Electronics **2015**, *4*, 1018-1032; doi:10.3390/electronics4041018

OPEN ACCESS

electronics

ISSN 2079-9292

www.mdpi.com/journal/electronics

Article

CDL, a Precise, Low-Cost Coincidence Detector Latch

Ralf Joost * and Ralf Salomon

Department for Informatics and Electrical Engineering, Institute of Applied Microelectronics and Computer Engineering, University of Rostock, Rostock 18051, Germany;
E-Mail: ralf.salomon@uni-rostock.de

* Author to whom correspondence should be addressed; E-Mail: ralf.joost@uni-rostock.de; Tel.: +49-381-498-7255; Fax: +49-381-498-118-7251.

Academic Editor: Ignacio Bravo-Muñoz

Received: 26 August 2015 / Accepted: 17 November 2015 / Published: 3 December 2015

Abstract: The electronic detection of the coincidence of two events is still a key ingredient for high-performance applications, such as Positron Emission Tomography and Quantum Optics. Such applications are demanding, since the precision of their calculations and thus their conclusions directly depend on the duration of the interval in which two events are considered coincidental. This paper proposes a new circuitry, called coincidence detector latch (CDL), which is derived from standard RS latches. The CDL has the following advantages: low complexity, fully synthesizable, and high scalability. Even in its simple implementation, it achieves a coincidence window width as short as 115 ps, which is more than 10 times better than that reported by recent research.

Keywords: optical instrumentation and technology; coincidence detection; FPGA coincidence logic

---

1. Introduction

Coincidence is a concept that is well understood on a theoretical level. All readers have probably a fair understanding about the coincidence of two events: they simply occur at the same time. However, things dramatically change, if it comes to physical, real-world events. Physical events are always associated with time, which is a continuous parameter by its very nature. Measuring a continuous parameter, such as time, requires some analog-to-digital conversion, which imposes certain quantization errors.

Technically, coincidence is referred to as the occurrence of two events, A and B, that happen within a defined time span, called coincidence window, T_c. Given the timing information of both events, t_A and t_B, coincidence is indicated when:

$$-T_c < t_A - t_B < T_c \quad (1)$$

The definition in Equation (1) allows for an arbitrary order of both events.

The detection of certain physical events requires a very high resolution-in-time. A good example is the emission of gamma-ray pairs from medical radioisotopes. These radioisotopes are injected into a human body. The detection of a pair of the emitted gamma rays allows for conclusions about a patient's medical status. For this task, positron emissions tomography (PET) is an approach that detects these rays [1]. The timing of the detection is crucial for determining the origin of the gamma rays, and subsequent processing stages can reconstruct the position of the event. These post-processing stages are usually implemented in software, and produce colored images that provide information about that specific area of the body.

Since the gamma rays travel with the speed of light, timing requirements are strict. It should be clear that the quality of the resulting image directly depends on the available time resolution, which is about 1 ns or below in state-of-the-art time-of-flight PET systems [2,3]. Since these resolutions are still quite challenging for software-based systems, the field of PET systems still enjoys significant activity in the development of hardware-based coincidence detectors. Some of these detection approaches are briefly summarized in Section 2.

A common characteristic of the approaches reviewed in Section 2 is that they all employ certain logic gates, e.g., AND gates, as their basic processing elements. It is well known that these gates process input signals based on their actual voltage *levels* rather than on signal *transitions*. Section 3 proposes a different approach that is based on a new type of RS latch: the latch processes the voltage changes of the input signals and is thus edge-triggered to a certain extend. Thus, it is able to save the edge event rather than losing it like an AND gate that goes low once its inputs go low. Section 3 illustrates how this special latch structure is able to operate as a coincidence detector with a coincidence window of width T_c in the range of about a few hundreds of picoseconds. Consequently, this type of latch is called *coincidence detector latch* (CDL).

The proposed CDL was implemented in a field-programmable gate array (FPGA) and thoroughly tested in a physical laboratory setup (*i.e.*, *not* in simulation). All the parameters of the implementation and the test procedure are summarized in Section 4.

The results are presented in Section 5 and indicate that the width of the coincidence window is T_c as small as $T_c \approx 115$ ps, which is at least about ten times smaller (more precise) than reported by other research [4–7]. Finally, Section 6 concludes this paper with a brief discussion.

2. Background

In general, many coincidence detection systems used in medical applications, quantum physics, and optics contain two major parts: single-event detectors and coincidence detectors (see Figure 1). PET systems, for example, employ scintillation radiation detectors, while certain areas of quantum mechanics research, on the other hand, resort to single photon counting modules (SPCM) that detect emitted

photons. The output ports of the detectors provide a signal pulse after the successful detection of an event, *i.e.*, photon or ray arrival.

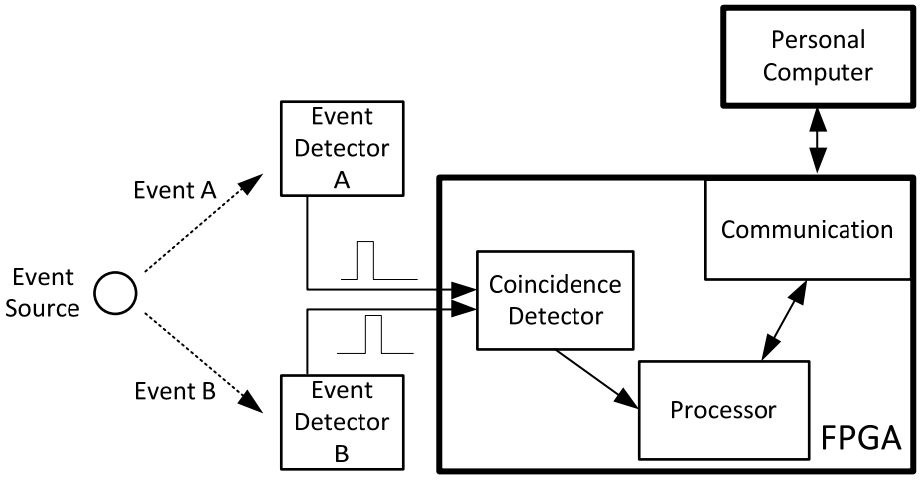

Figure 1. General hardware setup for coincidence detection.

The second major part of the coincidence detection is a subsequent processing stage that observes the output pulses of the aforementioned detectors. If pulses of two (or more) detectors arrive within an arbitrarily defined time window, the system considers them as coincidental. Recent research has strongly focused on implementing the second processing stage in field-programmable gate arrays (FPGAs), since they are cost-efficient and provide sufficient resources to implement multiple coincidence detectors on a single chip.

The basic method of determining coincidence is of surprisingly low complexity: The incoming events (*i.e.*, pulses that are emitted by the event detectors) are simply fed into an AND gate [8]. If two pulses are observed at the same time, the output port of the AND gate is activated. However, this approach faces several limitations. First of all, the length of the event pulses is critical with respect to the length of the coincidence window. Some research [7] utilized event detectors that provide output pulses of 30 ns. The pulses' lengths directly determine the coincidence window T_c. Thus, if one pulse arrives at $t = 0$ ns and the second pulse arrives at $t = 28$ ns, the AND gate indicates coincidence, even though the actual events that caused the emission of the event pulses are 28 ns apart and the overlay is merely 2 ns.

For many experiments in physics, shorter coincidence windows are required. To overcome the limitations imposed by the external event detectors, others [4] illustrated a pulse shaping system that shortens the length of the event pulses down to 7.5 ns. In this case, the signal propagation delays inside the FPGA-internal structures were exploited. Pooser *et al.* [7] refer to a pulse shaping approach that includes a digital latch that samples the incoming event pulse. Thus, the coincidence window is limited by the applied clock frequency, which was as high as 400 MHz, resulting in a coincidence window of 2.5 ns.

The systems reviewed above involve synchronously operating subparts, such as digital counters to evaluate the output information from the AND gates. Thus, if pulses from the external event detectors are too short to be synchronously processed, they have to be stretched. In [6], an incoming event pulse triggers a digital counter to count down to zero. For that count-down time, a high output value is generated as a modified event pulse. Since the counter was driven by a 250 MHz clock signal, the resulting

coincidence window is at least 4 ns wide. If switched to double data rate mode in which not only rising clock edges but also falling edges are utilized the coincidence window could be halved to 2 ns.

In summary, state-of-the-art research achieves coincidence windows as short as about 2 ns to 7.5 ns. Since these values are highly coupled to the employed clock frequencies, higher frequencies are required for shorter coincidence windows. Unfortunately, the clock generators (phase-locked loops and delay-locked loops) of currently available FPGAs are limited to about 400 MHz. In order to overcome this intrinsic limitation, the remainder of this paper proposes a new, asynchronously operating coincidence detector.

3. The Coincidence Detector Latch (CDL)

The coincidence detector latch (CDL) is basically a simple RS latch, which has undergone major changes. Subsection 3.1 gives with a brief review of the original RS latch, and discusses why it is unable to operate as a coincidence detector.

3.1. The RS Latch: A Brief Review

Figure 2 shows a simple NOR-gate-based implementation of the standard RS-latch along with its truth table. Furthermore, the right part of Figure 2 shows the latch's outputs as a response to its inputs. The input combination S = 1 and R = 0 is called *set*, since it activates the output Q = 1. The input combination S = 0 and R = 0 is called *store* as it saves the previous output state. The input combination S = 0 and R = 1 is called *reset* and results in the output Q = 0, $\overline{Q}$ = 1. Particular emphasis should be put on the forth input combination S = 1 and R = 1, which is commonly known as the illegal state, since both outputs are Q = $\overline{Q}$ = 1; this state is considered illegal, since the constraint Q = 1 − $\overline{Q}$ does not hold anymore.

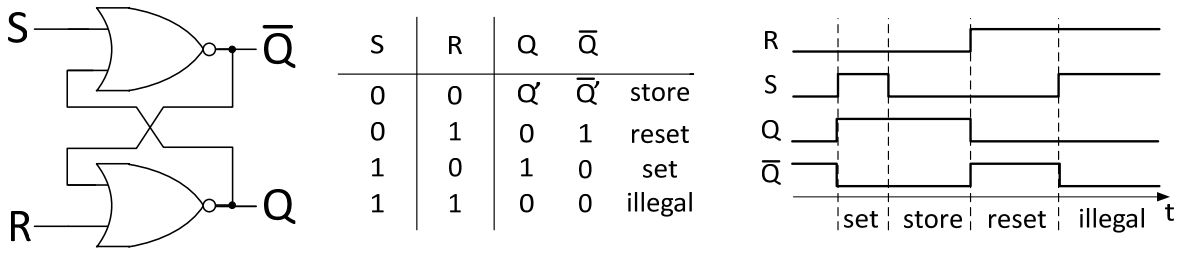

Figure 2. Original RS latch implementation utilizing NOR gates and its corresponding truth table.

Let us assume for a moment that an RS latch is used as an event detector in a PET system. If both sensors detected an event, they would activate both the R and S input of the latch. The latch would in turn end up in the illegal state. A subsequent processing stage would then be easily able to determine the occurrence of both events by evaluating Q = $\overline{Q}$ = 1.

This approach is plausible. However, the illegal state would not be able to provide *any* information of whether both events occurred coincidently or were separated in time. In other words, regardless of the actual timing of the two inputs, the latch would always end up in the same state Q = $\overline{Q}$ = 1, and would not provide any timing information about the occurrence of the two events. This information is essential for the type of applications under consideration.

Subsection 3.2 proposes some modifications with which the RS latch overcomes this limitation.

3.2. The Modified RS Latch

For the sack of simplicity, it is useful to modify the NOR-gate-based implementation so that the output negations are moved to the inputs of the subsequent gates. Figure 3 shows these two functionally identical implementations with Q and $\overline{Q}$ rewritten as Q_2 and Q_1.

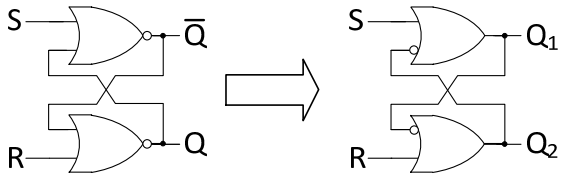

Figure 3. Two functionally identical RS latches. Moving the output inverters of both NOR gates to the input of the partner gate does not change the latch's behavior. Please note that the output labels Q and $\overline{Q}$ have been exchanged with Q_2 and Q_1.

In a further modification, two AND-gates are added to the inputs of every OR-gate as shown in Figure 4. The OR-gate's output is fed back to the input of the upper AND-gate. If the hold input is active, a Q value of 1 would sustain itself because of the positive feedback; only a deactivation of the hold input would lead to a reset. The lower AND-gate (which is connected to the second input of the OR-gate) combines an input and a disable signal. A value of 1 at the input would lead to Q = 1, provided that the disable input is not active.

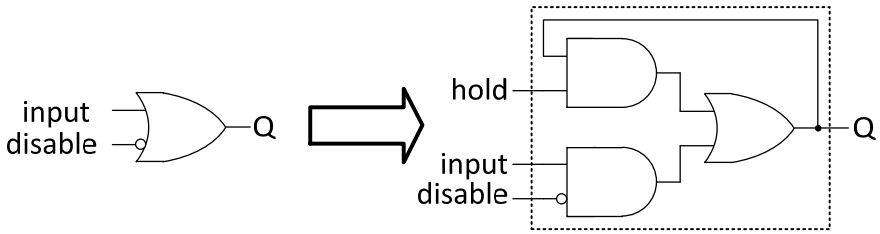

Figure 4. Modifying the OR gates to achieve self-sustaining functionality.

In the final CDL, two of these modified OR-gates are cross-coupled like the standard RS latch, as shown in Figure 5. Because of the cross-coupling, the CDL cannot assume the illegal state since the activation of either OR-gate *disables* the input (S_1 or S_2) of the other one. The only exception consists in the *simultaneous* activation of both inputs S_1 and S_2. In that case, they would force an activation of their OR-gates' outputs *before* their inputs are disabled by the partner gate.

The complete functionality of the coincidence detector latch is summarized in Table 1. After reset (*i.e.*, hold = S_1 = S_2 = 0), the latch is in its ground state Q_1 = Q_2 = 0. The coincidence detection functionality is activated when the hold input is set to hold = 1. Then, if input S_1 is activated prior to S_2, the CDL ends up in the state Q_1 = 1, Q_2 = 0. Conversely, if input S_2 is activated prior to S_1, the final output state is Q_1 = 0, Q_2 = 1. The forth output state, Q_1 = Q_2 = 1, is reached only if both inputs are activated within a narrow time window.

In summary, the final output state condenses the input behavior of the two input signals S_1 and S_2. This timing information remains in the detector even if the two input signals S_1 and S_2 are deactivated,

i.e., $S_1 = 0$ and $S_2 = 0$. In that regard, the proposed coincidence detector latch can be considered an *edge-triggered* circuit with a particular coincidence window. The experimental determination of this window is given below.

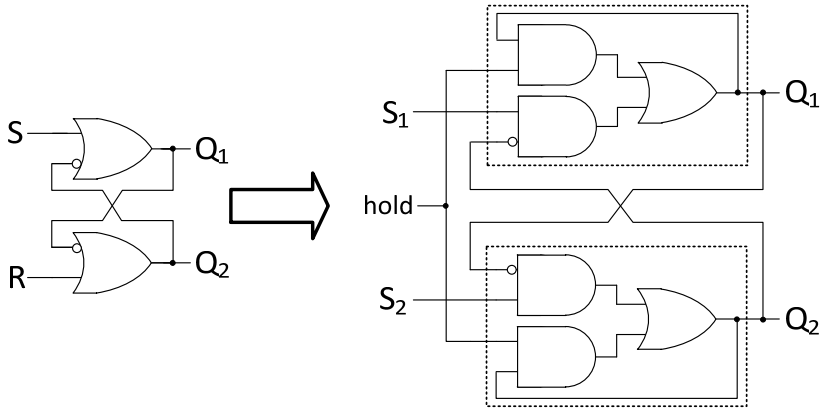

Figure 5. Complete Coincidence Detector Latch (CDL) based on the RS latch functionality, modified with a self-saving mechanism. The dotted lines indicate the border of one logic element.

Table 1. Functional behavior of the Coincidence Detector Latch (CDL).

Description	S_1	S_2	Hold	Q_1	Q_2
Reset	0	0	0	0	0
Armed	0	0	1	0	0
S_1 and not S_2	1	0	1	1	0
S_2 after S_1 and $\|\Delta t\| > T_c$	1	1	1	1	0
S_2 and not S_1	0	1	1	0	1
S_1 after S_2 and $\|\Delta t\| > T_c$	1	1	1	0	1
S_1 and S_2 and $-T_c < \Delta t < T_c$	1	1	1	1	1

4. Methods

In order to evaluate the timing characteristics of the CDL, an Altera DE2-70 Development Board serves as a test platform [9]. The board contains a Cyclone II FPGA that provides 70,000 configurable logic elements. Furthermore, the low-cost development board is equipped with various communication interfaces, memories, and display elements, such as a Seven-Segment-Display, an LCD, and various LEDs. This makes it perfectly suited to host the coincidence detector latch as well as a monitoring system to gather the output of the CDL. A NIOSII soft-core processor served as a monitor system. The processor stored the CDL-output during measurements and transferred it for further processing to a personal computer via a serial RS232 connection.

Beside this processing, the personal computer also controlled the characteristics of the signal generation, which was done by a Keysight 81150A Function Pulse Generator [10]. The function pulse generator offers an easy-to-use remote programming interface.

The function generator triggers the signal outputs on both channels. The generator's output signal is a single positive pulse with a width of 200 ns. The rising edge of that pulse is treated as an event in the coincidence detector system. In order to provide different event timings, the function generator applies different user-defined time offsets on every channel. The accuracy of these offsets is limited

to ±25 ps according to the data sheet [10]. Thus, the rising pulse edge on both channels can be set to different points in time. For the remainder of this paper, the event delay Δt is calculated as the timing difference between the rising edges of channels 1 and 2 as follows:

$$\Delta t = t_{\text{rise}} \text{ (Channel 2)} - t_{\text{rise}} \text{ (Channel 1)} \quad (2)$$

Thus, a positive Δt indicates that Channel 1 is activated prior to Channel 2, whereas negative values refer to an activation of Channel 2 prior to Channel 1. According to the definition of coincidence in Section 1, coincidence is given when the absolute value of Δt is smaller than the coincidence window T_c of the CDL.

Because of the system layout and its various processing stages, some further remarks should be made in order to help understand the results presented in Section 5. The actual event timing is given by the signal source (*i.e.*, the Keysight function generator) at the system input. However, the actual coincidence detection takes place at a certain geometric position *within* the FPGA. Due to additional signal paths, drivers, path transistors, *etc.*, the actual event timing controlled by the function generator differs significantly from the timing observed by a CDL.

According to the coincidence definition in Equation (1), one would expect the CDL to be 100% effective at detecting coincidence from $-T_c < \Delta t < +T_c$. In other words, if one plotted detection efficiency *vs.* Δt, the plot would look like a square of width $2T_c$ centered on $\Delta t = 0$. This is illustrated in Figure 6 as the idealized coincidence detection. However, the time of arrival of both signals at the CDL inside the FPGA is highly influenced by architectural signal delays. Figure 6 illustrates three different origins of timing modification: transmission line delays caused by wires, connectors, *etc.*; on-chip delays caused by input pins, drivers, pads, and routing; and logic delays that refer to the structural characteristics of the CDL. For the logic delay, the length of the feedback paths inside the CDL is of special importance. A minor influence might originate from slightly changing technical characteristics of the FPGA's logic elements, e.g., their specific propagation delays.

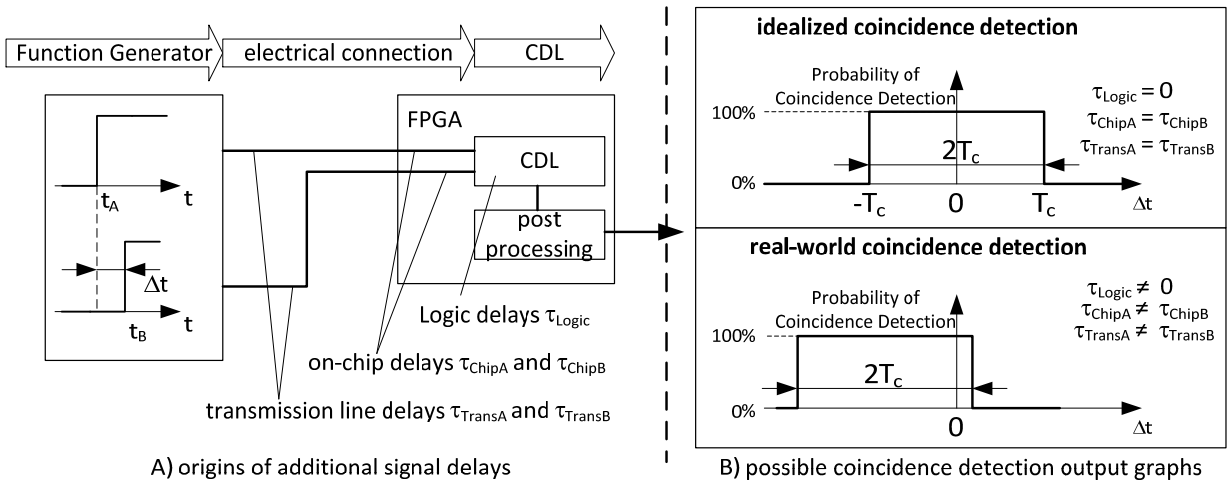

Figure 6. (**A**) The timing characteristics and the various types of delays; (**B**) The top panel shows the idealized coincidence detection with no logic delay and equal chip and transmission delays between the two signals. The bottom panel shows a more typical coincidence detection with non-ideal delays.

The combination of these additional delays leads to a shifted coincidence graph. The particular values of the delays depend on the physical location of the logic on the FPGA and therefore differ when the CDL is mapped to different regions within the FPGA.

Taking these additional delays into account, Equation (1) is augmented by a system-based timing offset that incorporates the delay variables shown in Figure 6:

$$-T_c < \Delta t + T_{system} < T_c \tag{3}$$

$$T_{system} = (\tau_{TransA} - \tau_{TransB}) + (\tau_{ChipA} - \tau_{ChipB}) \tag{4}$$

Equations (3) and (4) describe the real-world (*i.e.*, shifted) coincidence detection plots. The more the additional delays differ, the larger the shift of the coincidence detection plot.

Furthermore, the coincidence window width, T_c, itself can change with differing delay values. In particular, when the two logic elements that form the CDL are placed at different locations, the lengths of the feedback signals are changed. The longer the distance between those two logic elements, the larger the logic delay, τ_c, is. This is caused by the functional behavior of the RS latch. According to Figure 5, a signal change on input S_1 activates the output Q_1. The output of one logic element travels along the feedback path to its counterpart. There, Q_1 "locks" the second logic element. If signal S_2 is activated before Q_1 arrives, the CDL states coincidence. Thus, the larger the logic delay τ_{Logic}, the longer the coincidence window T_c. The actual width of the coincidence window for different CDL positions is evaluated in Section 5.

For the experiments, both signal channels were connected to the FPGA development board via the available general purpose input/output (GPIO) pins. To synchronize the signal generation with the CDL-readout, the NIOSII was programmed with the following measurement scheme. First, the processor reset the CDL by deactivating the hold input (see Figure 5). The hold input is then set again to activate the CDL. In the next step, the processor triggered the function generator to output a pulse on every channel. Finally, the processor read the output state of the CDL. After 1000 measurements, it sent the counted values of the four possible output states {(0,0), (0,1), (1,0), (1,1)} to the personal computer, which in turn wrote them into a file. After a user-defined number of received data sets, the personal computer changed the delay setting.

The coincidence detector latch shown in Figure 5 was implemented in VHDL. The actual synthesis and routing was automatically done by the software tool chain, *i.e.*, Altera Quartus II 13.0 SP1 [11]. The two logic elements forming the CDL were placed in the center of the FPGA in between the logic elements that belong to the NIOS II processor in order to minimize signal path lengths from the CDL to the processor.

Given that the determination of the coincidence of two events is affected by the path lengths involved, the experimental evaluation also considers different CDL placements on the FPGA device. These alternatives are labeled CDL 2 to CDL 6 in Figure 6 and resulted in different path lengths. These placements were obtained by manual position assignments in the Quartus project settings file. They were arbitrarily chosen and satisfied the following constraints:

1. Intermediate distance to the NIOS II processor for noise and cross-talk reduction (CDL 2)
2. Maximum distance to the NIOS II processor (CDL 3)
3. shortest path length from the FPGA's input ports to the CDL's input ports (CDL 4)

4. maximum path length from the FPGA's input ports to the CDL's input ports (CDL 5)
5. increasing the cross coupling paths' length between the CDL's two logic elements (CDL 6)

Figure 7 also shows the input pads for the two signals S_1 and S_2 that belong to those GPIO pins that are connected to the Keysight function generator. Since the FPGA is mounted on a development board, the options to choose FPGA pins as signal inputs are limited. All pins are hardwired to components on the board. The designer can only chose among those pins that connect to the on-board GPIO pin header.

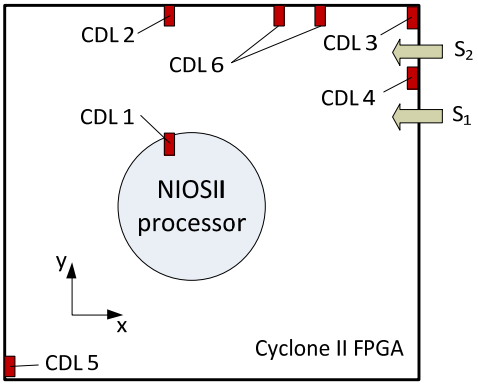

Figure 7. Test positions for the CDL on the Cyclone II field-programmable gate array (FPGA). The NIOS II processor and the signal inputs are shown for illustration.

5. Results

The first experimental objective was to provide a proof-of-concept for the functional behavior of the aforementioned CDL structure. The Keysight function generator generated a delay range from −200 ps to 400 ps in steps of 5 ps. At every delay setting, the FPGA evaluated the output of the CDL 1000 times. The result for the automatically placed CDL 1 (see, also, Figure 7) is shown in Figure 8.

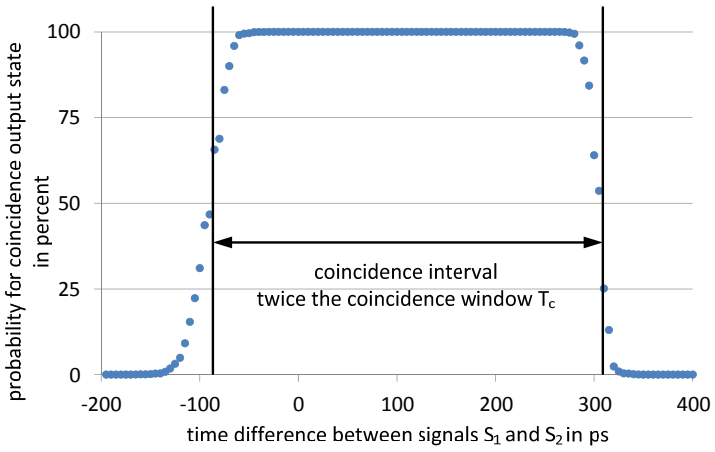

Figure 8. Timing behavior of the proof-of-concept implementation of a CDL.

As can be seen, starting from an event delay of −140 ps, *i.e.*, the signal on Channel 2 is emitted 140 ps ahead of the signal on Channel 1, the CDL begins to indicate coincidence in some of the measurements

at that particular delay setting. At −90 ps, the 50% level is reached; half of the 1000 measurements indicated coincidence, whereas the remaining 500 measurements did not. The 50% level is marked with the black bars in Figure 8. From −50 ps on, the CDL stated coincidence in all measurements. This behavior ends at 270 ps, where the output probability for coincidence starts to decrease down to zero. The 50% point is reached at 305 ps; delays larger than 340 ps never lead to coincidence indication. The remainder of this paper defines the duration of the coincidence window T_c as half of the time between the 50% points. In Figure 8 this length is $T_c = (305\text{ ps} - (-90\text{ ps}))/2 \approx 200$ ps.

The second set of experiments evaluated five different locations of the CDL within the FPGA. The results for these variants (*i.e.*, CDL 2 to CDL 6) are summarized in Figure 9. For illustration purposes, the time axis is identical throughout all graphs. Furthermore, the result for the automatically placed CDL 1 is included for comparison. The measurement range was extended to provide delays $-700\text{ ps} < \Delta t < 700$ ps in steps of 5 ps. Table 2 provides the detailed measurement results. The three columns T_{c100}, T_c, and T_{cmax} state the width of the coincidence window. The subscript indicates the boarders of the coincidence window: T_{c100} only refers to that part of the graph that shows a 100% probability for coincidence detection. The T_{cmax} values include all output values greater than zero into the coincidence window calculation. The coincidence window Tc is based on those output values that show a coincidence detection probability equal to or greater than 50%. The two columns Δt_{min} and Δt_{max} state the lowest and the highest function generator delay setting that caused a positive coincidence detection output. Thus, the term $(\Delta t_{max} - \Delta t_{min})/2$ provides the value for the maximum coincidence window width T_{cmax}.

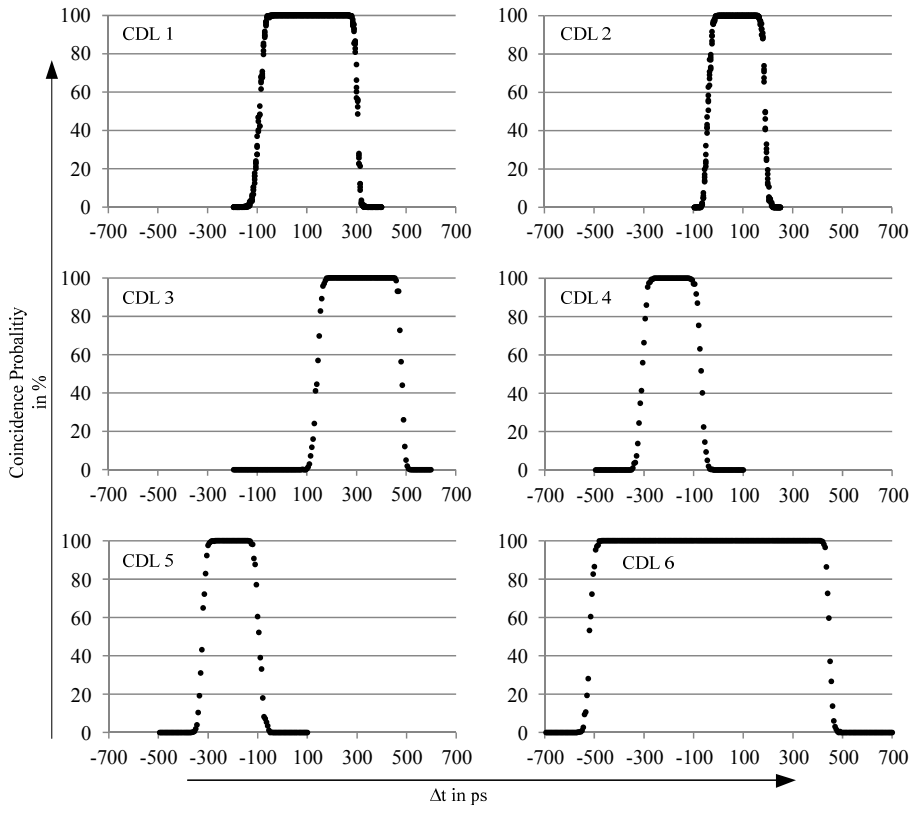

Figure 9. Coincidence windows for different setups of the CDL on a Cyclone II FPGA. The time axis for all tests is defined by the Keysight function generator.

Table 2. Coincidence window times and timing characteristics of different CDL implementations.

Name	T_{c100} (ps)	T_c (ps)	T_{cmax} (ps)	Δt_{min} (ps)	Δt_{max} (ps)
CDL 1	160	197.5	240	−140	340
CDL 2	72.5	115	157.5	−85	230
CDL 3	137.5	170	207.5	100	515
CDL 4	72.5	120	165	−350	−20
CDL 5	75	115	157.5	−365	−50
CDL 6	445	485	525	−565	485

As can be seen, the diagrams presented in Figure 9 are not all symmetric about $\Delta t = 0$. The reasons for this have already been discussed in Section 4. CDLs 3 and 4 provide good examples for asymmetric on-chip delays, τ_{chip}. The length of the signal path from S_1 to CDL 4 is nearly as long as the signal path from signal input S_2 to CDL 4. The center of the coincidence window of CDL 4 is at −185 ps. The path from S_1 to CDL 3 is significantly longer than its path from S_2. Thus, the absolute value of the system timing constant T_{system} (see Equation (4)) increases and the coincidence window is shifted to the right with the center at +270 ps.

The transmission line delays are unknown. However, they are constant throughout all experiments. Furthermore, the aforementioned CDL 4 is centered at −185 ps, although both of its signal paths do not differ that much in length. This indicates that the transmission lines from the function generator to the FPGA's input pins have caused this shift.

The width of the coincidence window T_c varies slightly from CDL 1 to CDL 5. The 50% values are in the range of 115–200 ps. CDL 2, 4, and 5 provide especially narrow coincidence windows. On the other end, the coincidence window T_c for CDL 6 is significantly larger.

CDLs 1 to 5 were designed such that their two parts were placed into the very same logic array block which ensures short feedback paths between them. By contrast, the two logic elements of CDL 6 were separated. The logic element that connects to signal S_1 (left part of CDL 6 in Figure 7) is placed in column 85 of the FPGA, whereas the other logic element (right part of CDL 6 in Figure 7) is placed in column 90. This heavily affects the system timing constant τ_{logic}, and thus affects the coincidence window T_c as announced in Section 4.

The separation of the two logic elements that form CDL 6 thus provide a design opportunity for arbitrarily formed coincidence windows. Figure 10 compares the coincidence windows of three variations of CDL 6. The signal input S_2 always connects to a logic element at position (90,50,0) (The coordinate of a logic element comprises: column, row, element number inside the logic array block), whereas the second logic element (that connects to S_1) was placed at positions (85,50,0), (80,50,0), and (75,50,0). Thus, the geometric distances within these three variants were 5, 10, and 15 columns of logic elements inside the FPGA. Transforming these geometric distances into propagation delay values would require exact knowledge about the on-chip structural elements, such as the layout of the utilized routing resources, the characteristics of the involved pass transistors, and the final signal mapping to local and global routing resources. Since this discussion would exceed the scope of this paper by far, a rule-of-thumb would be: Every traversed column of logic elements induces an additional propagation delay of about 50–100 ps.

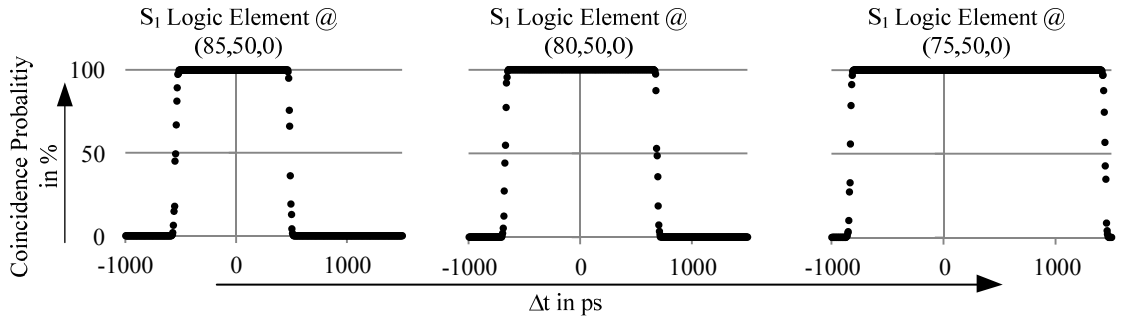

Figure 10. Comparison of different feedback path lengths for CDL 6. The S_2 logic element resides at position (90,50,0). Thus the feedback path length is 5, 10, 15 columns of logic elements inside the Cyclone IV FPGA (from left to right).

Figure 10 shows that the longer the geometric distance between the two logic elements is, the wider their coincidence windows will be. Since the logic element connected to S_1 was moved, the system timing constant, T_{System}, also changed. Thus, the coincidence windows were shifted. As already shown above, a significant elongation of the S_1 signal path leads to a shift in the positive direction. The detailed measurement results are presented in Table 3.

Table 3. Coincidence window *T*c depending on the distance between the CDL's logic elements.

S_1 Logic Element Position	**(85,50,0)**	**(80,50,0)**	**(75,50,0)**
S_2 logic element position	(90,50,0)	(90,50,0)	(90,50,0)
Feedback path length	5 columns	10 columns	15 columns
Center of coincidence window	−30 ps	10 ps	300 ps
T_c in ps	485 ps	680 ps	1150 ps

6. Discussion

This paper has proposed a coincidence detection latch "CDL" that provides a promising new option for achieving high-precision, low-cost coincidence detection with coincidence window times shorter than 400 ps. Another appealing aspect is that the hardware structure of the CDL is based on the well-known RS-latch. The CDL can be formed by standard VHDL constructs and is fully synthesizable by common synthesis tools. Despite utilizing six logic gates, it consumes only two logic elements on an off-the-shelf FPGA. Even a vintage Cyclone II FPGA development board could host tens of thousands of CDLs.

The smallest evaluated coincidence window was only 115 ps wide. The experiments indicate that on both ends, the coincidence window has some transient areas where coincidence detection is sporadic. The widths of these areas vary. At the beginning of the window, they usually last for approximately 90 ps to 100 ps. At the end of the window, they are slightly smaller, *i.e.*, approximately 70 ps to 90 ps. These transient areas might be caused by external effects. The Keysight function generator, for example, provides limited precision during its delay generation. The data sheet states an accuracy of ±25 ps ± 50 ppm [10]. Furthermore, on-chip as well as on-board noise might be sources of the observed effect. Therefore, the default TTL inputs of the FPGA were driven in LVDS mode to provide better robustness against noise and voltage variation.

When comparing the individual results, Figure 9 indicates an effect that might be considered a limitation by some: the location of the coincidence window on the *X*-axis (the time base) varies among the various CDL implementations. Since the CDL evaluates the signals at its gate inputs, the electrical connections to the CDL's inputs are part of the *external* wiring along with the cables to the function generator. As already illustrated in Figure 6, this external wiring affects the location of the coincidence window but not the duration of its window. The FGPA synthesis process assigns individual signal routing resources to every CDL. Thus, every particular CDL exhibits an individual coincidence window, with a timing that depends on the physical location on the FPGA device. But this is not a flaw of the developed system, as every timing system has to be calibrated within its actual physical environment.

For experiments that require multi-channel coincidence detection, the use of multiple instances of the CDL on the very same FPGA is possible. Future experiments will investigate this, especially the use of nested multi-channel CDL structures. Furthermore, CDL chains might lead to an even higher time resolution in coincidence detection with the benefit of eliminating the offset limitation discussed above. The experiments indicate that a CDL's coincidence window T_c and its center point, depends on structural and configurable FPGA parameters. Every single CDL can thus be configured to its required demands. When multiple CDLs are placed on the FPGA, they might be configured in such way that all CDLs evaluate the same signal channel pair. The combination of two CDLs with different center points and overlapping coincidence windows can be used to form a virtual coincidence window that is smaller than a single coincidence window. For example, two CDLs with $T_{c1} = T_{c2} =$150 ps and $t_{Center1} = 0$ ps and $t_{Center2} = 100$ ps form a virtual combined coincidence window of $T_c = 100$ ps and a center point at $t_{Center} = 50$ ps. This can be extended for nearly arbitrarily many CDLs.

Unfortunately, it is impossible to calculate precise timing characteristics of a single CDL in advance, *i.e.*, prior to the synthesis process. They depend on a wide variety of parameters. Beside the aforementioned geometric position, also other variables, such as the total number of CDLs, the size and placement of additional control logic, the placement of the input ports, *etc.*, have to be taken into account. Even the very same CDL might behave differently on different FPGAs of the same type. However, the possibility for multiple CDLs operating in parallel might overcome this limitation. The designer can implement hundreds of CDLs, and refer just to those CDLs that provide the desired timing behavior.

In conclusion, this paper has presented a possible approach to design coincidence detectors with a coincidence window of less than 200 ps. Even though the practical experiments were done on an Altera 90 nm Cyclone II FPGA, the results show significant progress towards high precision coincidence detection. The discussion presented above indicates that for a CDL's final timing characteristics, the involved path delays are more important than the look-up table's propagation delays. Thus, switching to modern FPGA families, e.g., the 28 nm Cyclone V, might have only minor impacts on a CDL's overall timing behavior.

Furthermore, the purpose of the practical experiments was to provide a *proof-of-concept* of the proposed approach. The integration of the proposed CDLs into a commercial system requires, for example, the cooperation with a selected PET system developer, which is beyond the scope of this paper. Nevertheless, such a system integration would include long-term stability tests in order to evaluate the effects of temperature changes, radiation exposure, the event detector precision. It might

well be that these long-term stability tests reveal some dependencies, which impose some sporadic re-calibration (e.g., one a week or once a month).

7. Conclusions

This paper has proposed a new circuitry for the detection of the coincidence of two (independent) signals. This detector originates from the well-known, standard RS latch, and provides four different, stable states at its two outputs. Due to its internals, this detector provides very precise coincidence windows as short as 120 ps. Its compact and regular architecture facilitates the massively parallel realization of a large number of these detectors on FPGAs, resulting in practically usable detector arrays.

Acknowledgments

The authors gratefully thank Matthias Hinkfoth for fruitful discussions and support. Furthermore, the authors' thanks are dedicated to Christian Haubelt for providing access to the Keysight Function Pulse Generator.

Author Contributions

Both authors were working together very tightly. The theoretical work on the design of the described coincidence detectors were mainly driven by Ralf Salomon. The transformation of the ideas into FPGA-synthesizable constructs was done by Ralf Joost. Ralf Joost was also responsible for performing the experiments. Both authors analyzed the experimental results and wrote the paper together.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Ziegler, S.I. Positron Emission Tomography: Principles, Technology and Recent Developments. *Nucl. Phys. A* **2005**, *752*, 679c–687c.
2. Conti, M. Improving time resolution in time-of-flight PET. *Nucl. Instrum. Methods Phys. Res. A* **2011**, *648*, 194–198.
3. Cates, J.W.; Vinke, R.; Levin, S.L. Analytical calculation of the lower bound on timing resolution for PET scintillation detectors comprising high-aspect-ratio crystal elements. *Phys. Med. Biol.* **2015**, *60*, 5141–5161.
4. Branning, D.; Bhandari, S.; Beck, M. Low-cost coincidence-counting electronics for undergraduate quantum optics. *Am. J. Phys.* **2009**, *77*, 667–670.
5. Branning, D.; Beck, M. An FPGA-based module for multiphoton coincidence counting. In Proceedings of the Advanced photon counting techniques VI, Baltimore, MD, USA, 25–26 April 2012; p. 83750F.
6. Ko, G.B.; Yoon, H.S.; il Kwon, S.; Hong, S.J.; Lee, D.S.; Lee, J.S. Development of FPGA-based coincidence units with veto function. *Biomed. Eng. Lett.* **2011**, *1*, 27–31.

7. Pooser, R.C.; Earl, D.D.; Evans, P.G.; Williams, B.; Schaake, J.; Humble, T.S. FPGA-based gating and logic for multichannel single photon counting. *J. Mod. Option* **2012**, *59*, 1500–1511.
8. Kim, T.; Fiorentino, M.; Gorelik, P.V.; Wong, F.N.C. Low-cost nanosecond electronic coincidence detector. arXiv:physics/0501141, 2005.
9. *Altera DE2-70 User Manual*, version 1.08; Terasic Inc.: HsinChu, Taiwan, 2010.
10. 81150A and 81160A Pulse Function Arbitrary Noise Generators—Data Sheet, version 1.2; Keysight Technologies: Santa Rosa, CA, USA, 2015.
11. *Altera Quartus II Handbook*, Version 13.0; Altera Corp.: San José, CA, USA, 2013.