターで比較値の素($\lceil d/2 \rceil \cdot \lfloor d/2 \rfloor$)にし、周期の最後(cnt = 2399)に取り込んで次の周期から使います(二重化)。

H ブリッジは相補のバイポーラ駆動で、A 相 = $1200 + u$ 、B 相 = $1200 - u$ 。モーターの平均電圧は $V_{bus} \cdot u/1200$ です。

3.4 デッドタイムと即時の遮断(deadtime)

理想の PWM が変わったら、OFF にする側は次のクロックですぐ OFF、ON にする側は dt クロック待ってから ON にします。したがって両方 OFF の時間は $\max(dt, 1)$ クロックで、ハイスайдとローサイドが同時に ON になることは構造上ありません。dt より短いパルスは ON になりません。

出力のフリップフロップの D 入力の直前で `kill_async` (過電流のピン、同期化しない)と `kill_sync` (ラッチした故障・非常停止)を見て、どちらかが 1 なら 4 本とも 0 にします。過電流のピンが変わってから**最初のクロックの立ち上がり**で出力が 0 になるので、遅れは 1 クロック(20.8 ns)以内です。同期化していないため、エッジのごく近くで変わると出力のフリップフロップが一瞬不安定になりえますが、その場合も次のエッジで必ず 0 になり、ON 側に倒れ続けることはありません。

3.5 速度の PI 制御(pi_ctrl)

$e = \text{sat16}(\text{target} - \text{speed})$ 、 $P = (K_p \cdot e) \gg 8$ 、 $I_{\text{cand}} = \text{clamp}(I + K_i \cdot e, \pm 1200 \cdot 256)$
 $u = \text{sat}(P + I_{\text{cand}}/256, \pm 1200)$ 。飽和の向きと e の向きが同じなら I を更新せず $u = \text{sat}(P + I/256)$

$K_p \cdot K_i$ は符号なし Q7.8(0~127.996)、積分は Q.8 です。乗算は 16×16 の符号付き 2 回で、iCE40 の DSP(SB_MAC16)に入ります。1 ms に 1 回しか計算しないので、**1 段に演算を 1 つだけ**にした 9 段のパイプラインで 48 MHz に余裕を持たせています (start を取り込んでから 8 クロック後に u)。途中で幅を絞っていますが、溢れるときは結局飽和するので結果は理想の整数演算と同じで、検証 Q1 で参照モデルとビット単位で一致することを確認しています。

段	内容
1	偏差 $e = \text{sat16}(\text{target} - \text{speed})$
2	$K_p \cdot e$ 、 $K_i \cdot e$ (SB_MAC16)
3	P を 16 ビット、 $K_i \cdot e$ を 24 ビットに飽和(結果は変わらない)
4	$I + K_i \cdot e$ 、 $P - 1201$ 、 $P + 1200$
5	和が ± 1201 を越えるか(上位 13 ビットで比較)
6	積分の候補を選ぶ
7	u の候補 $P + I/256$ と上限・下限の判定(加算の符号ビット)
8	積分を止めるか
9	$u \cdot I \cdot \text{done}$ を更新

3.6 UART とコマンド(uart_rx・uart_tx・cmd_parser)

受信は 2 段同期化のあと立ち下がり待ち、ビットの半分でもう一度見てからビットの中央で標本化します。コマンドの解釈は「文字の分類 → 解釈(状態機械) → 実行」と 1 段ずつ分け、文字の種類は受けた時点でワンホットに直しておきます。応答は「1 文字目 → 2 文字目 → 16 進の桁 → CR → LF」の状態機械で 1 文字ずつ組み立てます。

3.7 停止と故障(motor_ctrl)

- 運転の条件 $\text{run} = \neg \text{estop} \wedge \neg \text{oc_fault}$ 。停止中は P の積分と u を 0 にし、再開時に跳ばないようにします。
- oc_n は (1) 同期化せずに deadtime の kill_async へ、(2) 2 段同期化して故障のラッチ oc_fault へ。ピンが戻ってもラッチは残り、 C で消します。
- C は過電流のピンがまだ有効なら NG を返して解除しません。
- 起動直後は $\text{estop} = 1$ (停止)で始まり、 C を受けるまで PWM を出しません。

4. タイミングの図

すべてシミュレーションの VCD から描いた図です(PNG は変化の時刻をそのまま使う。同じ範囲を WaveDrom の JSON・SVG にもしてあり、<results/waves/index.html> で描き直せます)。

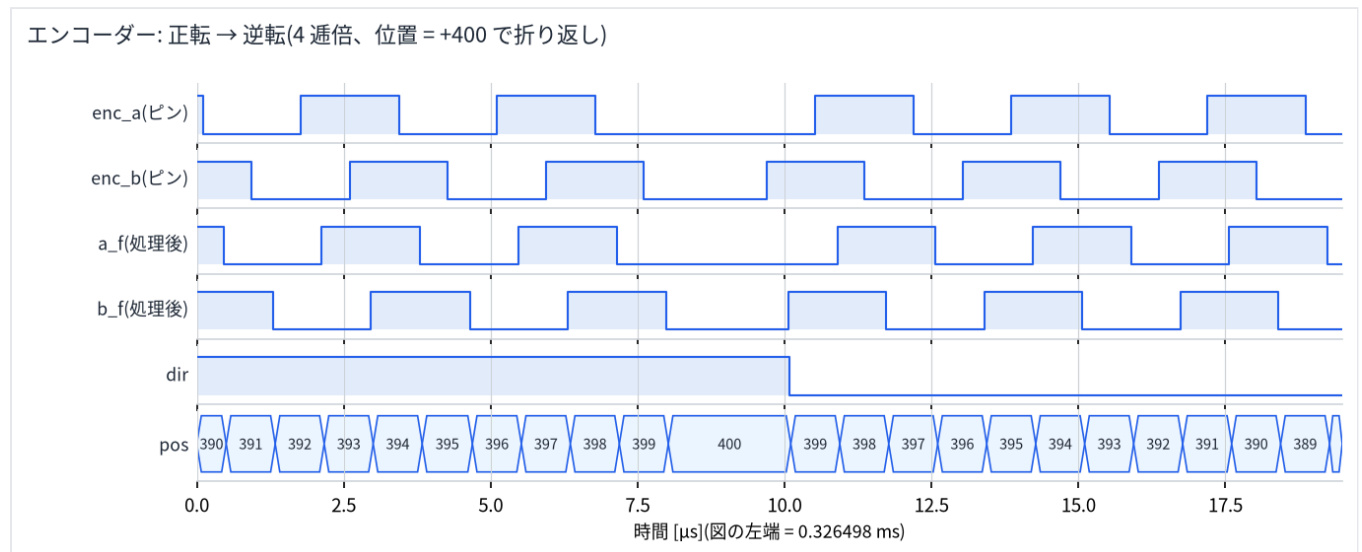


図 4-1 エンコーダーの正転 → 逆転。ピンから約 18 クロック遅れて処理後の信号が変わり、各エッジで位置が ±1

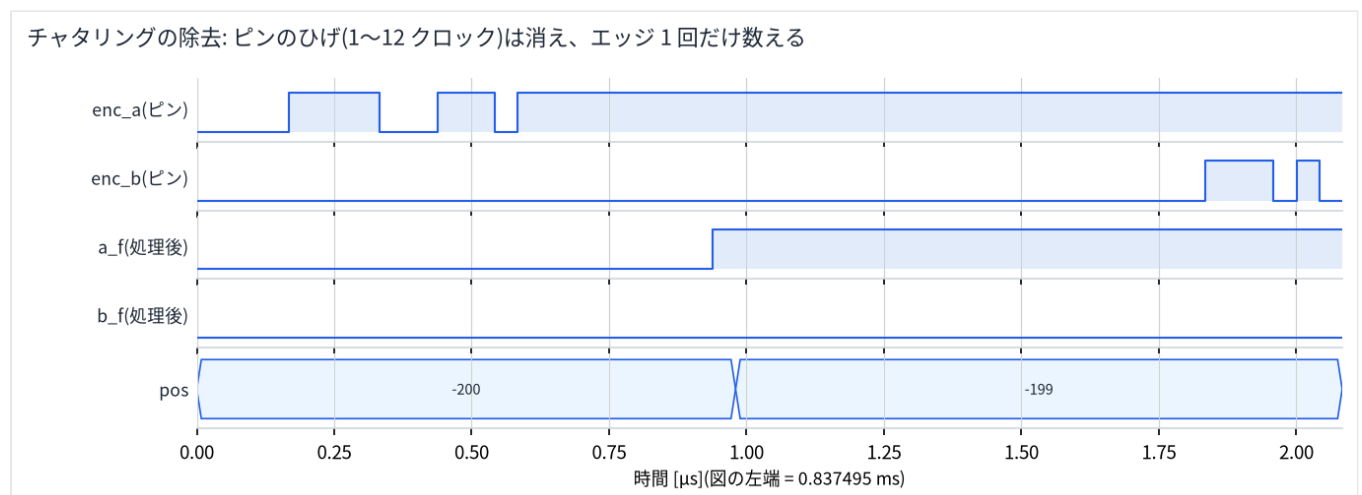


図 4-2 チャタリングの除去。ピンの短いひげは消え、エッジ 1 回だけ数える

中心揃え PWM 1 周期(20 kHz):A 相 d = 1700、B 相 d = 700、デッドタイム 48 クロック

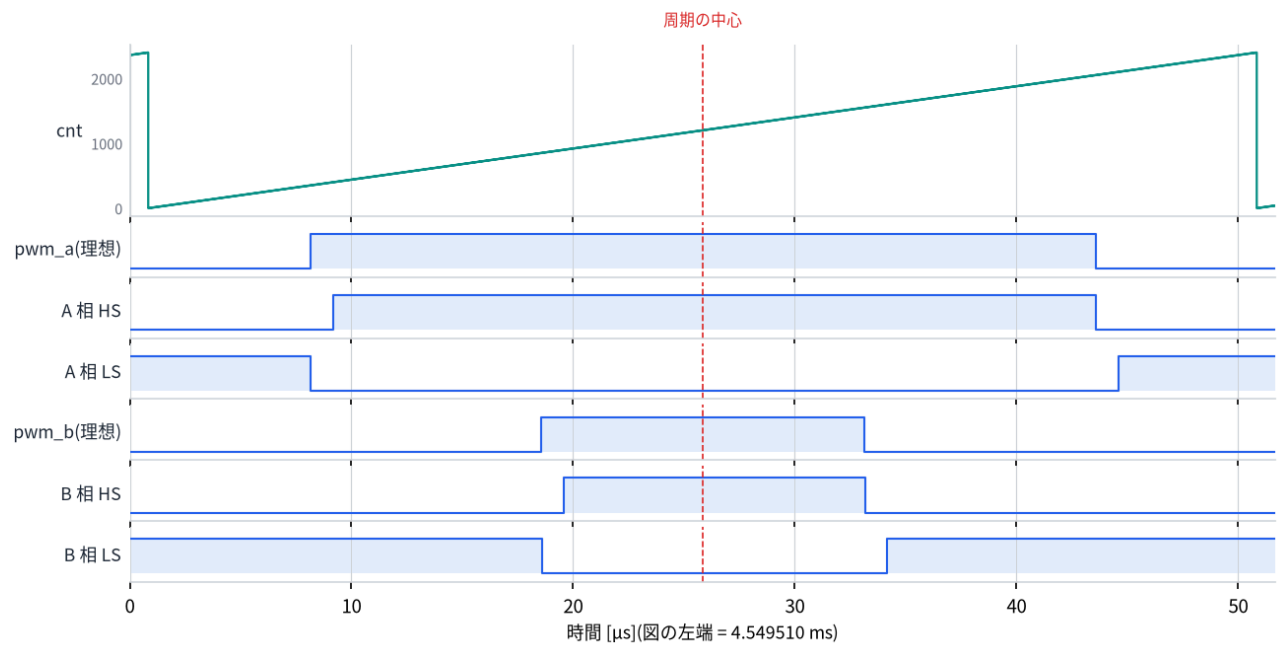


図 4-3 中心揃え PWM の 1 周期。A 相・B 相とも周期の中心(cnt = 1200)を挟んで対称

デッドタイム(dt = 48 クロック = 1.0 us):LS を切ってから 48 クロック後に HS を入れる

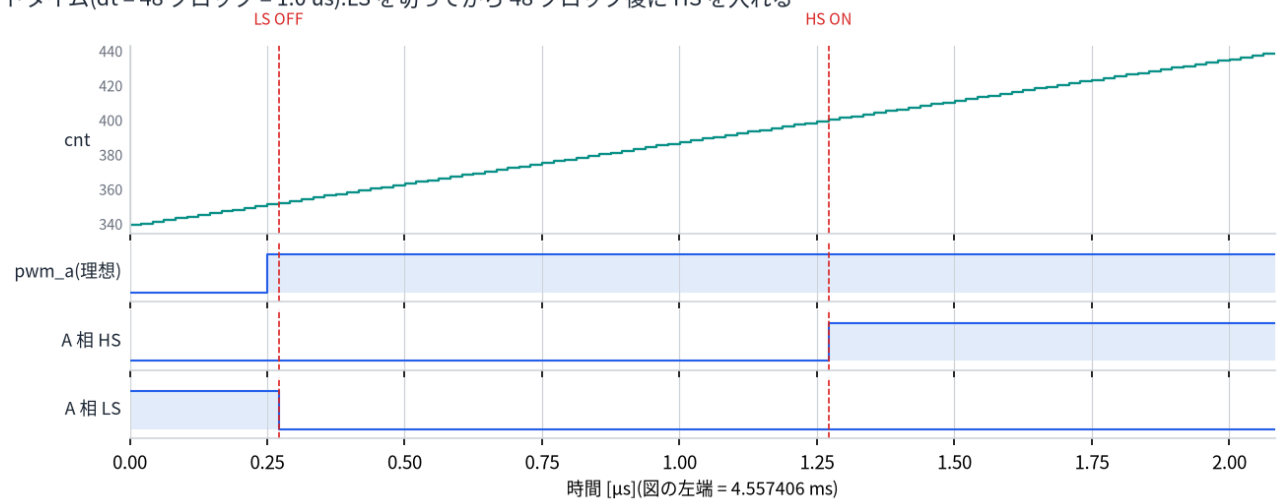


図 4-4 デッドタイム 48 クロック。LS を切ってから 48 クロック後に HS を入れる

過電流の停止: kill_async から最初のクロックの立ち上がりで 4 本とも OFF(遅れ 19.3 ns)

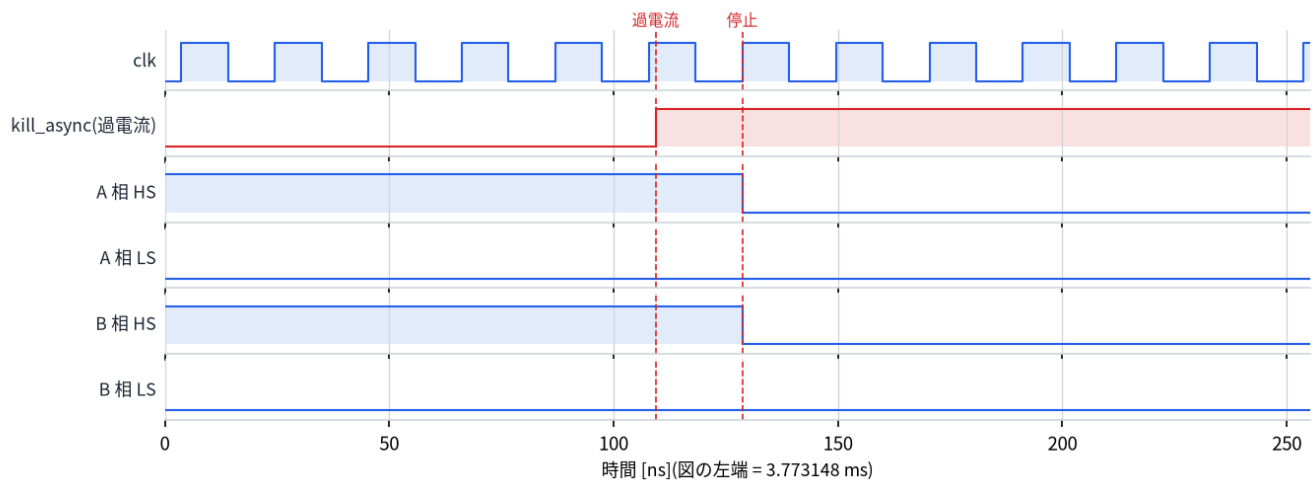


図 4-5 過電流の停止。過電流から最初のクロックの立ち上がりで 4 本とも OFF

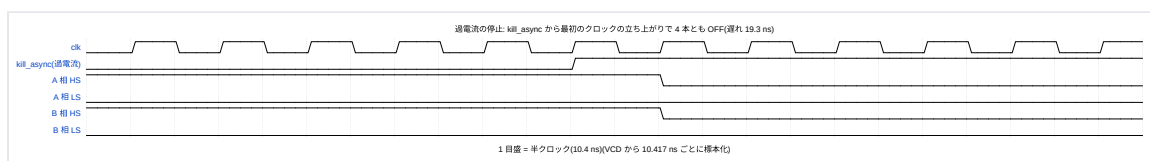


図 4-6 図 4-5 と同じ範囲を WaveDrom で描いたもの(半クロックごとに標本化)

UART のコマンド: PC → 「V CR」、FPGA → 「V0032 CR LF」(速度 = 0x32 = 50 カウント/ms)

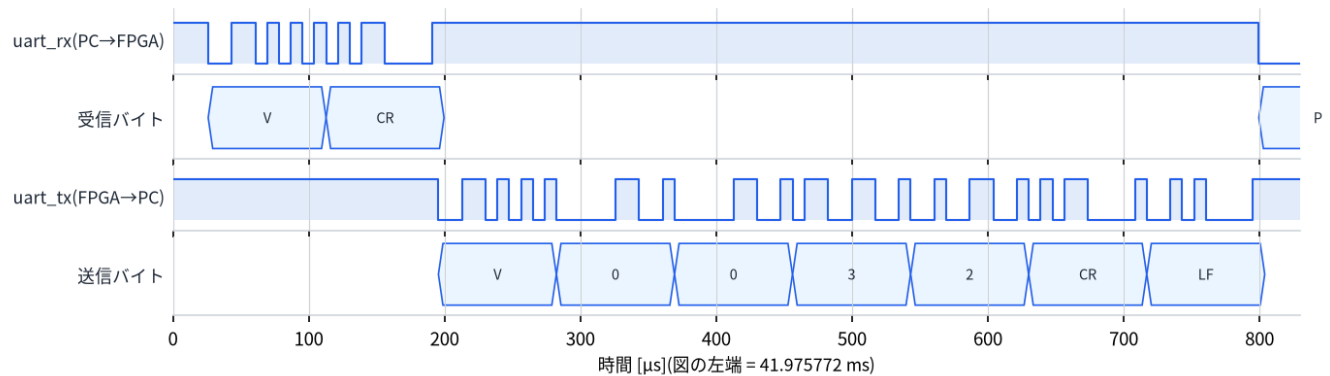


図 4-7 UART のコマンド V と応答 V0032

全体: 非常停止(E)→ 解除(C)→ 過電流 → 解除を断る(NG)→ 解除(C)

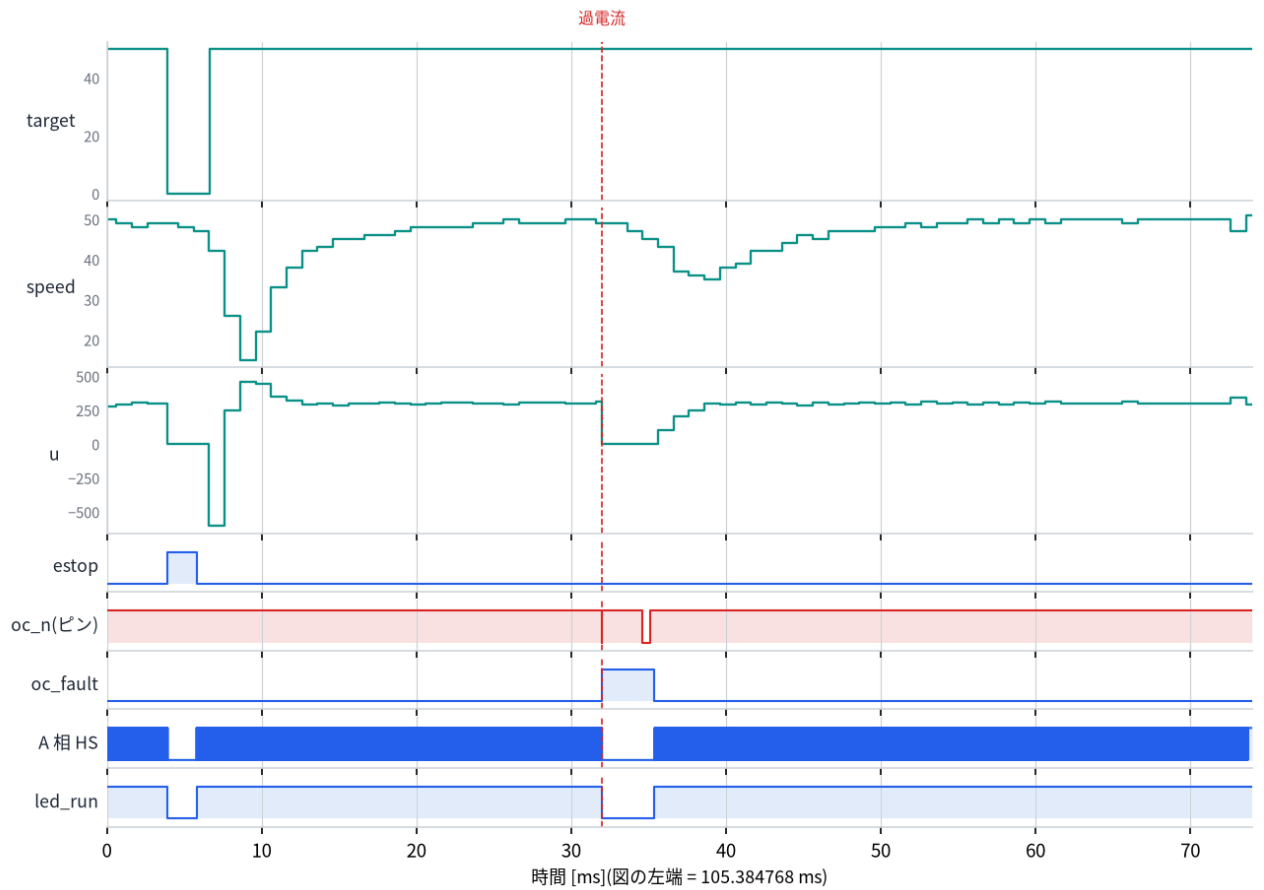


図 4-8 全体での非常停止と過電流。停止中は PWM が止まり(A 相 HS)、解除で再開する

5. 検証の項目と結果

Icarus Verilog で 4 つのテストベンチを流し、すべての項目を自動で判定します(可否は各ログの CHECK| の行、集計は results/sim/results.json)。乱数は種を固定しているので、流すたびに同じ結果になります。

テストベンチ	対象	合格	不合格	結果
tb_encoder	エンコーダー入力・速度の計測	11	0	合格
tb_pwm	中心揃え PWM・デッドタイム・過電流の停止	9	0	合格
tb_pi	速度の PI 制御	6	0	合格
tb_system	全体(閉ループ・UART のコマンド・非常停止・過電流)	13	0	合格

全項目

ID	検査の内容	結果	詳細(測った値)
E1	正転 100 周期(400 エッジ)で位置 = +400	合格	pos=400(期待 400)
E2	逆転 150 周期で位置 = -200	合格	pos=-200(期待 -200)
E3a	各エッジに 2~4 回のバウンド(1~12 クロック)をつけた正転 200 エッジ	合格	増分=200(期待 200)、数えた回数=200(期待 200)、不正遷移=0
E3b	バウンドつきの逆転 200 エッジ	合格	増分=-200(期待 -200)、数えた回数=200(期待 200)、不正遷移=0
E3c	停止中に A・B へ 1~15 クロックのひげを 200 回入れても数えない	合格	増分=0、数えた回数=0(どちらも期待 0)、不正遷移=0
E4	A・B の同時変化は不正遷移として検出し、位置は変えない	合格	err_pulse=1 回、位置の変化=0
E5a	正転 50 カウント/ms の速度を 1 ms ごとに計る	合格	3 回とも speed=50 か: 最後=50 カウント/ms(期待 50)
E5b	逆転 -120 カウント/ms の速度を計る	合格	3 回とも speed=-120 か: 最後=-120 カウント/ms(期待 -120)
E6	動作中に位置を 0 にしても速度は跳ばない	合格	クリアした周期=38(0~50)、次の 2 回=50 か(最後=50)、pos=226
E7	位置が 0 を挟むとき(下位 16 ビットの桁借り・桁上げ)も正しい	合格	0 → -5 → +5 の各エッジで一致、最後の pos=5 (0x00000005)
E8	全期間で位置は 1 クロックに ±1 しか変わらない	合格	1 クロックに 2 以上変わった回数=0
P1	PWM の周期 = 2400 クロック(20 kHz)	合格	周期=2400 クロック(20.000 kHz)
P2	パルス幅 = デューティ d クロック(d = 0, 1, 2, 3, 100, 1199~1201, 2398~2400)	合格	11 通りすべてで幅 = d(0~2400、2401 段階 = 11.2 ビット)
P3	パルスの中心が周期の中心(cnt = 1200)から 0.5 クロック以内	合格	中心のずれの最大 = 0.5 クロック(d が奇数のとき 0.5)
P4	周期の途中でデューティを変えても、その周期は前の値のまま(二重化)	合格	変えた周期の幅=800(期待 800)、次の周期=2000(期待 2000)
P5	デッドタイム = 設定値(クロック)、ハイサイドの幅は d - dt	合格	dt = 1, 24, 48, 100 で両方向の間 = dt、ハイサイド幅 = 1200 - dt
P6	デッドタイムより短いパルスはハイサイドを ON にしない	合格	d=30、dt=48 でハイサイドの幅=0(期待 0)

ID	検査の内容	結果	詳細(測った値)
P7	過電流の入力から 1 クロック以内に PWM の 4 本を止める	合格	20 回すべて次の立ち上がりで 4 本とも 0。遅れの最大 = 20.34 ns(1 クロック = 20.83 ns)
P8	故障のラッチ・非常停止(kill_sync)の間は 4 本とも 0	合格	3 周期のあいだ 4 本とも 0
P9	全期間で、同じ相のハイサイドとローサイドが同時に ON にならない	合格	ハイサイドとローサイドが同時に 1 のクロック数 = 0
Q6	計算の遅れ = 8 クロック(1 ms の制御周期に対して 0.02 %)	合格	start を取り込んだクロックから done まで 8 クロック
Q1	乱数 3000 回で u と積分が参照モデル(理想の整数演算)とビット単位で一致	合格	3000 回のうち不一致 0 回
Q2	大きな正の偏差で u = +1200 に飽和し sat_hi = 1	合格	u=1200 sat_hi=1
Q3	飽和中は積分を止め(増え続けない)、偏差が反転した最初の周期で $u \leq 0$ (すぐ止める向き)	合格	飽和中 200 周期で積分 110520 → 最大 110520(Q.8)、符号反転の最初の周期で u=-1
Q4	大きな負の偏差で u = -1200 に飽和し sat_lo = 1、積分は止まる	合格	u=-1200 sat_lo=1、100 周期で積分の変化なし
Q5	停止(enable = 0)で積分と u がすぐに 0	合格	enable=0 の 2 クロック後に u=0、積分=0
S1	起動直後は非常停止の状態が始まり、PWM を出さない	合格	応答 R01(期待 R01)、起動後 2000 クロックの PWM は 4 本とも 0
S2	C で解除して運転中になる	合格	C → OK、R → R20(期待 R20 = 運転中)
S3	目標速度 50 カウント/ms に落ち着く (± 2)。整定 30 ms 以内	合格	V → V0032(= 50 カウント/ms)、整定時間(± 2 に入って出ない)= 5 ms、最大 51
S4	P で読んだ位置が、模型が出したエッジの数と一致 (± 3)	合格	P → P00000782(= 1922)、模型のエッジ数 1922(差 0)
S5	目標 -50(FFCE)で逆転して落ち着く	合格	V → VFFCE(= -50 カウント/ms、期待 -50 ± 2)
S6	D コマンドでデッドタイムを変えられる(0x18 = 24 クロック = 0.5 μ s)	合格	D18 → OK、A 相の H→L 間 = 24、L→H 間 = 24 クロック(期待 24)
S7	K・I(ゲイン)の設定、小文字・空白入りのコマンドも受け付ける	合格	K0C00 / i 0266 → OK、v → VFFCE
S8	知らないコマンド・桁数の誤り・16 進でない文字には ? を返す	合格	X、S12、S12345、P1、@、SZZZZ の 6 つすべてに ?
S9	E(非常停止)は文字を受けた時点で PWM を止め、目標を 0 にする	合格	E のストップビットの中央から 167 ns で 4 本とも 0(行末の前)、1 ms 止まったまま、R → R01
S10	過電流の入力から 1 クロック以内に PWM を止め、故障をラッチする	合格	oc_n の立ち下がりから 13.53 ns 後のクロックで 4 本とも 0、ピンが戻っても 2 ms 停止のまま、R → R02
S11	過電流のピンが有効な間は解除を断り、戻ってから C で再開する	合格	ピン有効中の C → NG、戻してからの C → OK、35 ms 後 V = 50
S12	Z で位置を 0 にする	合格	Z → OK、直後の P = 25(Z から約 0.6 ms 分の移動 ≈ 30)
S13	全期間で貫通なし、バウンドつきのエンコーダーでも不正遷移なし	合格	貫通 0 クロック、エンコーダー不正遷移 0、R → R20

閉ループの応答

tb_system では、テストベンチの中に DC モーターの模型(1 次遅れ、 $K = 200$ カウント/ms、 $\tau = 5$ ms)とエンコーダーの模型 (4 回に 1 回バウンドつき)を置き、UART でコマンドを送って全体を閉ループで動かしています。

全体の閉ループ: 速度のステップ応答(モーターの模型 $K = 200$ カウント/ms、 $\tau = 5$ ms)

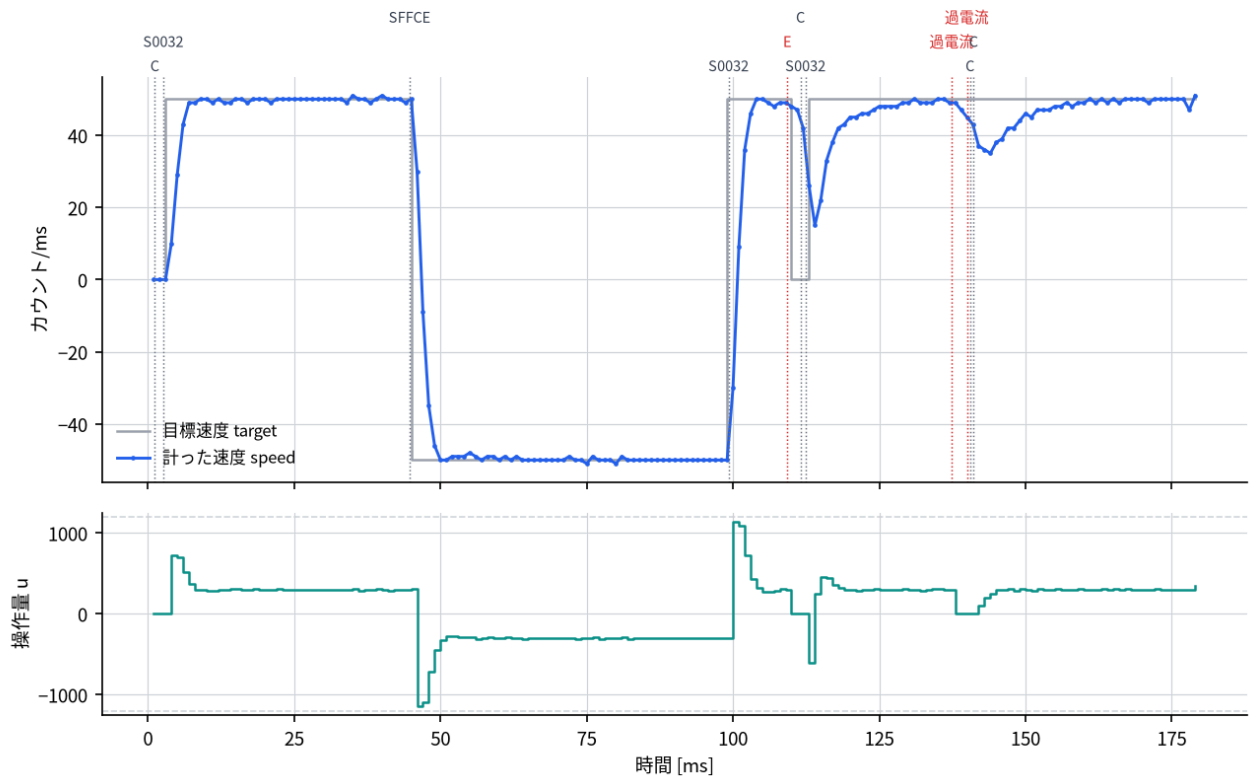


図 5-1 目標 50 → -50 → 50 カウント/ms、非常停止、過電流での速度と操作量

検査の強さの確認(ミューテーション)

RTL をわざと 1 か所だけ壊した版を作り、テストベンチがそれを不合格にできるかを確認しました (scripts/run_mutation.sh)。最初は M1(チャタリング除去を弱める)を見逃しました。ひげは「+1 して -1」で差し引き 0 になり、位置だけを見る検査では分からないためです。そこで「数えた回数」も検査に加え、M4 のために Q3 の測り始めと条件も厳しくしました。

版	壊したところ	テストベンチ	結果	不合格になった項目
M1	チャタリング除去を 16 → 3 クロックに弱める	tb_encoder	検出	E3a,E3b,E3c
M2	ハイサイドのデッドタイムを無くす	tb_pwm	検出	P5,P6
M3	過電流のピンの直結(即時の遮断)を外す	tb_pwm	検出	P7
M4	積分の止め(アンチwindアップ)を外す	tb_pi	検出	Q1,Q3

RTL の静的検査(Verilator -Wall)

すべてのモジュールを Verilator の `--lint-only -Wall` で検査し、警告が 0 件であることを確かめています。最上位は Yosys で接続を検査しています。

モジュール	結果
sync_filter	合格
quad_decoder	合格

モジュール	結果
encoder_if	合格
tick_gen	合格
speed_meter	合格
pwm_center	合格
deadtime	合格
pi_ctrl	合格
uart_rx	合格
uart_tx	合格
cmd_parser	合格
motor_ctrl	合格
top_up5k (yosys check)	合格

6. 論理合成・配置配線の結果

Yosys(synth_ice40 -dsp)で合成し、nextpnr-ice40 で配置配線(--freq 48 --seed 1 --threads 1)、icepack でビットストリーム(results/pnr/top_up5k.bin)を作りました。

使った資源

資源	使用	全体	割合
LUT4	1166	5280	22.1 %
FF	936	5280	17.7 %
CARRY	480	5280	9.1 %
LC (logic cells, after packing)	1701	5280	32.2 %
BRAM (EBR 4kbit)	0	30	0.0 %
SPRAM (256kbit)	0	4	0.0 %
DSP (SB_MAC16)	2	8	25.0 %
IO	12	96	12.5 %
PLL	1	1	100.0 %
Global buffer	8	8	100.0 %

LUT4・FF・CARRY は合成後の数、LC 以下は配置配線後の数。BRAM は使っていません(記憶は全てフリップフロップ)。Global buffer はクロックとリセット・高い fanout の信号に nextpnr が自動で割り当てたもの。

モジュールごとの内訳(階層を保った合成、目安)

モジュール	LUT4	FF	CARRY	DSP
pwm_center	184	130	122	0
pi_ctrl	268	297	173	2
sync_filter	10	8	3	0
tick_gen	22	17	14	0
uart_rx	44	36	8	0
uart_tx	33	23	9	0
cmd_parser	295	178	9	0
deadtime	32	11	14	0
motor_ctrl	35	28	16	0
quad_decoder	102	71	58	0
speed_meter	90	115	31	0

タイミング

目標	48 MHz(周期 20.833 ns)
最大の動作周波数	53.79 MHz(nextpnr、配線後)/ icetime の見積もり 55.17 MHz
最悪の余裕(slack)	+2.243 ns(目標より 12.1 % 速い)

最悪の経路	u_core.u_speed.speed_SB_LUT4_I3_15_LC → u_core.u_pi.e_neg_SB_DFFESR_Q_DFFLC 論理 9.15 ns + 配線 9.44 ns
-------	--

配置のシードを変えたときのばらつき

シード	最大周波数 [MHz]	余裕 [ns]	48 MHz
1	53.79	+2.24	合格
2	53.10	+2.00	合格
3	55.91	+2.95	合格
4	51.13	+1.28	合格
5	53.55	+2.16	合格

最初の版は 17 MHz でした。PI 制御・速度の計測・コマンドの解釈・PWM の比較値・位置カウンターの各所で、「1 段に演算は 1 つ」「比較の結果はレジスタに入れてから選ぶ」「大小比較を ASCII のビットの形や上位ビットの比較に置き換える」ことで 48 MHz を満たしました(7 章)。

合成後の回路

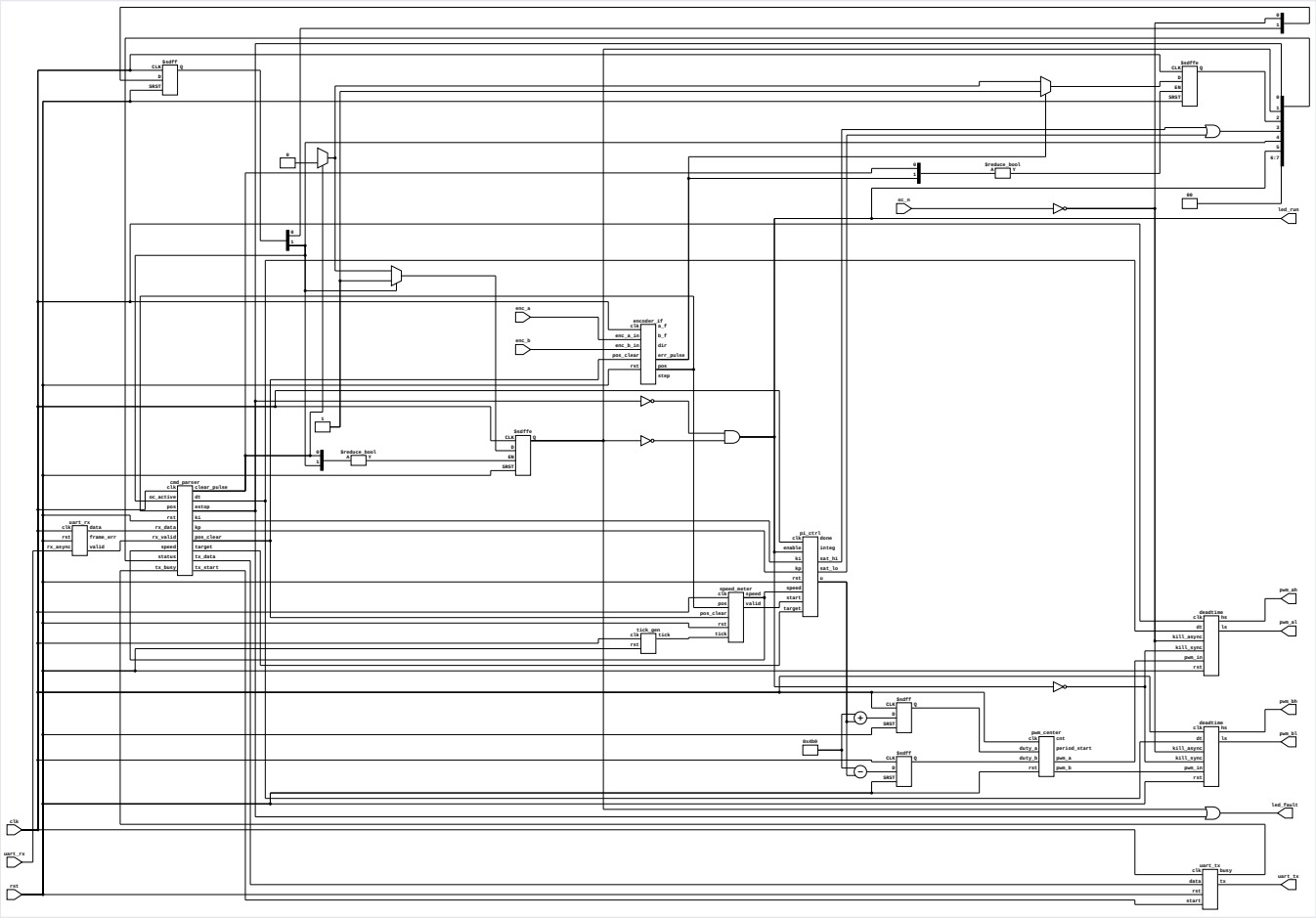


図 6-1 motor_ctrl の上位階層(語レベルで合成した直後。子モジュールのつながり)

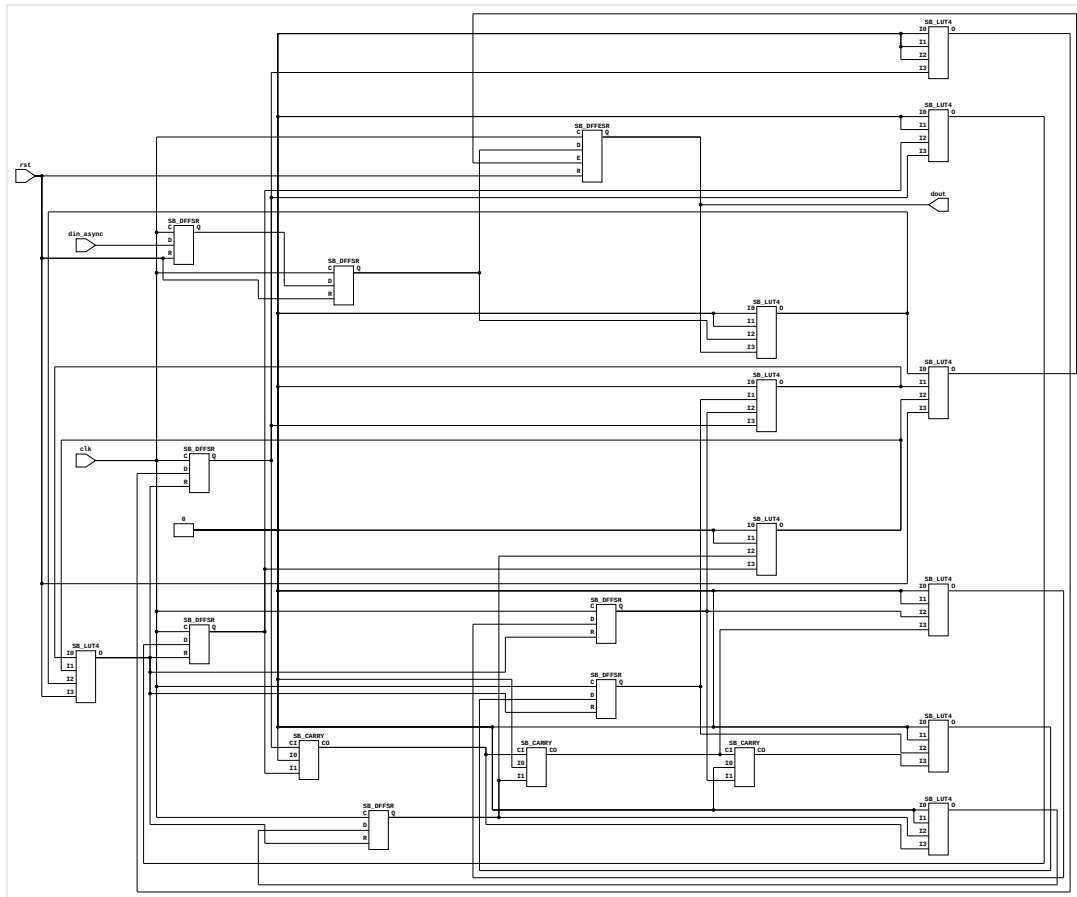


図 6-2 sync_filter を iCE40 の部品(SB_LUT4・SB_DFF*・SB_CARRY)まで落とした回路

deadtime・uart_tx の部品レベルの回路図と、配置の図(results/pnr/placement.svg)も同じフォルダーにあります。

7. 設計上の判断と制約

- **クロック源:** 内蔵の SB_HFOSC(48 MHz)は周波数の誤差が大きく(±10 % 程度)、UART に向かないため、外付け 12 MHz + PLL にしました。
- **過電流の経路:** 1 クロック以内の停止のため、ピンを同期化せずに出力のフリップフロップへ入れています(3.4 節)。ピンにノイズが乗ると誤って止まりうるので、基板側で RC フィルタかコンパレーターのヒステリシスを入れてください。止めたあとの状態はすべて同期化した信号で決めます。
- **出力はすべてレジスターから出すので、ゲート信号にひげは出ません。**リセット中・停止中は 4 本とも 0(全 OFF = 惰性)です。
- **速度の分解能**は 1 カウント/ms(4 通倍後)です。低速で滑らかに制御したいときは、制御周期を延ばすか、エッジの間隔を測る方式に替えます。
- **UP5K の速さ:** 桁上げの鎖と配線が遅く、32 ビットの演算 1 回で 20 ns 近くかかります。パイプラインを深くしても、1 ms に 1 回の制御では遅れ(8 クロック = 0.17 μ s)は問題になりません。
- **デッドタイム 0** は 1 クロックとして働きます(貫通を防ぐため)。使うパワー素子のターンオフ時間より長く設定してください。
- **実機では未確認**です。検証はシミュレーション(RTL)と、配置配線後の静的タイミング解析までです。配置配線後のゲートの遅れを入れたシミュレーションはしていません。
- PI のゲインの初期値は、テストベンチのモーターの模型に合わせた値です。実際のモーターでは K・I コマンドで調整してください。

8. 再現の手順とツール

```
cd lp-engineering/fpga
bash scripts/setup_tools.sh # ツールを入れる(初回だけ。Debian 12 の apt と npm)
make all                   # 検査 → 検証 → 波形 → 合成・配置配線 → 設計書
make repro                 # もう一度すべて作り、前回と同じかを sha256 で比べる
```

同じ版のツールで流せば、VCD・ログ・合成と配置配線の結果・ビットストリーム・図は毎回同じになります (results/MANIFEST.sha256)。PDF だけは作成日時が中に入るため、比較から除いています。

使ったツールの版

```
iverilog:      Icarus Verilog version 11.0 (stable) ()
verilator:     Verilator 5.006 2023-01-22 rev (Debian 5.006-3)
yosys:         Yosys 0.23 (git sha1 7ce5011c24b)
nextpnr-ice40: nextpnr-ice40 -- Next Generation Place and Route (Version 0.4-1+b1)
icestorm:      0~20230218gitd20a5e9-1~deb12u1
python3:       3.11.2 matplotlib 3.11.2
node:          v24.22.0 netlistsvg 1.0.2 wavedrom-cli 3.1.1
playwright:    1.63.0
```