Xilinx Spartan6 への Wave Union TDC の実装

大阪大学　RCNP
高橋智則

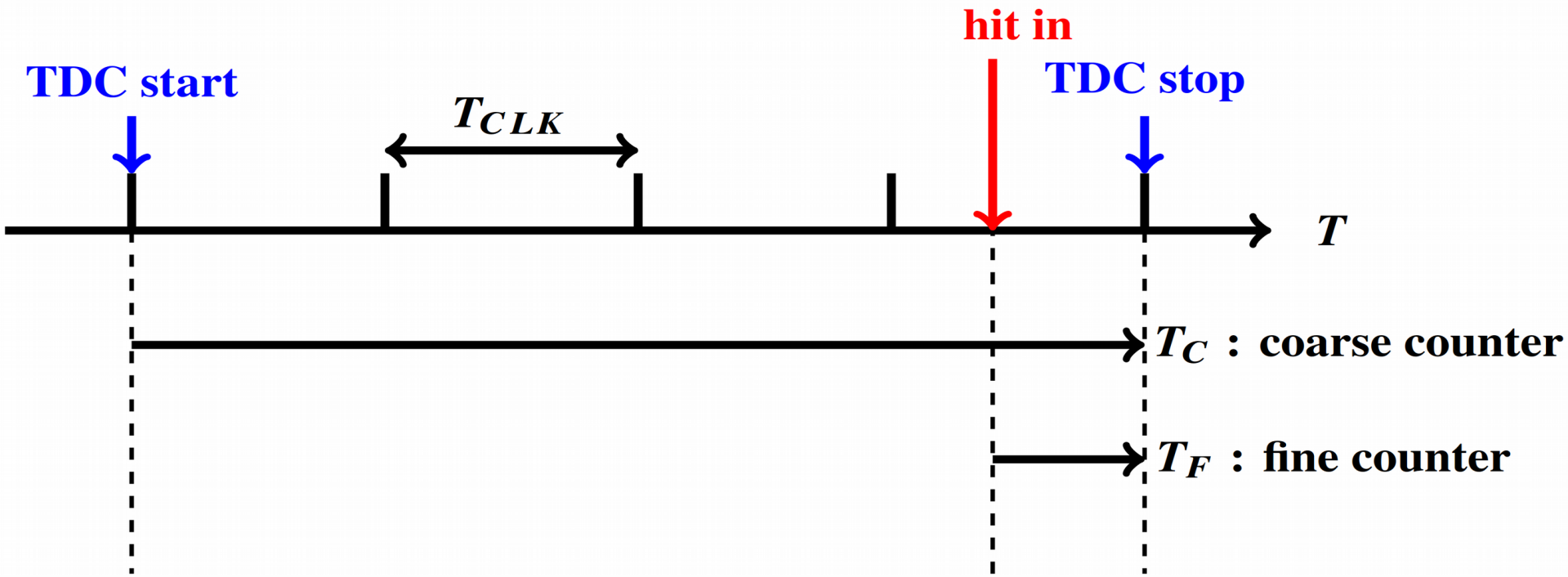
計測システム研究会　2017 @ 函館アリーナ

内容

- Tapped delay line 方式の TDC のおさらい
- 時間分解能を向上させるには
- Wave Union TDC
  - wave union launcher A の作り方
- Sparatan6 へ Wave Union TDC の実装
  - 分解能測定
  - 設計上の注意
- 今後の展望
- まとめ

復習：high resolution TDC

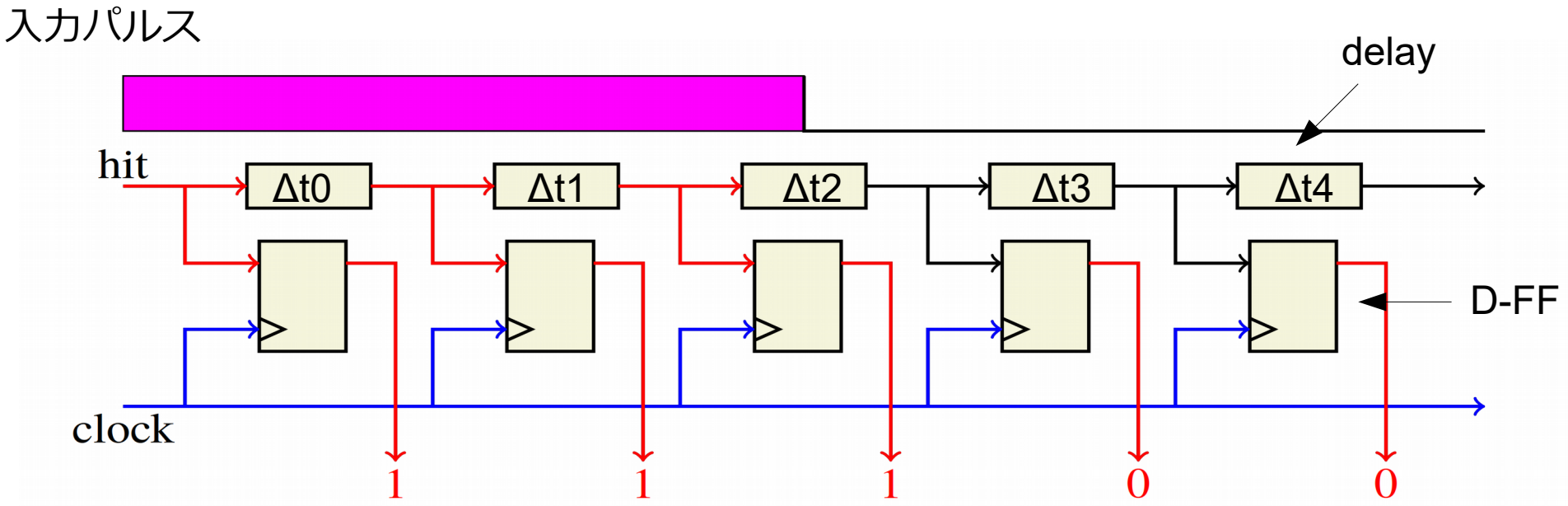
- clock をカウントするカウンタ　= coarse counter
- clock の間を内挿するカウンタ　= fine counter
  を使って時間計測



$$T = T_C - T_F$$

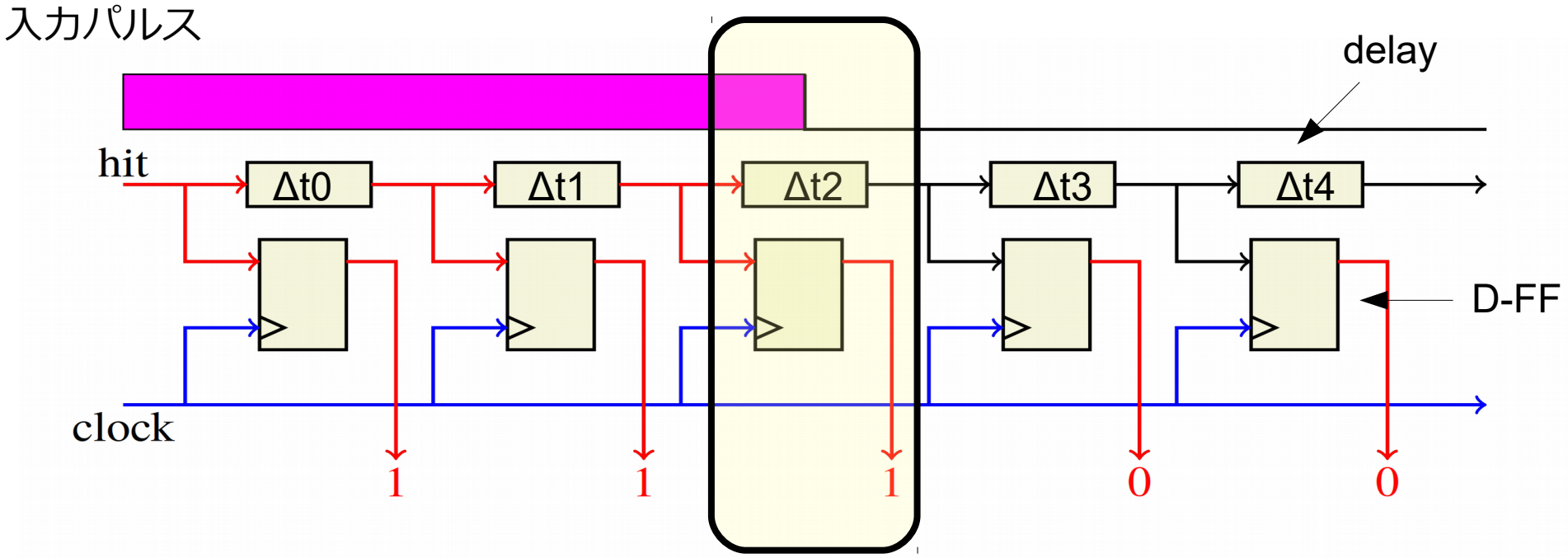
復習：Tapped delay line 方式の TDC

- **tapped delay line 上を信号がどこまで進んだか**
  - D-FF array 出力のビット列の中から 0→1 (or 1→0) に遷移するところを見つける
  - sampling clock のエッジからの時間に変換



復習：Tapped delay line 方式の TDC

- **tapped delay line 上を信号がどこまで進んだか**
  - D-FF array 出力のビット列の中から 0→1 (or 1→0) に遷移するところを見つける
  - sampling clock のエッジからの時間に変換



fine counter = Δt0 + Δt1 + Δt2/2

分解能を制限するのは何か？

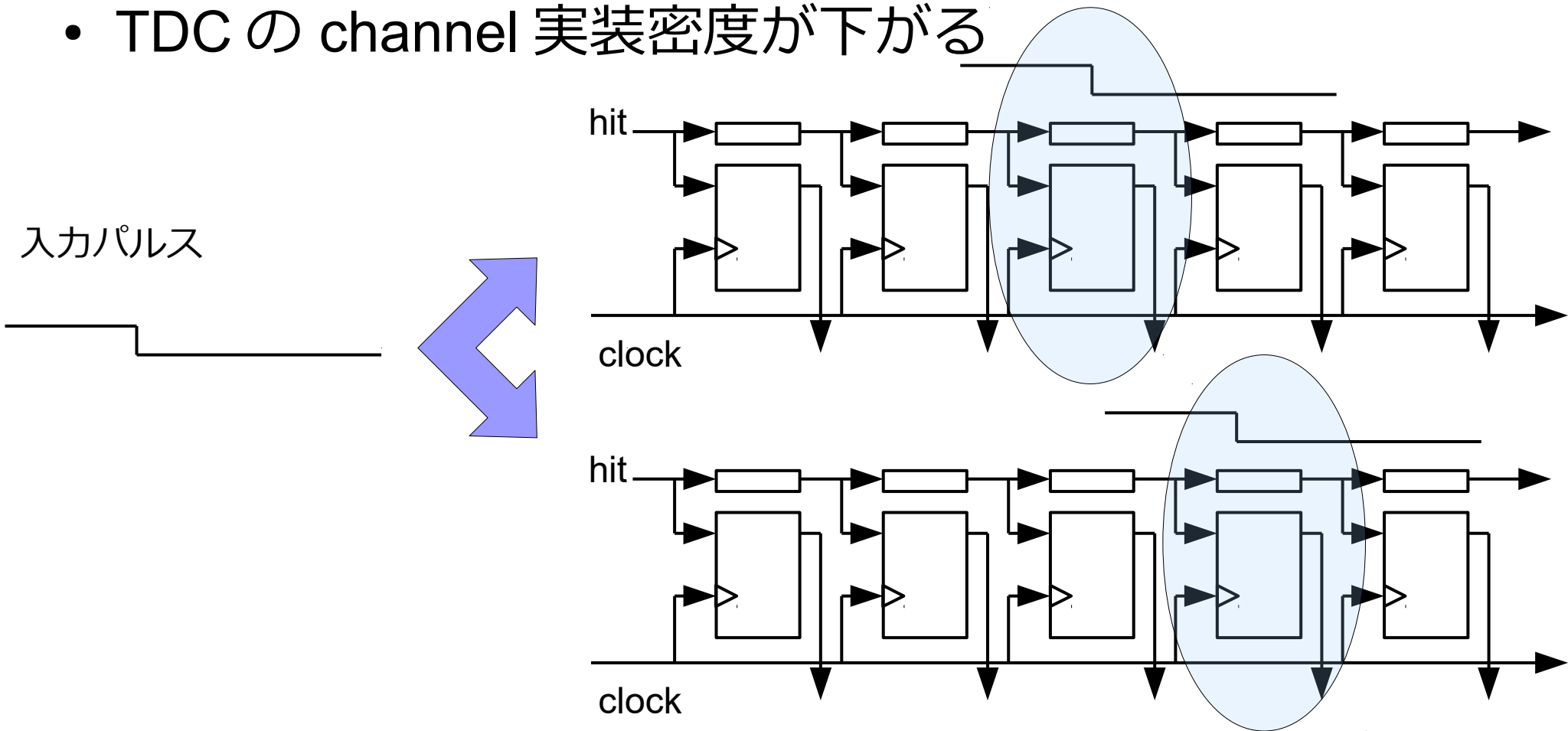
- delay line の**各 tap の遅延量**（**bin width**）
- **環境の影響**
  - 電源・グラウンドのノイズ、安定性
    - → 部品選び・基板設計の段階で対策
  - 温度（FPGA によっては影響が少ないものもある）
- **FPGA 内部の構造**に起因して遅延量が小さくならない箇所がある
  - 次の tap まで遠い
  - 例：clock region 境界をまたぐ delay line

FPGA への実装の仕方で分解能を改善する

- **sampling clock を速く**する　= delay line が短い
  - 環境の影響を受けにくくなる
  - FPGA によっては clock region 境界をまたがないのも可能
- 入力信号の遷移を複製して**複数回測定**する
  - リソース使用量、不感時間のトレードでいくつかの実装方法が存在
  - 例 :
    - 複数 delay line (空間的に複製)
    - Wave Union TDC (時間的に複製)

複数 delay line

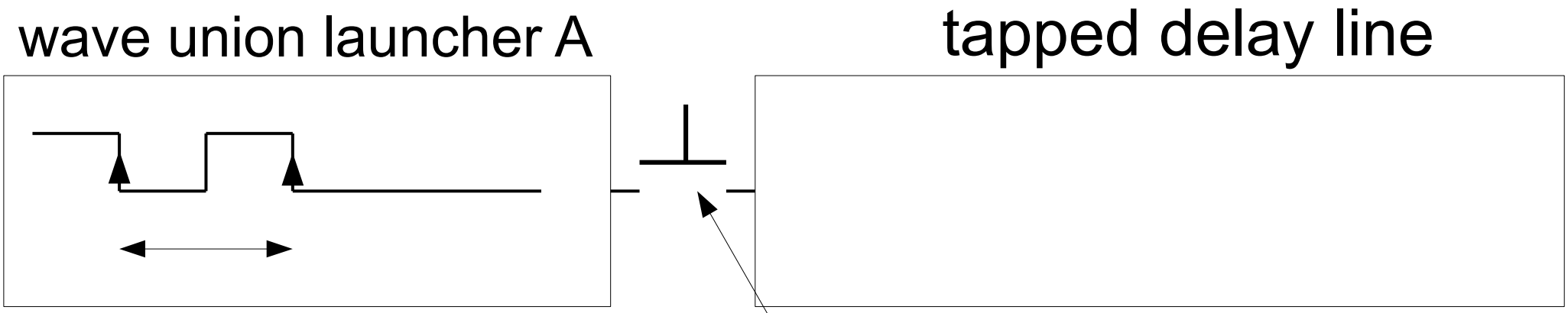
- 1 つの入力信号を FPGA 内部で fan-out させて N 個の delay line で測定
- 短所：リソース使用量が ~N 倍
  - TDC の channel 実装密度が下がる



Wave Union TDC

- Fermilab の Jin-Yuan Wu らが考案
  - J. Wu and Z. Shi, "The 10-ps wave union TDC: improving FPGA TDC resolution beyond its cell delay", IEEE NSS Conf. Rec. (2008) 3440. http://dx.doi.org/10.1109/NSSMIC.2008.4775079
- FPGA 内部で**入力パルスの 1→0(or 0→1) 遷移の timing を保ったまま時間方向に遷移回数を増やした pulse train (wave union) を作る**
- Wave Union Launcher: 2 通り
  - wave union launcher A(固定 pulse train)
  - wave union launcher B(帰還型，リングオシレータ)

wave union launcher A (固定 pulse train 入力) の概念図



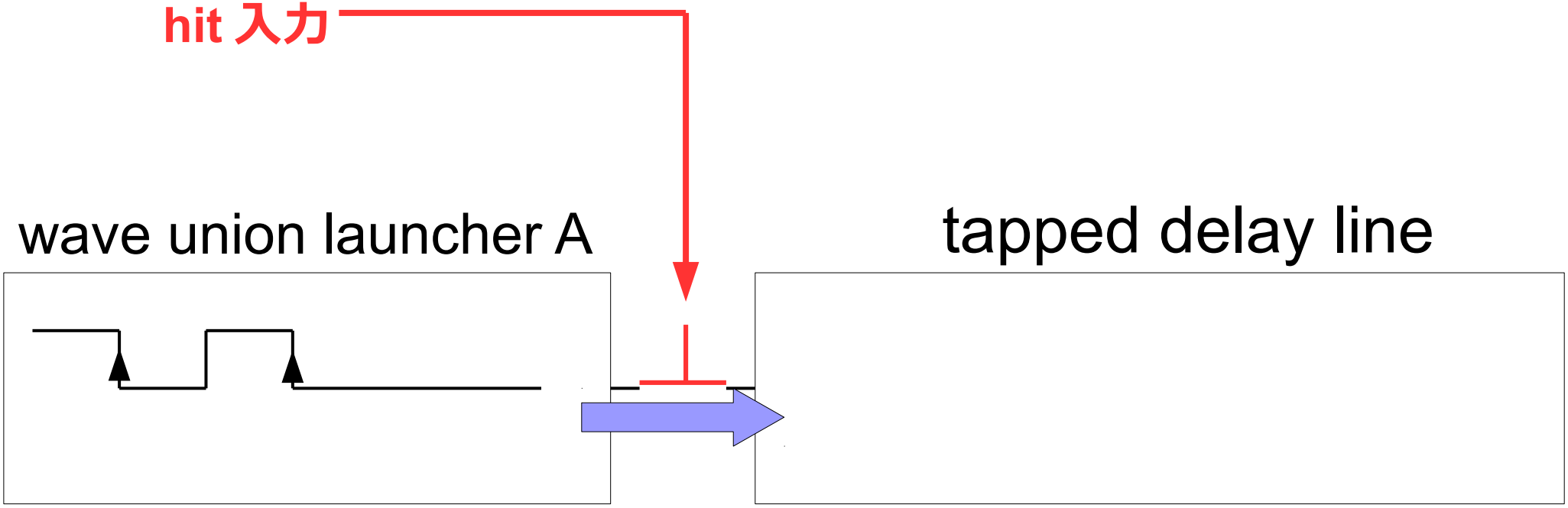
- **遷移箇所が 2 個 (以上)**
- **間隔が固定**

の pulse train を用意

パルス発射用スイッチ

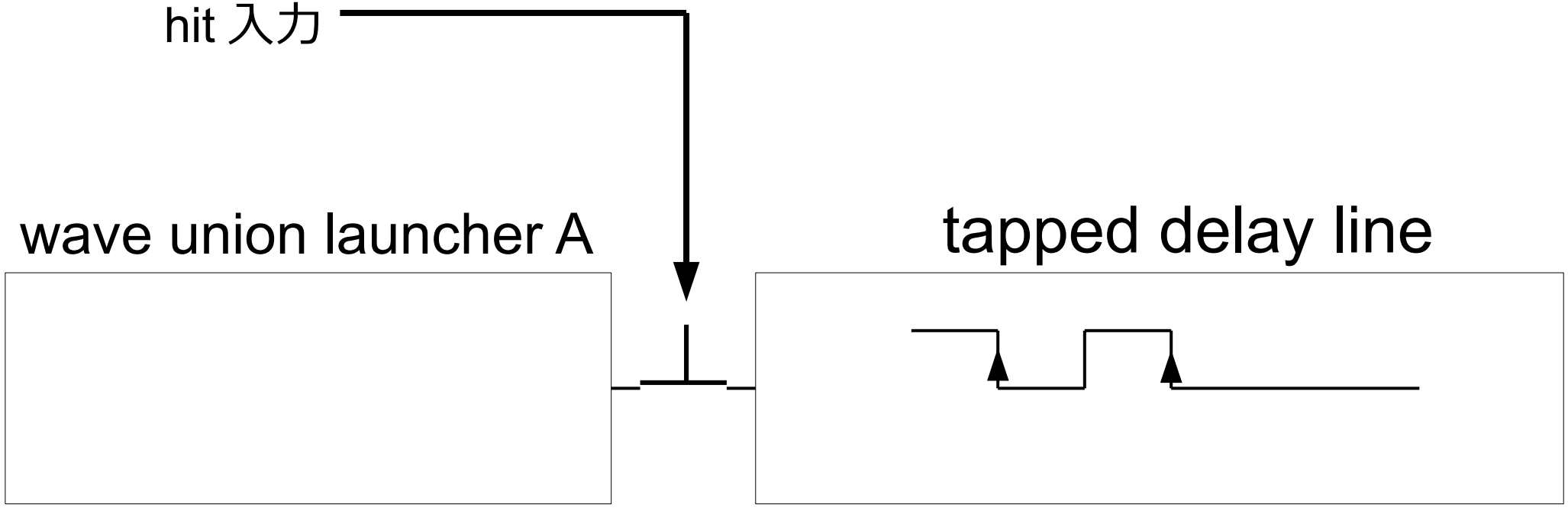
hit 入力がないときは
スイッチは off 状態

wave union launcher A (固定 pulse train 入力) の概念図 (2)



hit 入力があるとスイッチが on

wave union launcher A (固定 pulse train 入力) の概念図 (3)

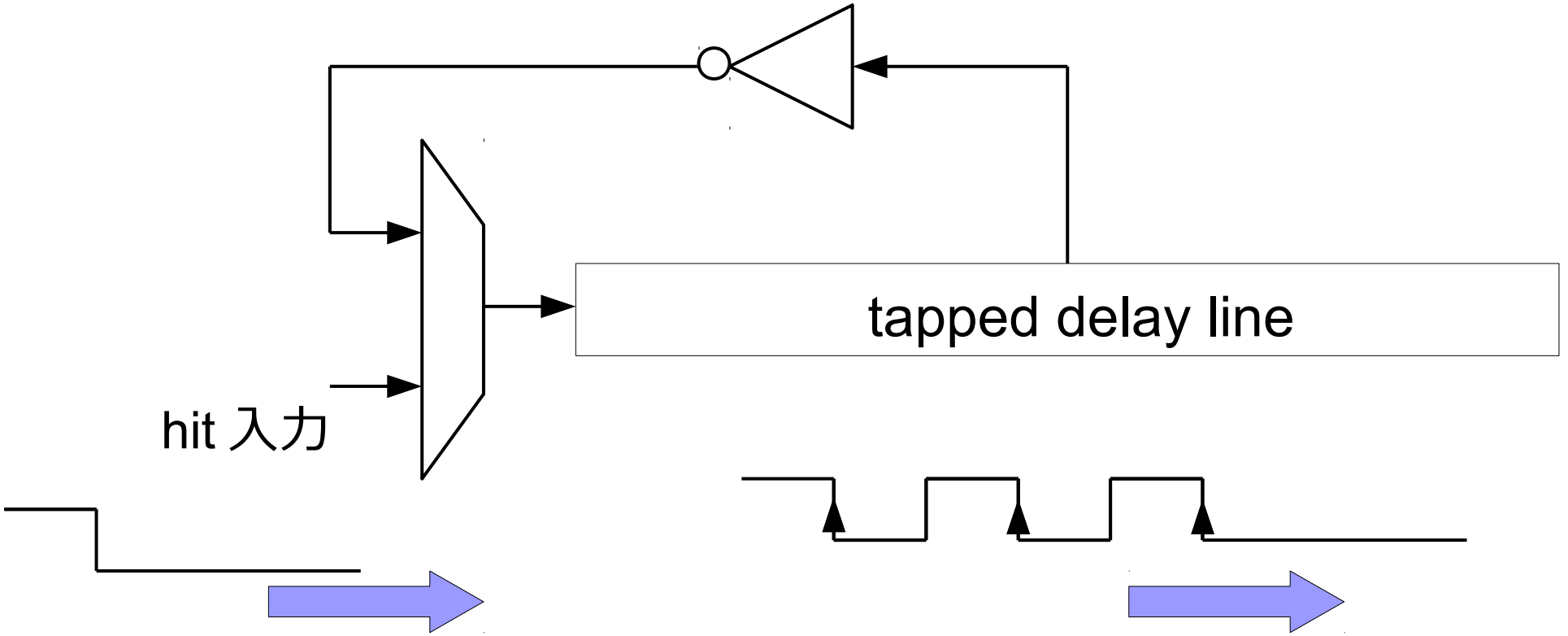


複数回遷移のあるパルスが
tapped delay line へ入る

どうやって pulse train を作るのか？→あとで説明

wave union launcher B (帰還型) の概念図

- tapped delay line の信号を途中で反転させて帰還させる (= ring oscillator)



実装方式による長所 (+) ・短所 (-)

	複数 delay line	WU-A (固定 pulse train)	WU-B (帰還型)
時間分解能	++	+	+++
不感時間		-	---
リソース使用量	---	-	-

wave union TDC に対するコメント

- tapped delay line の遷移位置を **2 進数に変換するエンコーダ**
  - → **遷移の数が増えると複雑化**
- **遷移回数を制御するために hit 入力信号は非同期ラッチを使って値を保持**
  - → 解除が必要→**不感時間の増加**

実装方式による長所 (+) ・短所 (-)

	複数 delay line	WU-A (固定 pulse train)	WU-B (帰還型)
時間分解能	++	+	+++
不感時間		-	---
リソース使用量	---	-	-

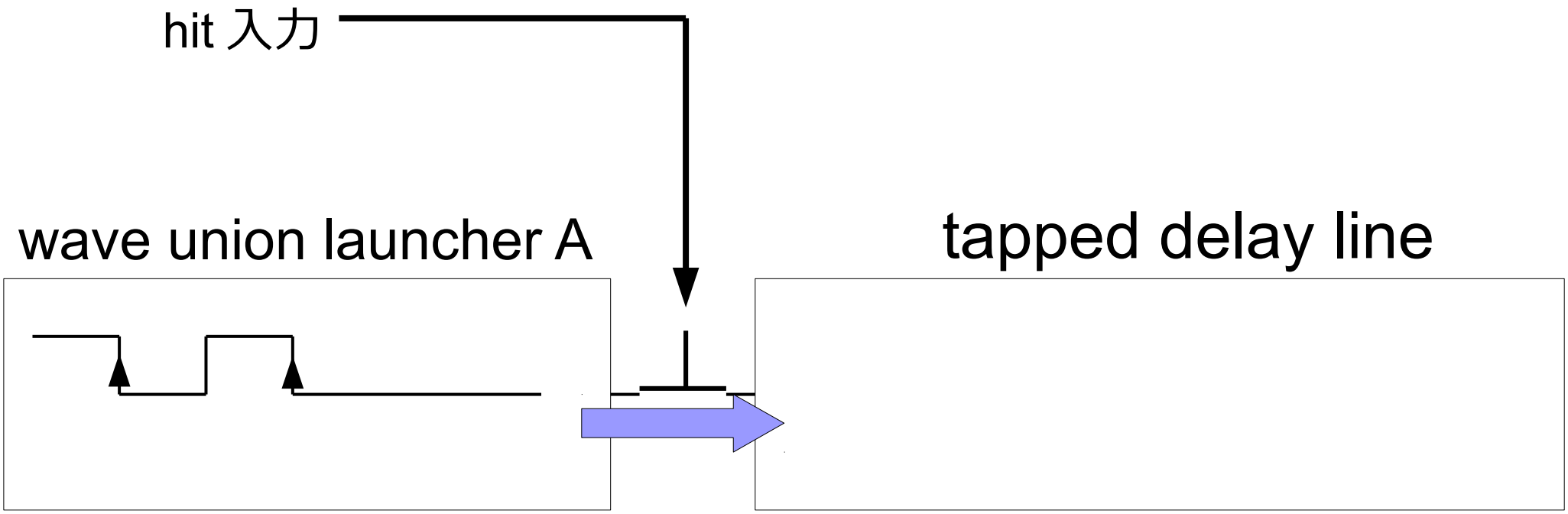
今回は

- high rate での使用 (= 不感時間を短く)
- TDC channel 数を稼ぐ (= リソース使用量を抑える)

を考慮して **wave union launcher A** (固定 pulse train 入力)

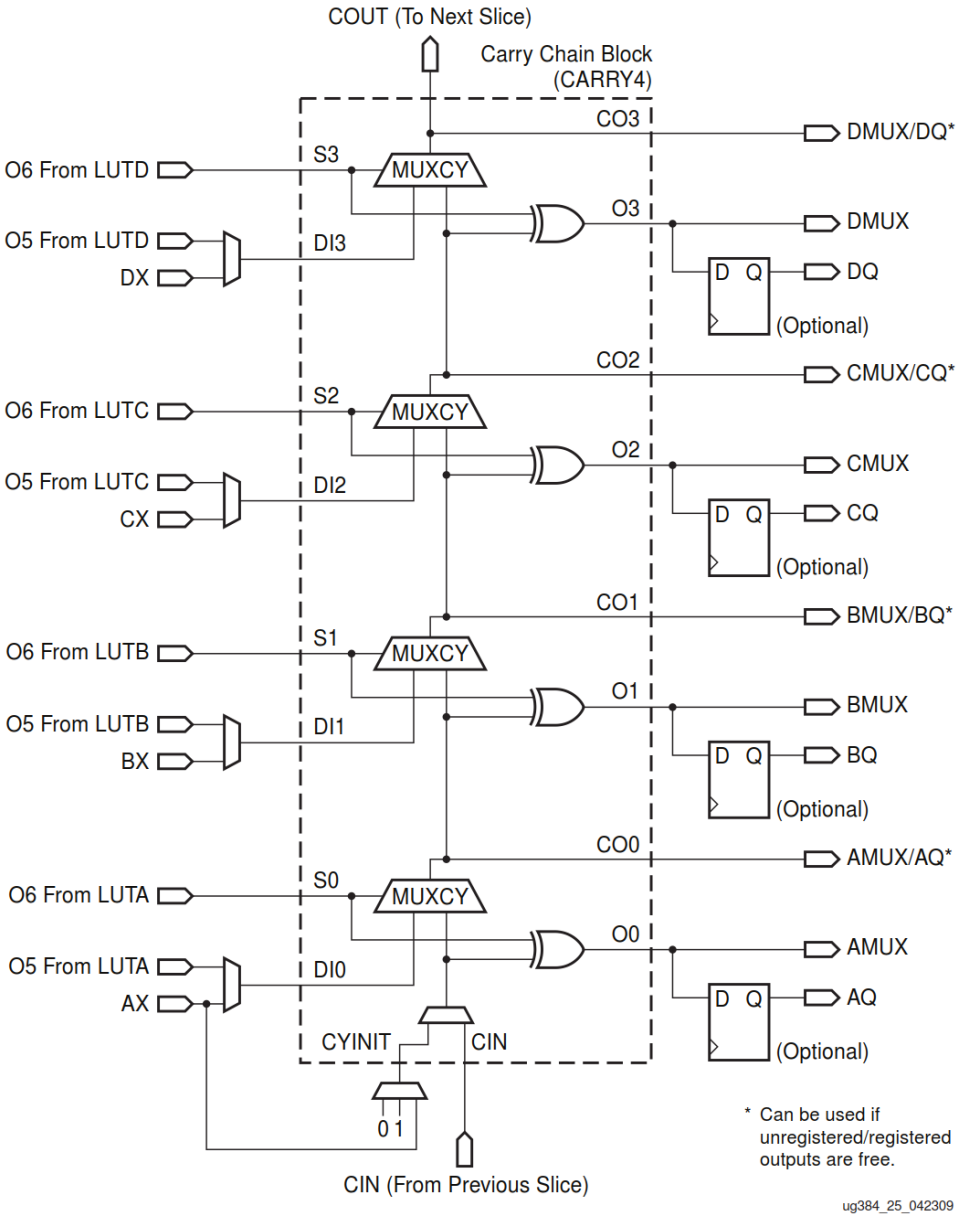
を Xilinx Spartan6 に実装した。

launcher A はどうやって作るのか？



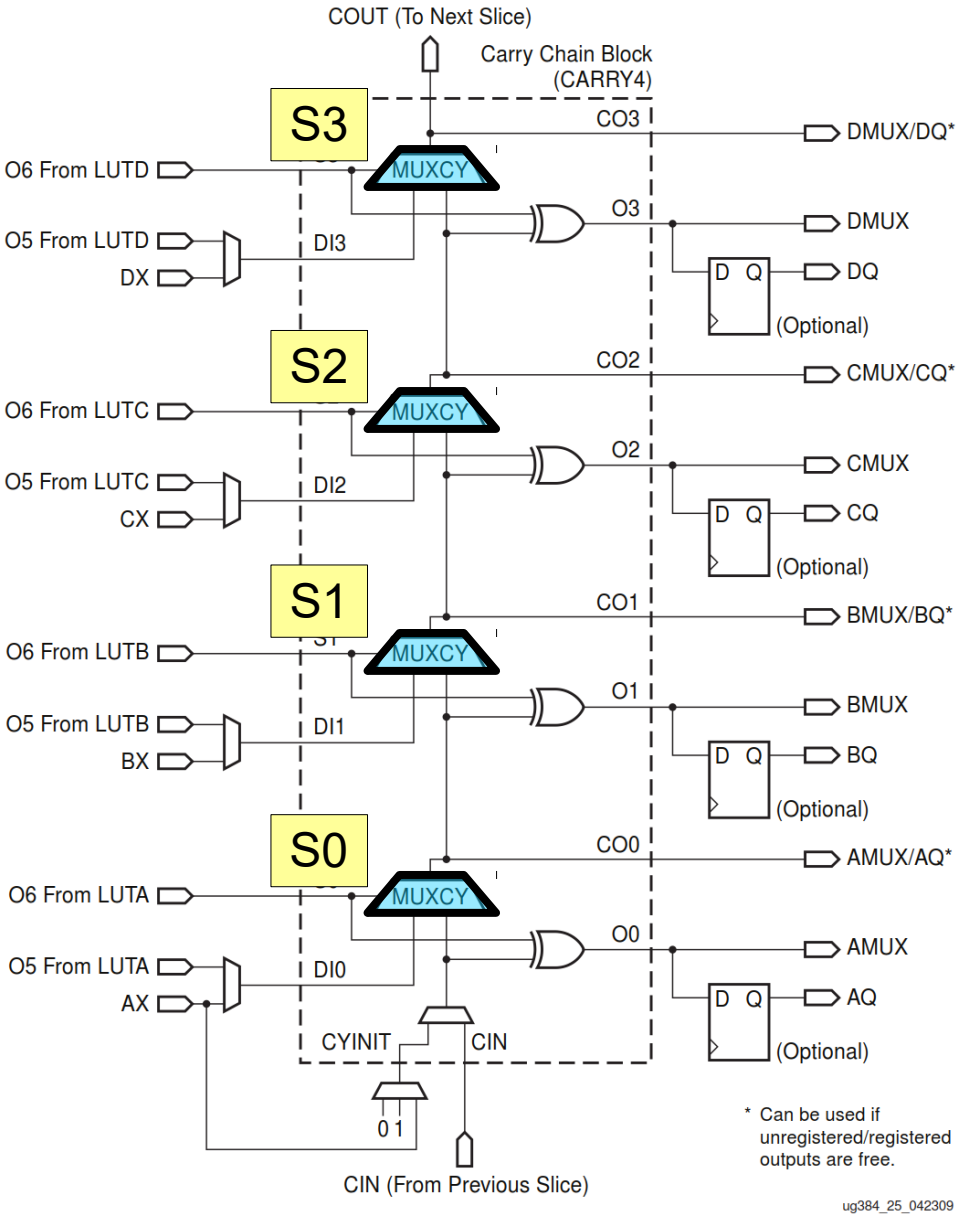
hit 入力があると
パルス発射スイッチが on

FPGA の CARRY4 をよく見てみると



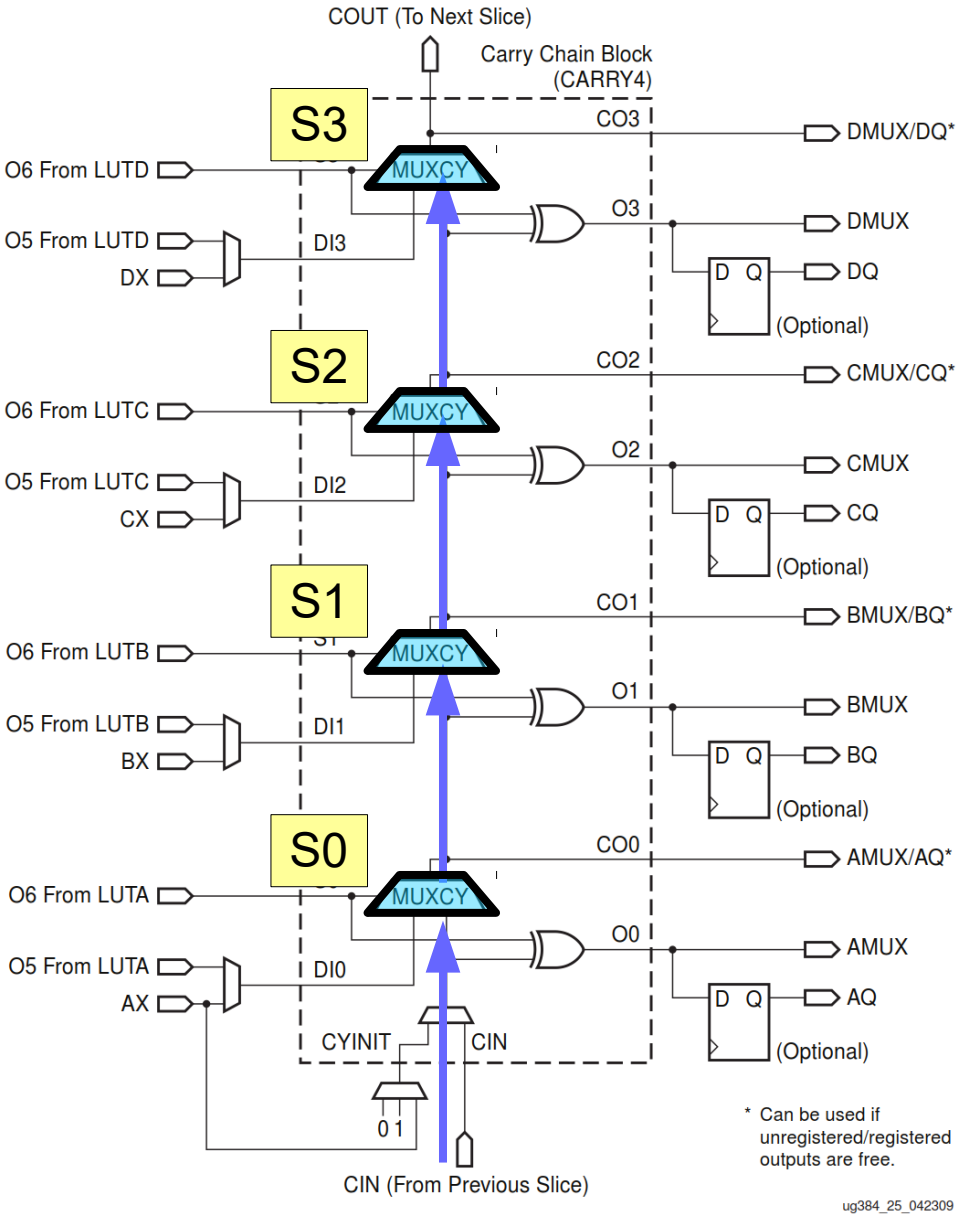
FPGA の CARRY4 をよく見てみると

- MUXCY が並んでいる
- 信号の選択をする　Si(i=0-3)



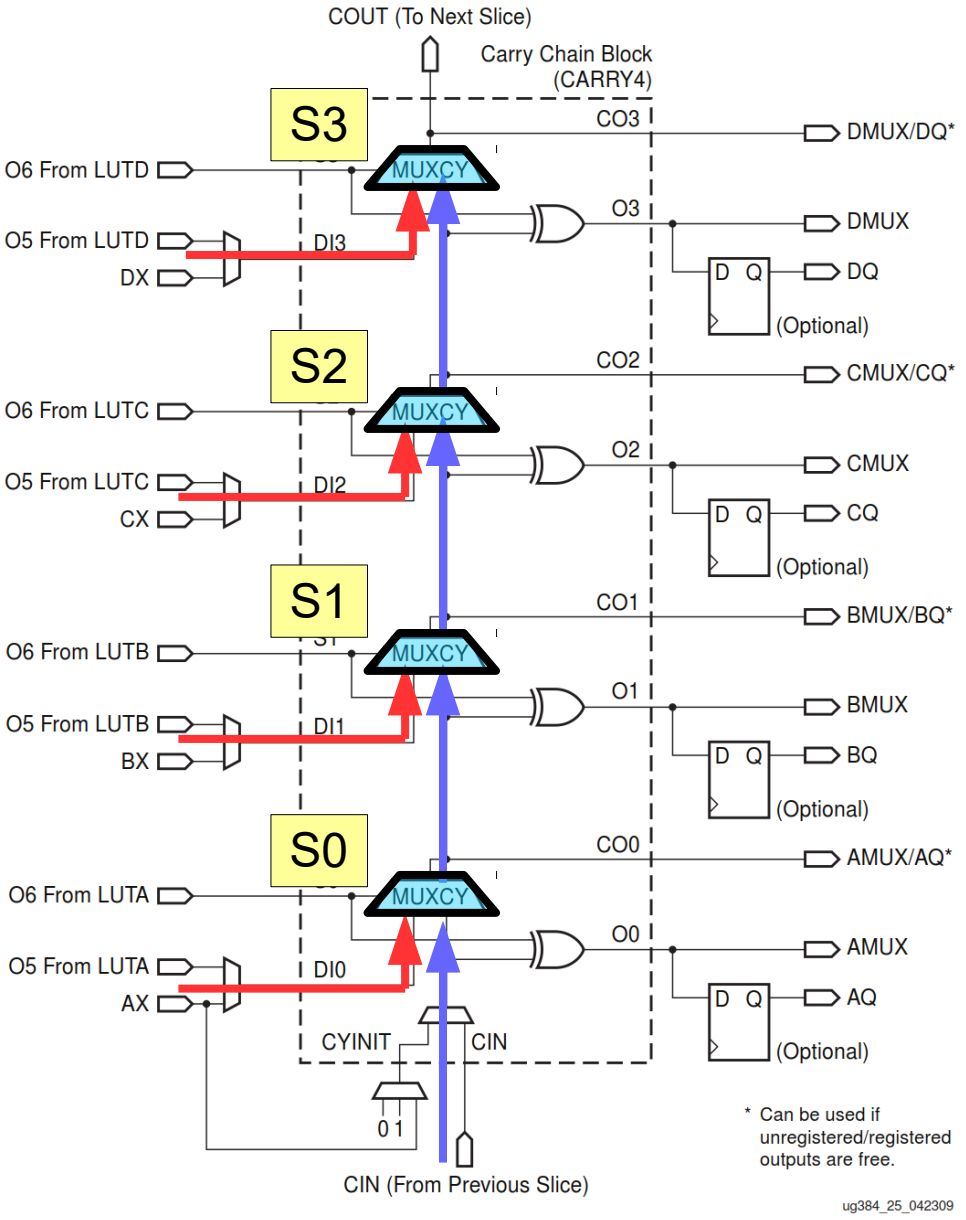
FPGA の CARRY4 をよく見てみると

- MUXCY が並んでいる
- 信号の選択をする　Si(i=0-3)
  - **Si=1: 一つ前の cell の信号**
    を次の cell に伝播する



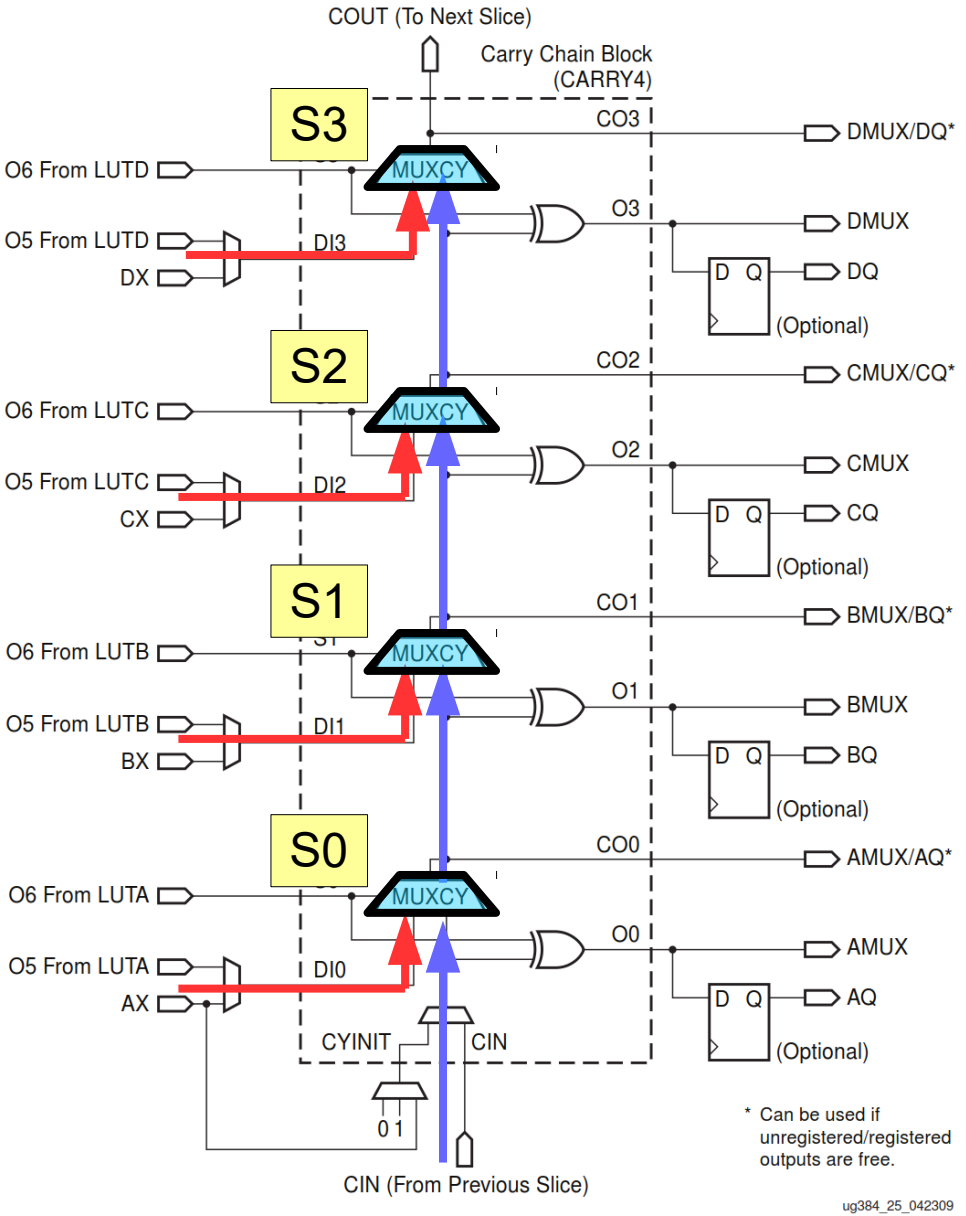
FPGA の CARRY4 をよく見てみると

- MUXCY が並んでいる
- 信号の選択をする　Si(i=0-3)
  - **Si=1: 一つ前の cell の信号**を次の cell に伝播する
  - **Si=0: 横から来る信号**を次の cell に伝播する

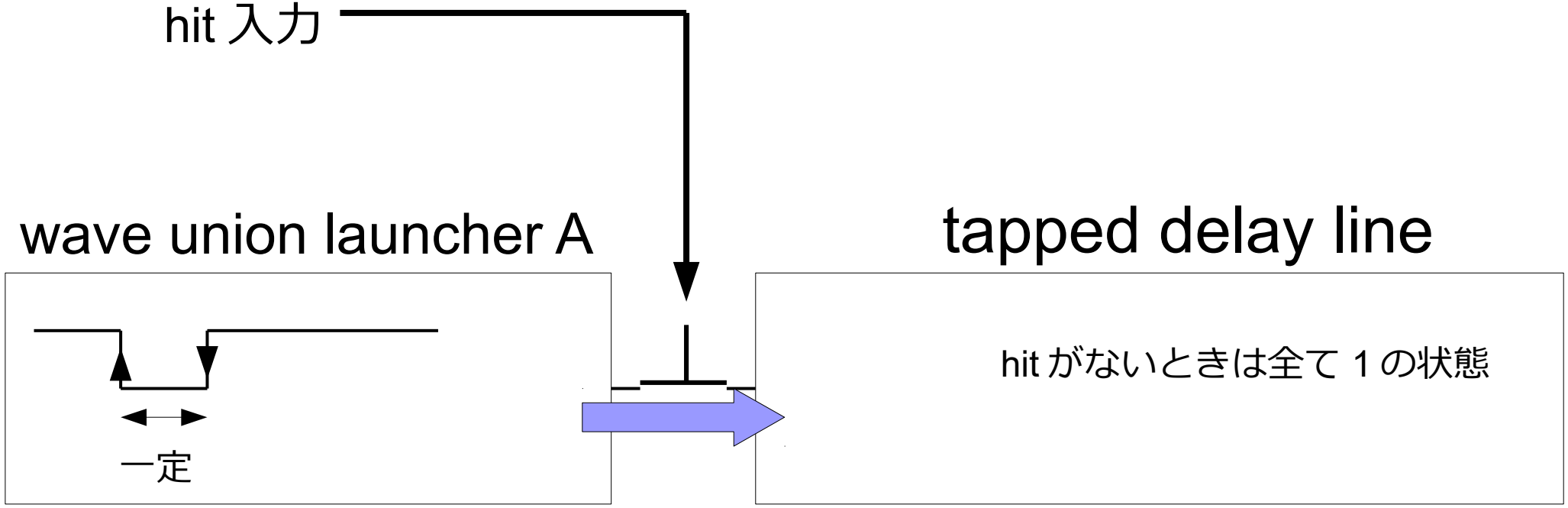


FPGA の CARRY4 をよく見てみると

- MUXCY が並んでいる
- 信号の選択をする　Si(i=0-3)
  - **Si=1: 一つ前の cell の信号**を次の cell に伝播する
  - **Si=0: 横から来る信号**を次の cell に伝播する
- これを上手い具合に使って **wave union launcher** を作る



例題：1→0→1 のパルス作成



1→0 : 1 回

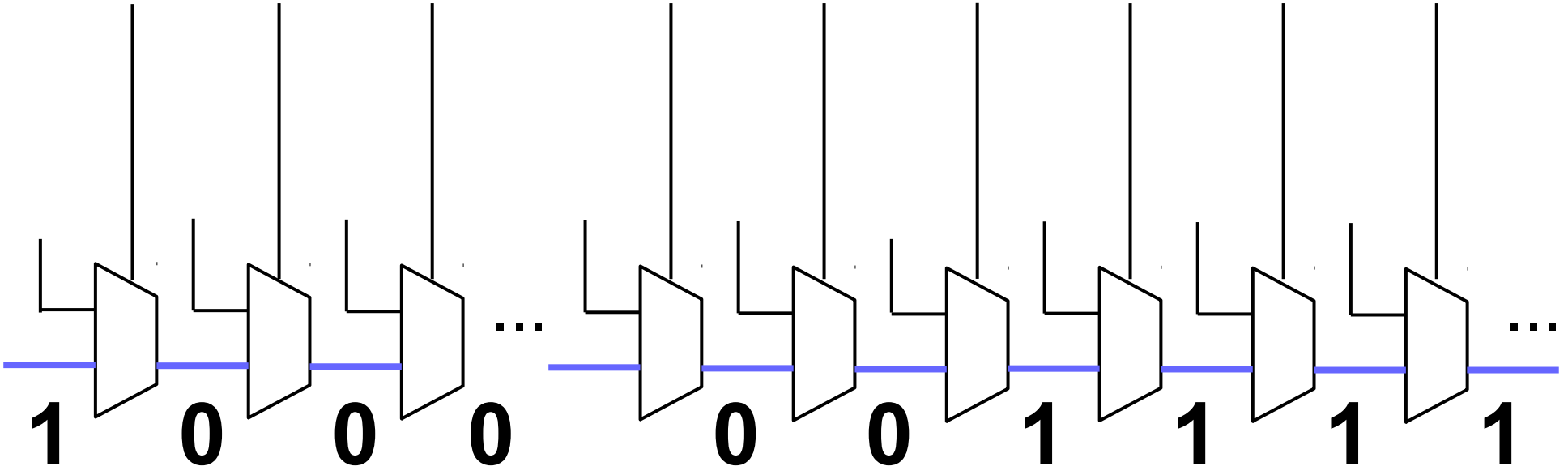
0→1 : 1 回

遷移の間隔は一定

hit 入力があると

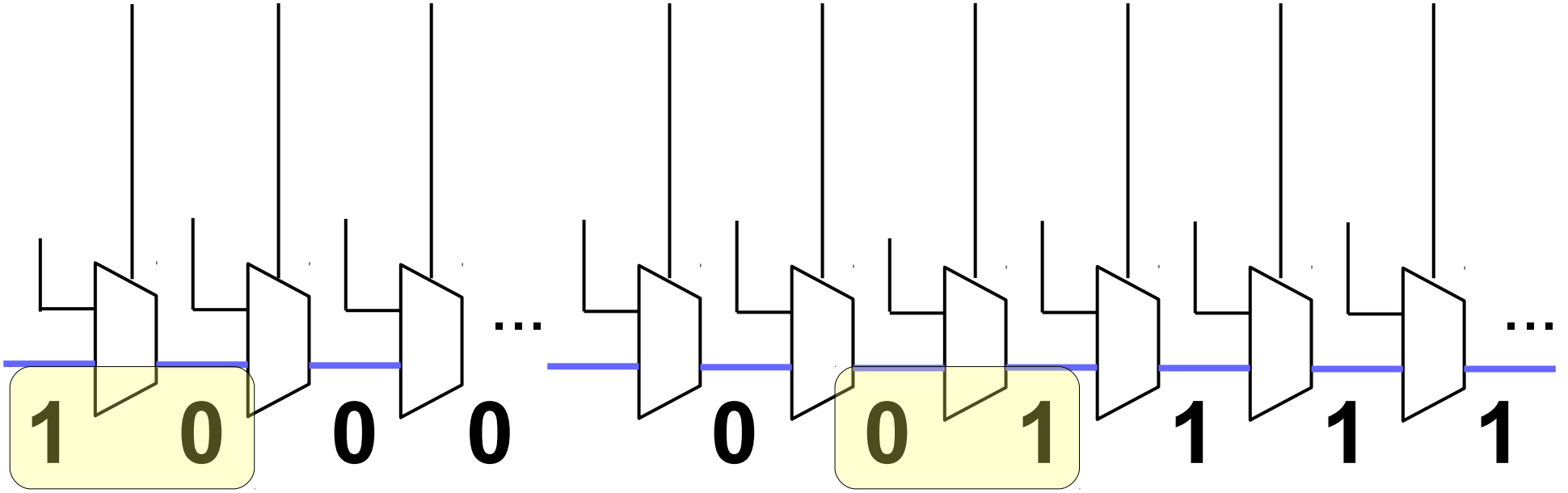
パルス発射スイッチが on

例：1→0→1 のパルスの launcher



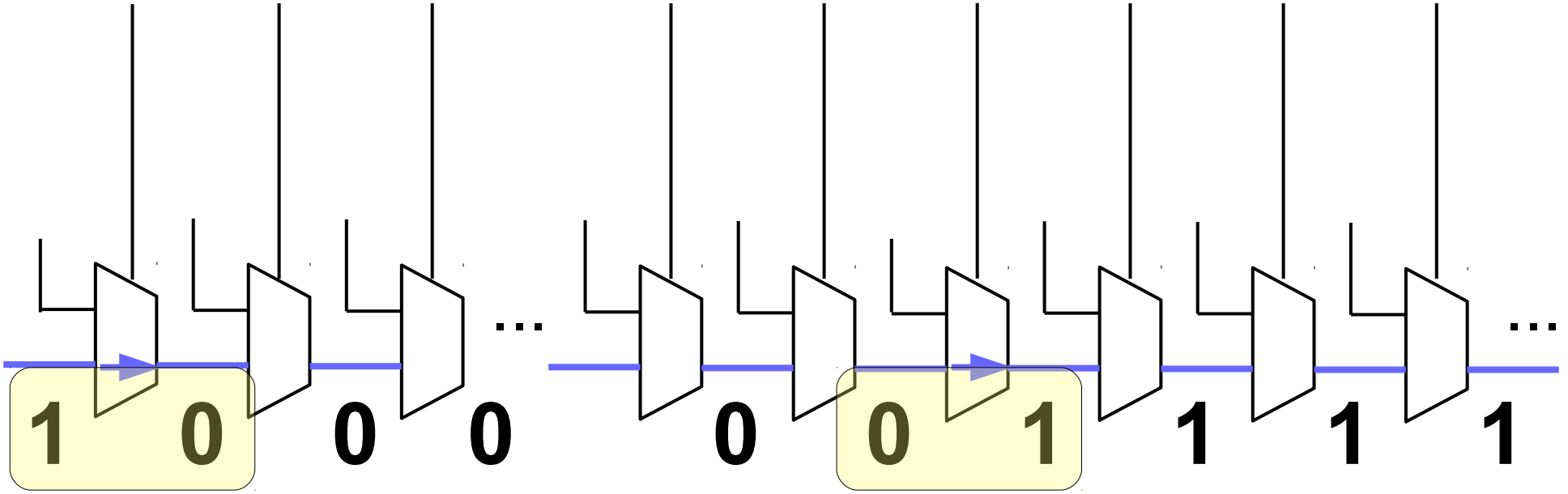
作りたいパルスの形

例：1→0→1 のパルスの launcher



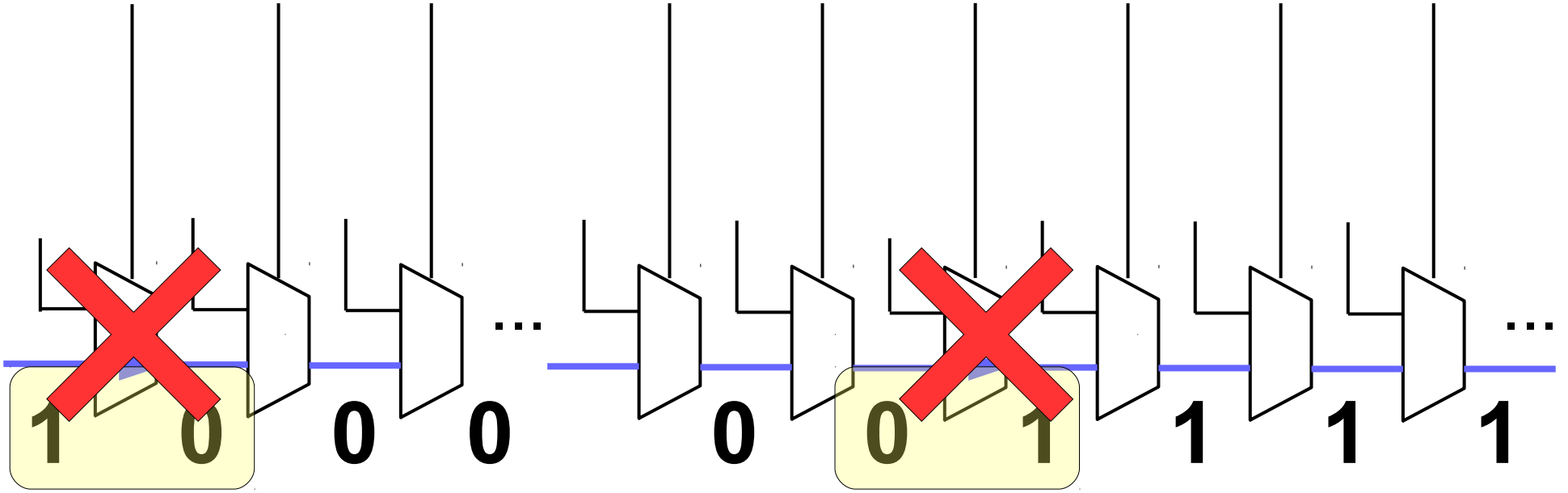
作りたいパルスの形

例：1→0→1 のパルスの launcher



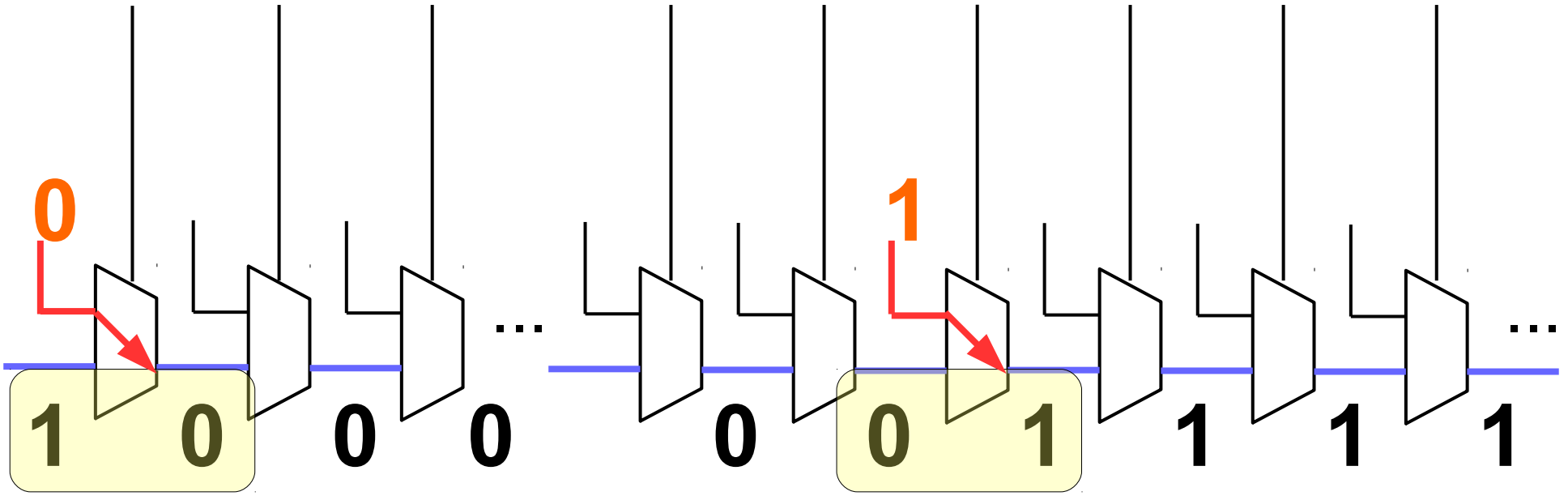
作りたいパルスの形

例：1→0→1 のパルスの launcher



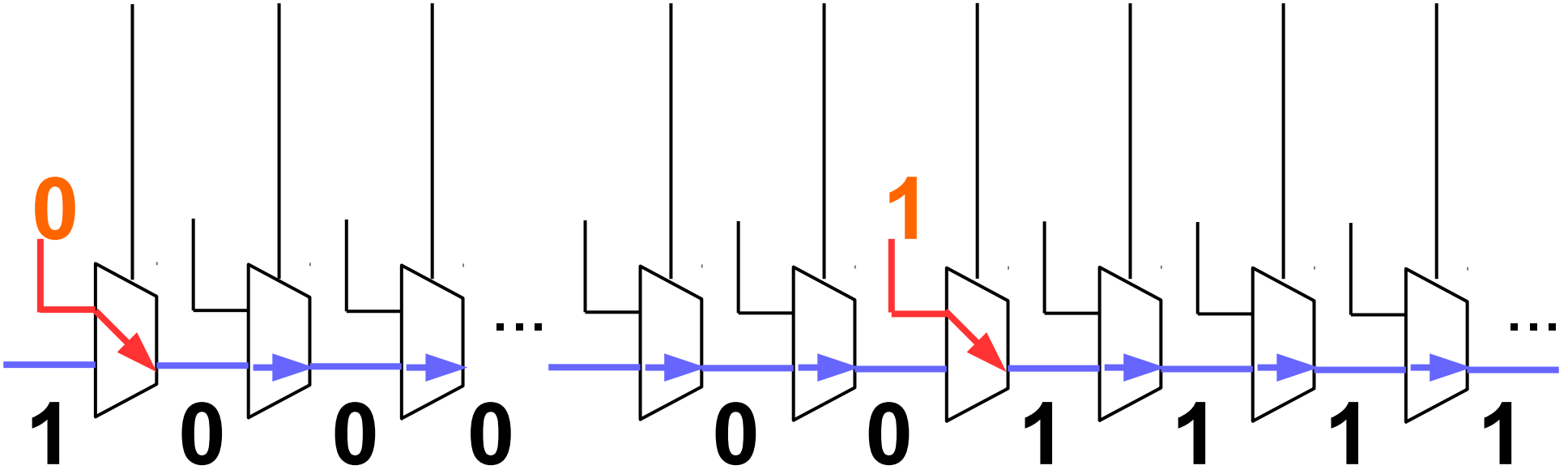
作りたいパルスの形

例：1→0→1 のパルスの launcher



作りたいパルスの形

例：1→0→1 のパルスの launcher

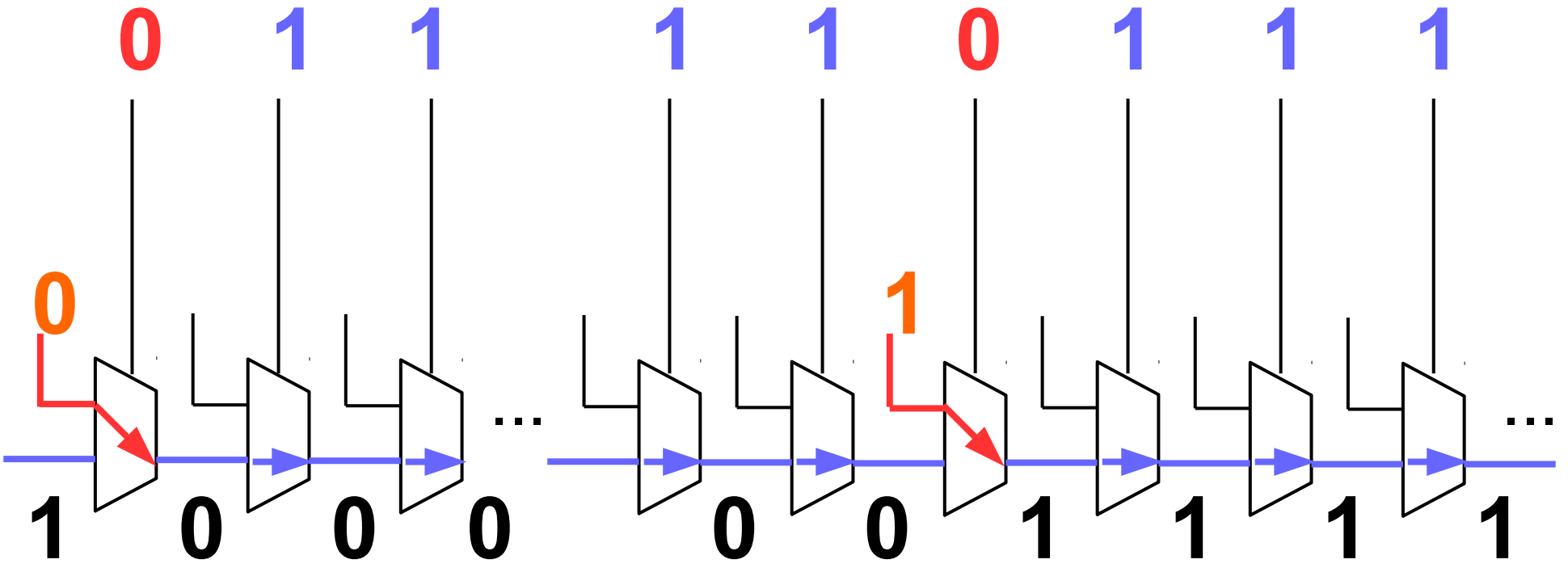


作りたいパルスの形

例：1→0→1 のパルスの launcher

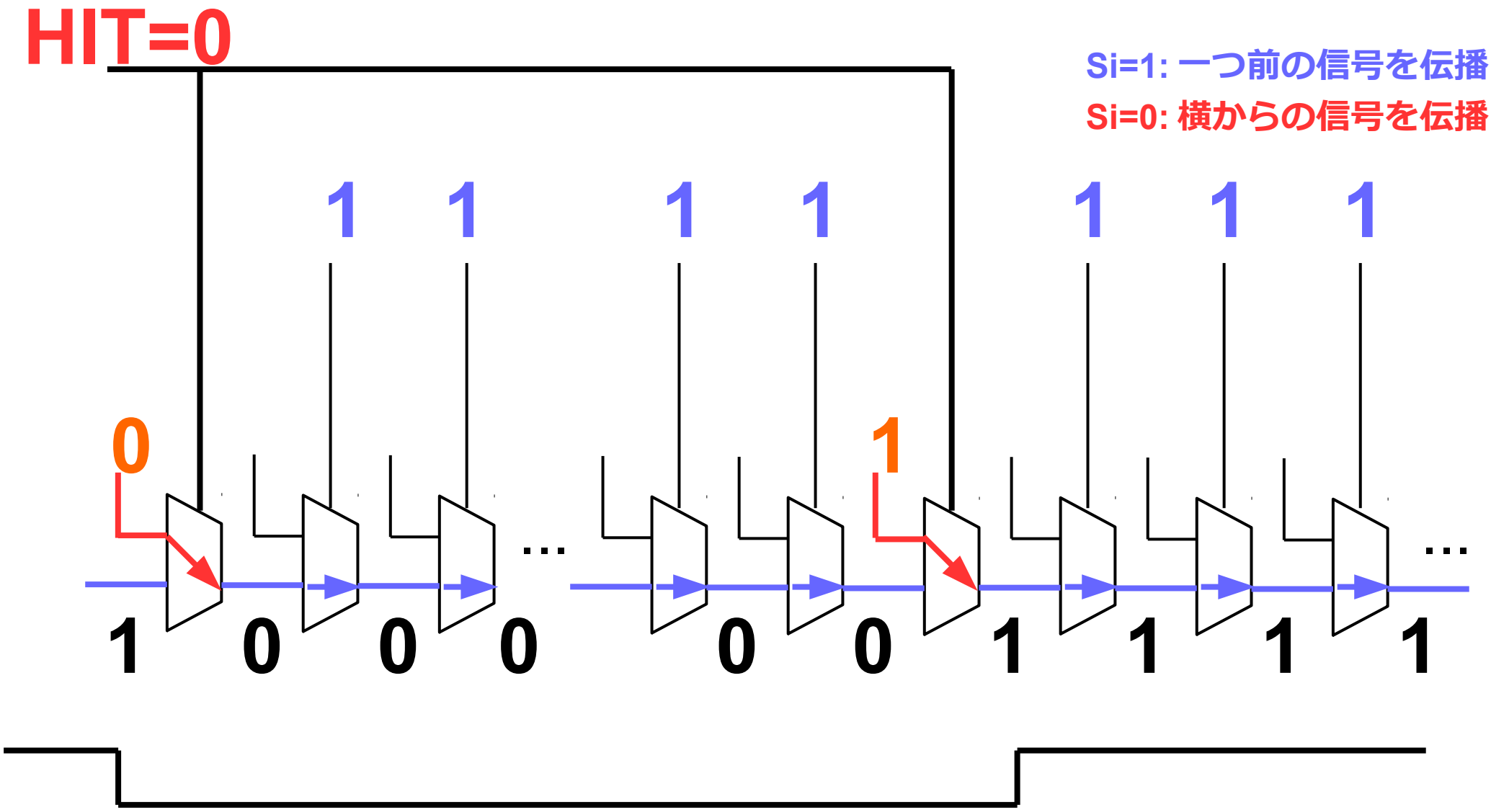
Si=1: 一つ前の信号を伝播

Si=0: 横からの信号を伝播

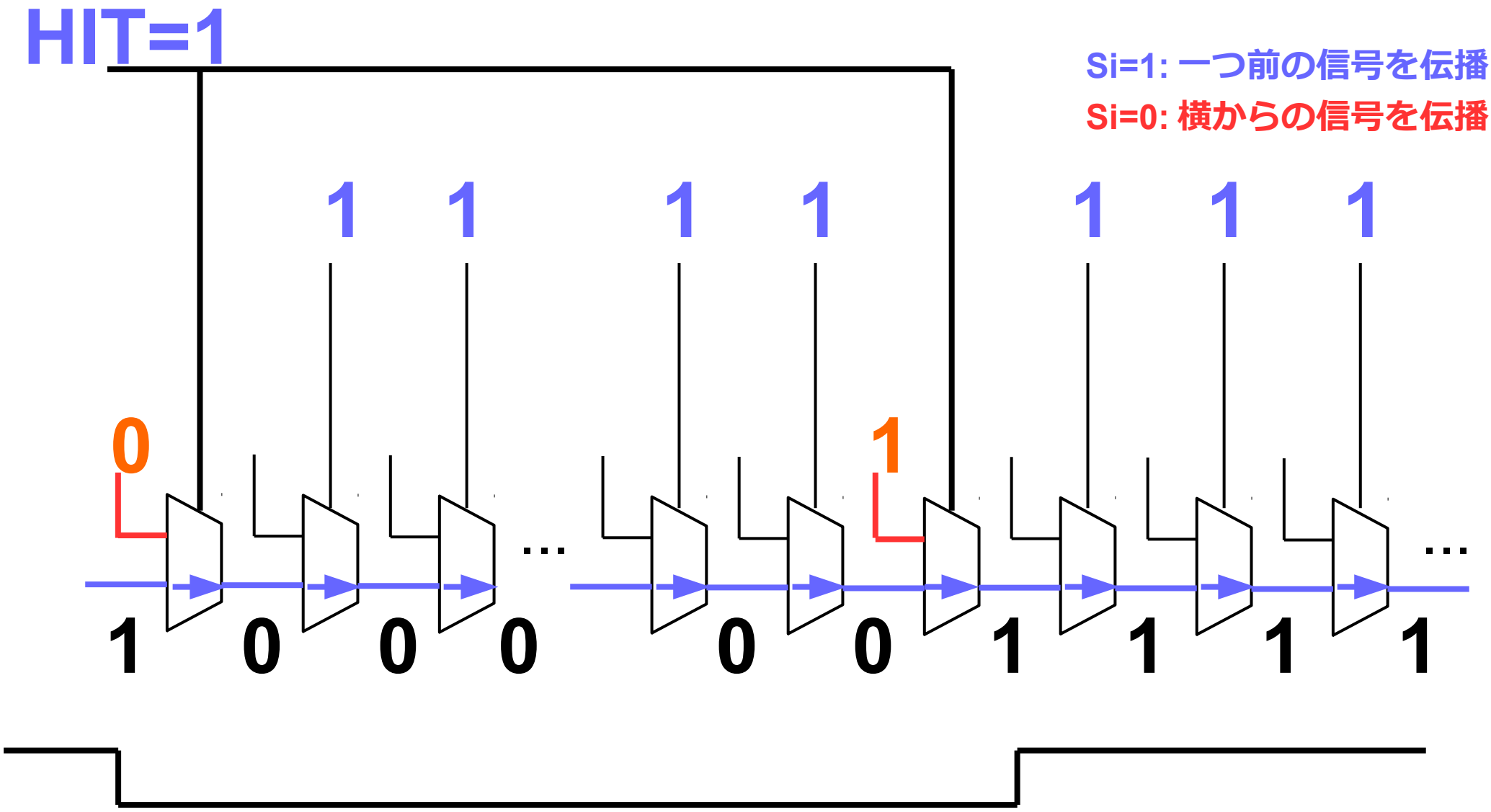


作りたいパルスの形

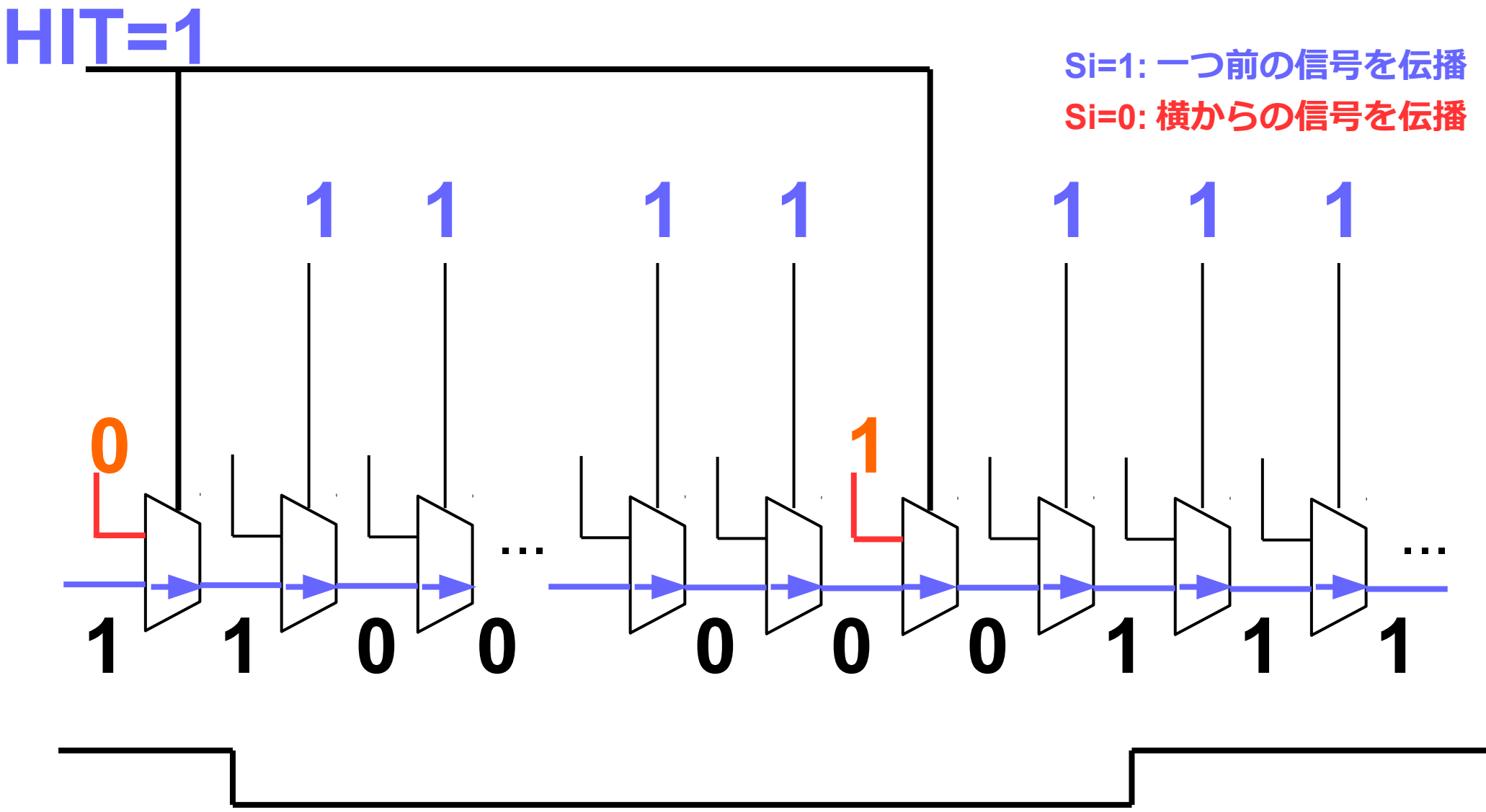
例：1→0→1 のパルスの launcher



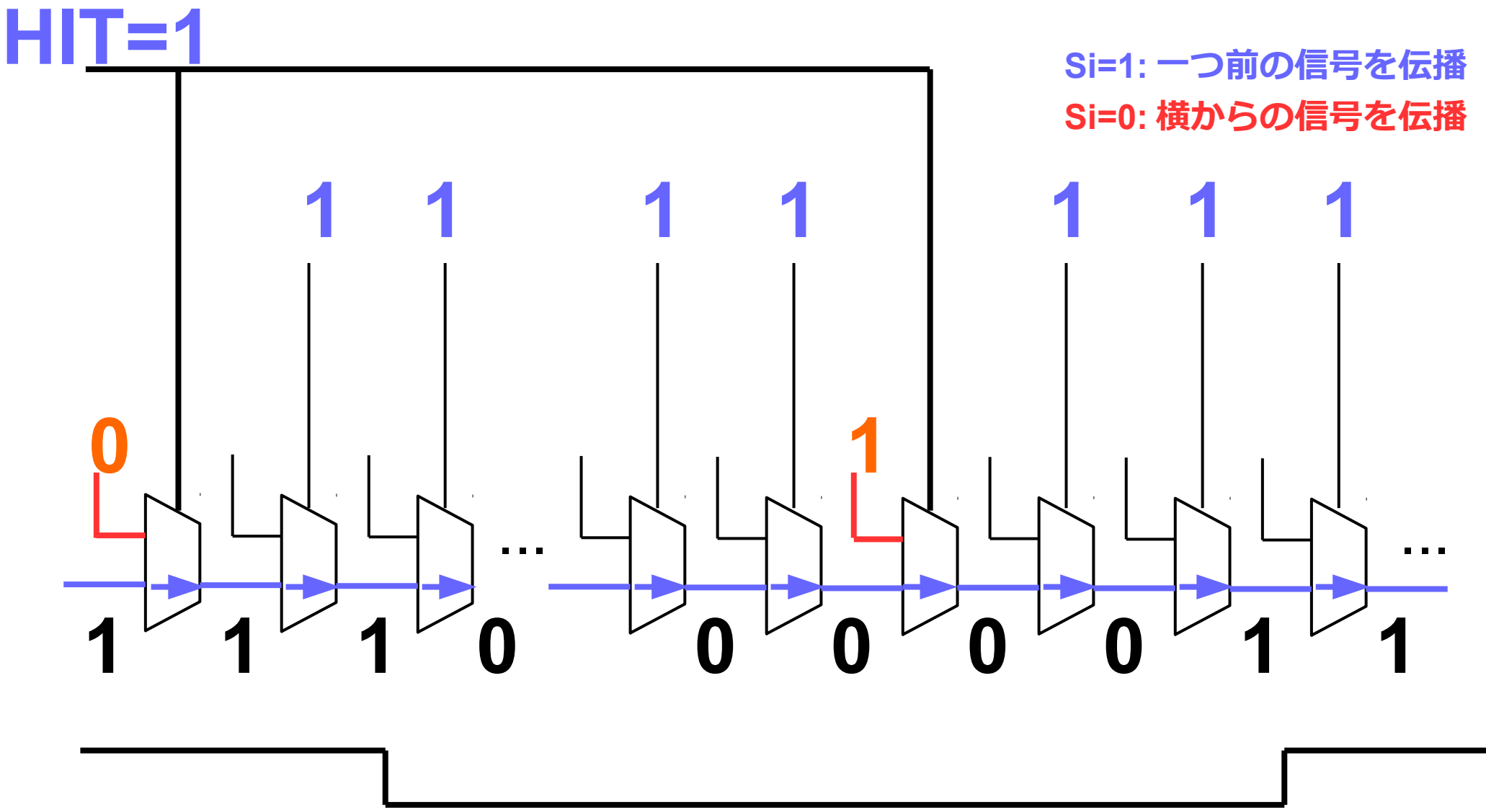
例：1→0→1 のパルスの launcher



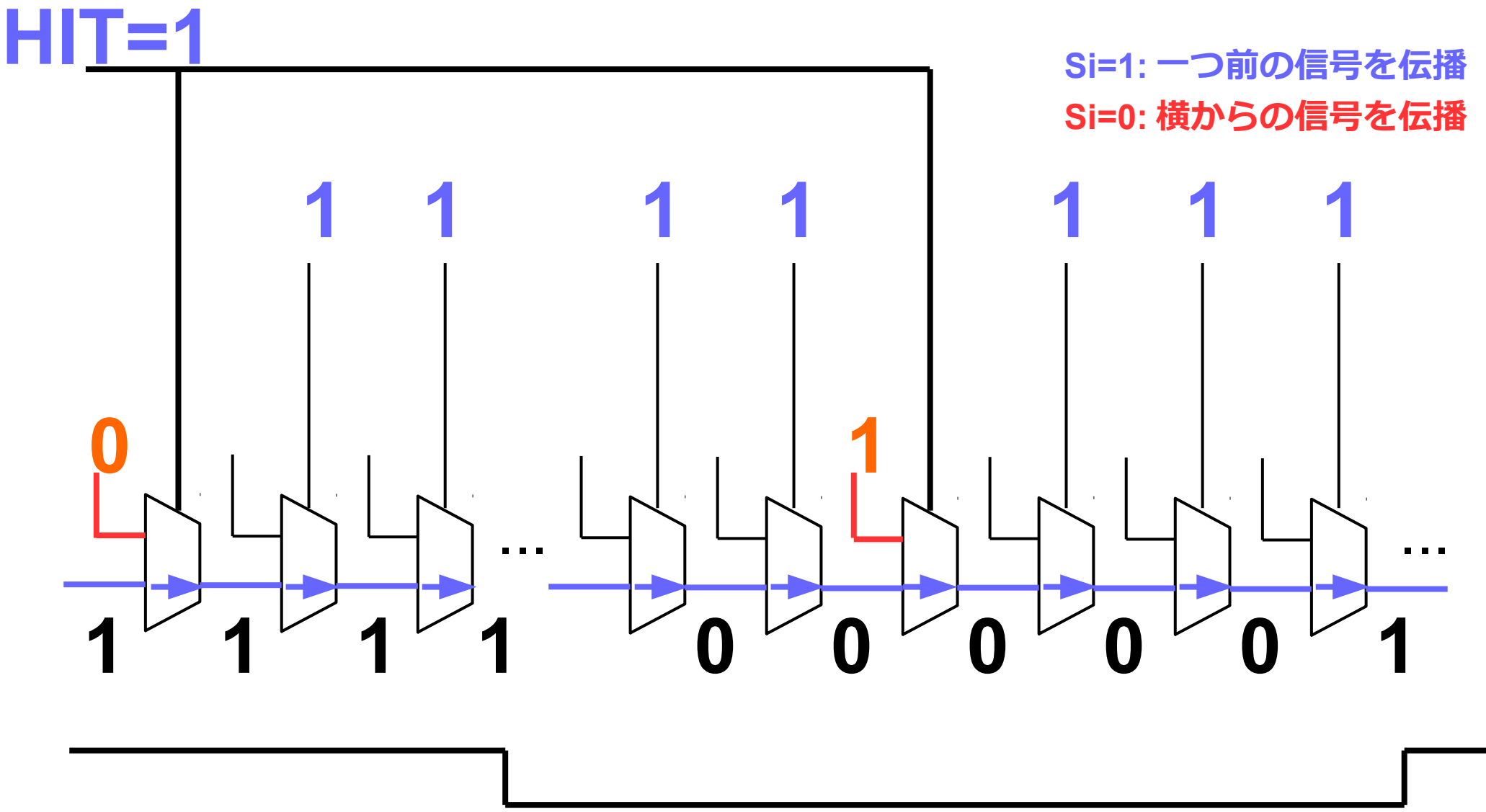
例：1→0→1 のパルスの launcher



例：1→0→1 のパルスの launcher



例：1→0→1 のパルスの launcher

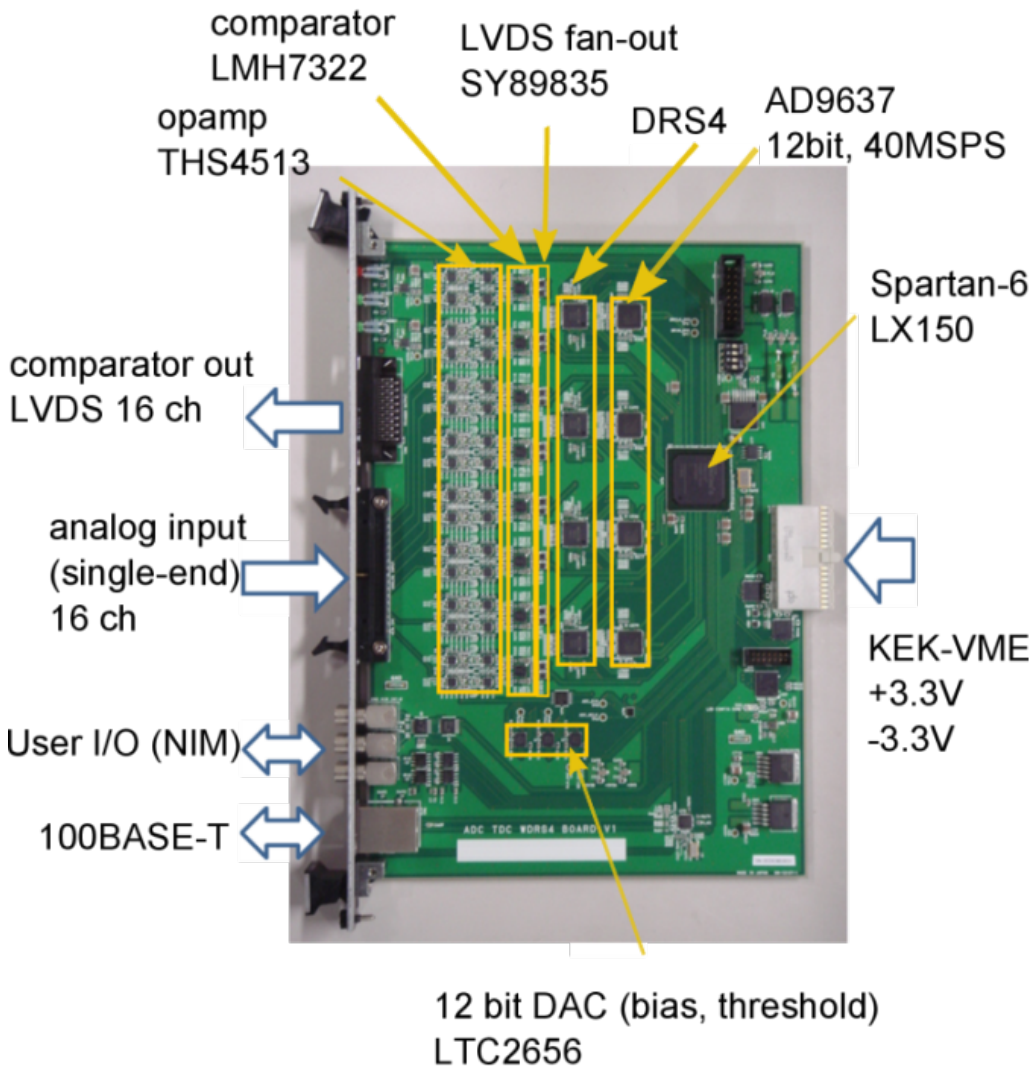


ここから実機の話

Xilinx Spartan6 への実装

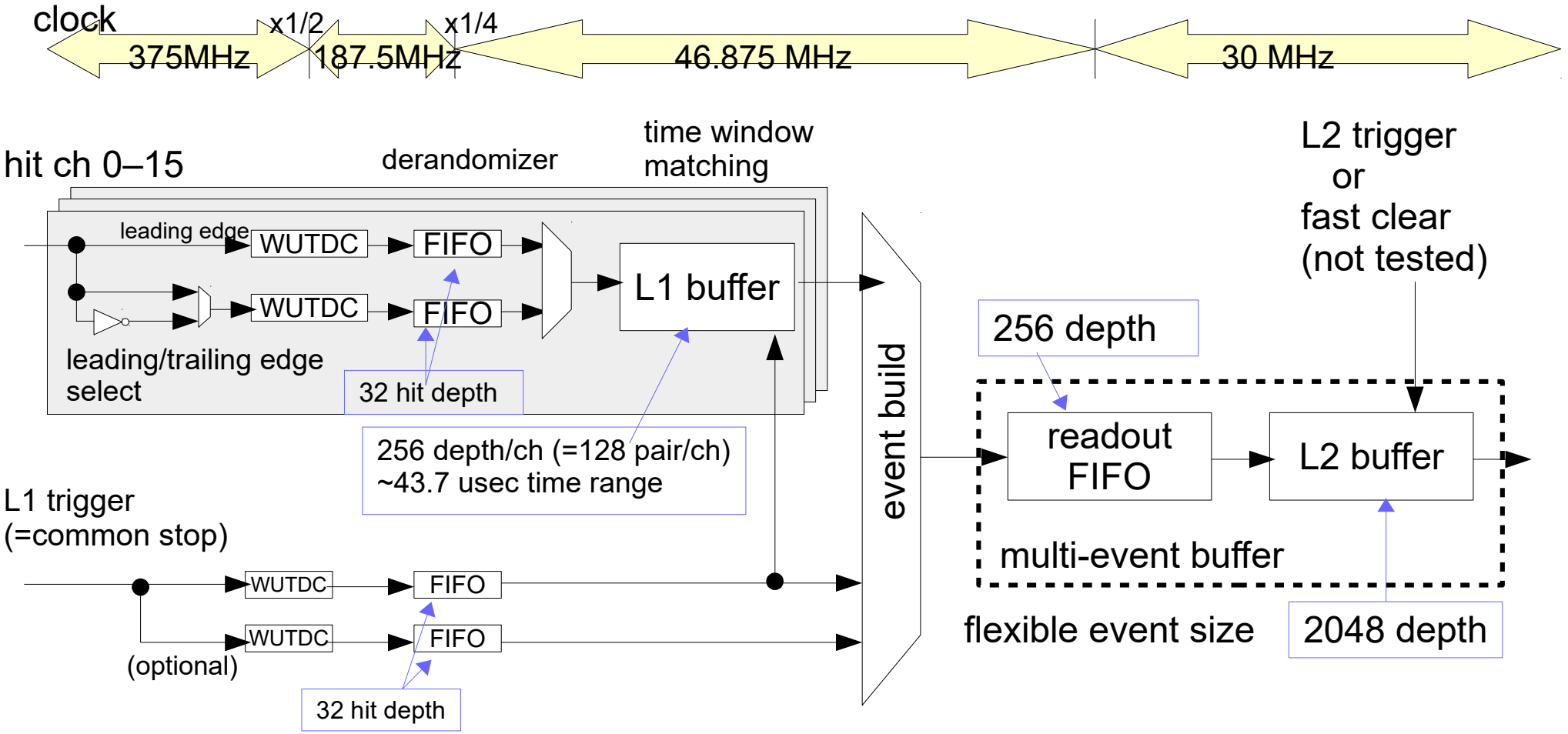
使用した基板

- DRS4QDC (KEKVME 6U)
- **XC6SLX150-2FFG484C**
  - ISE14.7
- **16 ch シングルエンド入力**
- on-board comparator
  - **LVDS→FPGA へ**
- DRS4 の波形取得ロジックと Wave Union TDC が同居
  - **(16+1)x2 delay line**



東北大 本多さん設計

TDC 部分のブロック図

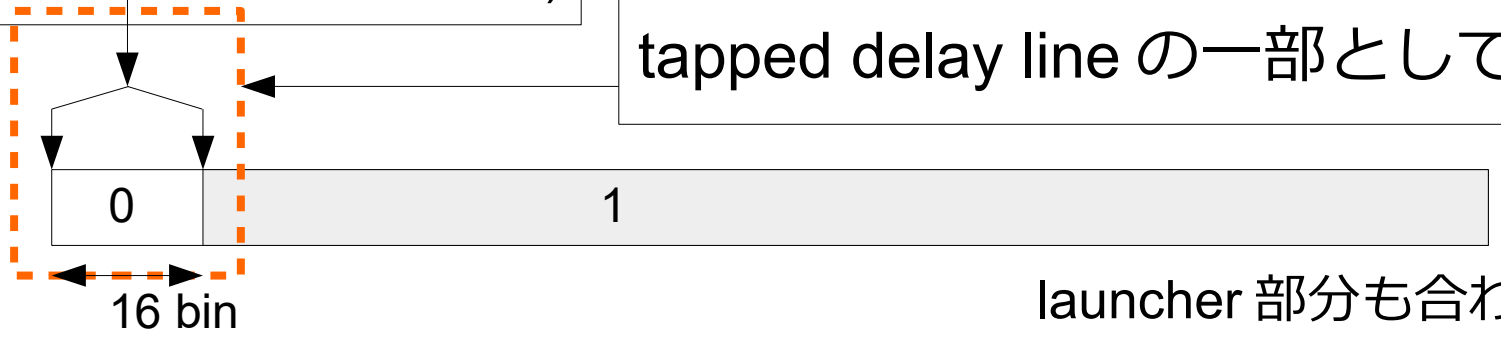


- 設計上の TDC 測定部の dead time は trailing edge 検出後、非同期ラッチクリアまでの +1 cycle (2.66ns/cycle) → 実測はまだ
- 他の部分はパイプライン処理
- TDC のデータは DRS4 の ADC データとマージされて SiTCP へ

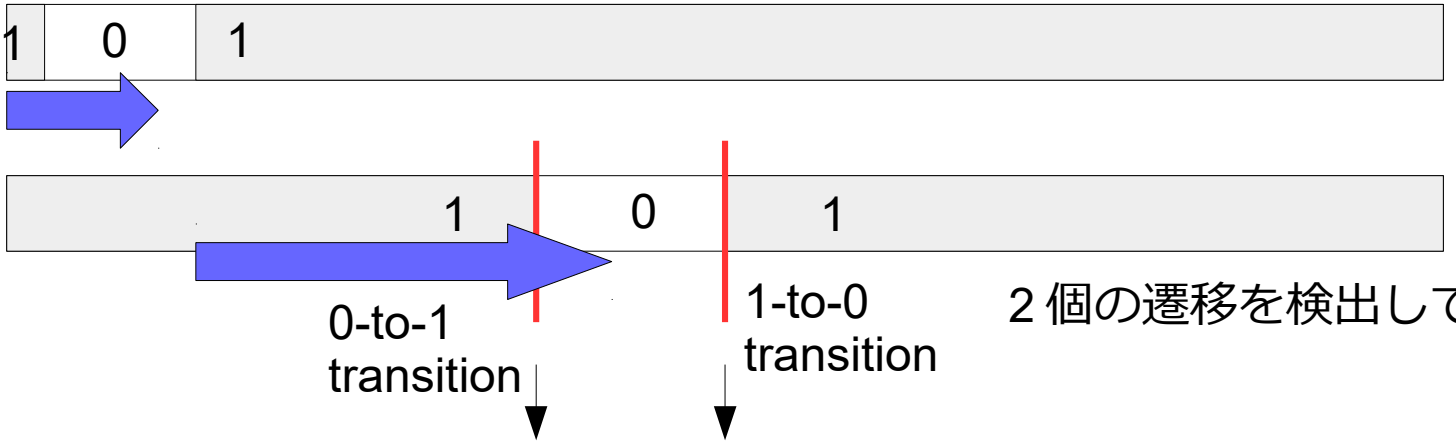
実装した Wave Union TDC

hit 入力
(非同期ラッチを通した信号)

wave union launcher A
実際には launcher 部分も
tapped delay line の一部として使用



launcher 部分も合わせて 200 bin

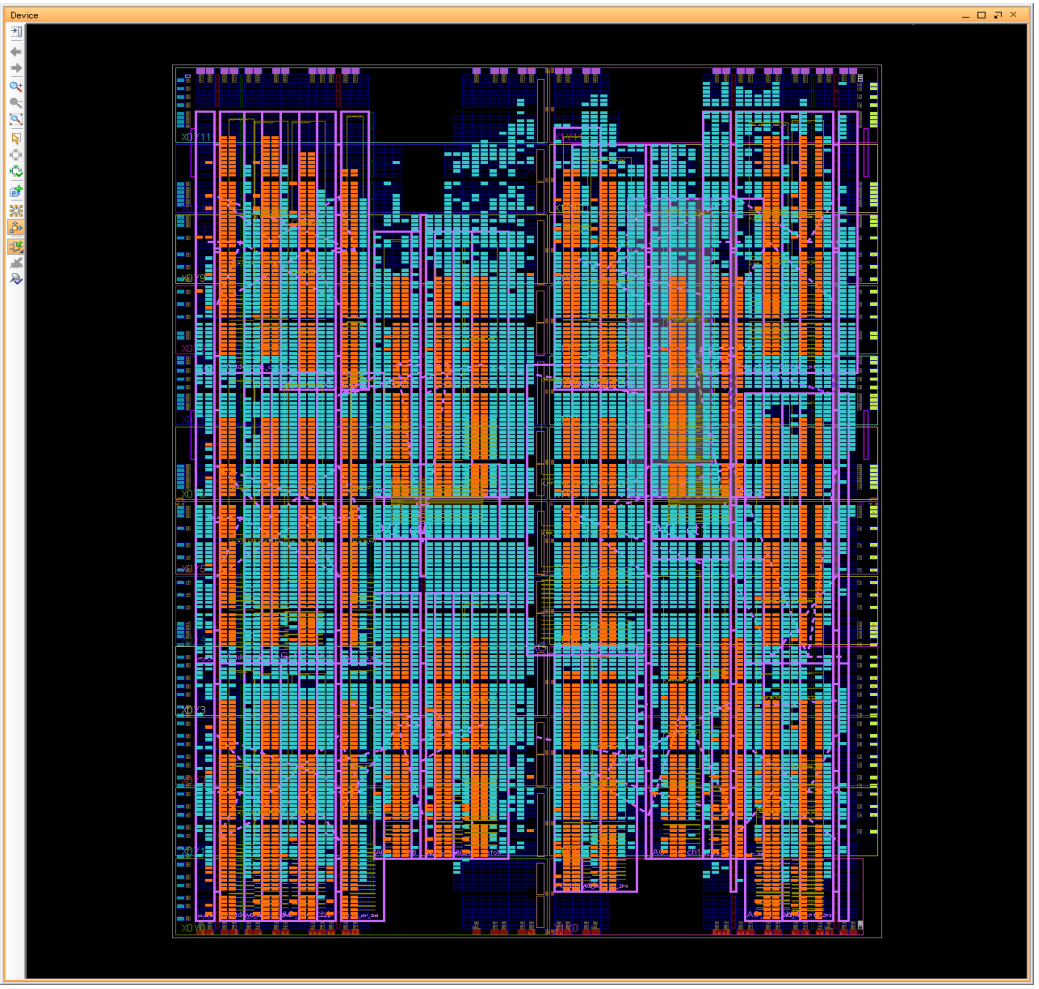


2 個の遷移を検出して 2 進数に変換

virtual bin ID = (bin ID A) + (bin ID B)

virtual bin ID に対して
bin width の calibration table
を作成

配置制約・ floor planning

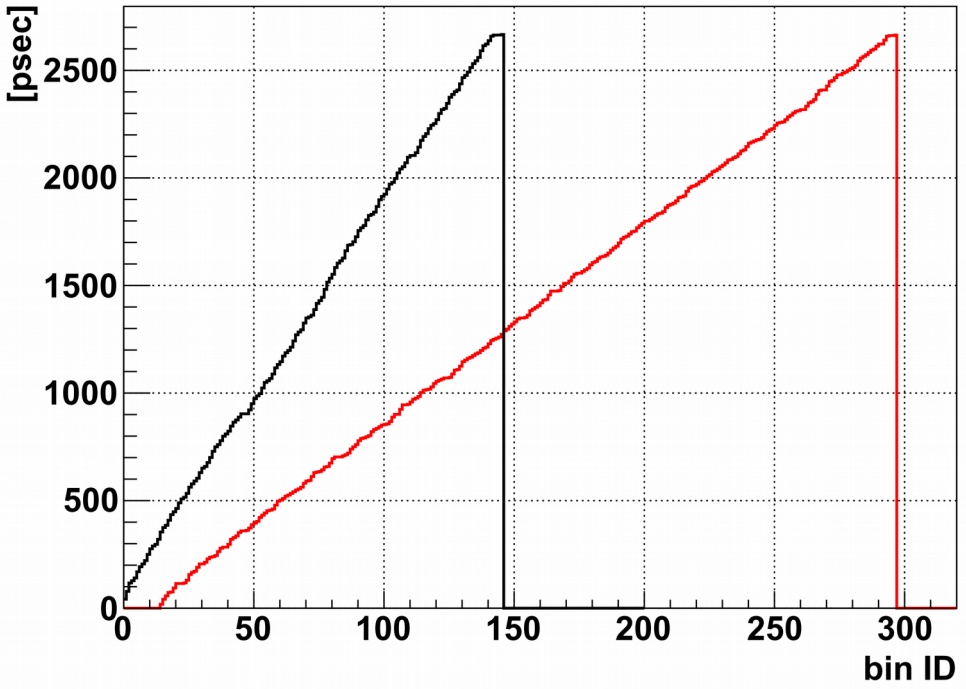


- delay line，遷移エッジ検出部の位置は固定
  - ***LOC*** 制約（左図のオレンジ色部分）
- channel FIFO に入る前の高速クロック (≧187.5 MHz) を使う部分
  - ***AREA_GROUP*** (Pblock) 制約
- タイミングクリティカルな配線では FF 挿入
- SLICE 使用率 ~ 80%
  - DRS4, ADC, SiTCP, etc. ~ 20%
  - TDC ~60%
- 合成・実装・ file 生成 ~ 1 hour
  - Xilinx ISE 14.7 (win7 pro 64bit)
  - core-i7 3770K 4.4GHz (OC)

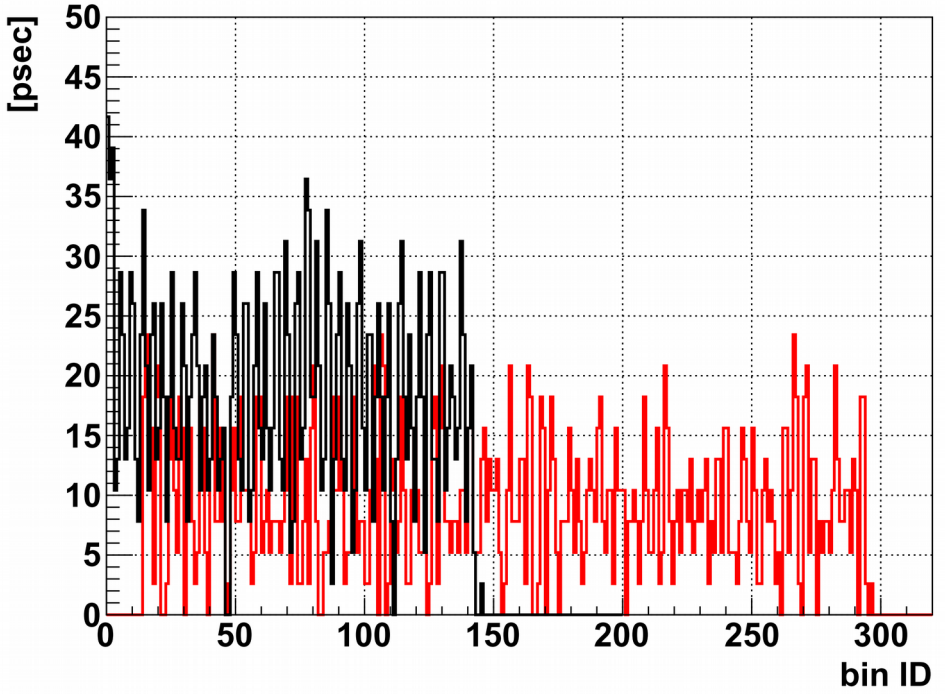
calibration table, bin width 分布

black: w/o wave union
red: w/ wave union

calibration table



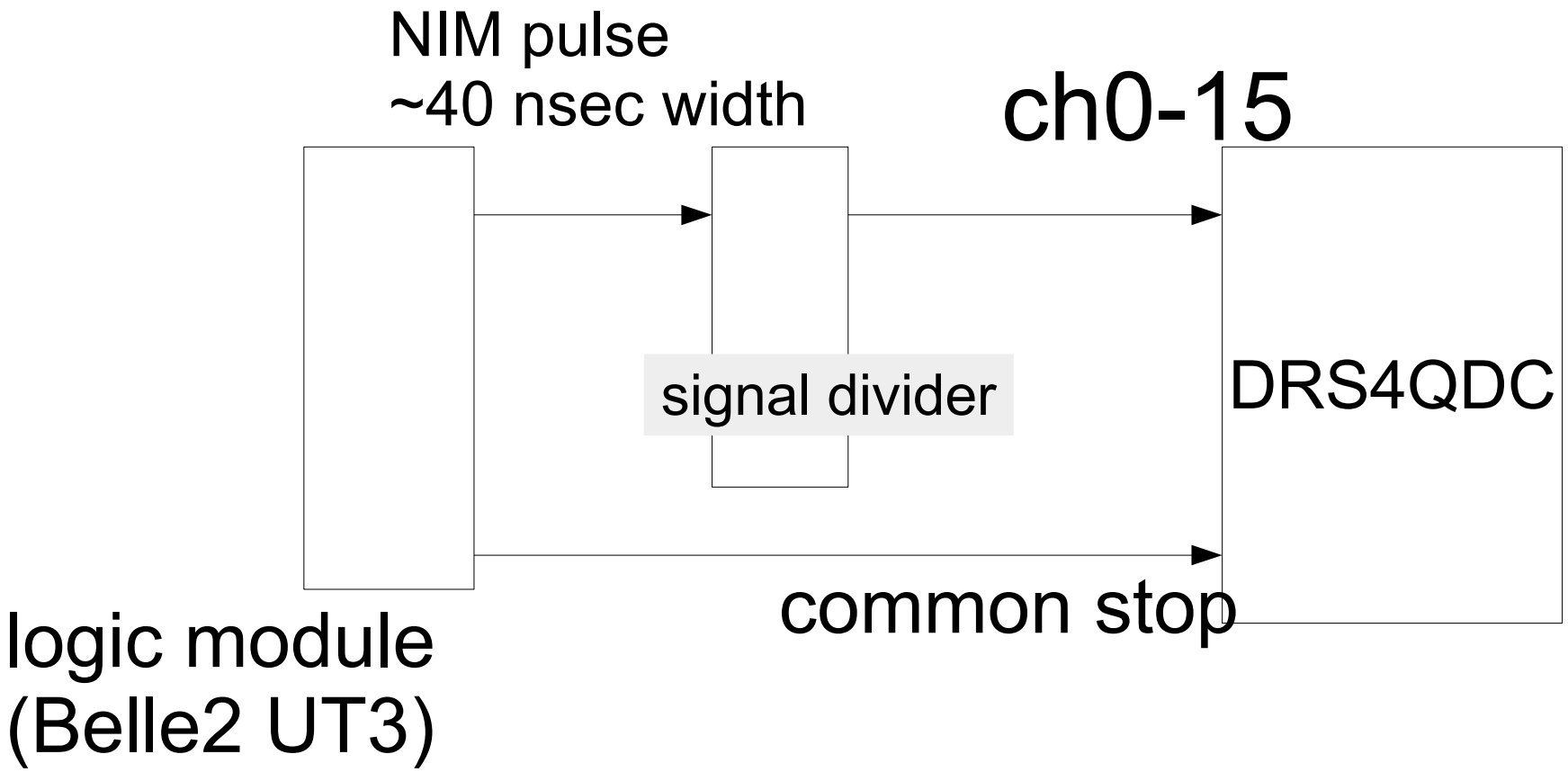
bin width



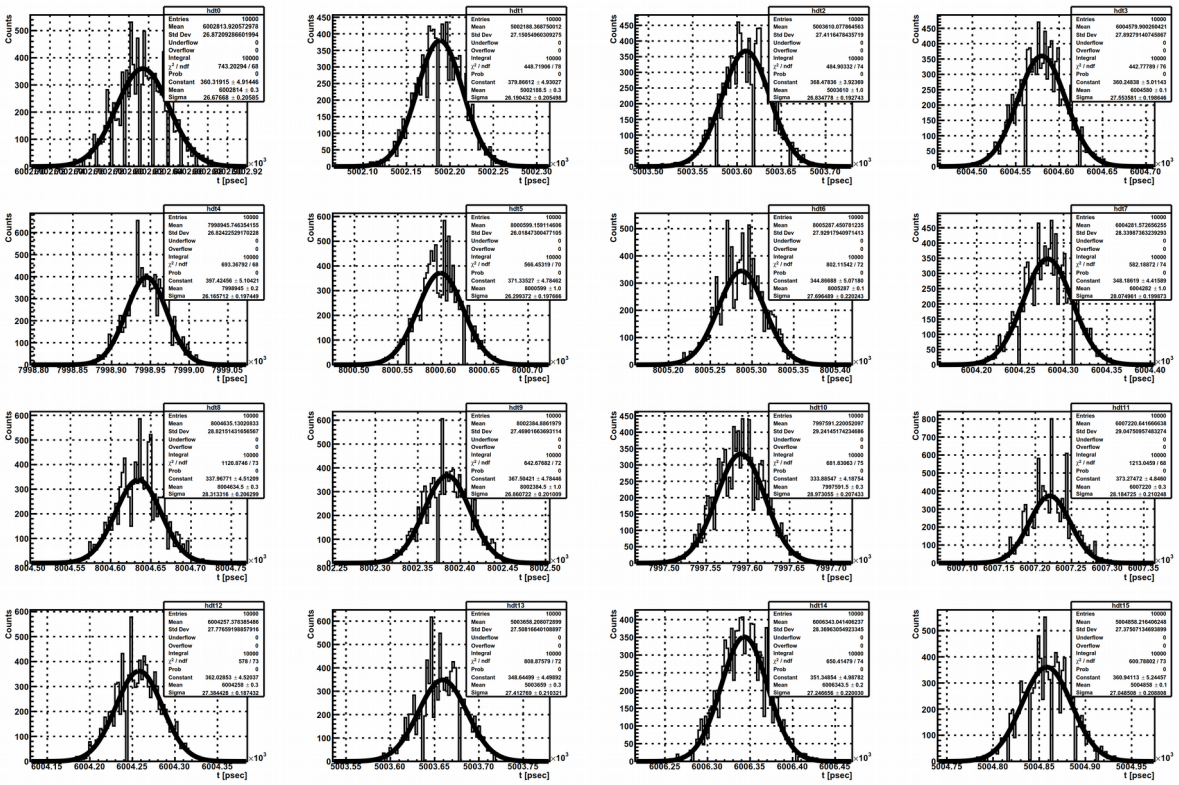
適用前　　適用後

~20 ps/bin → ~10 ps/bin

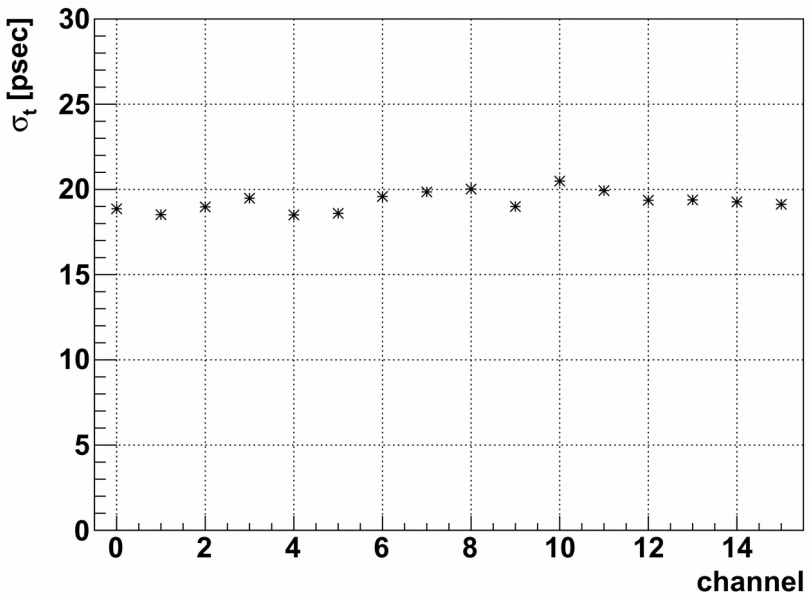
テストベンチでの分解能測定



leading edge の測定

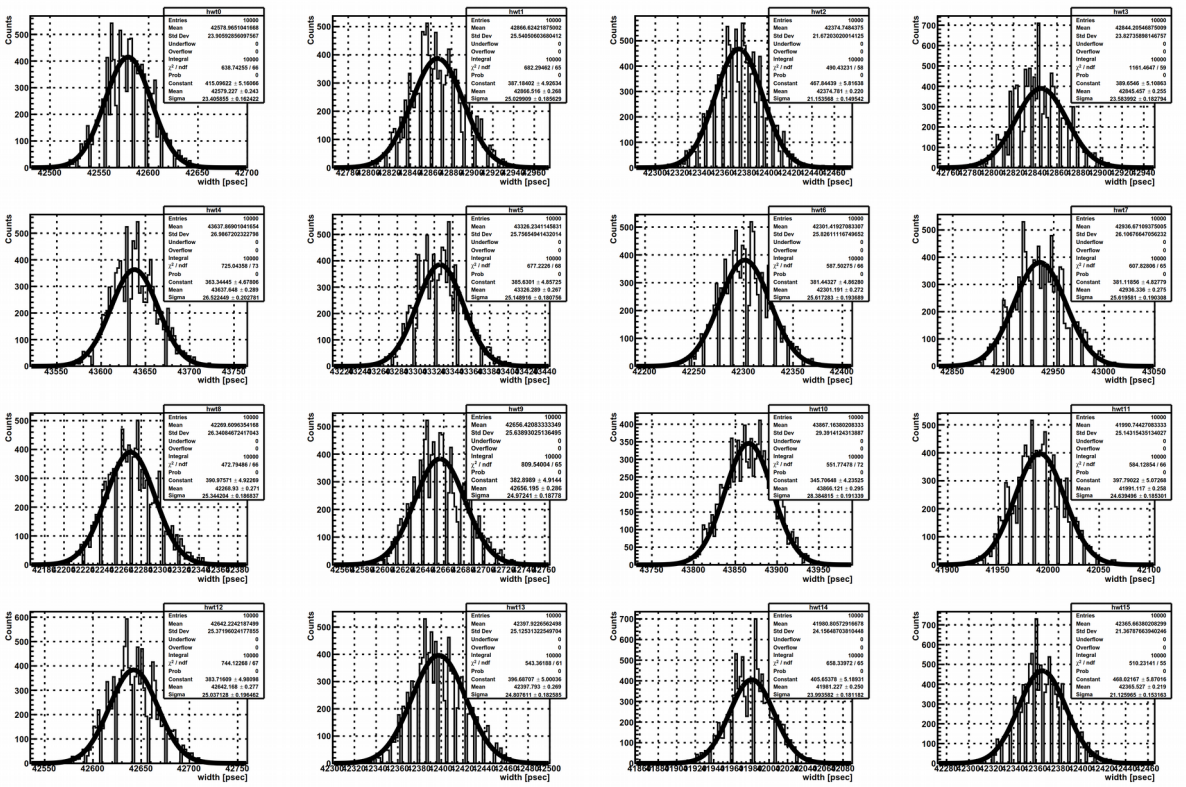


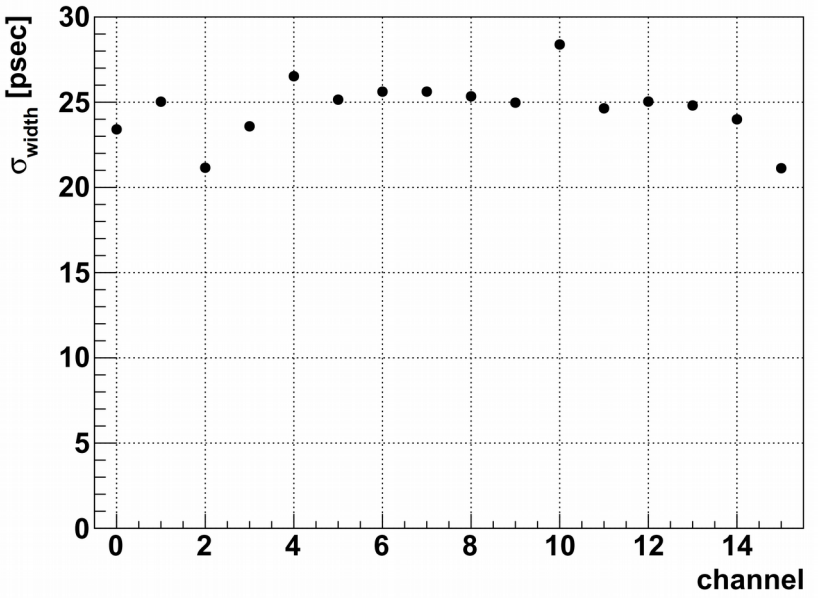
single channel resolution
(= σ/sqrt(2))



~20 ps

pulse width の分解能

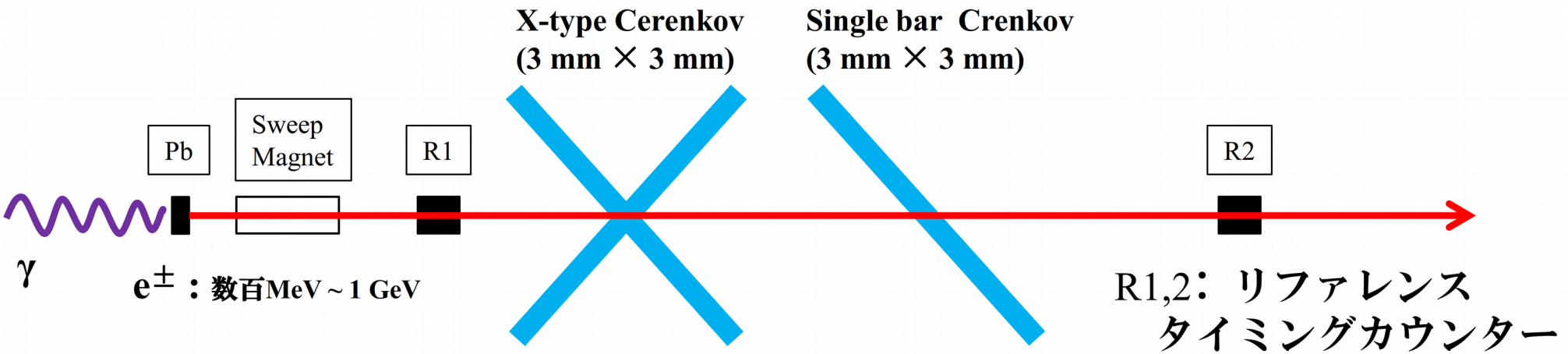




< 30 ps

検出器と接続して性能測定

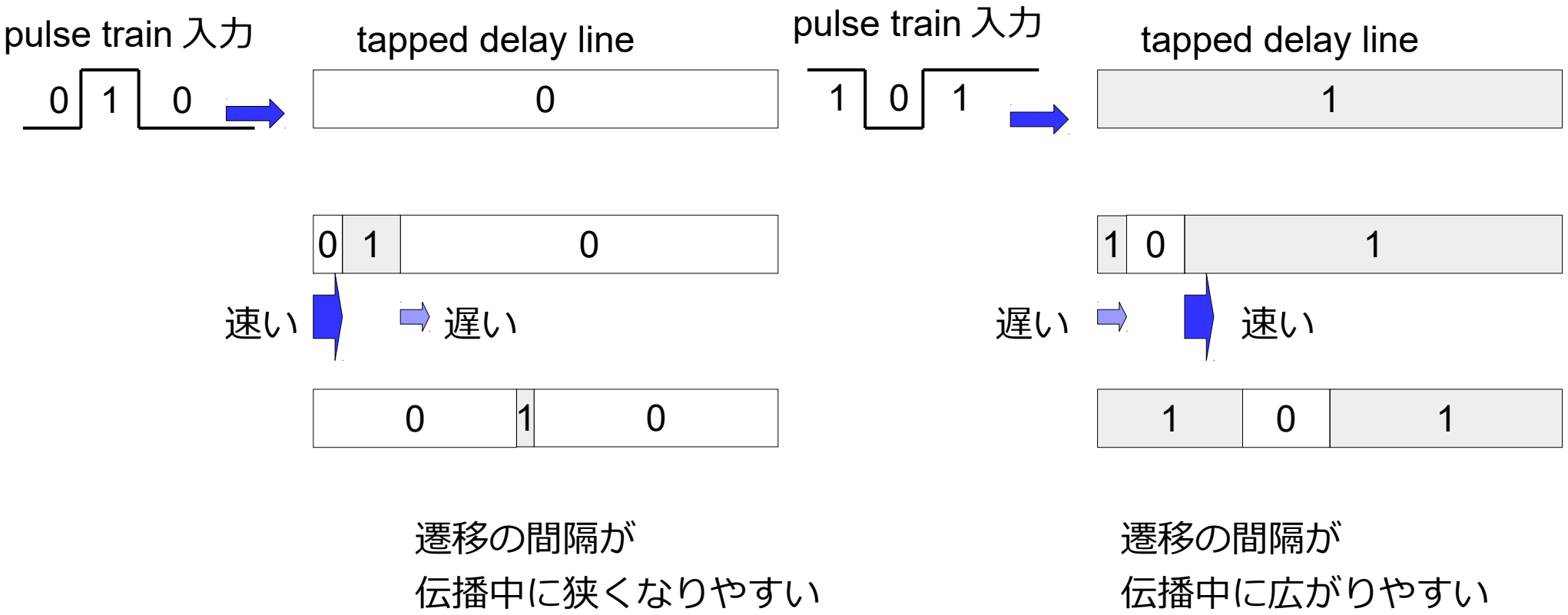
- 赤石 （阪大理）JPS 2017/09 17pK24-13



- テスト実験 @ LEPS2
- アクリルチェレンコフ輻射体 + MPPC
- Wave Union TDC
- DRS4 で取った ADC で slewing correction
- 検出器込みでの分解能 < 80 ps
  - FPGA TDC の分解能は CAEN の TDC V775 (分解能 35 ps) と同程度

設計上の注意点

- 1→0 と 0→1 で伝播速度が違う (~10%)



設計上の注意点

- 固定パルスの遷移間隔の tuning
  - 遷移間隔が狭すぎる
    - delay line 伝播中にくっついて遷移が消滅しまう可能性
  - 遷移間隔が広すぎる
    - delay line の長さが足りなくなる
- 高速クロックで動かす部分のデバッグは ChipScope が使えない
  - tapped delay line，エンコーダのデータ (の一部)
    - これらは BRAM の動作クロックより速いため

今後の展望

基板を変える

- たぶん FPGA ロジックに問題はない
- FPGA 入力についている部品 (opamp, comparator) ・もしくは基板設計を変更する
- ついでに FPGA も 7 series を試したい

データシート上での CARRY4 の伝播遅延

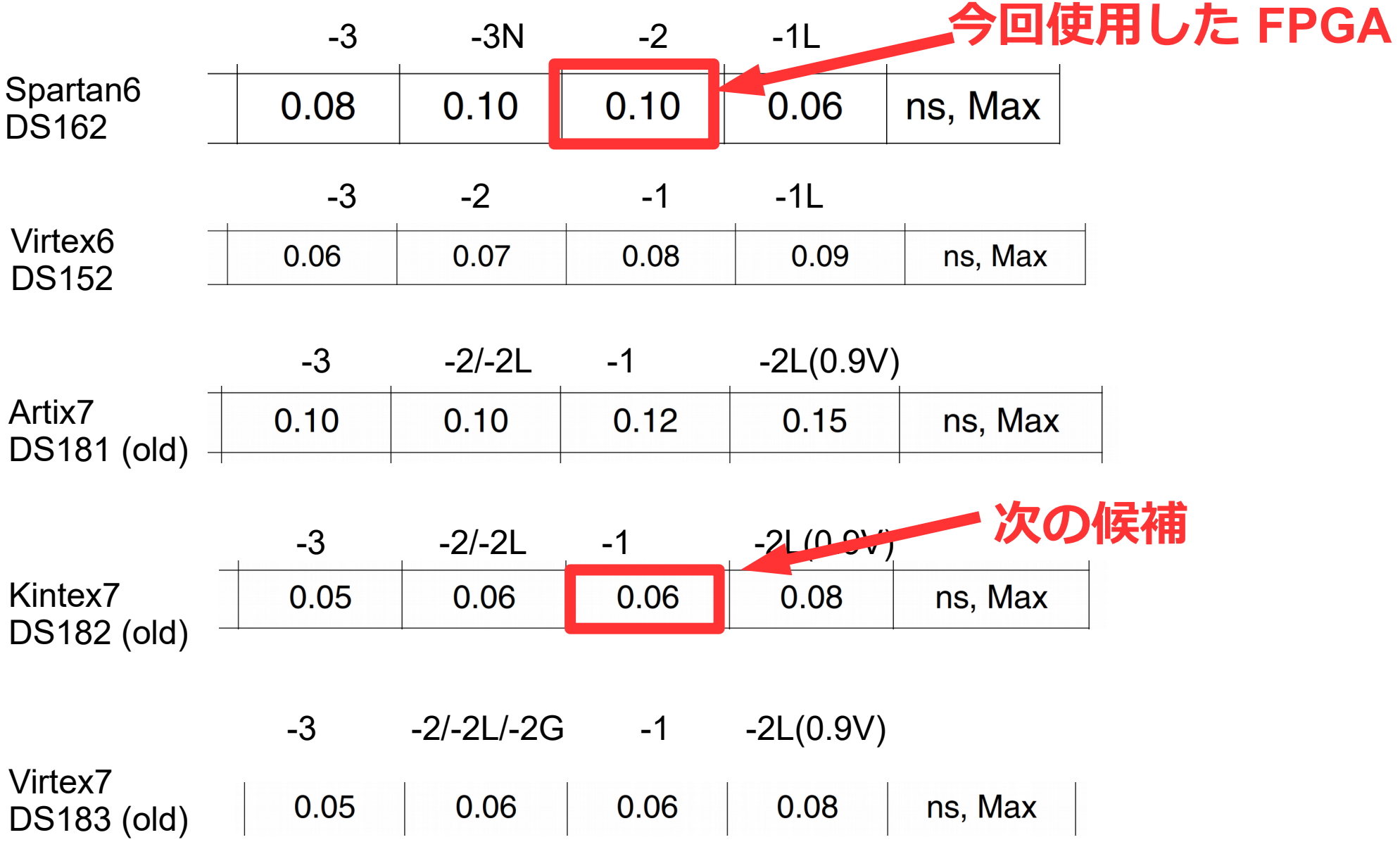
CLB Switching Characteristics (SLICEM Only)

Table 40: **CLB Switching Characteristics (SLICEM Only)**

Symbol	Description	Speed Grade				Units
		-3	-3N	-2	-1L	
Combinatorial Delays						
T_{ILO}	An – Dn LUT inputs to A to D outputs	0.21	0.26	0.26	0.46	ns, Max
	An – Dn LUT inputs through F7AMUX/F7BMUX to AMUX/CMUX output	0.37	0.43	0.43	0.77	ns, Max
T_{OPAB}	An – Dn LUT inputs through F7AMUX or F7BMUX and F8MUX to BMUX output	0.37	0.46	0.46	0.84	ns, Max
T_{ITO}	An – Dn LUT inputs through latch to AQ – DQ outputs	0.82	0.95	0.95	1.64	ns, Max
T_{TITO_LOGIC}	An – Dn LUT inputs to AQ – DQ outputs (latch as logic)	0.82	0.95	0.95	1.64	ns, Max
T_{OPCYA}	An LUT inputs to COUT output	0.38	0.48	0.48	0.69	ns, Max
T_{OPCYB}	Bn LUT inputs to COUT output	0.38	0.49	0.49	0.71	ns, Max
T_{OPCYC}	Cn LUT inputs to COUT output	0.28	0.33	0.33	0.55	ns, Max
T_{OPCYD}	Dn LUT inputs to COUT output	0.28	0.35	0.35	0.52	ns, Max
T_{AXCY}	AX input to COUT output	0.21	0.26	0.26	0.36	ns, Max
T_{BXCY}	BX input to COUT output	0.13	0.16	0.16	0.18	ns, Max
T_{CXCY}	CX input to COUT output	0.10	0.12	0.12	0.09	ns, Max
T_{DXCY}	DX input to COUT output	0.09	0.11	0.11	0.09	ns, Max
T_{BYP}	CIN input to COUT output	0.08	0.10	0.10	0.06	ns, Max
T_{CINA}	CIN input to AMUX output	0.21	0.22	0.22	0.47	ns, Max
T_{CINB}	CIN input to BMUX output	0.30	0.31	0.31	0.57	ns, Max
T_{CINC}	CIN input to CMUX output	0.29	0.31	0.31	0.58	ns, Max
T_{CIND}	CIN input to DMUX output	0.31	0.32	0.32	0.68	ns, Max

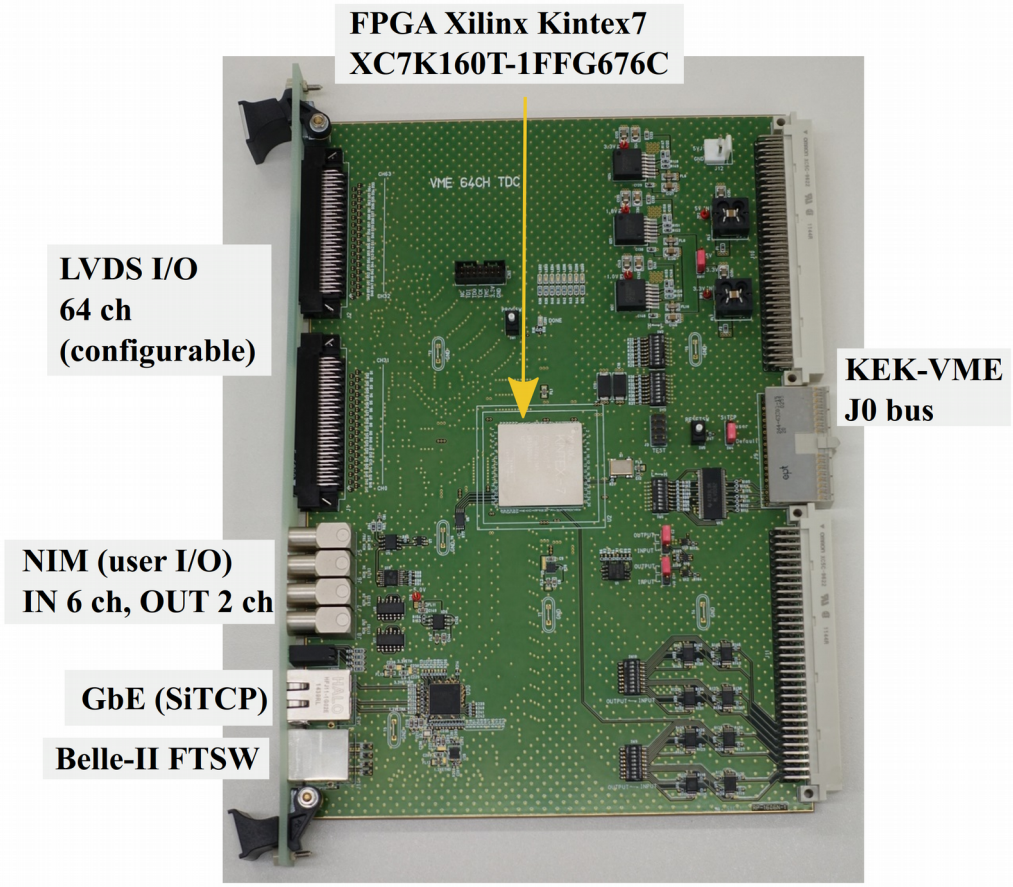
DS162

データシート上での CARRY4 の伝播遅延



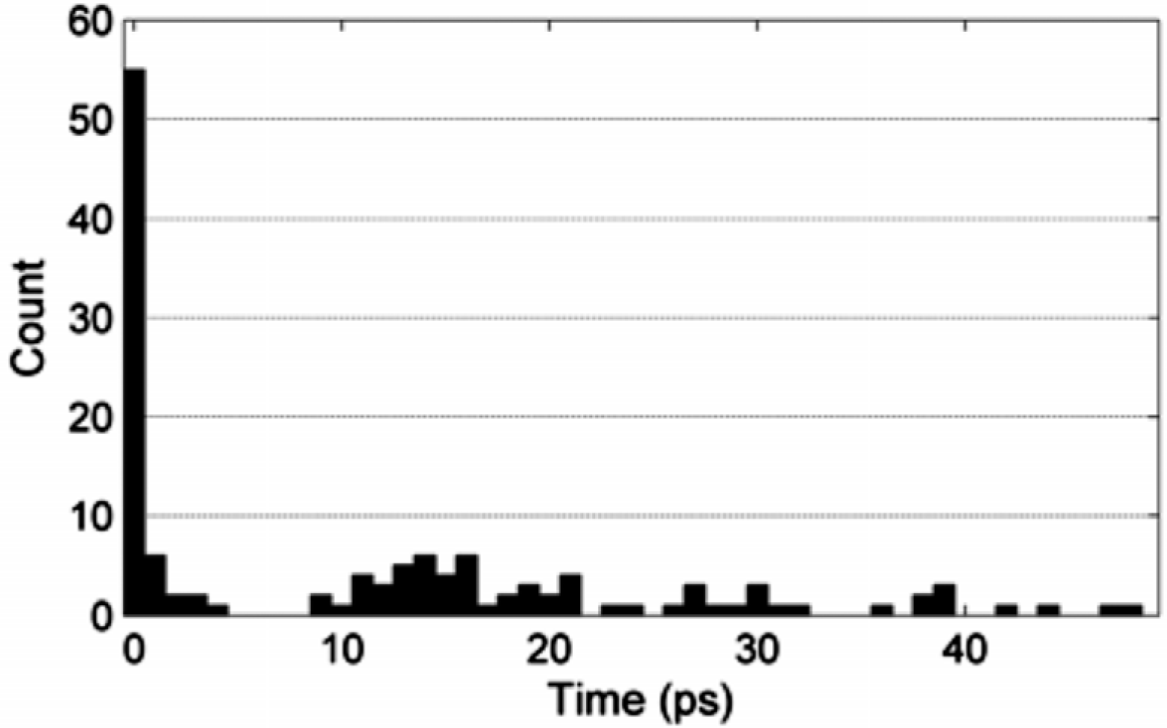
使用した FPGA モジュール

- RPV-260 (林栄精器)
- 本来の用途は LEPS2 の Drift Chamber 用 64 ch TDC (LSB=1 ns)
- **FPGA: Kintex7-160T-1**
- LVDS は FPGA に直結
  - 保護ダイオードのみ
- これに high resolution TDC を実装してみる



7 series でのロジック実装の問題

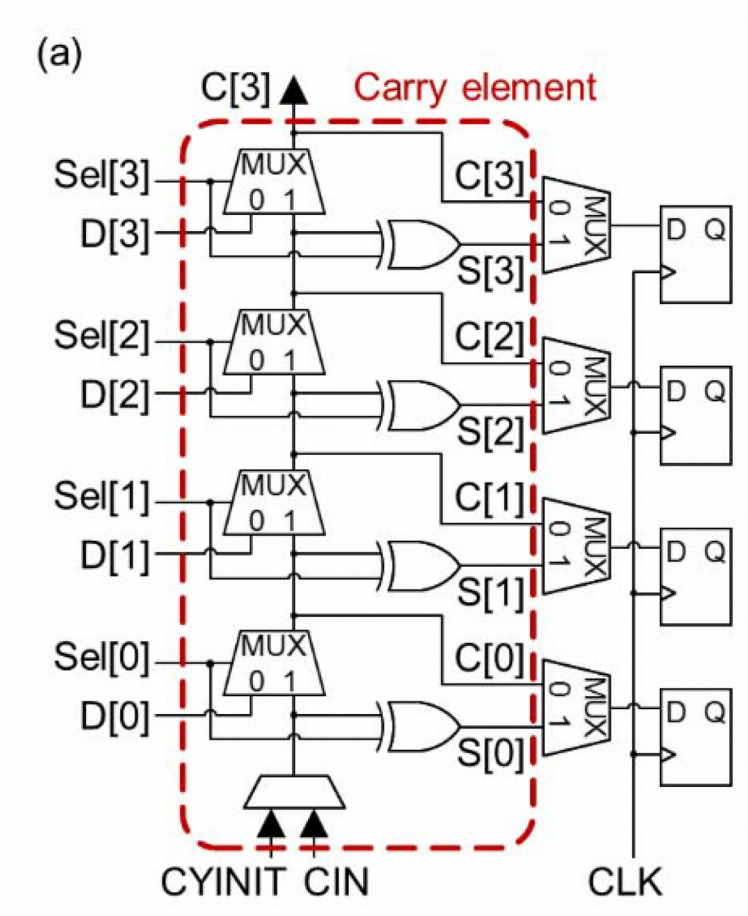
- C.Liu and Y. Wang, IEE TNS 62, No.3, 773 (2015)
- 約半分が zero-width の bin で tap の数を有効に使えてない



(b)

Tuned delay line

J.Y.Won, et al.,
"Time-to-Digital Converter Using a Tuned-Delay Line Evaluated in 28-, 40-, and 45-nm FPGAs",
IEEE TIM Vol.65,No.7,1678 (2016)



CARRY4 の各 tap 出力は

- XOR を通る
- XOR を通らない

の二種類から選べる

→ 適切な組み合わせを使うと

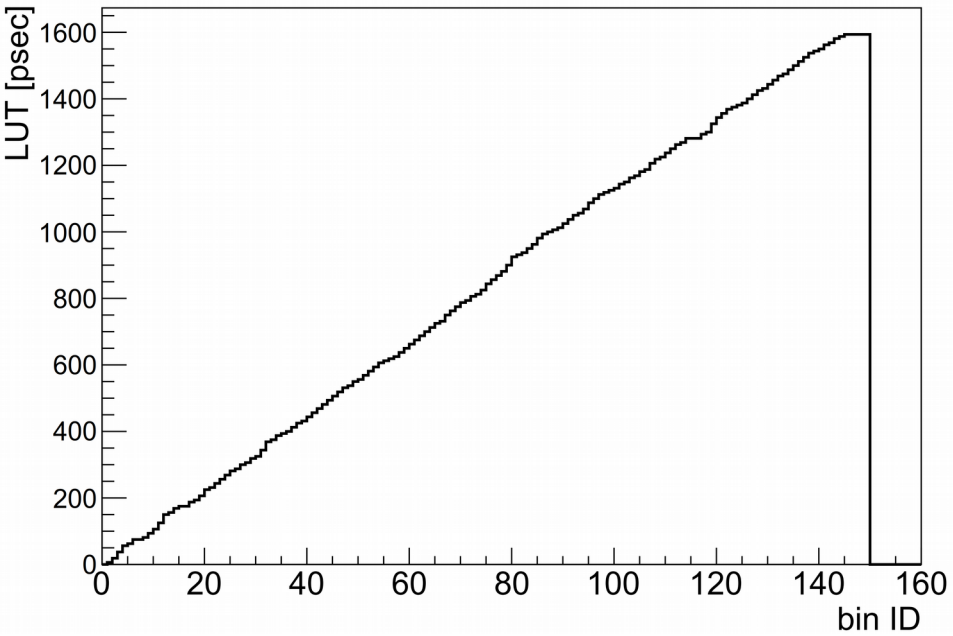
性能が向上する

(zero-width bin が減らせる)

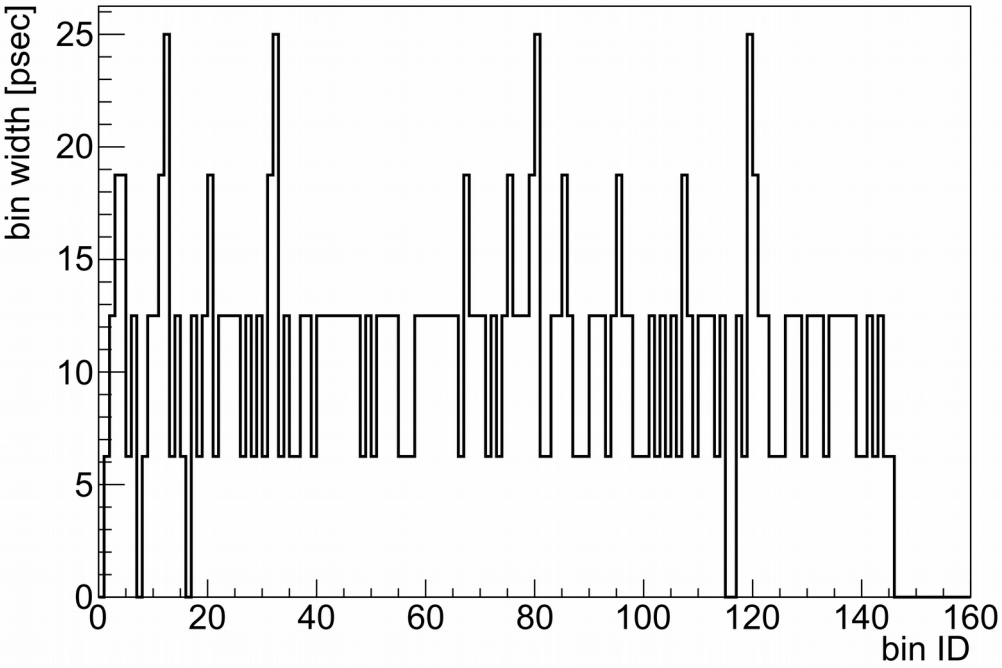
Kintex7 preliminary result

calibration table

(625MHz sampling)



bin width



wave union launcher なしでも

~11 ps/bin

まとめ

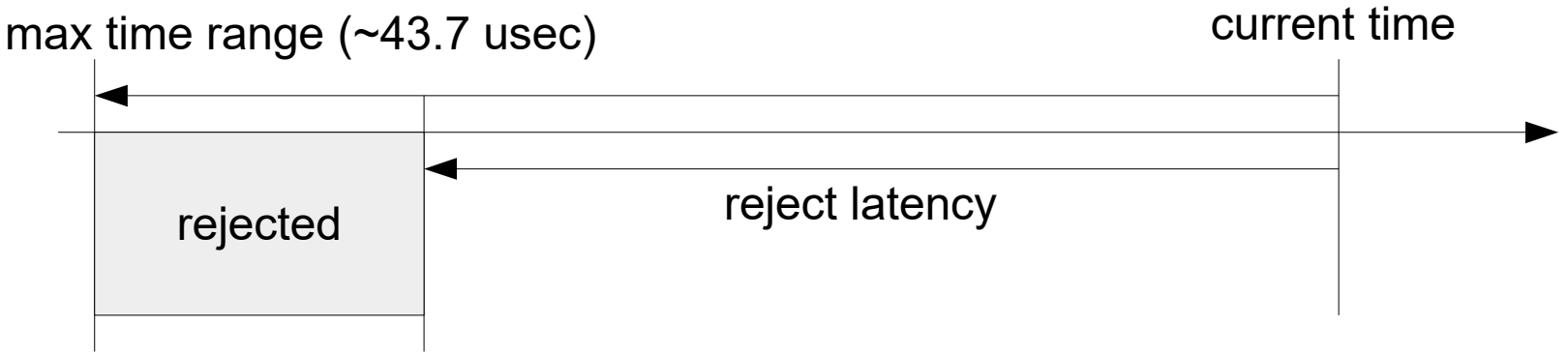
- Xilinx Spartan6 XC6SLX150-2 に Wave Union TDC を実装
  - launcher A (1→0→1 の遷移)
  - (16+1)x2 delay line
- calibration table : **~10 ps/bin**
  - 10 ps 分解能の FPGA TDC に手が届きそうなところに来た
- 一方、 **σ~30 ps**
  - 基板設計、回路の選定の見直し
- Kintex7 では wave union なしでも **~11 ps/bin** で calibration LUT が作れる
  - 分解能測定はこれから

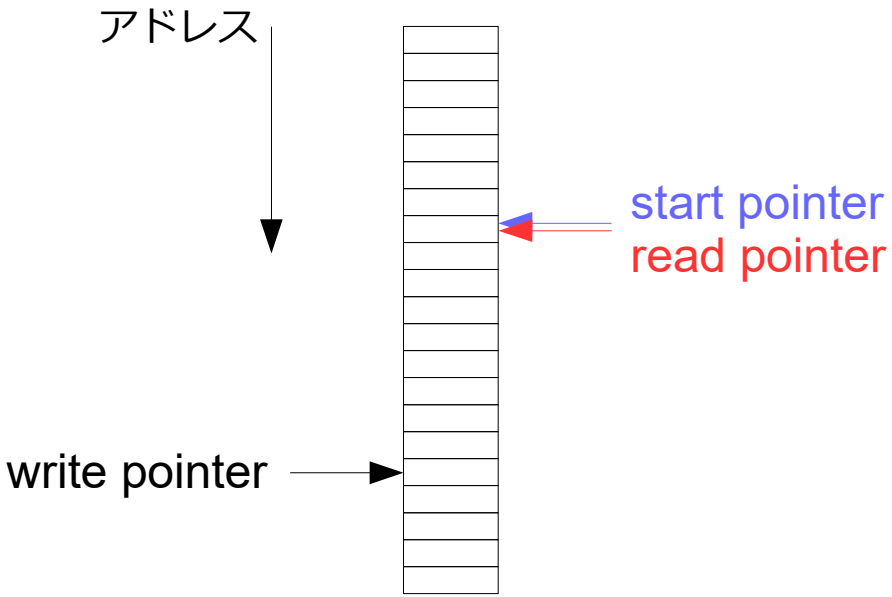
backup

L1 buffer の実装

- AMT, HPTDC とだいたい同じ (はず)
- simple dual port RAM を使用
  - アドレスが最大に達したら 0 に戻る (ring buffer 的動作)
- write port : write pointer
  - hit data 入力が来たら書き込んでアドレスを +1
  - start pointer と比較して buffer full 条件を判定
- read port : start pointer, read pointer
  - (後述)
- state : 大きく分けて 2 つ
  - trigger が来てないときに buffer 内の一番古い hit に対して time out を判定して除去する
  - trigger が来たときに time window 内の hit を探索する

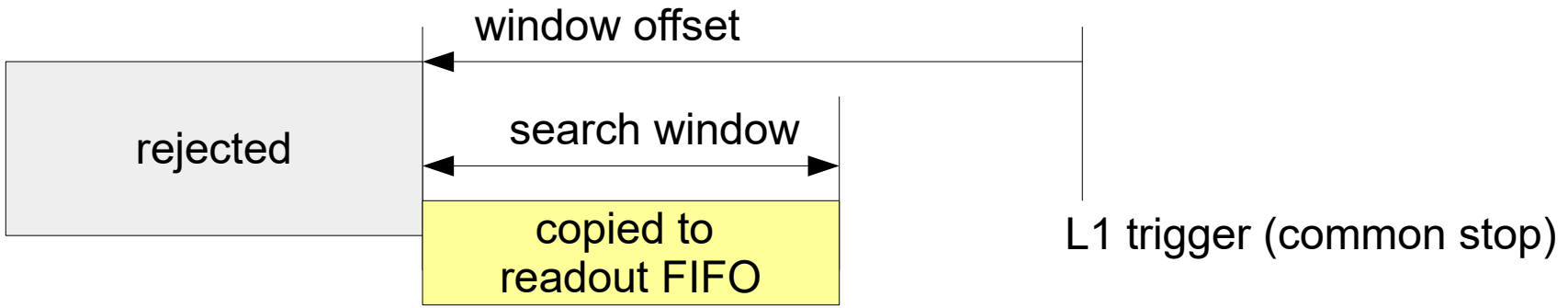
L1 buffer の実装 (2)
trigger が来てないとき



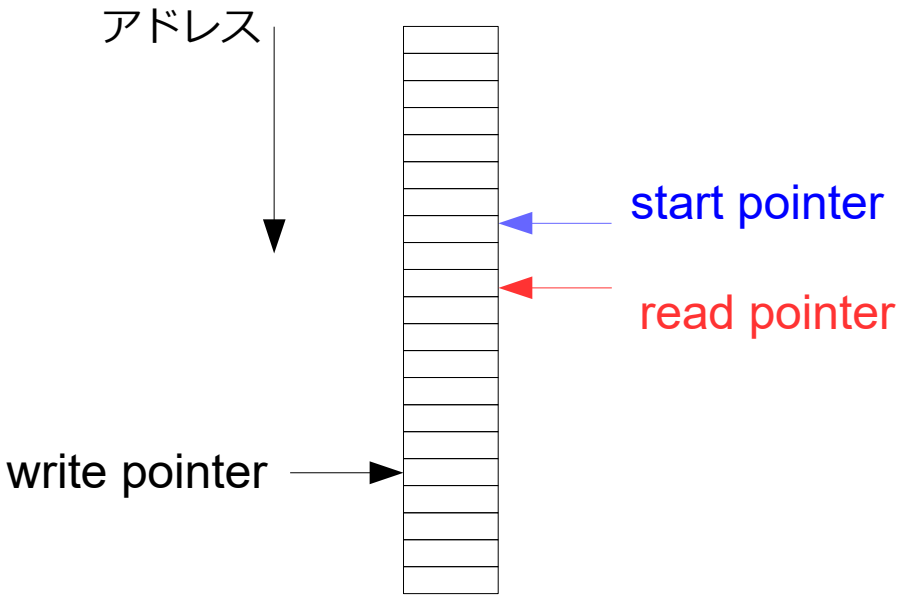


・ start pointer は buffer 内で最も古い hit を指す。
・ read pointer は start pointer と同じ位置
・現在の coarse time と比較して reject latency よりも古くなったら buffer から除去。
(アドレスを +1)

L1 buffer の実装 (3)
trigger が来たとき



(search window 内の hit は読み出し後も L1 buffer 内に残る)



- start pointer は buffer 内で最も古い hit を指す。
- read pointer は start pointer の位置から hit の探索を開始。
- window offset より古い hit は除去
  (=start pointer を +1)
- search window に入る hit は後段に送る
- 判定を下したなら read pointer を +1
- search window 終了で read pointer は start pointer の位置まで戻る