

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CITADEL SECURITIES LLC,
Petitioner

v.

HFT SOLUTIONS, LLC
Patent Owner.

Case No. IPR2026-00151

U.S. Patent No. 11,575,381

DECLARATION OF KEVIN B. STANTON, Ph.D.

TABLE OF CONTENTS

I. QUALIFICATIONS AND EXPERIENCE	6
II. SCOPE OF WORK.....	7
III. LEGAL STANDARDS.....	8
IV. LEVEL OF ORDINARY SKILL	13
V. TECHNICAL BACKGROUND.....	15
VI. OVERVIEW OF THE '381 PATENT.....	28
VII. CLAIM CONSTRUCTION.....	40
VIII.OVERVIEW OF THE PRIOR ART.....	44
IX. GROUND 1A: ALTERA ANTICIPATES CLAIMS 1, 4, AND 6-10 OF THE '381 PATENT.....	63
X. GROUND 1B: ALTERA, STRATIX, AND SI5345 RENDER CLAIMS 1, 4, AND 6-10 OBVIOUS	103
XI. GROUND 2: ALTERA AND LOCKWOOD RENDER CLAIMS 1-4 AND 6- 12 OBVIOUS.....	105
XII. CONCLUSION	128

LIST OF EXHIBITS

Exhibit No.	DESCRIPTION
1001	U.S. Patent No. 11,575,381
1002	Expert Declaration of Kevin B. Stanton, Ph.D.
1003	File History of U.S. Patent No. 11,575,381
1004	Synchronous Ethernet Solutions with Altera FPGAs and Silicon Labs Jitter-Attenuating PLLs (“Altera”)
1005	Altera Stratix V Handbook (“Stratix Handbook”)
1006	Any-Frequency, Any-Output Jitter-Attenuators/Clock Multipliers Si5345, Si5344, Si5342 Family Reference Manual, August 2015 (“Si5345 Manual”)
1007	A Low-Latency Library in FPGA Hardware for High-Frequency Trading (HFT) by John W. Lockwood et al., 2012 IEEE 20th Annual Symposium on High-Performance Interconnects (September 2012), https://doi.org/10.1109/HOTI.2012.15 (“Lockwood”)
1008	Plaintiff HFT Solutions’ Response in Opposition to Defendant Citadel Securities’ Motion to Dismiss
1009	Declaration of Mina Ching regarding website of Altera White Papers
1010	Declaration of Mina Ching regarding website of Stratix V Device Handbook
1011	Declaration of Mina Ching regarding website of Si5345/44/33 Data Sheet

Exhibit No.	DESCRIPTION
1012	Patent Owner's Initial Infringement Contentions for the '381 Patent
1013	Altera and IDT Synchronous Ethernet Solution for ITU-T G.8262
1014	"The Race to Zero Latency for High Frequency Trading" by Saul Sahoo
1015	U.S. Patent No. 10,169,814
1016	High-Frequency Trading, A Practical Guide to Algorithmic Strategies and Trading Systems by Irene Aldridge (2013)
1017	Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig Megafunctions, Application Note AN-661-3.0, Altera Corporation, November 2013
1018	Exploring Algorithmic Trading in Reconfigurable Hardware by Stephen Wray (2010)
1019	HDL Chip Design, A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog by Douglas J. Smith, Doone Publications (1999)
1020	UltraScale Architecture Clocking Resources – User Guide, Xilinx, April 9, 2018
1021	Chip Hall of Fame: XC2064 FPGA, IEEE Spectrum, June 30, 2017
1022	List and comparison of FPGA companies by Jeff Johnson, FPGA Developer, July 15, 2011
1023	Understanding Clock Domain Crossing (CDC) Checks and Techniques by Saurabh Verna, et al., EE Times, December 24, 2007

Exhibit No.	DESCRIPTION
1024	Phaselock Techniques by Floyd Gardner, Third Edition (2005)
1025	Interactive Brokers' Founder Changed Trading Forever. What He Sees Next by David Wignall, Barrons (October 2, 2025)
1026	Wall Street's Quest to Process Data at the Speed of Light by Richard Martin, Rubinow (April 21, 2017)
1027	FPGA accelerated low-latency market data feed processing by Gareth W. Morris, et al., 17 th IEEE Symposium on High Performance Interconnects (2009)
1028	The Once and Future Internet by Daniel Svernnson, Design Lines (May 1, 2001)
1029	An Introduction to Synchronized Internet, Embedded (March 3, 2009)
1030	Design of Control Systems by A. Frank D'Souza, Prentice-Hall (1988)
1031	Clock Networks and PLLs in Stratix V Devices, Altera (May 6, 2013)

DECLARATION OF KEVIN B. STANTON, PH.D.

I, Kevin B. Stanton, Ph.D., do hereby declare as follows:

I. QUALIFICATIONS AND EXPERIENCE

1. I received a Ph.D. in Electrical and Computer engineering from Portland State University in 1996, with a dissertation focused on clocking for telecommunications systems.

2. I worked at Intel Corporation for 27 years, from 1996 to 2023. I served as Intel's chief technologist for clock / time synchronization, CPU (central processing unit) and software timekeeping, and clock synchronization over network and I/O buses. In particular, I drove support for the Precision Time Protocol (PTP / IEEE 1588) and related standards and technologies into multiple silicon and system-level products.

3. I also designed critical portions of multi-port Ethernet switch silicon, was the architect for Intel's InfiniBand systems and served as chief silicon validation architect for Intel's network processing silicon.

4. External to Intel, I co-authored a profile of IEEE Std. 1588TM standard (IEEE Std. 802.1ASTM-2011) and contributed to others, founded and co-chaired a conference on Time Sensitive Networks and Applications (TSN/A), co-founded and chaired the industry alliance responsible for certification of devices implementing the IEEE 802.1 Time Sensitive Networking (TSN) standards (Avnu Alliance).

5. I hold 25 US patents and patent applications, over 20 internationally, and have published more than 20 papers and articles in the area of clock and time synchronization.

6. I have taught logic design at the university level and presently teach electrical/computer engineering part time. I also serve as a consultant to companies in need of my expertise. I am a Senior member of the IEEE.

II. SCOPE OF WORK

7. I have been retained by Citadel Securities LLC to provide and/or review the materials I discuss in this declaration, to provide my expert opinion in relation to what a person of ordinary skill in the art in relation to U.S. Patent No. 11,575,381 would have understood, and the actions that this hypothetical skilled engineer would have undertaken prior to November 5, 2018, given the state of the art and accumulated knowledge at the time.

8. In forming my opinions in this declaration, I relied on my training, education, and experience, as well as the materials referenced in this declaration.

9. In addition to the '381 patent, I have reviewed the record of the exchanges occurring between the USPTO and the applicant as that patent was originally examined. I have also reviewed many other documents that I understand to be prior art to the '381 patent in arriving at my opinions. Those documents are discussed in this declaration.

10. I am being compensated for the time spent working on this matter at my usual and customary hourly rate of \$500 per hour. My compensation is not dependent upon the outcome of, or my testimony in, the present proceeding. I have no other financial interest in the outcome of this proceeding.

III. LEGAL STANDARDS

11. I am not a lawyer and have no legal training. My understanding of the relevant legal principles was provided to me by counsel. A summary of these legal standards, as I understand them, and that I have applied in my analysis, are provided in this section.

A. Claim Construction

12. It is my understanding that the words of a claim are interpreted according to the plain and ordinary meaning as understood by a person of ordinary skill in the art.

13. It is my understanding that to understand the meaning of a claim term, one generally first looks to the intrinsic evidence: the patent claims, the written description and figures or diagrams, and the prosecution history. One may also look to extrinsic evidence to aid it in coming to the correct conclusion as to the true meaning of the language employed.

14. I also understand from counsel that all words in a claim are presumed to have meaning in a claim. In other words, claims terms generally should not be interpreted in a way that makes them unnecessary or superfluous.

B. Anticipation

15. I understand from counsel that a prior art reference anticipates a patent claim if all the claim's limitations are expressly or inherently disclosed in a single prior art reference. I understand that if a prior art reference does not expressly disclose a claim limitation, it may do so inherently if the limitation is necessarily present in the prior art reference. I understand that inherency may not be established by probabilities or possibilities. A prior art reference does not need to use the same language, but all of the requirements of the claims must be present. Each element in the claim must be arranged or combined in the same way that the claim recites. A prior art reference must also enable a person of ordinary skill in the art to make and use what is disclosed in the reference.

16. I also understand that it is appropriate to rely on additional evidence that shows that a feature is necessarily present in a prior art reference.

C. Obviousness

17. I understand that a patent claim is obvious if the differences between the subject matter sought to be patented and the prior art are such that the claimed subject matter as a whole would have been obvious to a person of ordinary skill in

the art (POSA) at the time the alleged invention was made. For the purposes of my analysis, I assume this timeframe to be on or before November 5, 2018.

18. I understand that the following factors must be considered in determining obviousness: (1) the scope of content of the prior art; (2) the difference or differences, if any, between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art at the time of the alleged invention; and (4) additional “secondary considerations,” if any, that indicate whether the alleged invention was obvious or not obvious. I further understand that evidence of “secondary considerations” may be advanced by a patent owner, if available, and that such evidence can overcome an initial showing of obviousness.

19. I understand a patent claim would have been obvious if all the claim elements were known in or suggested by prior art references, where a POSA would have had reason(s) to combine the teachings or suggestions from that art based on the references themselves and the background knowledge available to a POSA, so as to arrive at the claimed subject matter with a reasonable expectation of success.

20. I further understand that no guarantee or absolute predictability of success is required to establish obviousness. As such, obviousness cannot be avoided simply by a showing of some degree of unpredictability in the art, so long as the prior art establishes a reasonable probability that a POSA could have made the

subject matter embraced by a claim such that it would have worked for its intended purpose.

21. I understand that a conclusion of obviousness can be based on a single prior art reference or a combination of two or more prior art references. Where the POSA would have arrived at the claimed invention by selecting and/or modifying prior art disclosures and had a good reason (i.e., motivation) to do so with a reasonable expectation of success, the invention as claimed would have been obvious.

22. While I appreciate that the prior art references themselves may provide a POSA with a teaching, suggestion, or motivation to modify or combine them, I understand that motivation can also be based on the ordinary skill, ordinary creativity, and common sense of the POSA.

23. As I understand the law, an obviousness analysis considers the inferences and creative steps a POSA would employ under the circumstances, e.g., a need or problem to be solved in the field at the time of invention. I further understand that a need or a problem in the art can provide a motivation to modify and/or combine teachings and suggestions from the prior art.

24. I further understand that the law allows consideration of demands known to the design community or present in the marketplace. When a work is available in one field of endeavor, design incentives and other market forces can

prompt variations of it, either in the same field or a different one. As such, economic influences may also provide a motivation to combine prior art references. Moreover, I understand that when there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a POSA would have good reason to pursue the known options within his or her technical grasp. I understand such circumstances may lead to solutions that are not deemed inventive.

25. I further understand that obviousness requires a POSA to have a reasonable expectation of success when modifying the prior art in a particular way, e.g., if a POSA can implement a predictable variation of a known method using common sense or the teachings of another reference to arrive at the claimed invention, a patent claim is likely obvious.

26. I also understand that in determining the obviousness or non-obviousness of the subject matter of an alleged invention, secondary considerations such as unexpected results, long felt but unsolved needs, failure of others, commercial success, etc., to the extent they exist, should be considered. I am informed that secondary considerations are relevant where there is a nexus between the evidence and the merits of the claimed invention, and where the asserted secondary considerations are commensurate in scope with the claimed invention, and not just directed to one particular embodiment of a broader patent claim.

27. I am informed that the patent owner bears the burden to establish any secondary considerations indicating non-obviousness.

IV. LEVEL OF ORDINARY SKILL

28. This declaration refers throughout to a person of ordinary skill in the art, or a “POSA.” As I understand the law, this is a hypothetical person who is presumed to have a comprehensive knowledge of the available prior art and a level of skill commensurate with a typical practitioner in the relevant field at the time the patent application at issue was originally filed. For the ’381 patent, I understand that date to be November 5, 2018.

29. I understand that a POSA is a person of ordinary creativity. He or she generally thinks along the line of conventional wisdom in the art and is not one who undertakes to innovate. However, when another work is available for use by the POSA, I understand the law recognizes that design incentives and market forces can also prompt that POSA to seek variations of that work in the same field.

30. In determining the level of skill of a POSA, I understand that I may consider the following factors: (1) the type of problems encountered in the art; (2) the prior art solutions to those problems; (3) the rapidity with which innovations are made; (4) the sophistication of the technology; and (5) the educational level of active workers in the field.

31. In my opinion, a POSA in the field of the '381 patent, as of November 5, 2018, would have a bachelor's degree in electrical or computer engineering and 3-5 years of academic or industry experience working with clocking techniques for FPGAs with high-speed transceivers. This person could also be someone with a lower degree of education but with greater practical experience. It could also be someone with a higher degree of education and lower practical experience.

32. The following are examples of knowledge or experience that the POSA would possess: (1) electrical engineering fundamentals, including logic and circuit analysis and design, including a solid understanding of combinatorial and sequential digital logic; (2) synthesis of RTL to netlists, and the mapping of the logic design / computational logic onto a specific FPGA, including use of specialized IO blocks like I/O cells, phase-locked loops (PLLs), clock trees, clock skew, clock domain crossings (CDCs)—where and when they are required, their impact on latency and system design characteristics, and what is required for their elimination; (3) the latency that results from clock speeds, pipeline depths, and CDCs, and the interrelationships between these. Metastability and the various approach of managing or eliminating it; (4) high-speed transceivers containing serializers, deserializers, CDR; (5) the primary internal components of a PLL and its overall operation, and the operational use of a PLL to generate clocks with certain frequency and phase properties among its inputs and outputs; (6) various tradeoffs, including

those between components and system cost; (7) the distinction between hard IP and soft IP; and (8) familiarity with the features of at least one family of FPGA such as described above, as documented by its manufacturer.

V. TECHNICAL BACKGROUND

33. Below, I present background information about the state of the art that is relevant to the '381 patent.

A. FPGAs

34. The FPGA (“field programmable gate array”) is a versatile integrated circuit that fills the performance gap between a computer system running software and an application-specific integrated circuit (ASIC) that is designed for a specific purpose. An FPGA design can be modified (it’s “field programmable”) and is therefore more flexible than an ASIC. It can be deployed quickly, in hours or days instead of the months or years required for an ASIC. Like an ASIC, an FPGA design can achieve dramatically reduced latency compared with a general-purpose computer system. “FPGAs offer many benefits over software solutions, including a reduction in latency. . . Experiments show that the peak performance of our hardware architecture for algorithmic trading is 133 times faster than the corresponding software implementation.” *See* Wray.¹

¹ Ex. 1018, S. Wray, W. Luk and P. Pietzuch, “Exploring algorithmic trading in reconfigurable hardware,” ASAP 2010 - 21st IEEE International Conference on

35. For commonly used or complex functions, like high-speed transceivers, FPGAs contain ASIC-like implementations on-chip, for efficiency and cost effectiveness. But the primary value of an FPGA is the large regions of logic functions and interconnect / clock networks that are user-programmable, allowing the designer to realize combinatorial and sequential application logic which is specified using a hardware description language (HDL) such as Verilog or VHDL. *See* Smith.² And while the unit cost of an FPGA is higher than the equivalent ASIC, their flexibility for lower-volume applications makes them an attractive choice over ASICs. Likewise, compared with a general-purpose computer, FPGAs are a better choice for applications that require very low latencies, massively parallel operations, or other customizations not feasible or economical in a general-purpose CPU. *See* Ex. 1016, Aldridge at 23.³

36. The programmable portion of the FPGA has multiple names, including: logic fabric, FPGA core, FPGA logic core, computational circuitry, computational logic, FPGA fabric, core fabric, and programmable logic, among others.

Application-specific Systems, Architectures and Processors, Rennes, France, 2010, pp. 325-328, doi: 10.1109/ASAP.2010.5540966.

² Ex. 1019, HDL Chip Design. A practical guide for designing, synthesizing, and simulating ASICs and FPGAs using VHDL and Verilog, by Douglas J Smith, 1996 by Doone Publications

³ Ex. 1016, High Frequency Trading, Irene Aldridge, 2013

37. For FPGA-based systems directly attached to a network, one or more FPGA transceivers (each of which is typically composed of both a receive and transmit path) pass received network packets of data to the programmable logic fabric, which then performs computations and other logic operations on the contents of the packets, potentially also referencing and/or updating other state stored in local SRAM memories. This avoids relying on a host computer to mediate all communications. When the application logic decides to transmit a network packet, it passes the packet contents, some number of bits at a time, to one or more transmit interfaces of the appropriate transceiver(s), which send the information over the fiberoptic or electrical cable to be delivered to the destination by the network equipment. *See* Ex. 1001, '381 patent at Figs. 9A-C. The functions that can be implemented in the FPGA's computational logic are effectively infinite in number and include arithmetic, pipeline, and logic operations. And when additional data storage is needed, designs write to and read from the various embedded SRAM memories of the FPGA.

38. Because clocking is fundamental to any modern digital design, and since FPGAs are intended for use in a wide spectrum of designs, FPGAs provide capabilities that allow customization of where the clocks come from (internally generated or externally sourced), customization of the path through which the clocks

are routed within the chip, support for multiple clocks, and the option of using phase locked loops (PLLs) to transform clocks such that desired characteristics are met.⁴

39. FPGAs became commercially available in 1985 when Xilinx released the XC2064⁵ and by the early 2010s the two largest FPGA vendors, Xilinx and Altera, made up a substantial portion of the global market, 49% and 40% in 2011, respectively, for a combined revenue of \$4.3B.⁶ Both support highly configurable transceivers on-chip as well as large arrays of programmable logic gates and sophisticated clock networks. The extensive capabilities of these devices led FPGA vendors to publish hundreds of pages of technical documentation for the engineer to use while designing an FPGA-based system.⁷

40. Of the many applications in which FPGAs are employed, their use in financial services and high-frequency trading in particular is well established. This, in large part, is a result of the potential reduction of latency between, for example, receipt of market information on an Ethernet port (an FPGA transceiver configured

⁴ Ex. 1020, UltraScale Architecture Clocking Resources, User Guide, Xilinx April 9, 2018

⁵ Ex. 1021, <https://spectrum.ieee.org/chip-hall-of-fame-xilinx-xc2064-fpga>

⁶ Ex. 1022, <https://www.fpgadeveloper.com/2011/07/list-and-comparison-of-fpga-companies.html/>

⁷ Ex. 1005, Stratix V Device Handbook

to implement the Ethernet standard) and the transmission of trade requests to exchanges and other trading venues.⁸

B. Clock Domain Crossing (CDC)

41. Synchronous digital logic is the foundation of all modern integrated circuits. It uses a clock and a set of logical operations arranged in a pipeline where each pipeline stage collects the intermediate results from the prior state and passes them to the next stage at the next clock.

42. As discussed above, FPGA circuits receive data, execute operations, and transmit the resulting data off the device. For an FPGA to operate properly, one must take care when data is transferred from one clock domain to another. A clock domain is a region where all logic elements operate synchronously using the same clock signal. A clock-domain crossing circuit is one way to ensure data crosses between clock domains without metastability, data loss, or other issues. However, CDC operations typically have a latency of several clock cycles.

43. As background, a clock domain crossing circuit is used to mitigate the potential negative effects of metastability in synchronous digital logic—the

⁸ Ex. 1018, Stephen Wray, et al, Imperial College London, “Exploring Algorithmic Trading in Reconfigurable Hardware,” IEEE International Conference on Application-specific Systems, Architectures and Processors, 2010

condition where the output of a flip-flop persists at an intermediate voltage that is neither a one nor a zero.⁹

44. A common storage element is the “D” flip-flop, a circuit that has a data input, a clock input, a reset input, and a data output. When the voltage at its clock input changes from zero to one, known as a rising edge, the flip-flop stores and transfers to its output the one-bit value presented at its data input. In this context, a bit is a single binary digit, either a “zero” or a “one,” designated by a voltage below one threshold or a voltage above a different threshold, respectively. Metastability in a flip-flop can occur if the data input changes too close to the clock transition. The duration before the clock transition during which the data must be stable is called the “setup time” and the time during which the clock must remain stable after the clock is called the “hold time.” Violating either can result in metastability of that flip-flop, which can cause metastability in subsequent logic, resulting in malfunctioning computational logic.

45. Consider two consecutive flip-flops. The first generates an output soon after seeing the rising edge of its clock, and that output value passes through a variety of logic gates, arriving eventually at the data input of a second flip-flop. Where both flip-flops are clocked by the same clock, metastability is avoided because the amount

⁹ Ex. 1023, <https://www.eetimes.com/understanding-clock-domain-crossing-issues/>

of logic is adjusted to ensure that the input of the flip-flop (with respect to the clock) violates neither setup time nor hold time.

46. However, if the second flip-flop is clocked by a different clock, one that is not carefully controlled, a clock domain crossing (CDC) circuit is typically instantiated to deal with the possibility that the second flip-flop will go into metastability. The canonical CDC consists of two or more back-to-back flip-flops which copy the bit from one stage to the next, adding on average one clock cycle per stage to the overall latency of the design, and dedicating the full period of the clock to allow the output of the flip-flop to settle to a one or a zero. Since metastability is statistically like to happen when clocks are independent, the use of two or more flip-flops as a clock domain crossings circuit isolates the metastability, providing a high probability that the metastability, if it persists for one clock period on the output of the first flip-flop, will settle to a stable state on the output of the second flip-flop within a clock period, ensuring that no computational logic is in the path of the potentially metastability. Sometimes the term “synchronizer” is used for a CDC.

C. PLLs

47. Information can be moved from one clock domain to another without a CDC. For example, if the frequencies of the clocks are synchronized, the phase relationship between the clocks is constant over time and the phase of the second

clock with respect to the first clock can be adjusted, e.g. by a PLL, such that metastability is avoided altogether.

48. Within the field of digital system design (as opposed to, for example, RF or Radio Frequency system design), a phase locked loop (PLL) is an electronic circuit or device used to generate or condition a clock signal. A PLL “contains three essential elements . . . (1) a phase detector (PD), (2) a loop filter (LF), and (3) a voltage-controlled oscillator (VCO).”¹⁰

49. In this context, a clock signal is modeled as a square wave of some frequency oscillating between two voltage levels and used to trigger a set of flip flops or other storage elements to capture the zero-or-one bit values on their inputs, transferring their input to their output after some delay, among other uses.

50. In some designs (e.g., those containing high-speed transceivers), it is necessary to generate a clock that has some desired relationship or alignment to another clock and for this, a PLL is often employed.

51. The PLL can multiply the frequency of a clock by some whole number ($f_{\text{OUT}} = f_{\text{IN}} \times N$) or by a factor that is the sum of a natural number and the ratio of two natural numbers ($f_{\text{OUT}} = f_{\text{IN}} \times (N + K/F)$). The latter device is commonly referred to as a Fractional PLL or fPLL. PLLs are also used to attenuate certain ranges of

¹⁰ Ex. 1024, Phase Lock Techniques, 3rd Edition at page 1.

frequencies in their output, e.g. attenuate frequencies above some “cut-off” frequency.

52. In addition to frequency attenuation, PLLs also adjust the phase of the output such that some criteria are met. For example, the phase of a clock made to align with that of another clock. “The dynamic phase shifting feature allows the output phases of individual fractional PLL outputs to be dynamically adjusted relative to each other and to the reference clock.”¹¹

53. Structurally, the canonical PLL consists of several functions arranged into a closed negative feedback loop. This includes a phase detector, which compares the input signal to the feedback signal and generates an error signal that is a function of their difference in phase. This error signal passes through a filter to suppress, or filter out, certain frequencies (e.g. a low-pass filter suppresses frequencies higher than the cutoff frequency, or bandwidth), and the resulting signal is used to generate the output clock, which is fed back to the feedback input of the phase comparator through a block that, for example, divides the frequency by two if the design calls for an output frequency that is twice the input frequency. An fPLL contains additional circuitry to allow non-integer (but rational) frequency multipliers.

¹¹ Ex. 1017, *Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig Megafunctions* at 9 (referenced in the Stratix Handbook at 363).

54. In short, the PLL is a control loop that uses negative feedback to generate an output clock with frequency and phase that drives the error signal to zero.

D. Techniques for Low-Latency Trading

55. The first computer-generated trade was performed in 1987. “Thomas Peterffy, founder and chairman of Interactive Brokers Group, created the first fully automated algorithmic trading system in 1987. It was a computer, plugged into a Nasdaq terminal, that could place orders faster than any human. Soon after the system started running, a Nasdaq representative told Peterffy to shut it down. All orders had to be manually typed, the representative said.”¹² Once the use of computers to execute trades was inevitable, a race ensued to both increase the sophistication of the trading algorithms and to reduce the latency between market knowledge and trade initiation.¹³

¹² Ex. 1025, https://www.barrons.com/articles/interactive-brokers-peterffy-automated-securities-trading-4b3c5a19?reflink=desktopwebshare_permalink

¹³ Ex. 1026, <https://rubinow.com/2007/04/21/wall-streets-quest-to-process-data-at-the-speed-of-light/>, Wall Street’s Quest To Process Data At The Speed Of Light, 2007, an article referenced in “Exploring Algorithmic Trading in Reconfigurable Hardware, IEEE International Conference on Application-specific Systems, Architectures and Processors, 2010 below.

Exploring Algorithmic Trading in Reconfigurable Hardware

Stephen Wray, Wayne Luk, Peter Pietzuch
Department of Computing
Imperial College London
London, United Kingdom
{sjw06,wl,pp}@doc.ic.ac.uk

Abstract—This paper describes an algorithmic trading engine based on reconfigurable hardware, derived from a software implementation. Our approach exploits parallelism and reconfigurability of field-programmable gate array (FPGA) technology. FPGAs offer many benefits over software solutions, including a reduction in latency, while increasing overall throughput and computational density. All of which are important attributes to a successful algorithmic trading engine. Experiments show that the peak performance of our hardware architecture for algorithmic trading is 133 times faster than the corresponding software implementation. Six implementations can operate simultaneously on a Xilinx Vertex 5 xc5vk30 FPGA on average, maximising performance and available resource usage.

Keywords—Algorithmic Trading; Reconfigurable Hardware;

I. INTRODUCTION

Electronic trading enables trades to be completed in a virtual environment by computers with orders being communicated across computer networks. This revolution started in the 1970s and by 2007, 60% to 70% of the volume traded on the New York Stock Exchange (NYSE) was done so electronically [1]. In 1961, the average daily traded volume on the NYSE was four million, however by January 2001, daily volume was already topping two billion trades a day [2].

The increase in electronic trading had a large effect on how the equity markets reacted and systems designed to operate in these environments had to possess certain attributes to be successful:

- Orders now take a fraction of the time they previously took to reach the market, reduced from minutes to milliseconds, making system latency critical.

These attributes align with those that reconfigurable hardware provides making it an ideal solution, it has previously been discussed in some detail [4], but also explains why there hadn't been much market penetration. Some of those issues have now been solved and our approach maps a selection of algorithms into reconfigurable hardware to take advantage of these qualities. We have used a software solution to provide a comparison against our results. The contributions of this paper include the following:

- An introduction to algorithmic trading and the specifics examined in this paper; in Section II.
- An algorithmic trading architecture based on reconfigurable hardware that processes available information to decide whether an order should trade; in Section III.
- Implementation details and initial results from the hardware architecture proposed, compared against a comparative software implementation; in Section IV.
- Evaluation of the proposed approach and comparison with a related software solution, illustrating its potential; in Section V.

Currently, Credit Suisse has a joint venture with Celoxica to use reconfigurable hardware in their electronic trading systems [5], showing that these technologies are used in industry. Academic research on this subject has been limited however. One study [6] which is particularly relevant explores parsing market data feeds directly from a network interface in hardware. The study found that the hardware implementation could process 3.5 million updates per second, or roughly 12 times the current real-world maximum rate as well as improved latency.

II. ALGORITHMIC TRADING

56. The logic of early trading systems was constructed using standard PCs and custom software. A lot of effort was made to reduce the delay between a packet arriving at the PC from the network and the processing of that packet by software, but software solutions are incapable of providing the low latency necessary for many modern trading strategies. As a result, some trading firms were driven to find other solutions to augment or replace the software. “In almost all such applications, a key

requirement is low latency processing, as the market is continually changing, so decisions should be made based on the most up to date data. For example, an opportunity for instrument arbitrage only exists until the first trader is able to exploit it, so it is critical to minimise the time between the information becoming available and the execution of the appropriate response.”¹⁴ FPGAs have provided “a very attractive means of acceleration, as they are a mature technology, with low power and space requirements.”¹⁵ They are able to connect directly to the network, to parse and filter the data feed, and then push messages to a processor. This significantly reduces the latency of initiating a trade.

57. FPGA vendors added features into their FPGAs to reduce the latency, by allowing users to customize the logic applied to certain functions of the FPGA’s transceivers or to skip those functions entirely. Transceivers deserialize received data streams into bytes and multi-byte ‘words,’ and conversely serialize the outgoing data words into bits that are transmitted out the same or separate network interface. Therefore, eliminating or customizing the functions of the transceivers can reduce the time necessary for data to be processed.

58. For example, the Stratix V FPGA used in the Altera white paper (discussed more below) includes a “PMA Direct” feature that “is used to support

¹⁴ Ex. 1027, FPGA accelerated low-latency market data feed processing” by Gareth W. Morris, et al., 17th IEEE Symposium on High Performance Interconnects, 2009

¹⁵ *Id.*

protocols that require extremely low or zero transceiver PCS latency.” Stratix Handbook at 231.

59. For background, the PMA is the physical medium attachment, which “receives and transmits off-chip high-speed serial data streams.” Ex. 1005, Stratix Handbook at 14. The PMA also converts received serial data to parallel data and then converts parallel data to serial data for transmission off the chip. *Id.* at 6. The PCS is the physical coding sublayer, which “performs digital processing logic between the PMA and the FPGA core.” *Id.*

60. As noted in the Stratix Handbook, “there are no PCS blocks in the PMA Direct configuration, so clock phase compensation must be designed in the fabric core. Data and clock signals are interfaced directly to the transceiver PMA. Consequently, you must also compensate for the timing and clock phase differences from the core fabric interface of the FPGA to the transceiver PMA,” Ex. 1005, Stratix Handbook at 231. Similarly, the ’381 patent acknowledged that “[i]n some FPGAs, the PCS of the transceiver may be bypassed and [sic] so that the corresponding PCS functions may be done in the FPGA core.” Ex. 1001, ’381 patent at 10:7-9.

61. Companies were in fierce competition to obtain even a slight edge because even the smallest latency advantage can be extremely profitable. “The goal of coding is to accurately translate the logic of the algorithm into computer ‘speak’

and, in the process, to create as little delay as possible during ‘run time,’ when the computer will read and interpret the code. The code written directly to FPGA chips is presently the fastest.”¹⁶

VI. OVERVIEW OF THE ’381 PATENT

A. The Specification of the ’381 Patent

62. The ’381 patent, titled “Field Programmable Gate Array with External Phase-Locked Loop,” names Nima Badizadegan as the inventor. The ’381 patent was filed on April 18, 2022, and claims an earliest filing date of November 5, 2018. The ’381 patent issued on February 7, 2023. Its abstract says “[t]he present invention relates to a field programmable gate array system that provides phase control with minimal latency.” Ex. 1001.

63. Figures 2, 7A, 7B, and 7C provide exemplary block diagrams of FPGA systems containing an FPGA and an external PLL. The example shown in Figure 2 has a single deserializer and a single serializer, whereas the examples in 7A-7C include multiple deserializers and multiple serializers.

¹⁶ Ex. 1016, *High-Frequency Trading: A practical Guide to Algorithmic Strategies and Trading Systems, Second Edition*, by Irene Aldridge (2013).

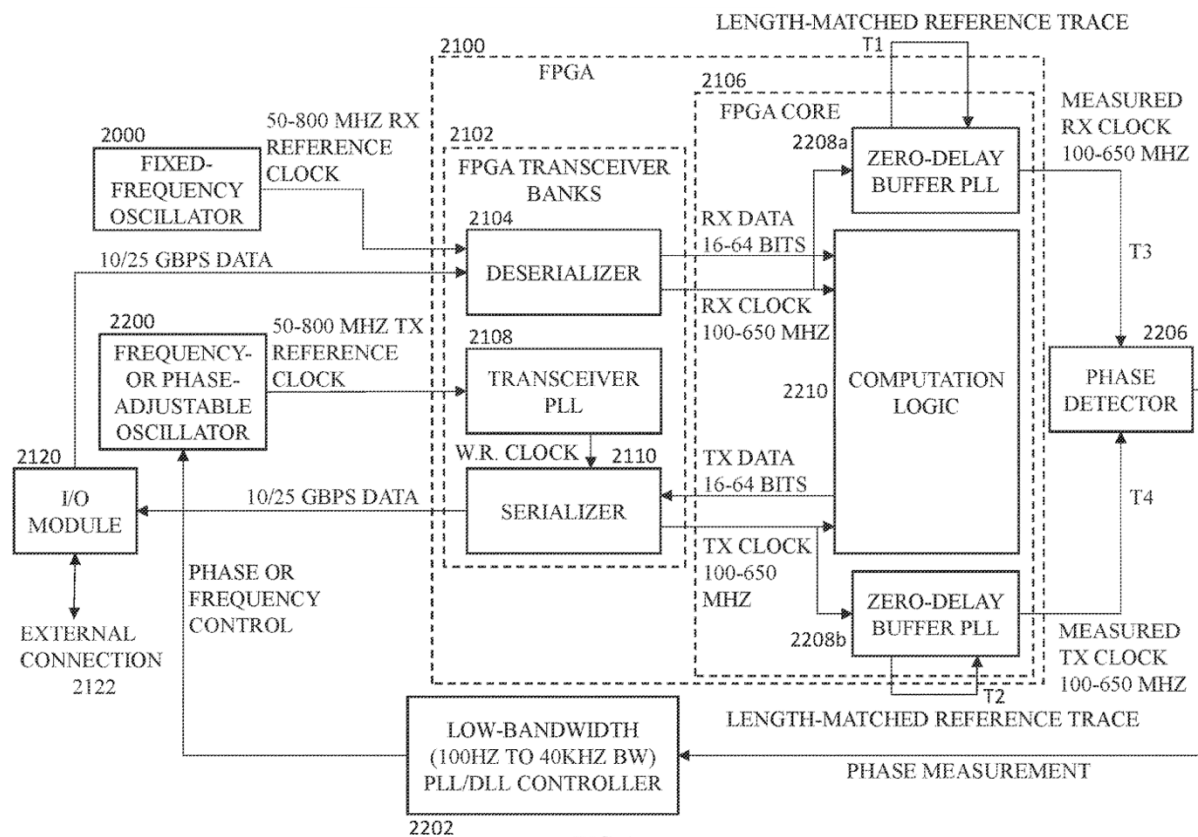


FIG. 2

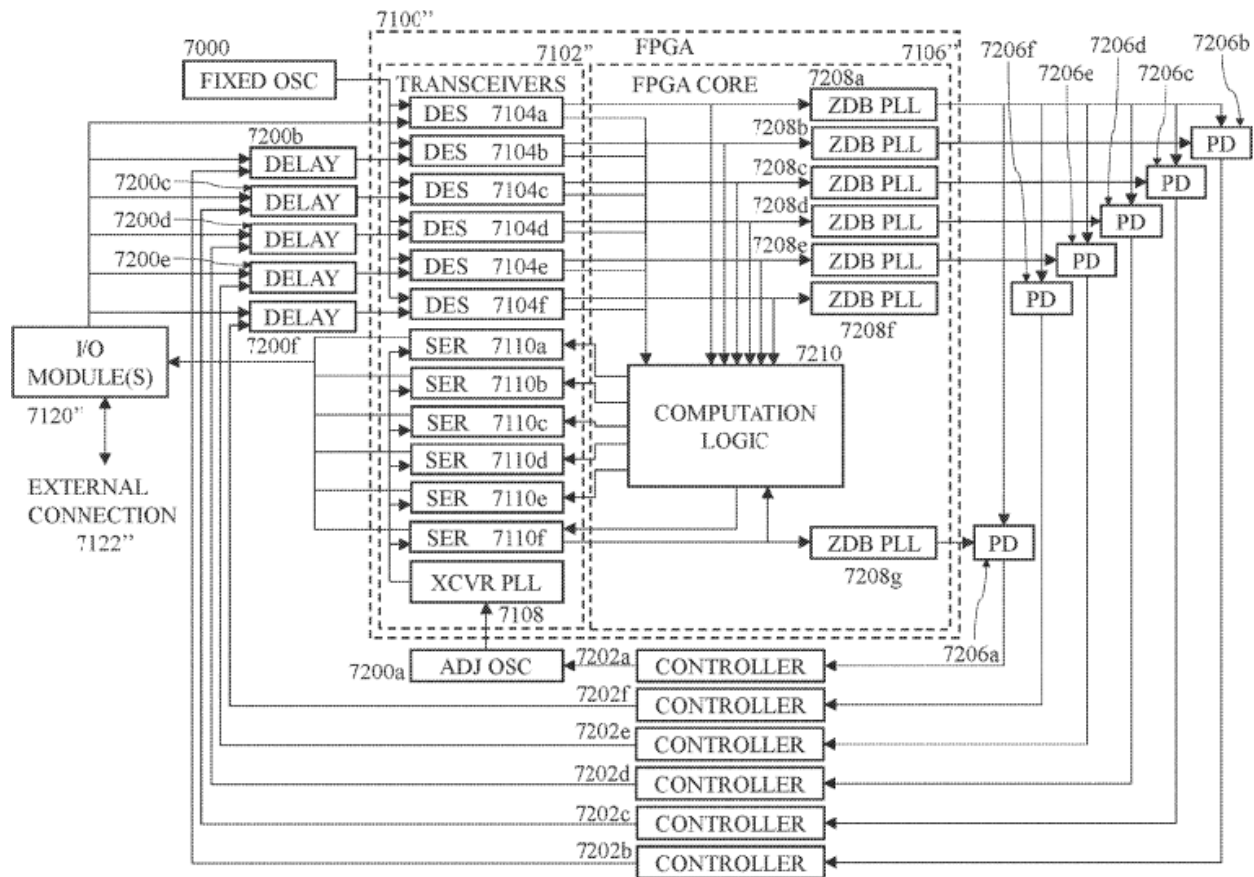


FIG. 7C

64. The patent also includes examples of deserializers in Figures 3A-3E. “FIG. 3A is an exemplary block diagram of a deserializer suitable for use in the field programmable gate array system of FIG. 2 in accordance with an embodiment of the present invention. Deserializer 2104', in embodiments, may include an input buffer 3104, a continuous time linear equalizer (CTLE) 3106, a variable gain amplifier (VGA) 3108; voltage adder 3110; decision feedback equalizer (DFE) 3110, clock and data recovery (CDR) 3116, Serial to parallel box 3118, decision circuit 3112, and clock divider 3120. In embodiments, deserializer 2104' may not include one or

more of the following components: CTLE 3106, DFE 3114, and/or VGA 3108, to name a few.” Ex. 1001, '381 patent at 16:15-26.

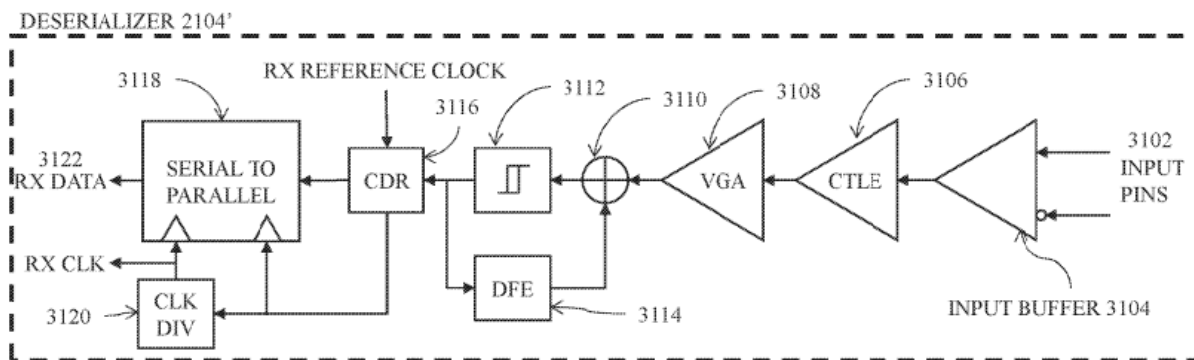


FIG. 3A

65. The patent explains the operation of the deserializer as follows: “A deserializer circuit accepts a data stream from an input buffer and converts it into a parallel format at a lower bit rate. First, data comes into the chip from the input pins 3102 into an input buffer 3104. In embodiments, the input buffer receives a transmission from an I/O module, such as I/O module 2120. A continuous time linear equalizer (CTLE) 3106 may be used after the input buffer 3104 to improve signal quality by placing a zero near the dominant pole of the transmission medium to cancel the first pole of the system. A variable-gain amplifier (VGA) 3108 may then be used to scale the signal up for further processing. This output is then turned into a binary 0-1 decision by the decision circuit 3112. In embodiments, the decision circuit 3112 includes adder 3110. A decision feedback amplifier may be included to further improve signal integrity by taking the results of the decision circuit, applying a weighting filter, and adding the result to the output from the VGA 3108. When the

signal is in the digital domain, a clock and data recovery circuit (CDR) 3116 recovers the phase of the data and create [sic] a wire-rate clock that is phase-aligned to the received data. The CDR 3116 may use a reference clock and a priori knowledge of the nominal frequency of the data stream to set the frequency of the recovered clock. With a clock aligned to the data stream, which in some embodiments may be a high speed data stream (e.g. 10 Gbps, 25 Gbps, a range from 9-25 Gbps, 10-25 Gbps, 22-33 Gbps, 33-45 Gbps, 45-60 Gbps, 60-80 Gbps, 80-120 Gbps, or 1 Gbps-100 Gbps, to name a few) the receiver can use a clock divider 3120 and a serial-to-parallel circuit 3118 to convert the serial data stream into a lower-frequency than the [sic] data stream (e.g. $\frac{1}{4}$ th of the data rate of the high speed data stream or $\frac{1}{256}$ th of the data rate of the high speed data stream, to name a few) parallel data stream [sic].” Ex. 1001, ’381 patent at 16:27-59.

66. “FIG. 3E illustrates an exemplary embodiment of deserializer 2104”. In embodiments, the deserializer 2104” may include a receiver PMA 3512 and receiver PCS 3508. The receiver PMA 3512 may include receiver buffer 3504, CDR 3516, and serial to parallel box 3118’. In embodiments, the receiver buffer 3504 receives serial differential input data 3502. In embodiments, this data is received by an I/O module, such as the I/O module 2120 of FIG. 2. In embodiments, the data is then received by the CDR 3516, from the receiver buffer 3504. The CDR, in embodiments, transmits the inputted serial data, a serial clock signal, and a parallel

clock signal to the serial to parallel box 3506. The CDR may also transmits [sic] the parallel clock data to the receiver PCS 3508. The serial to parallel box 3118', in embodiments, then converts the serial data into parallel data, transmitting the parallel data to the receiver PCS 3508. The receiver PCS 3508 transmits parallel data to the FPGA Core 2106." Ex. 1001, '381 patent at 17:33-56.

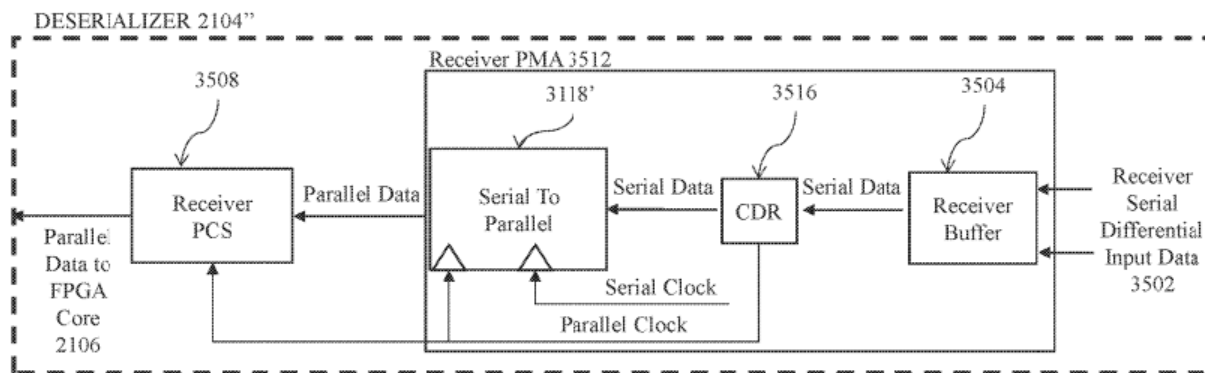


FIG. 3E

67. The patent similarly provides examples of the serializer in Figures 4A and 4B. "FIG. 4A is an exemplary block diagram of a serializer suitable for use in the field programmable gate array system of FIG. 2 in accordance with an embodiment of the present invention. The serializer circuit accepts a clock running at wire rate 4112 (from transceiver PLL 2108) and accepts parallel data (e.g. TX data 4102) to serialize the data. Data is converted from a parallel format to a serial data stream in the parallel to serial box 4104. A feed-forward equalizer (FFE) 4106, which uses the history of bits serialized on the wire to filter out inter-symbol interference, may be included to improve signal integrity. Finally, the serializer

outputs the serial data stream to a transmission medium using an output buffer 4114. The serializer also generates a parallel clock (TX Clk 4110), that operates at a lower speed than the data stream (e.g. 100-400 MHz, 100-600 MHz, to name a few), for use by other components using either a simple clock divider 4108 or a more complex clock generation block which can contain dividers and phase shifting elements for aligning multiple transmitters.” Ex. 1001, ’381 patent at 17:57-18:29.

68. “FIG. 4B illustrates an exemplary embodiment of serializer 2110”. As discussed above with respect to FIGS. 2 and 4A, serializer 2110” receives parallel data streams from the FPGA core 2106 and serializes them for transmission out of the FPGA 2100. In embodiments, serializer 2110” may include transmitter PCS 4206, clock generation block 4210, and transmitter PMA 4216. In embodiments, transmitter PMA 4216 may include parallel to serial box 4104’ and transmitter buffer 4214. The transmitter PCS 4206 may receive parallel data from FPGA core 2106. In embodiments, the parallel data is then transmitted to the parallel to serial box 4104’. The clock generation block 4210 may receive a wire-rate clock signal 4112. The clock generation block 4210, in embodiments, may split the signal in to [sic] a parallel clock signal and serial clock signal, both the parallel and serial signals being transmitted to the parallel to serial box 4104’. The parallel to serial box 4104 may covert [sic] the data from parallel to serial, transmitting the serial differential output data 4218 to the transmitter buffer 4214. Then, in embodiments, the transmitter

buffer 4214 outputs the serial differential output data 4218.” Ex. 1001, ’381 patent at 17:57-18:29.

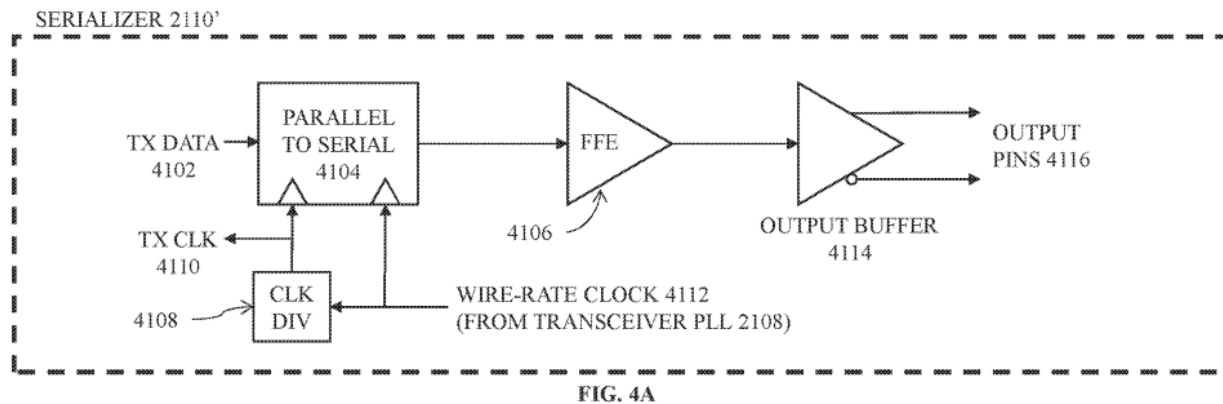


FIG. 4A

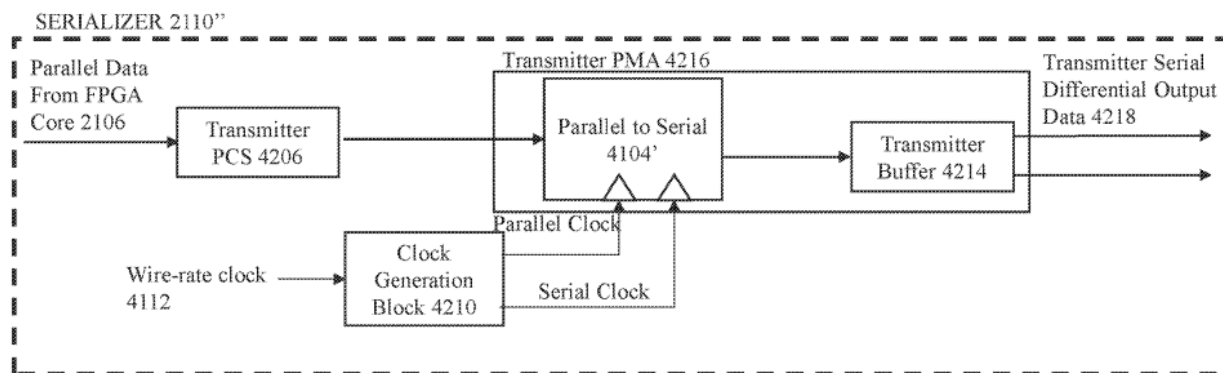


FIG. 4B

69. The patent illustrates a design containing a clock domain crossing (CDC) circuit within the FPGA CORE of an FPGA. “In embodiments, a REFERENCE CLOCK signal is provided, by Oscillator or Clock Generator 122, to both the deserializer 104 and the serializer 110. The REFERENCE CLOCK signal is received by the serializer 110 via the transceiver PLL 108 (Phase-Locked Loop). The receiver side clock signal RXCLOCK, however, is different in both frequency and phase from the REFERENCE CLOCK based on the deserialization process that

occurs in the deserializer 104. Similarly, the transmitter clock signal TXCLOCK, while based on the REFERENCE CLOCK signal varies in frequency and phase based on processing that occurs in the serializer 110. As a result, the RXCLOCK signal and TXCLOCK signal will be out of phase. A known solution to this issue is the inclusion of the RX to TX clock domain crossing circuit 112. An exemplary Clock Domain Crossing circuit is suitable for use in FPGA 100 is shown as clock domain crossing circuit 112' in connection with FIG. 1B. As is noted above, the inclusion of the RX to TX clock domain crossing circuit 112 creates a technical problem, introducing an inherent delay in the FPGA 100, which is undesirable. The technical problem of an inherent delay is illustrated in FIG. 1C. FIG. 1C illustrates an example of a phase difference between the receiver side clock and the transmitter side clock of the conventional field programmable gate array of FIG. 1.” As stated, the RXCLOCK clock and the TXCLOCK that would result from a configuration shown in FIG. 1 would “be out of phase.”

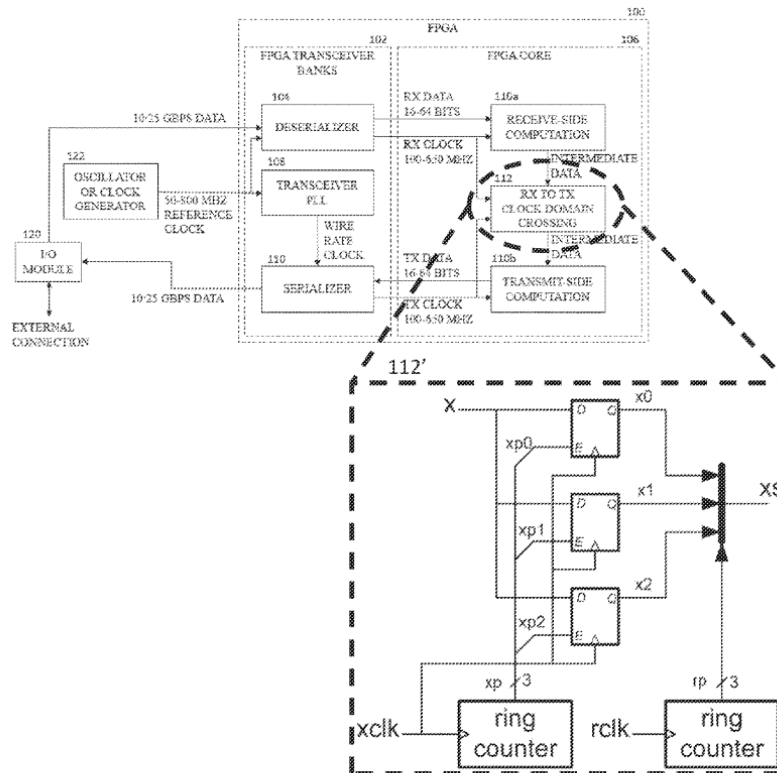


FIG. 1B

70. The patent states that “a significant drawback of the clock domain crossing circuit is that it adds latency . . . such that it slows the effective processing speed of the FPGA.”

B. The Claims of the '381 Patent

71. Claims 1-4 and 6-12 of the '381 patent are the focus of my analysis. Those claims recite the following:

1. A method for processing a first serial data stream, using a field programmable gate array system, to generate a second serial data stream, wherein the method comprises the steps of:
 - (a) receiving, by a deserializer in a field programmable array, a clock signal;

- (b) receiving, by the deserializer, the first serial data stream;
- (c) generating, by the deserializer, a receiver side clock signal;
- (d) converting, by the deserializer, the first serial data stream into a first plurality of parallel data streams;
- (e) transmitting, from the deserializer to computational circuitry in the field programmable gate array, the first plurality of parallel data streams;
- (f) transmitting, from the deserializer to a phase lock loop of the field programmable gate array system that is not within the field programmable gate array, the receiver side clock signal;
- (g) generating, using the phase lock loop, a second clock signal;
- (h) generating, within the field programmable gate array, a transmitter side clock signal derived from the second clock signal;
- (i) performing, by the computational circuitry, a set of operations on at least a portion of the first plurality of parallel data streams to generate a second plurality of parallel data streams; and
- (j) transmitting, from the field programmable gate array system, the second serial data stream, derived from the second plurality of parallel data streams,

wherein said method does not use clock domain crossing operations that delay processing of the first set of parallel data streams.

2. The method of claim 1, wherein the first serial data stream comprises market data, the second serial data stream comprises order entry data, and the set of operations are associated with a trading algorithm.
3. The method of claim 1, wherein the first serial data stream includes market data and the second serial data stream includes trading data.
4. The method of claim 1, wherein the set of operations includes at least one of the following:
 - (i) an arithmetic operation;
 - (ii) a logical operation;

- (iii) a pipeline operation; and
 - (iv) a memory access operation.
6. The method of claim 1, wherein at least a portion of the set of operations performed in step (h) are performed after step (g).
 7. The method of claim 1, wherein all of the set of operations performed in step (h) are performed after step (g).
 8. The method of claim 1, wherein step (g) further comprises:
 - (i) generating, by the phase lock loop, a feedback clock signal associated with the transmitter side clock signal by performing the following steps until a first output of a phase detector of the field programmable gate array system is below a first threshold level:
 - (1) generating, by an adjustable oscillator in the field programmable gate array system, the second clock signal;
 - (2) generating, by the phase detector, the first output based on a comparison of the receiver side clock signal and the feedback clock signal obtained from the second clock signal;
 - (3) transmitting, from the phase detector to a phase controller of the field programmable gate array system, the first output;
 - (4) determining, by the phase controller, interim adjustment information based on the first output; and
 - (5) transmitting, from the phase controller to the adjustable oscillator, the interim adjustment information;
- wherein, the adjustable oscillator adjusts the second clock signal based on the interim adjustment information,
- wherein steps (1) through (5) are repeated until the first output of the phase detector is below the first threshold level.
9. The method of claim 1, wherein the first plurality of data streams has the same number of data streams as the second plurality of data streams.

10. The method of claim 1, wherein the first plurality of data streams and the second plurality of data streams comprise one of the following:

- (i) eight (8) data streams;
- (ii) ten (10) data streams;
- (iii) sixteen (16) data streams;
- (iv) twenty (20) data streams;
- (v) thirty-two (32) data streams;
- (vi) forty (40) data streams;
- (vii) sixty-four (64) data streams;
- (viii) eighty (80) data streams;
- (ix) one hundred twenty-eight (128) data streams; and
- (x) one hundred sixty (160) data streams.

11. The method of claim 1, wherein the transmitter side clock signal and the receiver side clock signal have the same frequency and phase.

12. The method of claim 1, wherein the transmitter side clock signal and the receiver side clock signal have the same frequency and different phases.

C. The Prosecution History of the '381 Patent

72. I reviewed the prosecution history of the '381 patent. Ex. 1003. I understand from that review that there were no anticipation or obviousness rejections of the claims.

VII. CLAIM CONSTRUCTION

73. I understand that the words used in patent claims are to be given their plain and ordinary meaning, as would be understood by the POSA at the time of the

claimed invention, unless an applicant provides a particular definition of the words used in the specification of the patent.

74. I understand that the plain and ordinary meaning should be apparent from the content of the '381 patent, and (if necessary) the record of correspondence between the applicant and the USPTO, which I understand to be the intrinsic record. I have followed these principles in my analysis to the extent necessary throughout this declaration

75. Where the challenged claims refer to a “deserializer,” I have adopted the specification’s description that the deserializer may include a receiver PMA and receiver PCS, specifically as depicted in the diagrams of the patent.

76. Most importantly, Figure 3E provides a depiction of the deserializer and specifically includes the receiver PMA and receiver PCS.

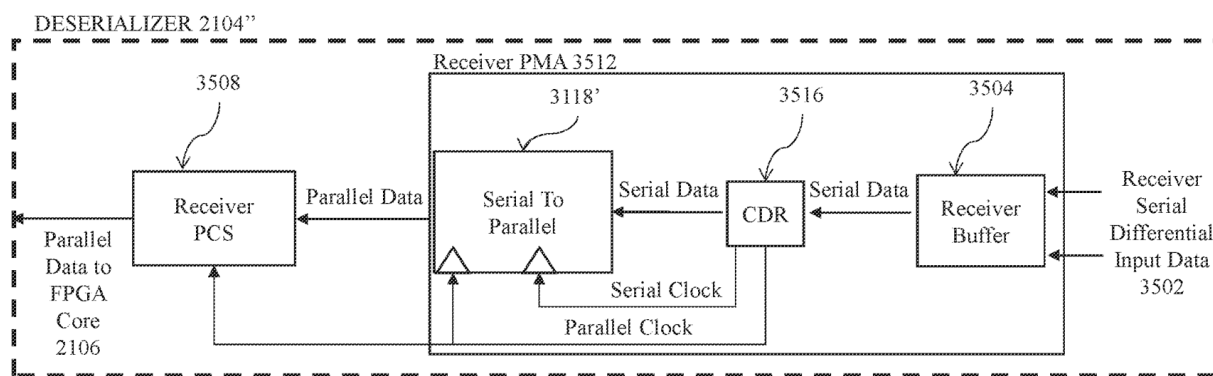
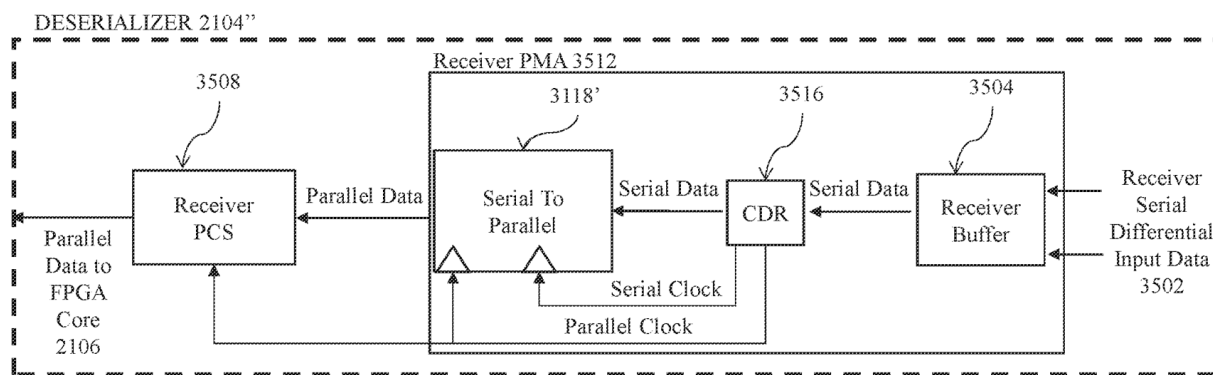


FIG. 3E

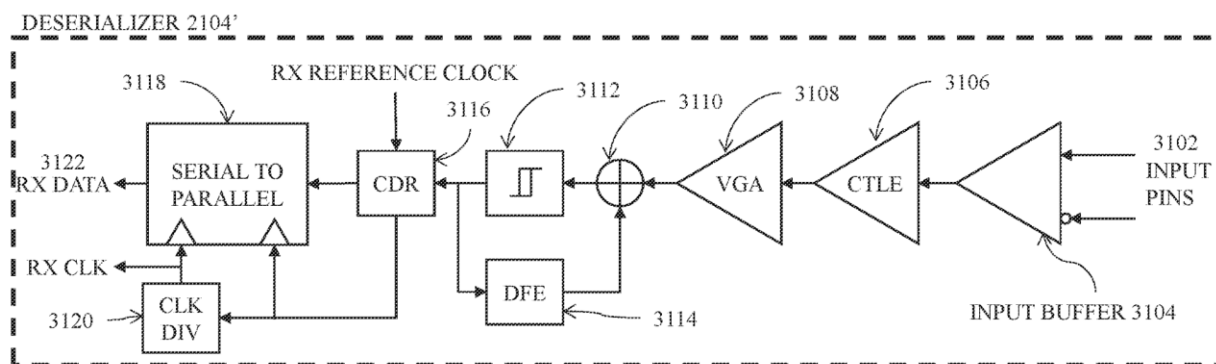
77. As described above, the PMA (Physical Medium Attachment) is the layer that processes the physical electrical signal and outputs parallel data into the receiver PCS (Physical Coding Sublayer), which processes the parallel data and

prepares it for user logic. The '381 patent explains that the “deserializer circuit accepts a data stream from an input buffer and converts it into a parallel format at a lower bit rate.” Ex. 1001, '381 patent at 16:27-29. The deserializer circuit includes, but is not limited to, “a serial-to-parallel circuit.” *Id.* at 16:27-59.

78. With respect to Figure 3E, the patent explains that Figure 3E provides an “exemplary block diagram of a deserializer suitable for use in the field programmable gate array system of FIG. 2.” *Id.* at 8:23-24. In this embodiment, “the deserializer 2104” may include a receiver PMA 3512 and receiver PCS 3508. The receiver PMA 3512 may include receiver buffer 3504, CDR 3516, and serial to parallel box 3118’.” *Id.* at 17:34-37. As the patent explains: “In embodiments, the receiver buffer 3504 receives serial differential input data 3502. In embodiments, this data is received by an I/O module, such as the I/O module 2120 of FIG. 2. In embodiments, the data is then received by the CDR 3516, from the receiver buffer 3504. The CDR, in embodiments, transmits the inputted serial data, a serial clock signal, and a parallel clock signal to the serial to parallel box 3506. The CDR may also transmits [sic] the parallel clock data to the receiver PCS 3508. The serial to parallel box 3118’, in embodiments, then converts the serial data into parallel data, transmitting the parallel data to the receiver PCS 3508. The receiver PCS 3508 transmits parallel data to the FPGA Core 2106.” *Id.* at 17:37-49.



79. Figure 3A also provides “an exemplary block diagram of a deserializer suitable for use in the field programmable gate array system of FIG. 2 in accordance with an embodiment of the present invention. Deserializer 2104’, in embodiments, may include an input buffer 3104, a continuous time linear equalizer (CTLE) 3106, a variable gain amplifier (VGA) 3108; voltage adder 3110; decision feedback equalizer (DFE) 3110, clock and data recovery (CDR) 3116, Serial to parallel box 3118, decision circuit 3112, and clock divider 3120.” *Id.* at 16:15-23.



80. A POSA would understand from the descriptions and figures that the term “deserializer” as used in the ’381 patent refers to more than just the serial to parallel circuit (e.g., box 3118)—which is sometimes also referred to as a

deserializer in this field. Instead, a “deserializer” in the context of this patent includes additional components, such as everything contained within the receiver PMA and PCS. Likewise, the term “serializer” received a similar treatment.

VIII. OVERVIEW OF THE PRIOR ART

81. I have reviewed in detail the prior art that I discuss in this section. I understand each of these publications qualify as prior art to the '381 patent. As I explain in more detail below, it is my opinion that these prior art references invalidate the '381 patent.

A. Synchronous Ethernet Solutions with Altera FPGAs and Silicon Labs Jitter-Attenuating PLLs (Altera)

82. In November 2015, Altera Corporation published a white paper describing an “a multiple-port 1G/10 Gbps Ethernet SyncE [Synchronous Ethernet] synchronization solution jointly developed by Altera and Silicon Labs, with a complete stable clock loop.” Ex. 1004, Altera at 1.

83. As the telecommunications industry transitioned from the use of Synchronous Optical NETworks (SONET) to Ethernet,¹⁷ there remained a need to distribute synchronization information to equipment—a capability that wasn’t natively part of the IEEE Std. 802.3™ (Ethernet) standard. As a result, in the early

¹⁷ Ex. 1028, <https://www.eetimes.com/the-once-and-future-ethernet/#:~:text=The%20year%202001%20will%20likely,Ethernet%2C%20these%20advantages%20can%20be>

2000s, a method of distributing a clock from device to device in a tree structure using native Ethernet signaling was defined.¹⁸ This came to be known as Synchronous Ethernet, or SyncE, and was standardized within the International Telecommunications Union (ITU) under the United Nations (UN). The resulting ITU standards include ITU-T G.8262.

84. The solution described in the Altera white paper “consists of an Altera® SyncE reference design implementing up to ten 1G/10 Gbps Ethernet (1G/10GbE) ports in Altera’s Stratix® V FPGA interoperating with Silicon Labs’ Si5345 jitter-attenuating phase-locked loop (PLL) to provide a complete SyncE clock recovery, clock selection, and synchronization sub-system. This solution can be used in wireless or wireline networking equipment to realize an accurate SyncE clock synchronization.” Ex. 1004, Altera at 2. The Altera solution consisted of an off-the-shelf FPGA combined with an off-the-shelf external PLL for high-performance clock synchronization for use with Ethernet networking. The Altera system—using a Stratix FPGA and Si5345 PLL—is shown in Figure 1. Altera also describes test results using the Stratix FPGA and Si5345 system. *Id.* at 5-7.

85. A SyncE-capable telecommunications network, like an Ethernet network, contains IEEE 802.1Q bridges (also known colloquially as “switches”) which forward information from one ingress port to one or more egress ports as a

¹⁸ Ex. 1029, <https://www.embedded.com/an-introduction-to-synchronized-ethernet/>

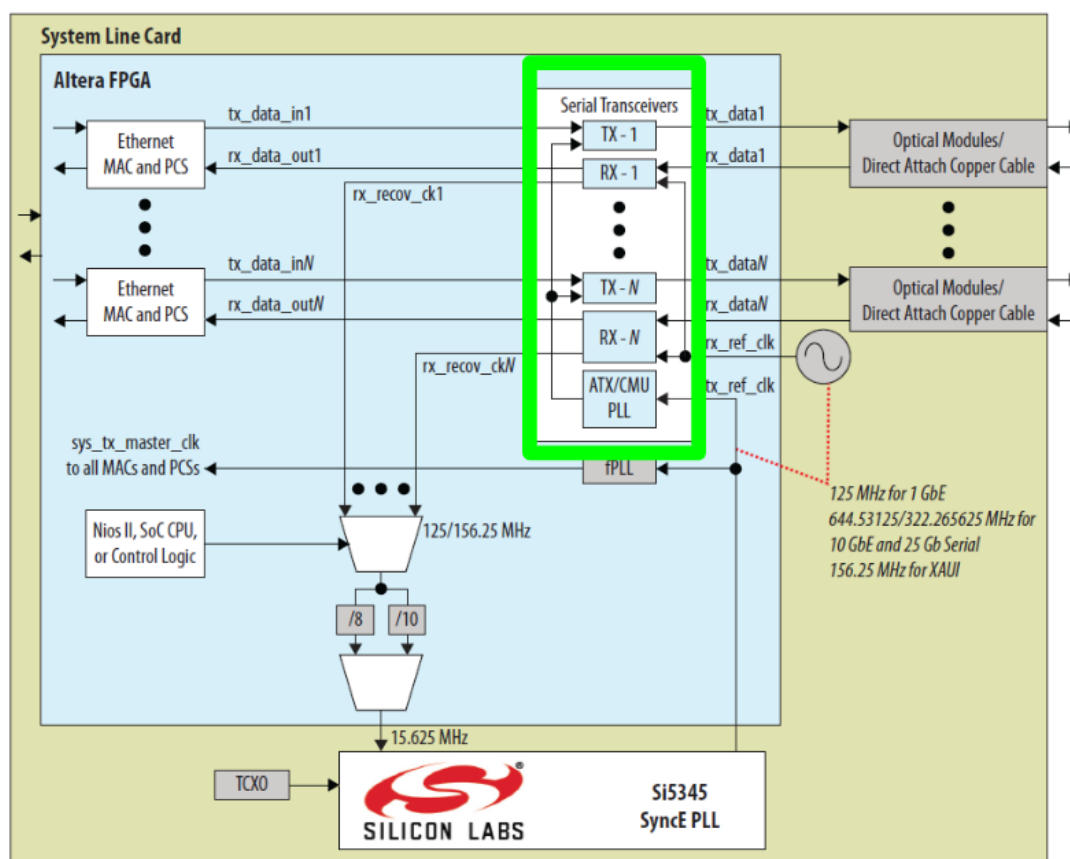
function of a variety of static and dynamic factors, and these packets are typically communicated bi-directional. When SyncE is added as an overlay to such a network, a root node is selected to be the source of frequency and/or time, and a tree is constructed. In this way, the (primary master) clock of the root node is passed through intermediate bridge nodes to all of the leaves of the tree such that they contain their own copy of the clock in the root node, which has certain characteristics required for the proper operation of the network.

86. It is critically important to the operation of SyncE that the received clock in each bridge is manipulated such that the jitter (and wander) of the clock used to transmit data meets certain stringent requirements. The component typically used to reduce clock jitter is the phase locked loop (PLL), which is the focus of this white paper written by leading FPGA vendor (Altera) and leading PLL vendor (Silicon Labs).

87. The Altera white paper provides basic background information, such as the multitude of ways in which FPGAs can be utilized, specifically calling out telecommunications applications and “other time-critical networking applications,” among others. They describe using, for example, 1 Gigabit Ethernet (GbE) and 10 GbE in such applications, Ex. 1004, Altera at 1.

88. The system diagram, in Figure 1 of Altera, shows a set of N transceivers on the right, highlighted in green in the figure below.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



89. The recovered Rx clocks are labeled “rx_recov_clkN,” and are therefore numbered 1 through N . The multiplexer (shown as a trapezoid) selects among the recovered receiver clocks to use as the reference clock to be sent, through a divider, off the FPGA chip, and into the broader FPGA system to an input of the Si5345 PLL by Silicon Labs.

90. The PLL includes a phase detector, which compares the input clock signal to a feedback signal generated by the PLL. The output of the PLL is a clock signal. This output clock is sent from the FPGA system back to the FPGA chip, where it goes to all Tx transceivers via the ATX/CMU PLL internal to the FPGA.

The same clock output of the Si5345 is also passed (through the fPLL block) to the MAC and PCS functions of the FPGA.

91. In any such system, it is critical that the loss or degradation of the signal selected to generate other clocks can be quickly detected, so the system can enter holdover (where a local reference is used in the interim—typically handled by a device such as the Si5345 PLL) until the proper alternative receiver (Rx) port can be selected to drive the external PLL.

B. Stratix V Device Handbook

92. The Stratix V FPGAs used in the Altera white paper are off-the-shelf products sold by Altera before November 2018 (then an Intel company), and numerous publications describe in detail the FPGA's operation. The Stratix V Device Handbook—a lengthy reference manual published by Altera—is one such example.

93. The Stratix V family of devices are FPGAs designed by Altera for use in a broad range of applications. They are based on 28nm silicon manufacturing technology and contain serial transceivers, 10 Gigabit Ethernet PCS, embedded memory blocks, DSP blocks, fractional PLLs (fPLLs), memories, clock networks (up to 717-MHz fabric clocking), and many other features. *See* Altera Stratix V Handbook Volumes 1, 2.

94. Many features of the FPGA are described in detail over several volumes of Handbooks, many composed of hundreds of pages, but in the context of this declaration, the high-speed transceivers, the clock networks, input/output features, and PLLs are the focus.

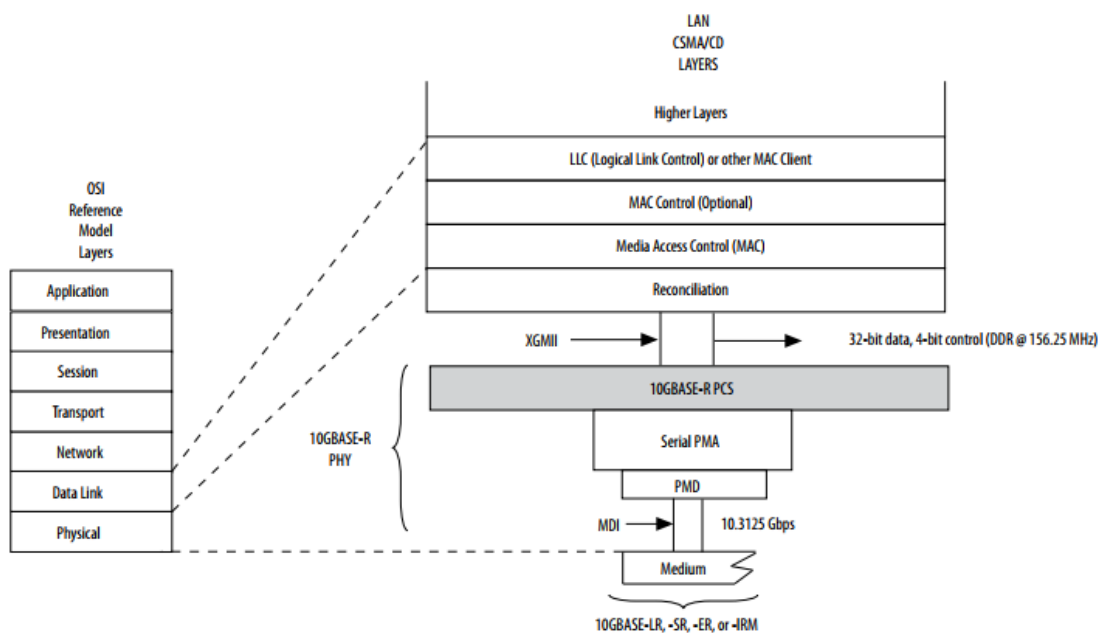
95. Stratix V transceivers are designed to support multiple data rates and protocols, many of which share a number of common characteristics. For example, high-speed serial data communications protocols require clock and data recovery (CDR)—a capability that accepts a reference clock, finds the transition times of the incoming serial data, generates a high-speed clock aligned to those transition times, and uses the clock to sample the data, producing a serial data stream. The serial data stream is converted into a parallel data stream, resulting in a correspondingly lower clock rate, is transformed in several other ways, ultimately being presented by the transceiver's receive path to the user-defined logic within the fabric of the FPGA.

96. Once the processing of the data packets is complete, parallel data is presented to the transmit interface of one or more transceivers. The parallel data is transformed, largely an inversion of the transformation performed when packet data is received, and is ultimately converted to serial from parallel using a multiplied clock, and the serial data sent off the FPGA.

97. The clock network of the Stratix V allows various clocking schemes for the programmable logic, which can be clocked at rates up to 717 MHz.

98. For example, the Stratix V can run at the same frequency described in the patent. The “receive side clock” mentioned in the patent runs at a frequency of 156.25 MHz. The parallel interface of the Stratix V transceiver operates Double Data Rate (DDR) at 156.25 MHz, assuming 32 bits of data and 4 bits of control. *See* Ex. 1005, Stratix Handbook at 140.

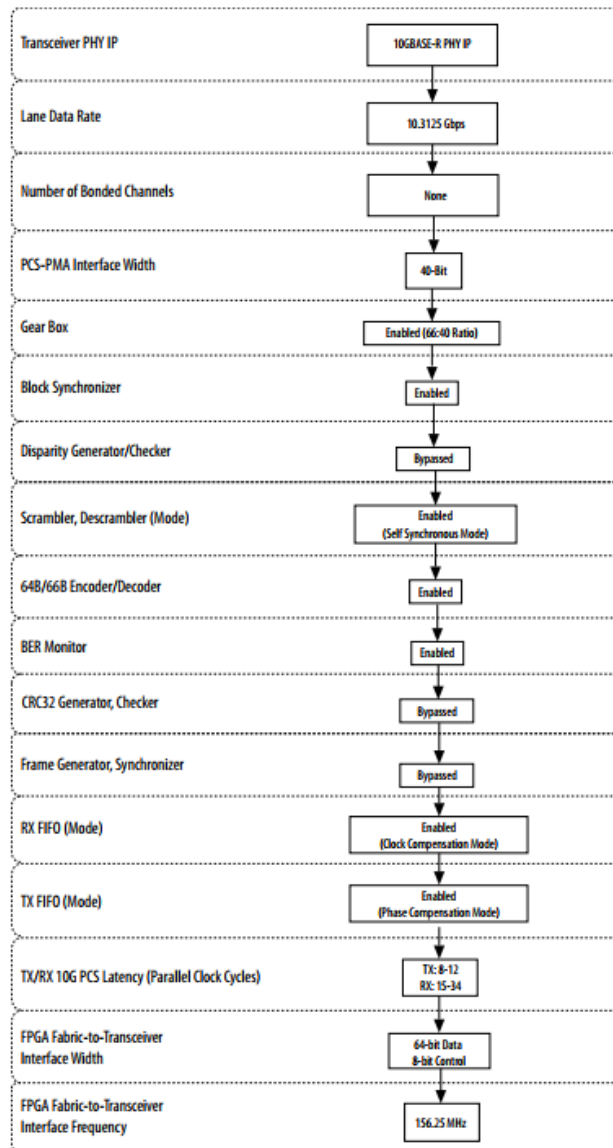
Figure 4-1: 10GBASE-R PHY Connection to IEEE802.3 MAC and RS



99. *See also id.* at 142:

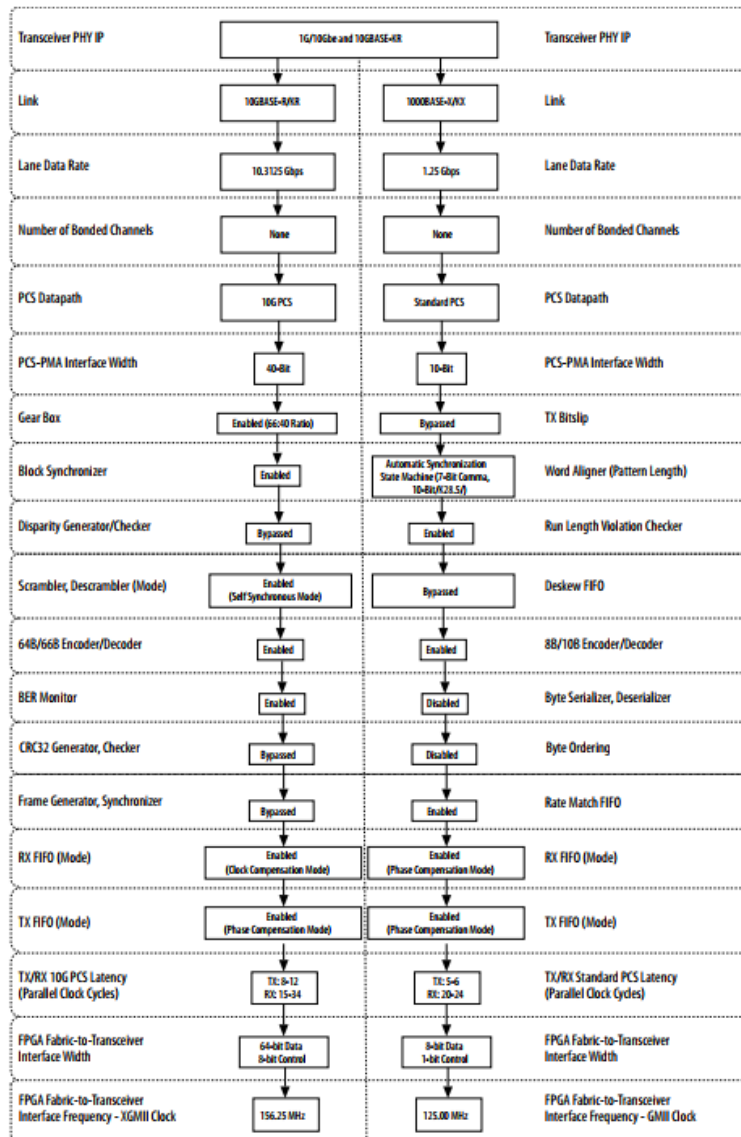
Figure 4-3: 10GBASE-R Datapath Configuration

The blocks shown as "Disabled" are not used, but incur latency. The blocks shown as "Bypassed" are not used and do not incur latency.



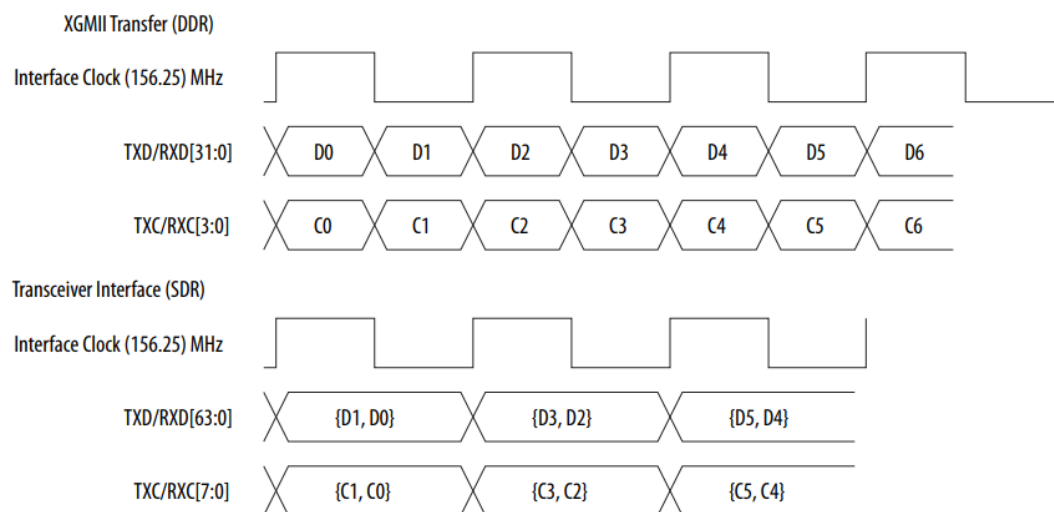
100. When 1GbE is used, it operates at 125 MHz frequency. *See id.* at 144:

Figure 4-5: 10GBASE-R/KR and 1000Base-X/KX Datapath Configuration



101. The data rate can be a single data rate (SDR) or double data rate (DDR), depending on the configuration. *See id.* at 146:

Figure 4-7: XGMII Interface (DDR) versus Stratix V Transceiver Interface (SDR) for 10GBASE-R and 10GBASE-KR Configurations



102. The relationship between input reference clock frequencies and interface speed specifications for different Stratix V configurations is summarized in the Stratix Handbook at 151:

Transceiver Clocking in 10GBASE-R, 10GBASE-KR, 1000BASE-X, and 1000BASE-KX Configurations

The CMU PLL or the auxiliary transmit (ATX) PLLs in a transceiver bank generate the transmitter serial and the fractional PLL for the parallel clocks for the 10GBASE-R, 10GBASE-KR, 1000BASE-X, and 1000BASE-KX channels. The following table lists the configuration details.

Table 4-3: Input Reference Clock Frequency and Interface Speed Specifications for 10GBASE-R, 10GBASE-KR, and 1000BASE-KX Configurations

PHY IP Type	PHY Type	Input Reference Clock Frequency (MHz)	FPGA Fabric-Transceiver Interface Width	FPGA Fabric-Transceiver Interface Frequency (MHz)
10GBASE-R PHY IP	10GBASE-R	644.53125, 322.265625	64-bit data, 8-bit control	156.25
1G/10GbE and 10GBASE-KR PHY IP	10GBASE-R and 10GBASE-KR	644.53125, 322.265625	64-bit data, 8-bit control	156.25

103. When attached to a XAUI interface (one option discussed in the Altera white paper), the FPGA uses twice the data width and SDR instead of DDR. *See id.* at 194:

Supported Features

Stratix V transceivers support the following features in a XAUI configuration.

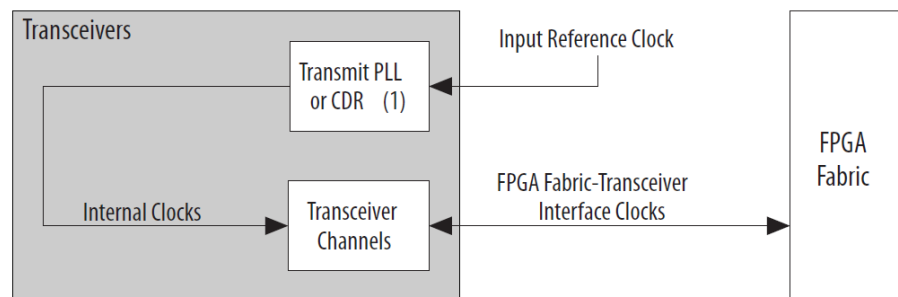
64-Bit SDR Interface to the MAC/RS

Clause 46 of the IEEE 802.3-2008 specification defines the XGMII interface between the XAUI PCS and the Ethernet MAC/RS. The specification requires each of the four XAUI lanes to transfer 8-bit data and 1-bit wide control code at both the positive and negative edge (DDR) of the 156.25 MHz interface clock.

Stratix V transceivers in a XAUI configuration do not support the XGMII interface to the MAC/RS as defined in IEEE 802.3-2008 specification. Instead, they allow the transferring of 16-bit data and 2-bit control code on each of the four XAUI lanes, only at the positive edge (SDR) of the 156.25 MHz interface clock.

104. The Altera Transceiver allows several types of PLLs to be used to generate transmit clocks. “The transmit phase-locked loop (PLL) can be a CMU PLL (channel PLL), fPLL (fractional PLL Clock), or an ATX PLL (Auxiliary Transmit PLL). *See id.* at 69.

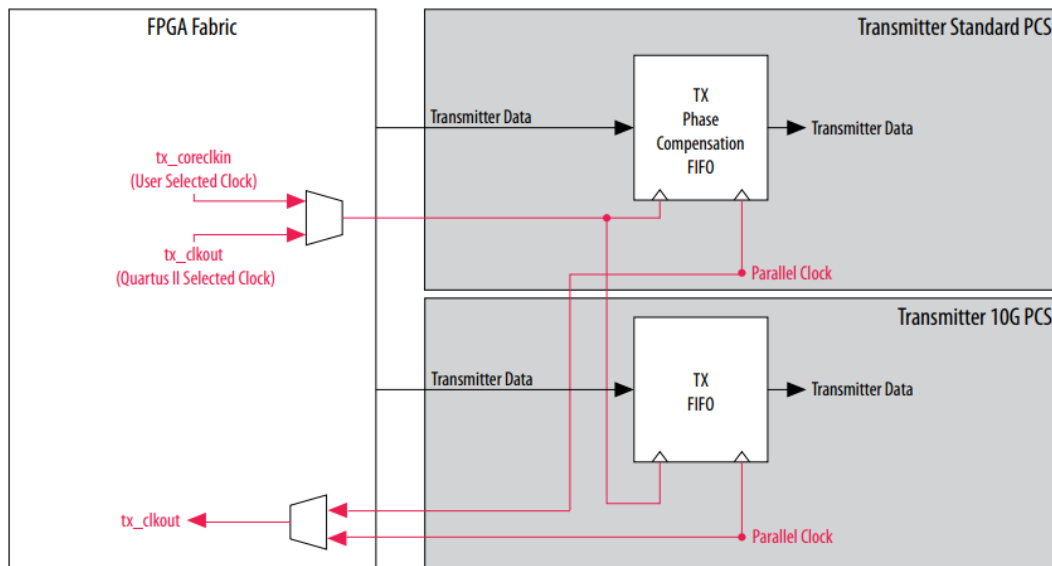
Figure 2-1: Transceiver Clocking Architecture Overview



Note: (1) The transmit phase-locked loop (PLL) can be a CMU PLL (channel PLL), fPLL (fractional PLL Clock), or an ATX PLL (Auxiliary Transmit PLL).

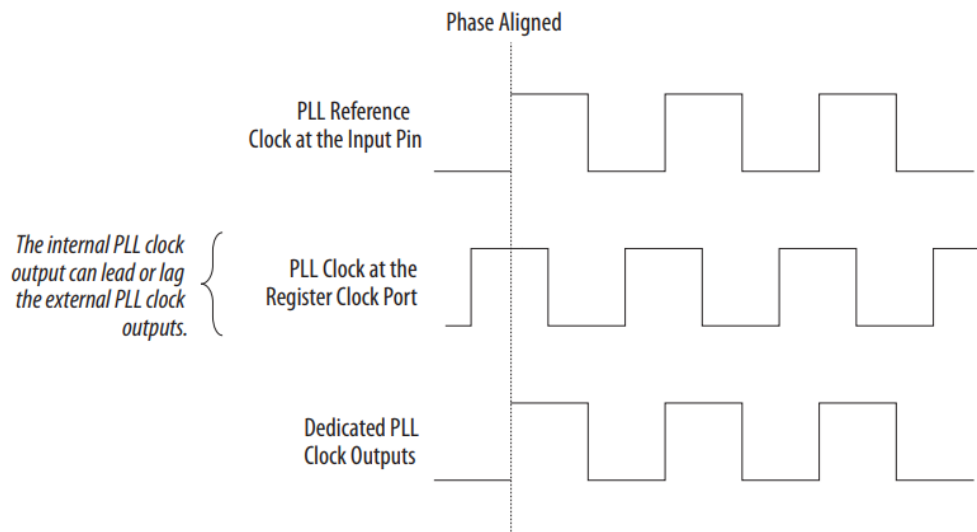
105. As seen here, the parallel transceiver clock may be fed from the transceiver to the FPGA fabric, and back into as the parallel clock of the serializer. *See id.* at 106.

Figure 2-28: Transmitter Datapath Interface Clocking



106. Regarding the clocking capabilities of the Stratix FPGA, the FPGA can eliminate the phase difference between a clock input pin and the PLL clock output. *See id.* at 353 (Figure 4-13):

Figure 4-31: Example of Phase Relationship Between the PLL Clocks in ZDB Mode



107. The fPLL can also zero out the phase error introduced by traces on the circuit board. *See id.* at 354 (Figure 4-32):

External Feedback Mode

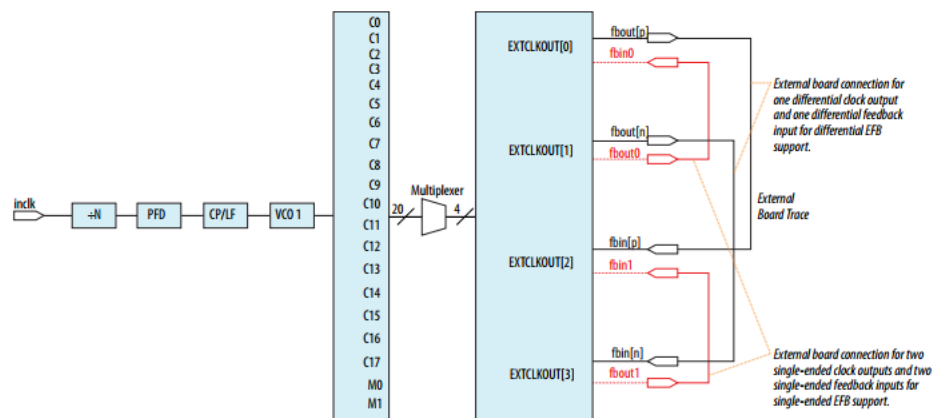
In EFB mode, the output of the M counter (f_{out}) feeds back to the PLL f_{bin} input (using a trace on the board) and becomes part of the feedback loop.

One of the dual-purpose external clock outputs becomes the f_{bin} input pin in this mode. The external feedback input pin, f_{bin} is phase-aligned with the clock input pin. Aligning these clocks allows you to remove clock delay and skew between devices.

When using EFB mode, you must use the same I/O standard on the input clock, feedback input, and clock outputs.

This mode is supported only on the center and corner fractional PLLs in Stratix V devices.

Figure 4-32: EFB Mode in Stratix V Devices



108. In the Stratix Handbook at 94, the fPLL is described synthesizing an input reference clock with the same frequency as the parallel clock of the transceiver.

Bonded Channel Configurations Using the PLL Feedback Compensation Path

You can bond channels across multiple banks by using the PLL feedback compensation path.

The PLL feedback compensation path loops the parallel clock, which is used by the PCS blocks, back to the transmitter PLL. The PLL feedback compensation path synchronizes the parallel clock used to clock the PCS blocks in all transceiver banks with the `refclk`. You can use the PLL feedback compensation path to reduce channel-to-channel skew, which is introduced by the clock divider in each transceiver bank.

To bond channels using the PLL feedback compensation path, the input reference clock frequency used by the transmitter PLL must be the same as the parallel clock that clocks the PCS of the same channel. If the input reference clock frequency is not equal to the parallel clock frequency, use a fractional PLL to synthesize an input reference clock with the same frequency as the parallel clock.

Notes:

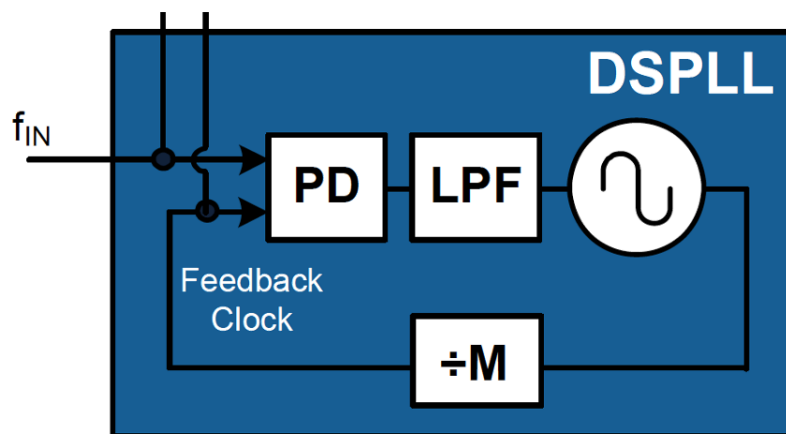
- fPLL does not support PLL feedback compensation when used as a TX PLL.
- Every transceiver bank with a bonded channel configured using the PLL feedback compensation path consumes a transmit PLL.

C. Reference Manual for the Si5345 PLL: Any-Frequency, Any-Output Jitter-Attenuators/Clock Multipliers Si5346, Si5344, Si5342 Family Reference Manual (Si5345 Manual)

109. The Si5345 PLL used in the Altera white paper is an off-the-shelf product sold by Silicon Labs (now Skyworks) before November 2018. Like the Stratix FPGA, numerous publications detail the operation of the Silicon Labs PLLs. The Si5345 Manual, from August 2015, is a Family Reference Manual that describes three PLLs made by Silicon Labs, the Si5345, Si5344, and Si5342, all of which are any-frequency, any-output jitter-attenuators/clock multipliers. Ex. 1006, Si5345 Manual at 1. “This Family Reference Manual is intended to provide system, PCB design, signal integrity, and software engineers the necessary technical information to successfully use the Si5345/44/42 devices in end applications.” *Id.* at 9. As the

Si5345 Manual describes, “[t]he Si5345/44/42 jitter attenuating clock multipliers combine 4th generation DSPLL and MultiSynth™ technologies to enable any-frequency clock generation for applications that require the highest level of jitter performance.”

110. Each PLL described in the Manual has a digital phase-locked loop (DSPLL, shown in blue box below) that includes a phase detector, low pass filter, and voltage-controlled oscillator. *See, e.g., id.* at 27. The DSPLL uses those components in an iterative process to synchronize an output clock to an input clock. *See id.* at 17 (describing “Acquisition Mode”).



Si5345/44/42 Figure 13 Excerpt

111. The startup behavior of the Si5345 includes several phases. After initialization, and without any clock on its clock inputs, a clock is nonetheless generated by the PLL using the connected oscillator as its reference. This is referred to as “Free Run Mode.” It can later enter the following modes, as described below:

Acquisition Mode (when at least one valid clock is available for synchronization), Locked Mode (where the DSPLL generates output clocks that are frequency and phase locked to the selected input clock), and Holdover Mode (when the selected input clock becomes invalid and the DSPLL uses an averaged input clock frequency).

Si5345-44-42-RM

4.4. Free Run Mode

The DSPLL will automatically enter freerun mode once power is applied to the device and initialization is complete. The frequency accuracy of the generated output clocks in freerun mode is entirely dependent on the frequency accuracy of the external crystal or reference clock on the XA/XB pins. For example, if the crystal frequency is ± 100 ppm, then all the output clocks will be generated at their configured frequency ± 100 ppm in freerun mode. Any drift of the crystal frequency will be tracked at the output clock frequencies. A TCXO or OCXO is recommended for applications that need better frequency accuracy and stability while in freerun or holdover modes. Because there is little or no jitter attenuation from the XAXB pins to the clock outputs, a low-jitter XAXB source will be needed for low-jitter clock outputs.

4.5. Acquisition Mode

The device monitors all inputs for a valid clock. If at least one valid clock is available for synchronization, the DSPLL will automatically start the lock acquisition process. If the fast lock feature is enabled, the DSPLL will acquire lock using the Fastlock Loop Bandwidth setting and then transition to the DSPLL Loop Bandwidth setting when lock acquisition is complete. During lock acquisition the outputs will generate a clock that follows the VCO frequency change as it pulls-in to the input clock frequency.

4.6. Locked Mode

Once locked, the DSPLL will generate output clocks that are both frequency and phase locked to its selected input clock. At this point any XTAL frequency drift will typically not affect the output frequency. A loss of lock pin (LOL) and status bit indicate when lock is achieved. See "5.3.3. Loss of Lock Fault Monitoring" for more details on the operation of the loss of lock circuit.

4.7. Holdover Mode

The DSPLL will automatically enter holdover mode when the selected input clock becomes invalid and no other valid input clocks are available for selection. The DSPLL uses an averaged input clock frequency as its final holdover frequency to minimize the disturbance of the output clock phase and frequency when an input clock suddenly fails. The holdover circuit for the DSPLL stores up to 120 seconds of historical frequency data while locked to a valid clock input. The final averaged holdover frequency value is calculated from a programmable window within the stored historical frequency data. Both the window size and the delay are programmable as shown in Figure 6. The window size determines the amount of holdover frequency averaging. The delay value allows ignoring frequency data that may be corrupt just before the input clock failure.

112. The feedback path from VCO to PD can be completed within the device, as shown in Figure 13 Excerpt above, or through a path external to the

device. *See id.* at 44 (Figure 21). This latter configuration provides additional options for phase compensation between input clocks and output clocks of the PLL.

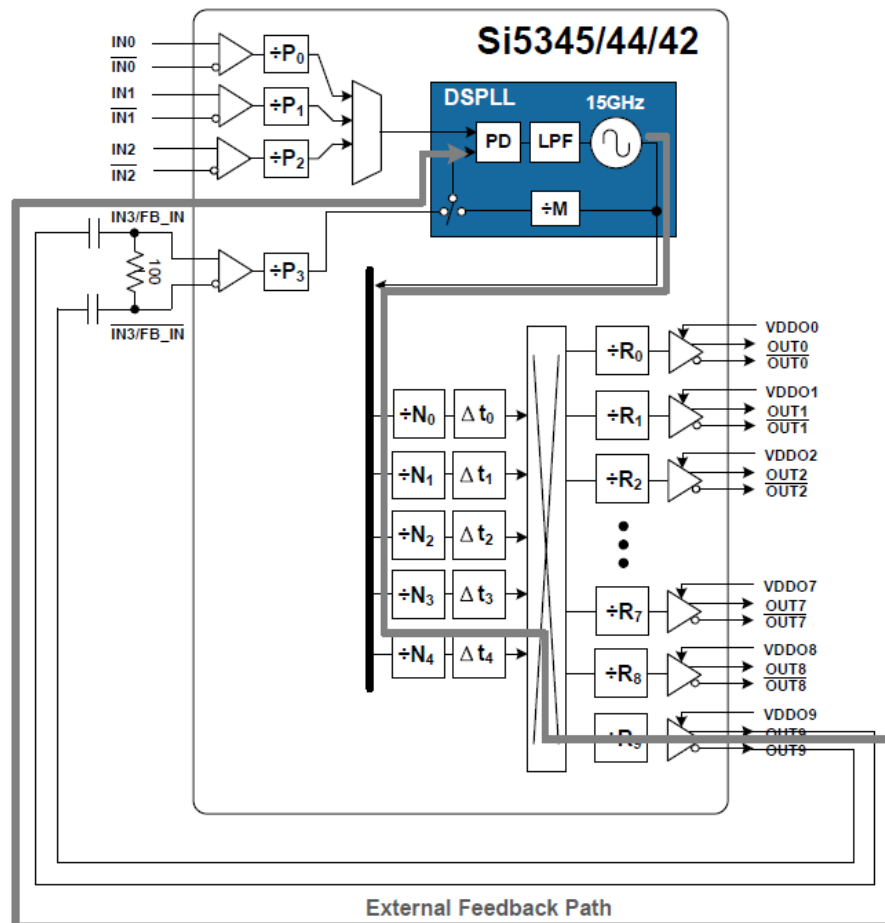


Figure 21. Si5345 Zero Delay Mode Set-up

D. A Low-Latency Library in FPGA Hardware for High-Frequency Trading (HFT) by Lockwood

113. By 2018, it was common knowledge that FPGAs were being “used in the financial industry in high frequency trading where the rapid processing of the FPGA is desired.” Ex. 1001, ’381 patent at 1:42-44. Lockwood is a paper that describes the utility of FPGA-based systems for low-latency trading and provides examples of how such systems could be implemented.

114. One option, according to Lockwood, is to use an FPGA-based “Smart Network Interface Controller,” or a Smart NIC. “FPGA-based ‘Smart NIC’ offer [a] way to apply programmable logic to the acceleration of financial applications. Smart NICs typically bring together high-speed network interfaces, a PCIe host interface, memory and an FPGA. The FPGA implements the NIC controller, acting as the bridge between the host computer and the network and allows user-designed custom processing logic to be integrated directly into the data path.” Ex. 1007, Lockwood at 3. A Smart NIC system is shown in Figure 2 of Lockwood. When using a Smart NIC, market data from an exchange enters the FPGA system, where it is processed in the FPGA to result in orders back to the market.

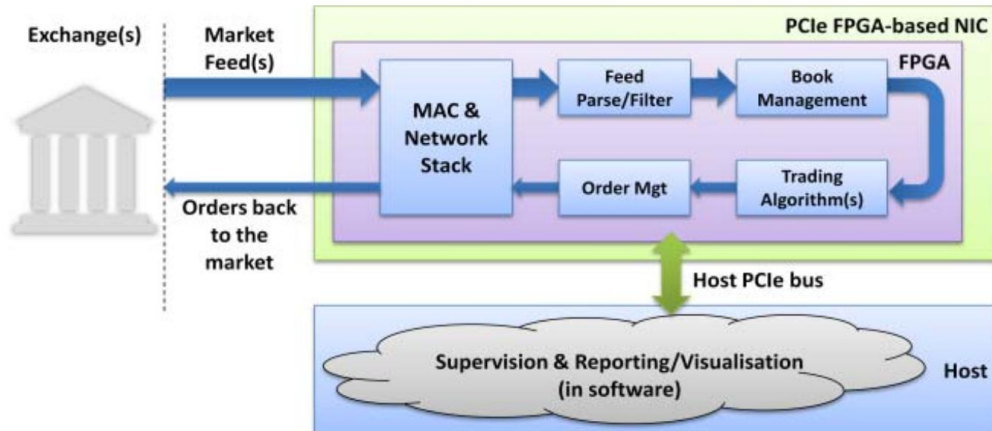


Figure 2 Typical Smart NIC in a supervised financial trading application

115. Lockwood also explains that “some financial processing systems are completely implemented on FPGA-based platforms without any host CPU yielding best possible latencies. As oncoming traffic no longer needs to make a round trip

over the system's PCIe interface for processing in the CPU, the inherent latency and jitter introduced by the host's operating system are entirely avoided.” Ex. 1007, Lockwood at 3. Such a system is shown in the bottom portion of Figure 3.

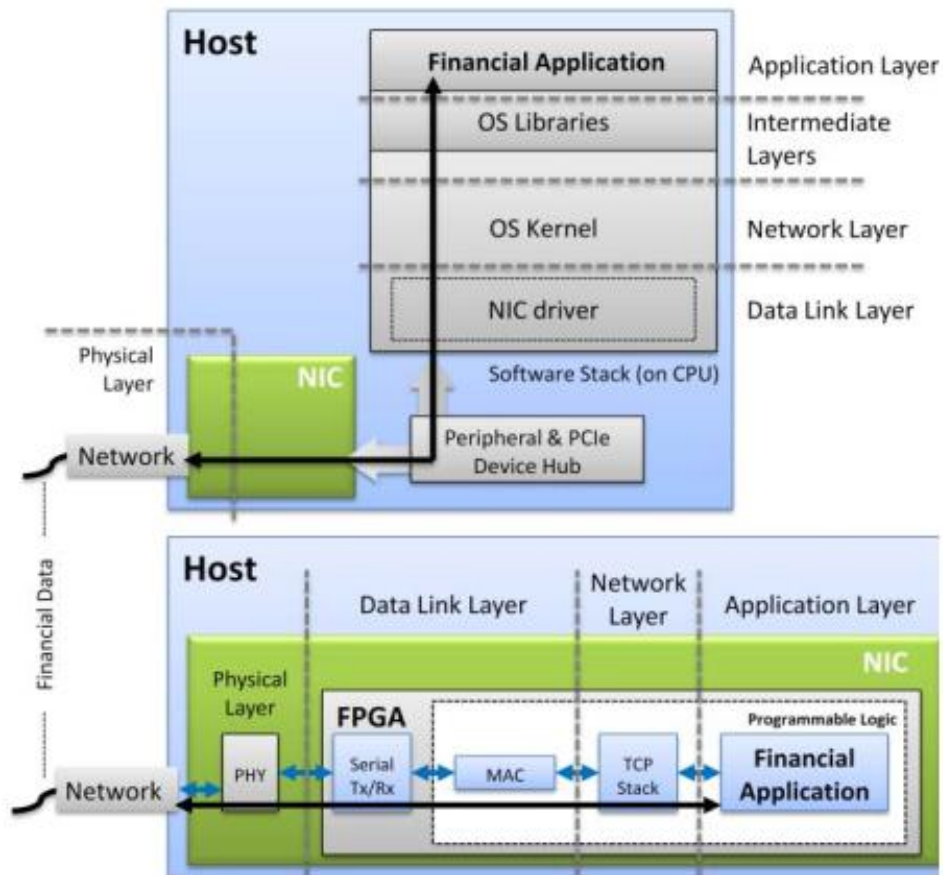


Figure 3 Software-based (top) and FPGA-based (bottom) trading platforms

116. Lockwood describes numerous advantages to using FPGAs for low-latency trading. For example, “[m]odern FPGAs can implement most aspects of any HFT trading application,” such that “[i]ncoming market data can typically be processed completely on the FPGA without needing to travel over and back to a host CPU.” *Id.* at 3. Additionally, “in the FPGA implementation, the incoming network

data is fed directly into a custom-designed, highly-optimized and application-specific processing pipeline via hardware PHY and MAC blocks.” *Id.* at 4. “Along with lower latency and minimal jitter, FPGAs can achieve significantly higher throughput using parallelism.” *Id.* And “FPGAs offer the[se] performances benefits of custom hardware while retaining programmability” so that “[t]rading algorithms” implemented on the FPGAs “can be continually improved.” *Id.*

IX. GROUND 1A: ALTERA ANTICIPATES CLAIMS 1, 4, AND 6-10 OF THE '381 PATENT

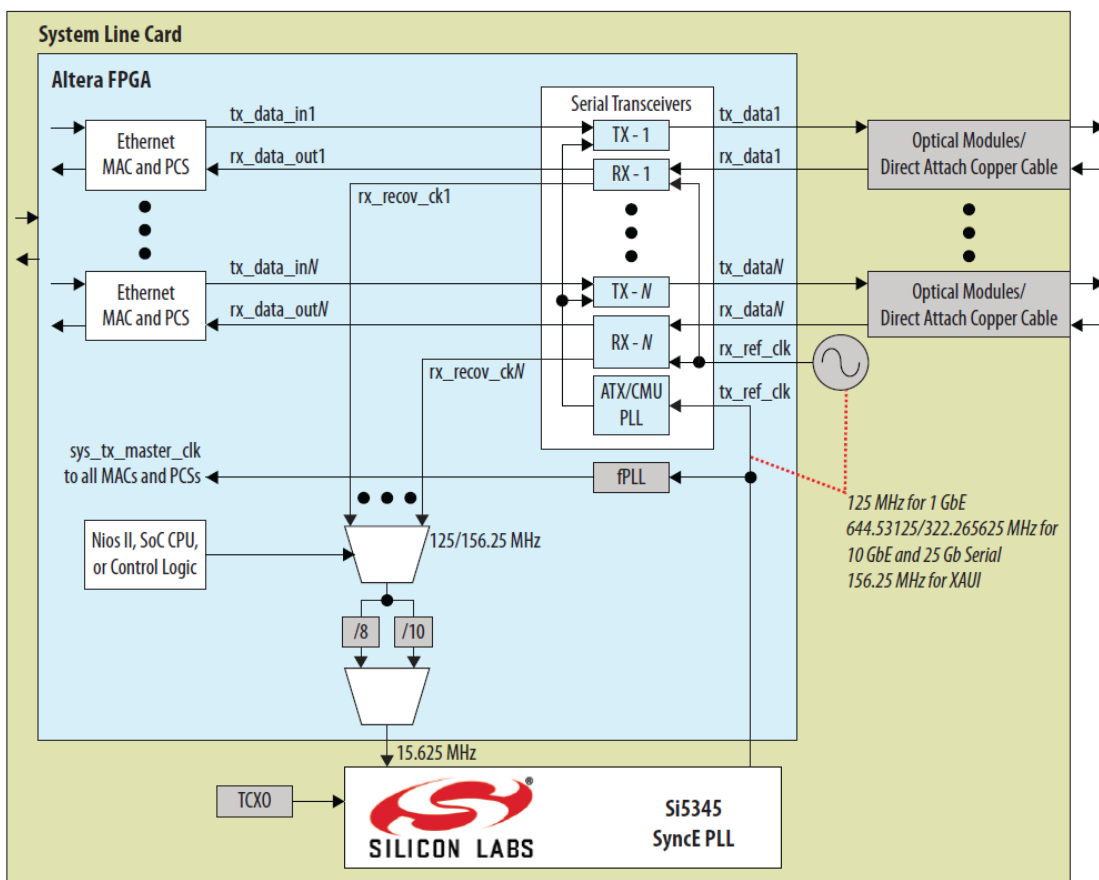
A. A Person of Skill in the Art Reading Altera Understands that the Features of the Altera Stratix FPGA and Si5345 PLL Are Necessarily Present

117. Altera is a white paper that describes the operation of “a high-performance SyncE solution based on . . . Altera’s Stratix® V FPGA interoperating with Silicon Labs’ Si5345 jitter-attenuating phase-locked loop (PLL).” Ex. 1004, Altera at 2. A POSA reading Altera would understand that the features of the Stratix FPGA and Si5345 PLL are necessarily present in the solution described in Altera. Altera provides both a “SyncE reference design” and a test setup for “interoperability” that utilizes the Stratix V FPGA and Si5345 PLL. *Id.* at 3, 5. The white paper does not describe in depth every aspect of the operation of the FPGA and PLL but instead assumes that the reader can and will locate that information separately. In my experience, this is common to do.

118. Here, a POSA would be intimately familiar with the fact that if they wanted to learn more about the Altera FPGA and/or Si5345 PLL, they would easily be able to find and rely on extensive information about those devices available from Altera and Silicon Labs. I commonly seek and find information about integrated circuits (ICs) when in need of detailed information on their capabilities and operation. For example, the Altera white paper encourages customers “to obtain the latest version of device specifications before relying on any published information.” *Id.* at 1. Altera and Silicon Labs made specifications and data sheets available for download from their respective websites, including the Stratix Handbook and Si5345 Manual used here. *See* Ex. 1009 – Ex. 1011, Mina Ching Declarations regarding the Internet Archive for the Stratix Handbook and Si5345 Manual. In many cases these specifications also point to other published documentation that covers various elements of the device and their use in more detail.

119. The following Figure 1 is unmodified from the Altera at 2, and is replicated herein, often with annotations and modifications to emphasize important and relevant teachings.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



B. Claim 1

1. Claim 1 preamble: “A method for processing a first serial data stream, using a field programmable gate array system, to generate a second serial data stream, wherein the method comprises the steps of”

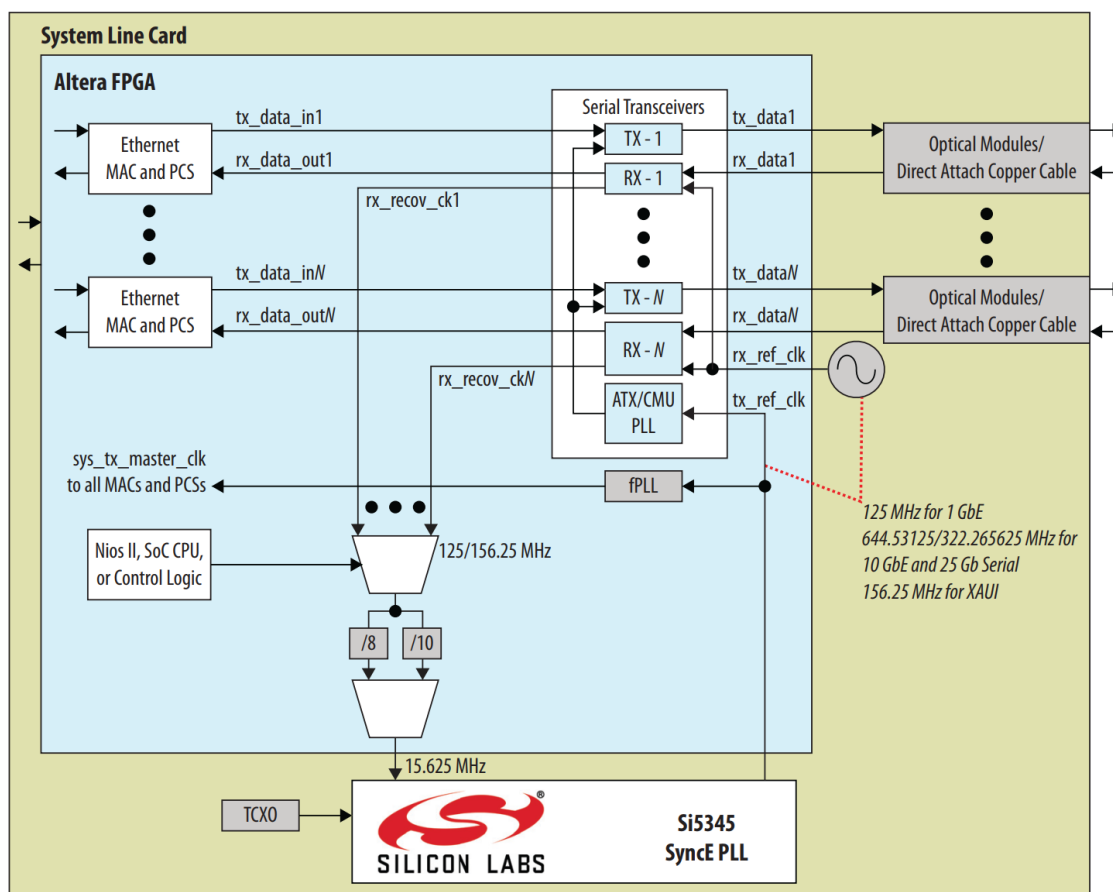
120. To the extent the preamble is limiting, Altera discloses “[a] method for processing a first serial data stream, using a field programmable gate array system, to generate a second serial data stream.”

121. Altera is a system using “Altera’s Stratix® V FPGA interoperating with Silicon Labs’ Si5345 jitter-attenuating phase-locked loop (PLL) to provide a

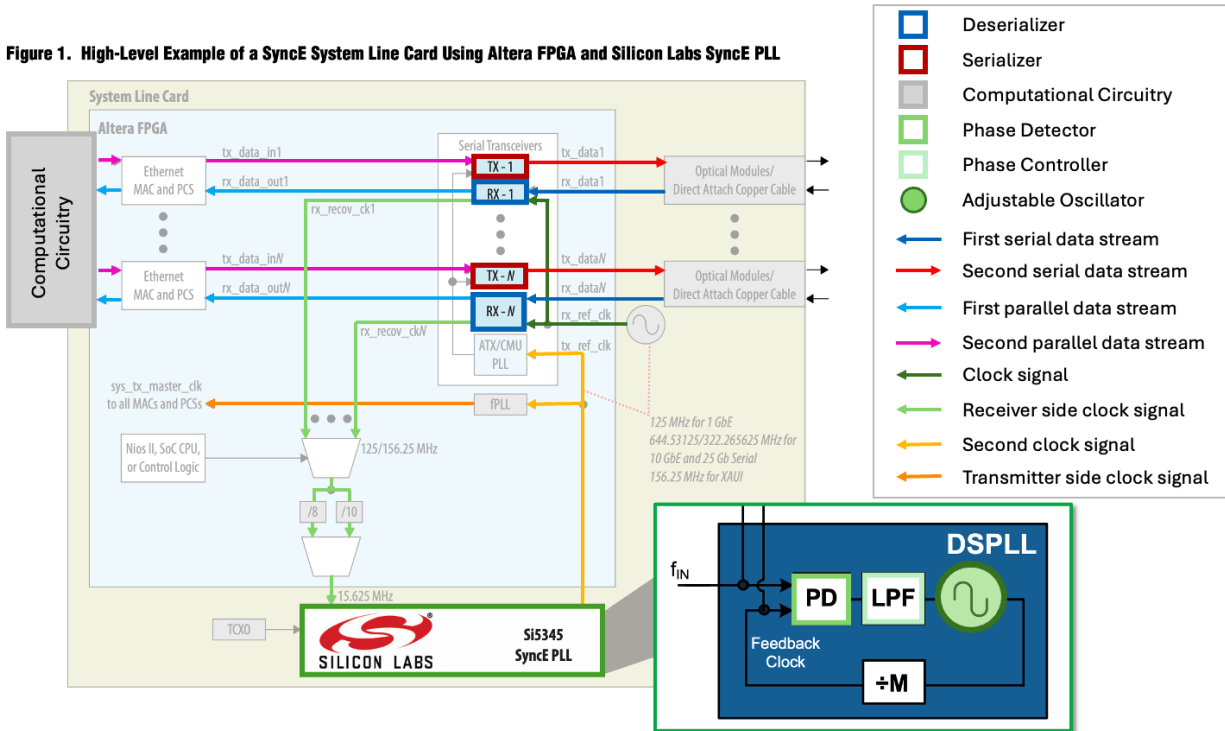
complete SyncE clock recovery, clock selection, and synchronization sub-system.”

Ex. 1004, Altera at 2. In Altera, first serial data streams (rx_data1-rx_dataN) are received and second serial data streams (tx_data1-tx_dataN) are transmitted. A POSA would understand that N is an integer equal to 1 or more. The Altera system is designed for wireless or wireline networking, *Id.* at 2, indicating that the received data is processed for those networking applications. “Networking equipment,” *id.*, receives a first serial data stream and generates a second serial data stream.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



122. I have annotated Figure 1 of Altera below to aid in my analysis of the '381 patent:

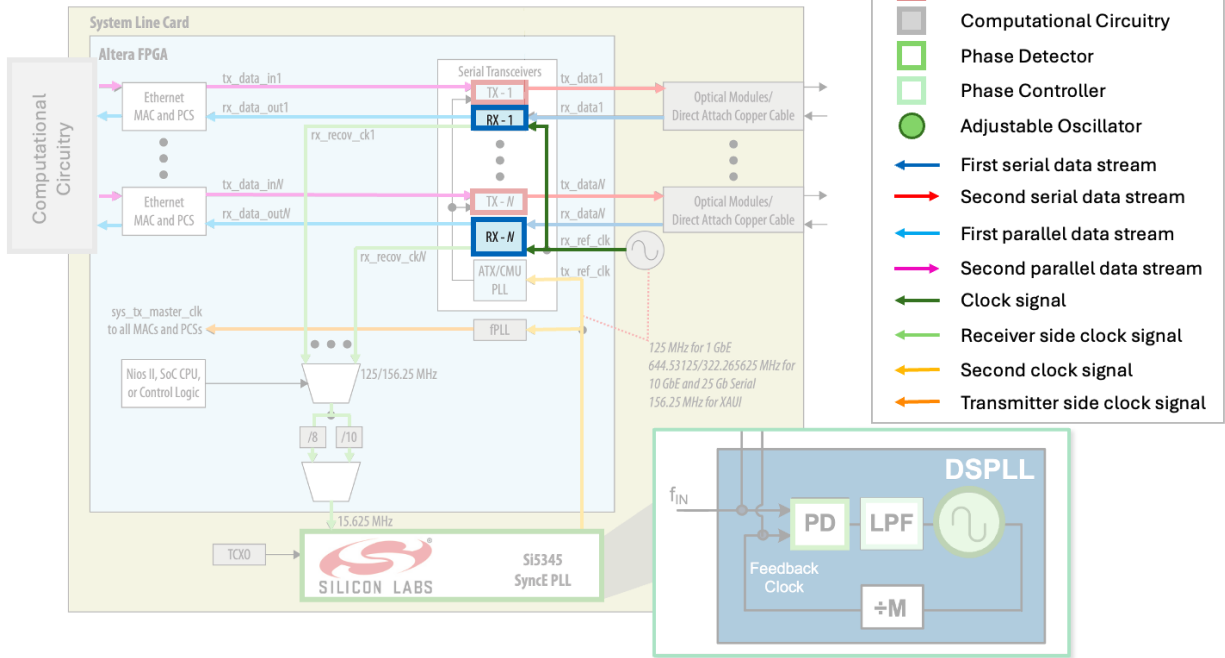


2. Claim 1(a): “receiving, by a deserializer in a field programmable array, a clock signal”

123. Altera discloses “receiving, by a deserializer in a field programmable array, a clock signal.”

124. Altera discloses a receiver reference clock signal, “rx_ref_clk,” that is received by receivers (RX-1 through RX-N) of the serial transceivers. Altera at Figure 1.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



125. Each receiver block (RX-1 through RX-N) within the Altera FPGA performs various operations of the “deserializer” claimed in the ’381 patent. The receiver-side portion of the block labeled Ethernet MAC and PCS performs various other operations of the “deserializer” claimed in the ’381 patent. The ’381 patent provides example deserializers that demonstrate that the claimed deserializer can include a receiver PMA and receiver PCS. *See, e.g.,* Ex. 1001, ’381 patent at 17:34-35 (“the deserializer 2104” may include a receiver PMA 3512 and receiver PCS 3508. The receiver PMA 3512 may include receiver buffer 3504, CDR 3516, and serial to parallel box 3118’.”), Fig. 3E.

126. As demonstrated in the comparison below, Altera Figure 1-7 and Figure 3E of the patent are very similar. Like the '381 patent, each receiver of Altera includes a receiver PMA and a receiver PCS. Ex. 1005, Stratix Handbook at 14.

Altera Figure 1-7

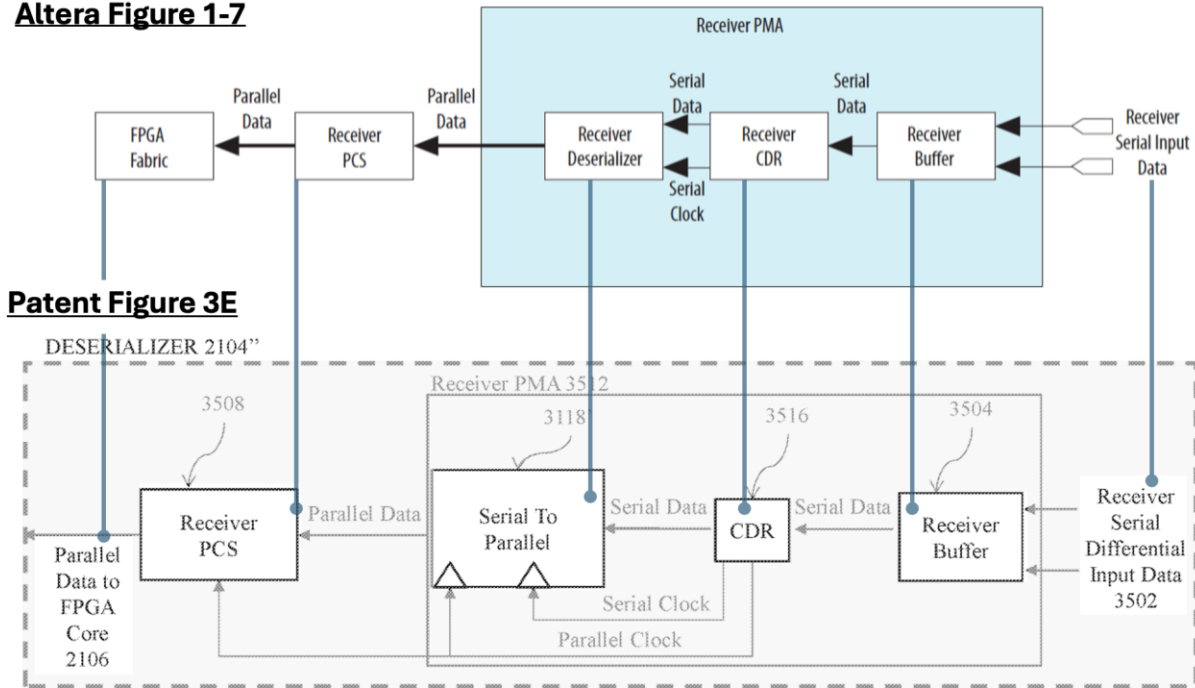


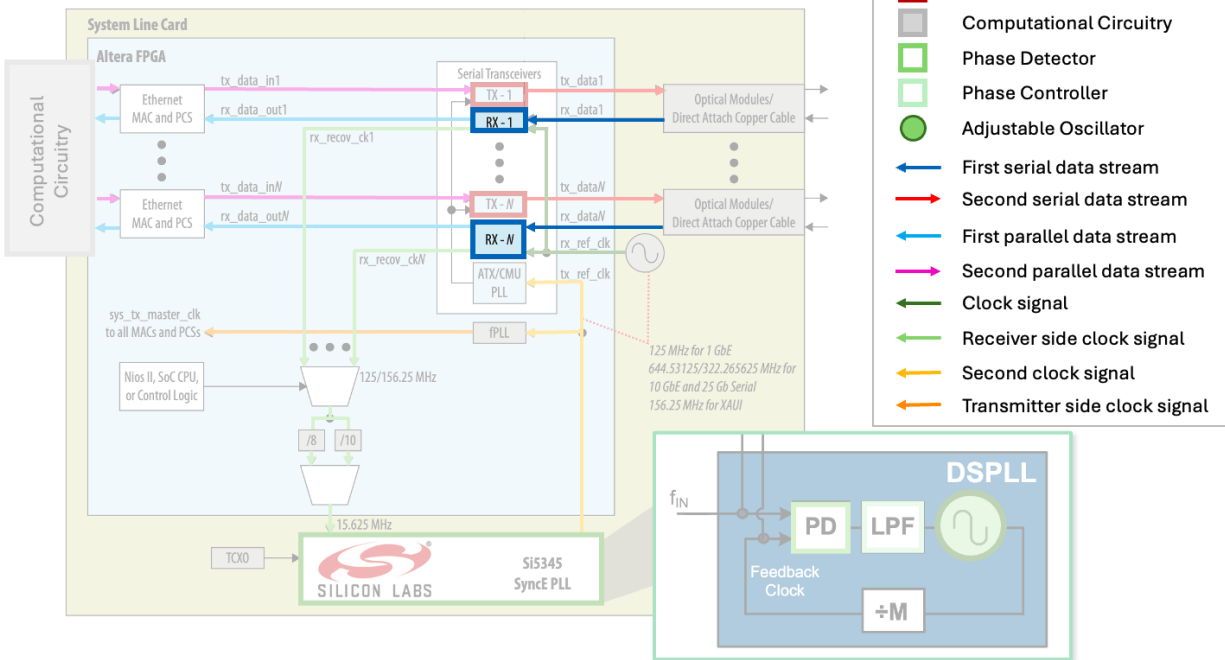
FIG. 3E

3. Claim 1(b): “receiving, by the deserializer, the first serial data stream”

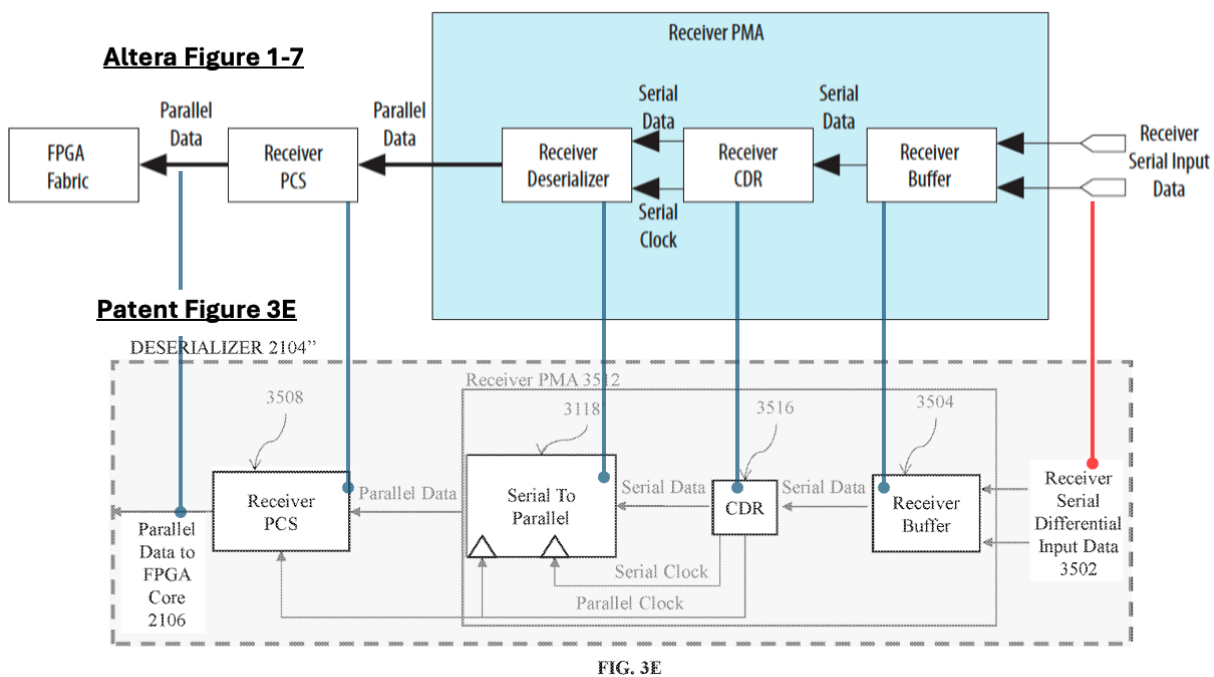
127. Altera discloses “receiving, by the deserializer, the first serial data stream.”

128. As depicted below, Altera discloses that the receivers (RX-1 through RX-N) of serial transceivers receive serial data (rx_data1-rx_dataN) from Optical Modules/Direct Attach Copper Cables. Ex. 1004, Altera at Figure 1. The “serial” nomenclature indicates that the serial transceiver receives serial data.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



129. In addition, the Stratix Handbook depicts the receivers (the RX blocks shown in Altera) in more detail. The Stratix Handbook confirms that the receivers receive serial data. Ex. 1005, Stratix Handbook at 14 (“The PMA receives . . . off-chip high-speed serial data streams.”).

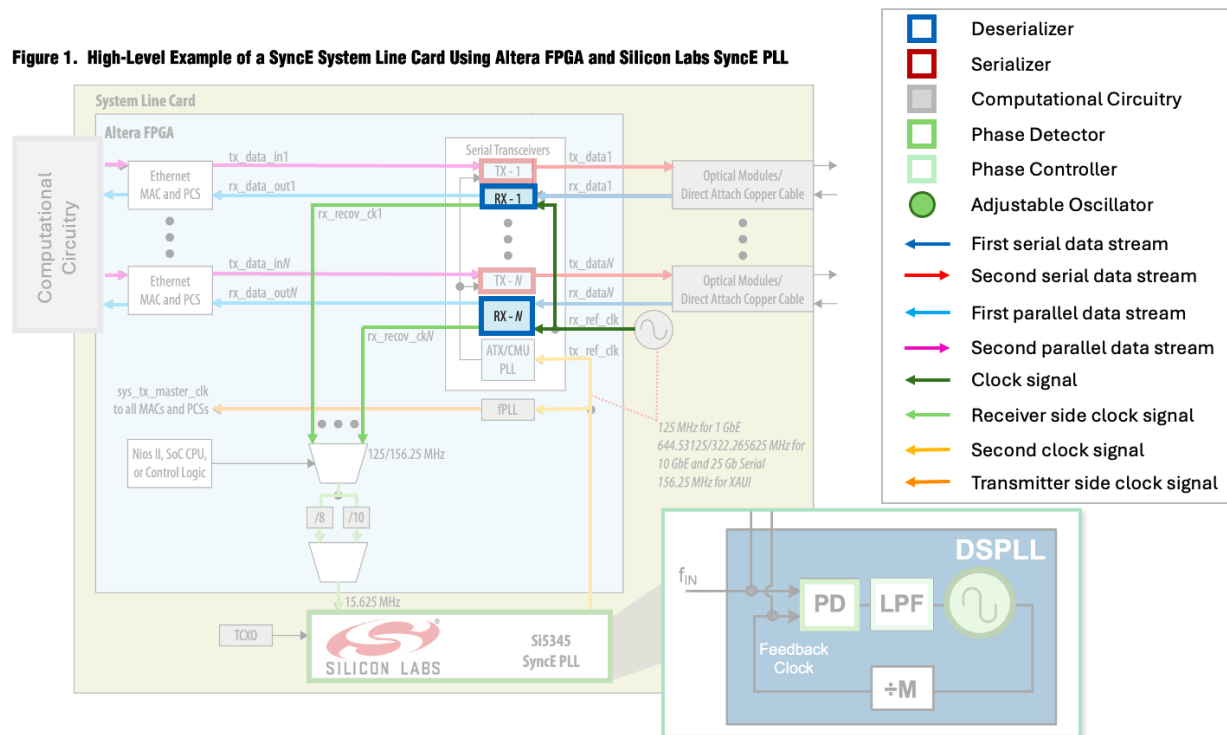


4. Claim 1(c): “generating, by the deserializer, a receiver side clock signal”

130. Altera discloses “generating, by the deserializer, a receiver side clock signal.”

131. In Altera, the “PHY IP cores recover the receiver (RX) clocks from their respective 1G/10G Gbps Ethernet ports.” Ex. 1004, Altera at 3. The PHY IP core refers to the serial transceivers—the PHY IP core is the “physical coding layer (PCS) and physical medium attachment (PMA) (PHY) intellectual property (IP) instances.” *Id.* Accordingly, a POSA would understand that in Altera each receiver, e.g. in the PMA, recovers the receiver (RX) serial clock and generates the corresponding receiver side clock signal associated with the parallel received data,

as depicted with the green lines below. *See also* Ex. 1005, Stratix Handbook at 104 (Table 2-5).



132. The receive PMA's role in recovering the receiver serial clock and generating the corresponding receiver-side parallel clock is confirmed by the Stratix Handbook, as shown in Figures 2-22 and 2-23. Ex. 1005, Stratix Handbook at 97; *see also id.* at 6 ("The PMA block . . . generates the required clocks"); Stratix Figure 1-19. These figures illustrate that the clock and data recovery circuit (CDR) within the Receiver PMA generates two recovered clocks, the parallel receiver side clock (shown in red) and the serial clock (shown in blue). For ease of reference, I have excerpted the relevant portion of the Figures below.

GX Channel Receiver Clocking

Figure 2-22: Receiver 10G PCS and PMA Clocking

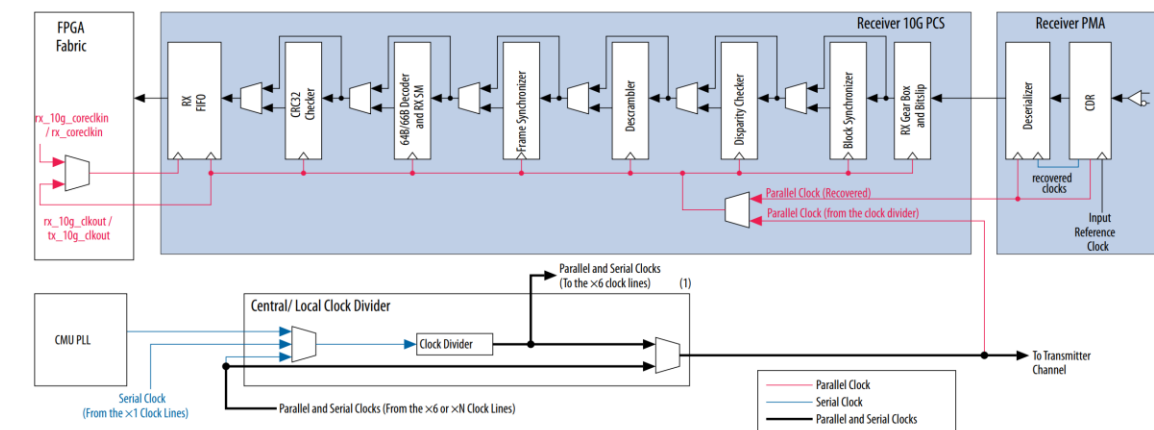
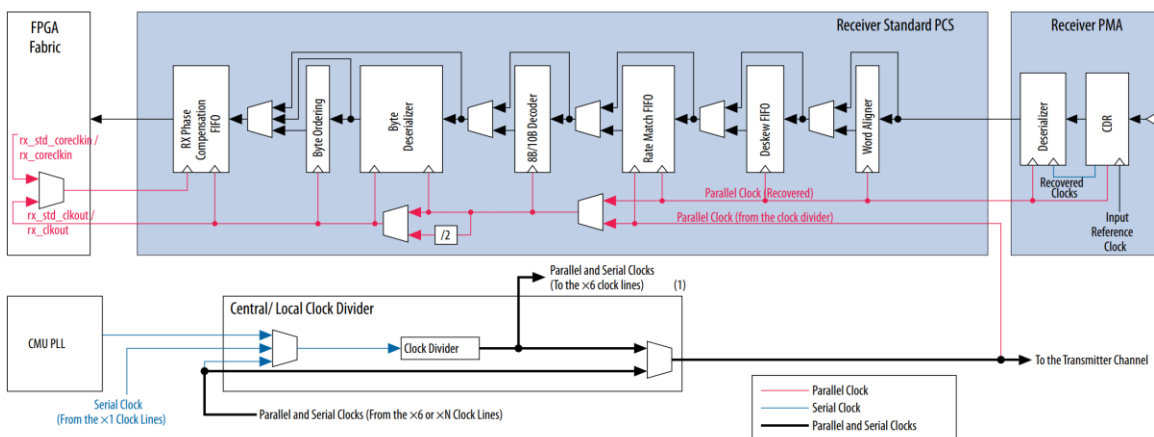
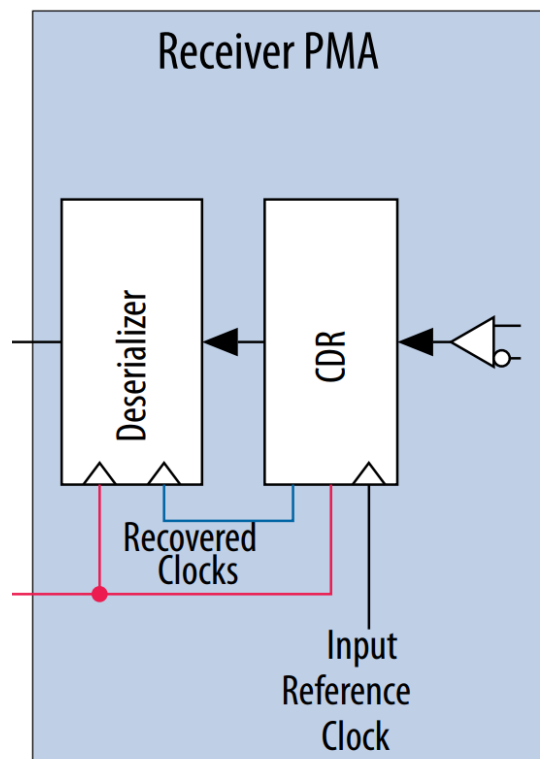
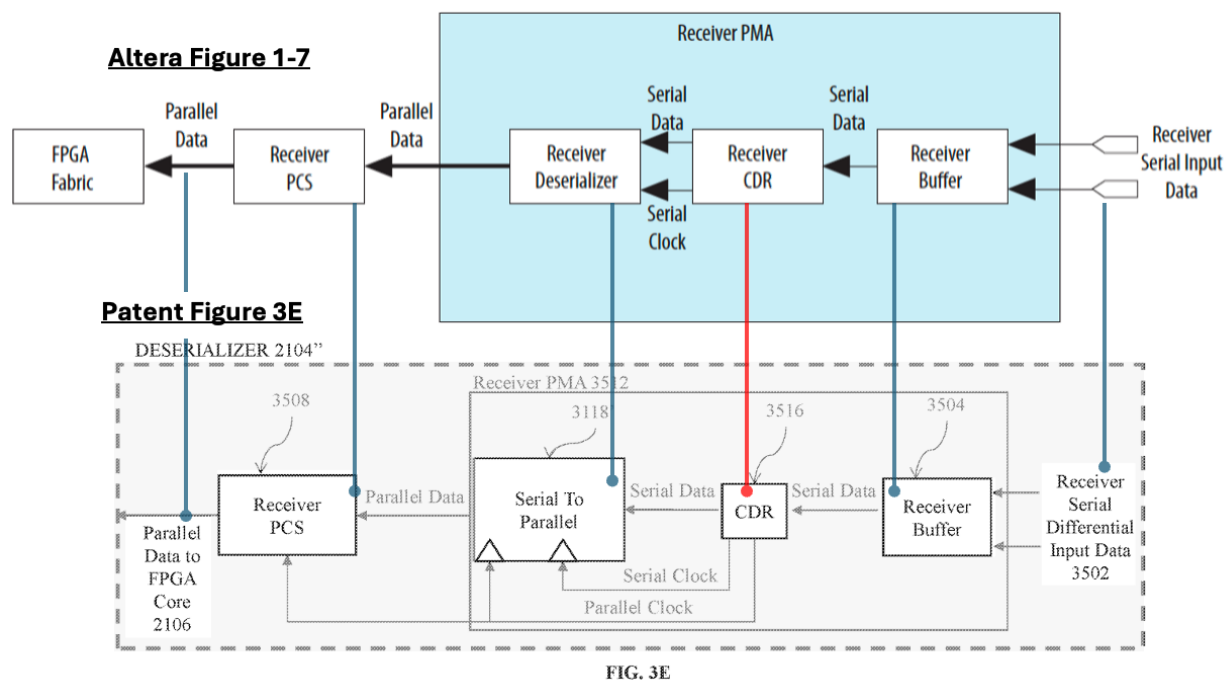


Figure 2-23: Receiver Standard PCS and PMA Clocking





133. Like Altera, the '381 patent explains that the claimed deserializer can include a CDR for generating a clock signal. *See, e.g.*, Ex. 1001, '381 patent at 8:16-18 (“FIG. 3C is an exemplary block diagram of a clock and data recovery (CDR) circuit suitable for use in the deserializer”), 10:56-58 (“The receiver clock domain is typically generated by a clock and data recovery (CDR) circuit from the incoming data stream.”), 16:18-22, 17:34-35 (“the deserializer 2104” may include a receiver PMA 3512 and receiver PCS 3508. The receiver PMA 3512 may include receiver buffer 3504, CDR 3516, and serial to parallel box 3118’.”). A CDR—a clock data recovery circuit—is a common circuit that can be used to generate a clock signal from a data stream.

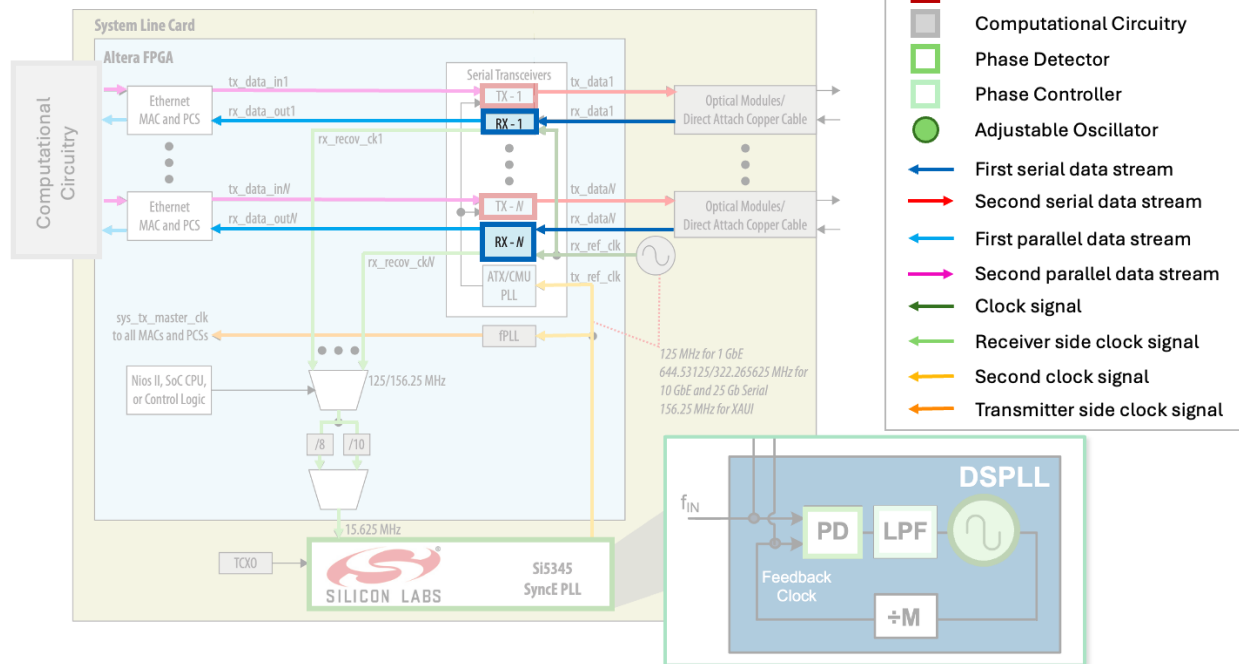


5. Claim 1(d): “converting, by the deserializer, the first serial data stream into a first plurality of parallel data streams”

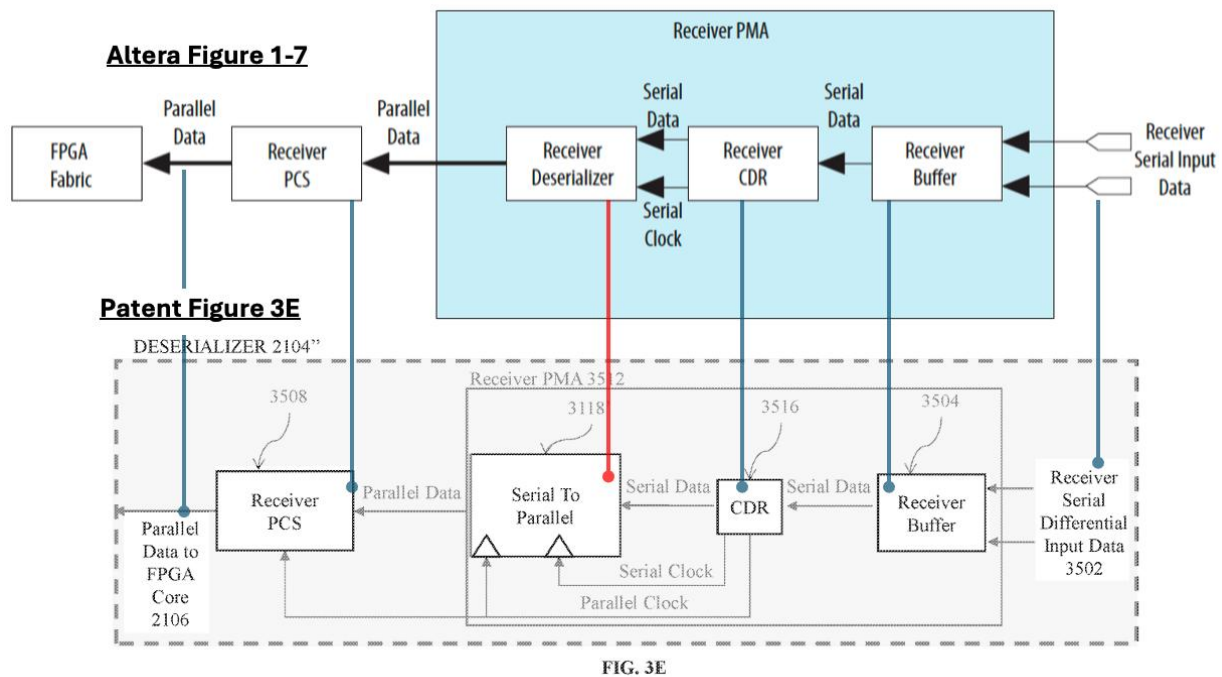
134. Altera discloses “converting, by the deserializer, the first serial data stream into a first plurality of parallel data streams.”

135. A POSA would understand from Figure 1 of Altera that serial rx_data is converted into parallel data, labeled rx_data_out, in the receiver (RX), as depicted with the blue lines. In addition, Altera discloses that there are as many of these parallel data streams as there are receivers (N in number), each of which converts a serial data stream to one among a plurality of parallel data streams.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



136. The Stratix Handbook confirms that the receiver converts serial data to parallel data. The Stratix Handbook explains that “[t]he PMA block connects the FPGA to the channel, generates the required clocks, and converts the data from parallel to serial or serial to parallel.” Ex. 1005, Stratix Handbook at 6. Figure 1-7 shows that the “Receiver Deserializer” of the Receiver PMA converts serial data to parallel data. *Id.* at 14.

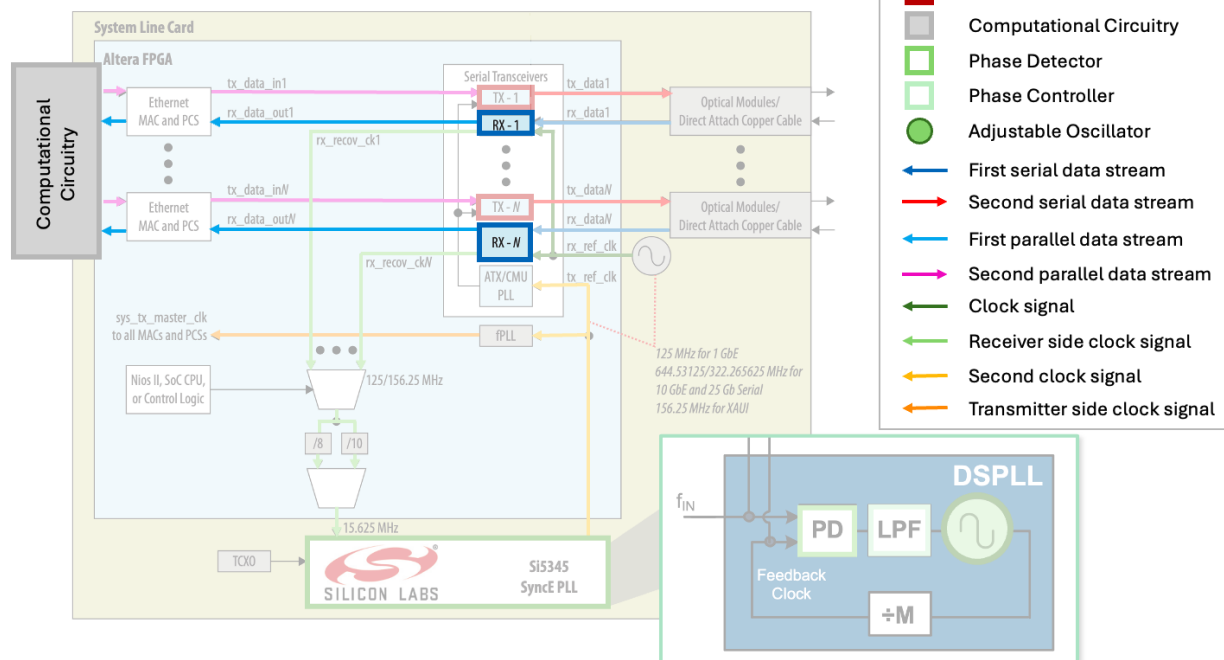


6. Claim 1(e): “transmitting, from the deserializer to computational circuitry in the field programmable gate array, the first plurality of parallel data streams”

137. Altera discloses “transmitting, from the deserializer to computational circuitry in the field programmable gate array, the first plurality of parallel data streams.”

138. As shown in Figure 1, a POSA would recognize that in Altera, the receivers transmit parallel data streams to computational circuitry within the FPGA. Although the Altera white paper does not explicitly draw a representation of the computational circuitry, a POSA would understand that the computational circuitry is present in the Altera FPGA as shown in the grey box below.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



139. The Stratix Handbook confirms that parallel data is transmitted from the Receiver PCS of the receiver to the FPGA fabric. Ex. 1005, Fig 1-7 of Stratix Handbook at 14. A POSA would understand that FPGA fabric refers to computational circuitry in the FPGA.

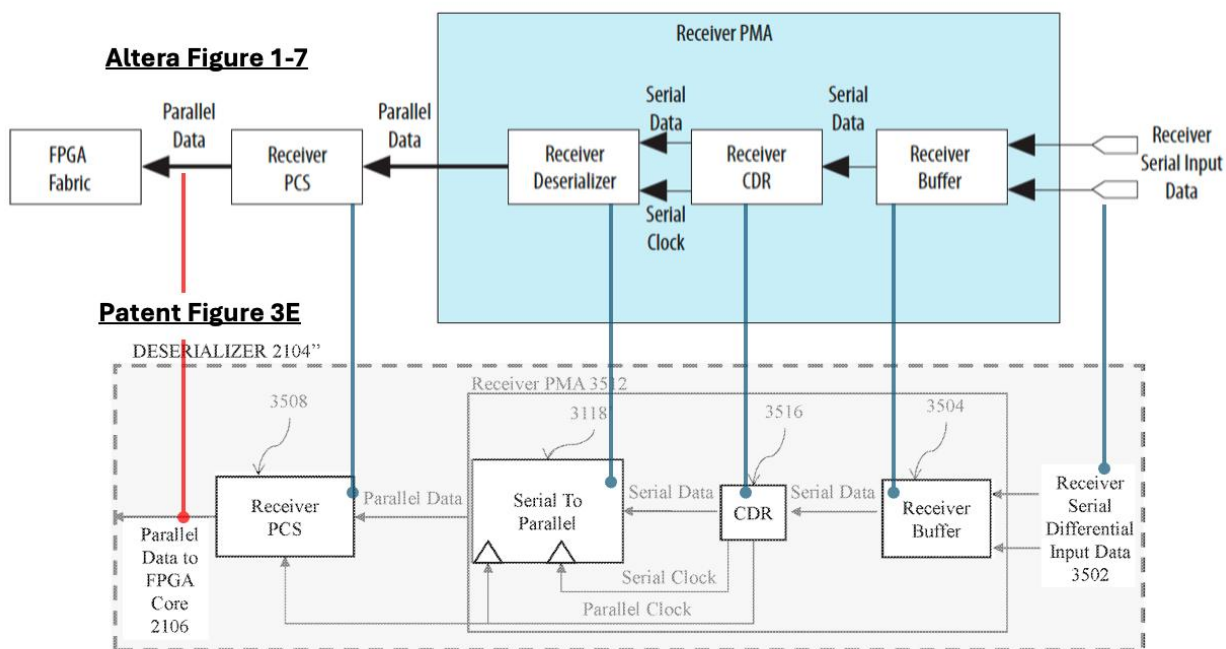


FIG. 3E

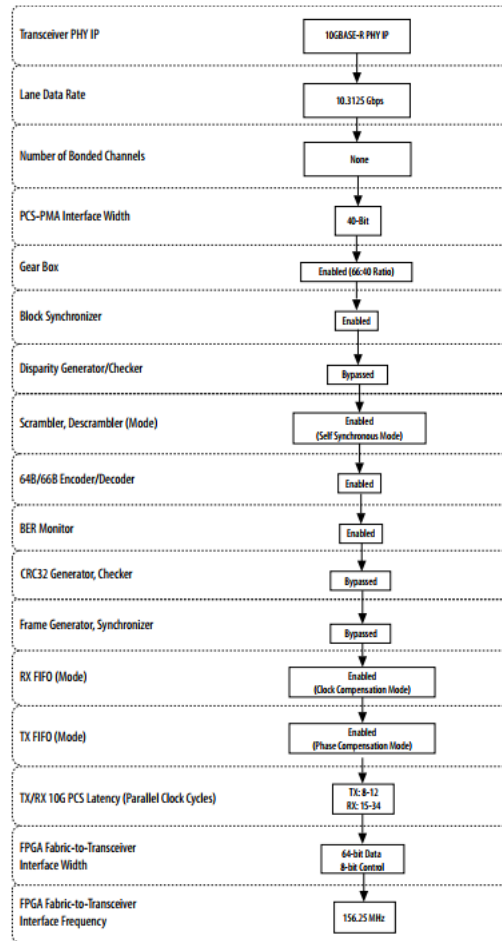
7. Claim 1(f): “transmitting, from the deserializer to a phase lock loop of the field programmable gate array system that is not within the field programmable gate array, the receiver side clock signal”

140. Altera discloses “transmitting, from the deserializer to a phase lock loop of the field programmable gate array system that is not within the field programmable gate array, the receiver side clock signal.”

141. Altera Figure 1 depicts the receiver-side clock signal (e.g., rx_recov_ck1) being transmitted from the receiver to the external Si5345 PLL. These clocks are shown in Ex. 1004, Figure 1 at Altera 2 to connect to a multiplexer with a clock frequency of, e.g., 156.25MHz. It is shown at the bottom of Ex. 1005, Figure 4-3 of Handbook at 142 that this frequency is used for parallel data passing from the 10 Gigabit Ethernet Transceiver to the FPGA Fabric.

Figure 4-3: 10GBASE-R Datapath Configuration

The blocks shown as "Disabled" are not used, but incur latency. The blocks shown as "Bypassed" are not used and do not incur latency.

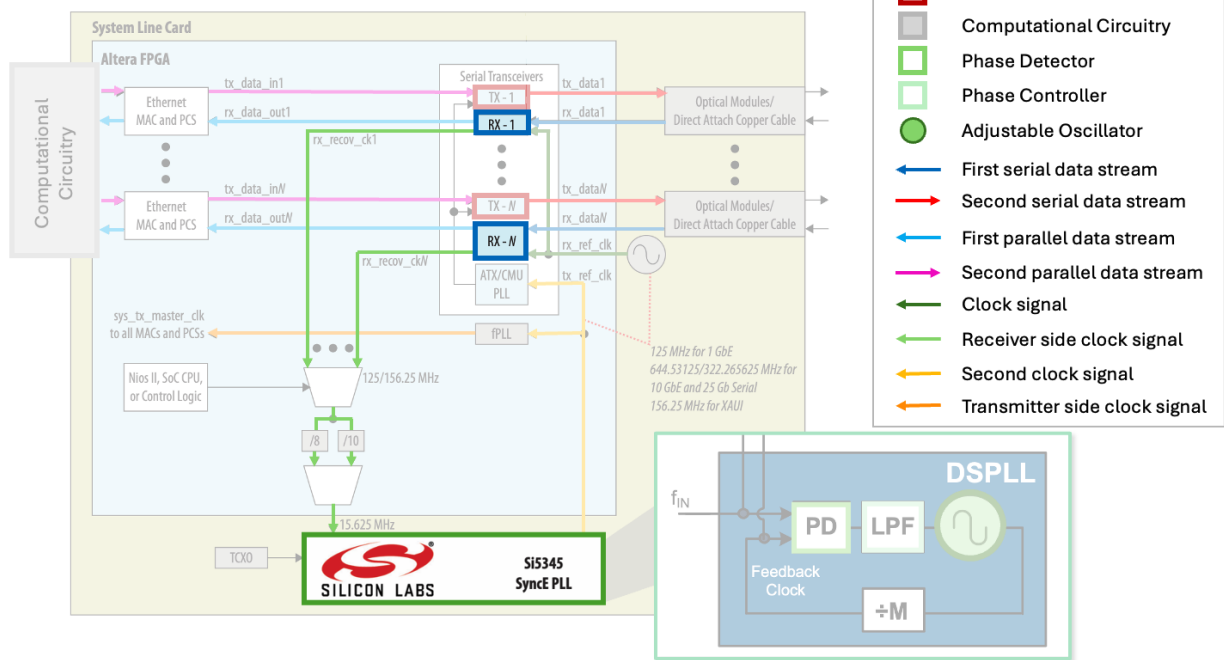


The last box in the flow is expanded below.



142. Altera also explains that “the RX clocks from multiple Ethernet ports are multiplexed in the FPGA fabric to the external jitter-attenuating PLL.” Ex. 1004, Altera at 4.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL

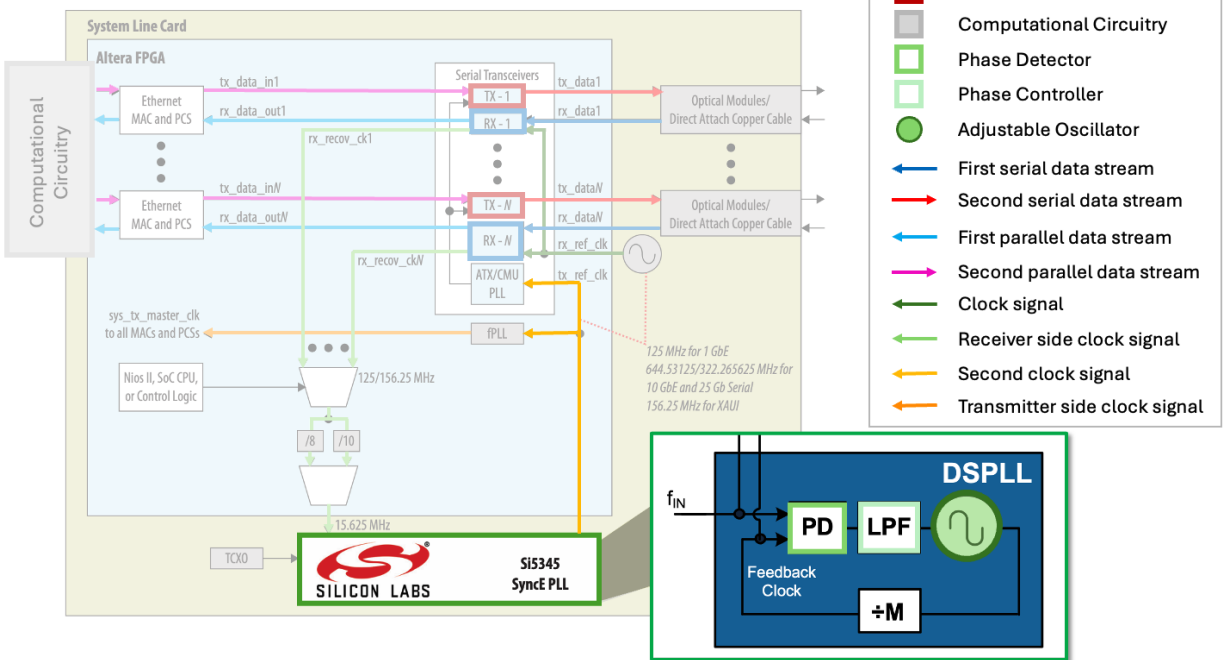


8. Claim 1(g): “generating, using the phase lock loop, a second clock signal”

143. Altera discloses “generating, using the phase lock loop, a second clock signal.”

144. Altera Figure 1 discloses that the external Si5345 SyncE PLL generates a second clock signal, tx_ref_clk. Altera explains that “The Si5345 ITU-T G.8262 SyncE standard-compliant PLL from Silicon Labs then attenuates the selected clock signal’s wander and jitter to within the SyncE specification,” thereby generating a “stabilized clock.” *Id.* at 3. A POSA would understand that the clock would also meet the Ethernet requirements.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL

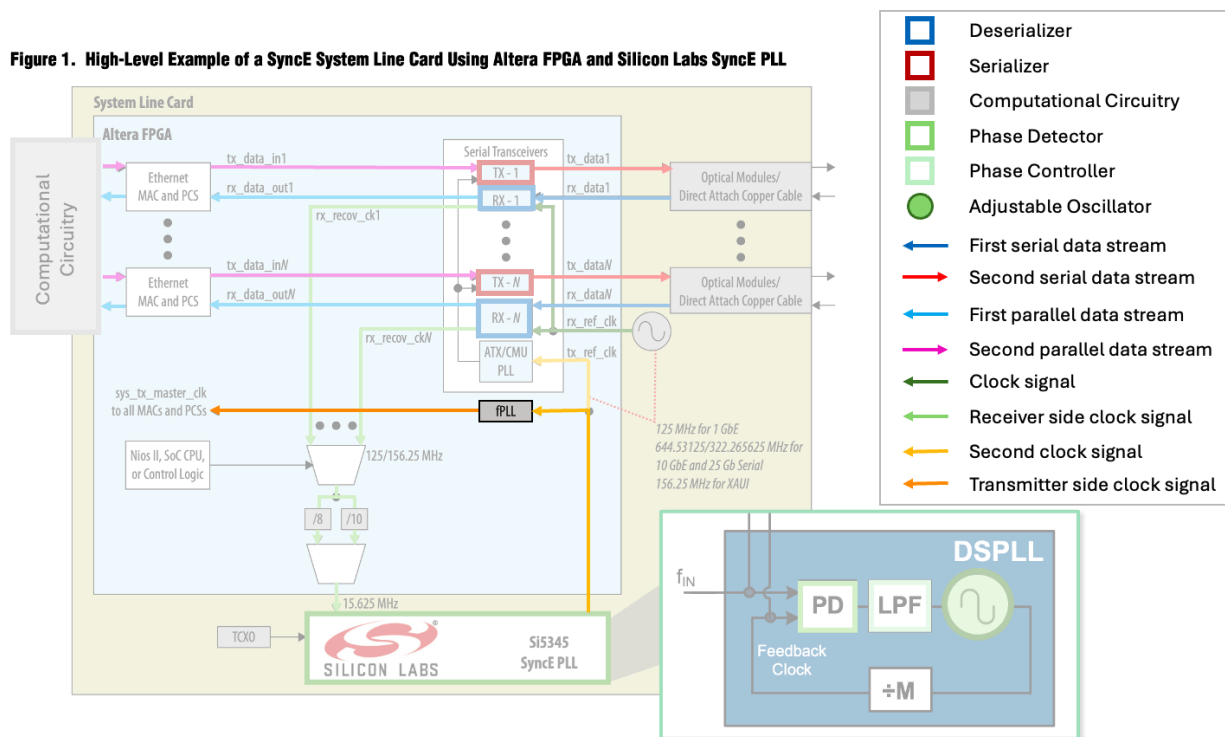


9. Claim 1(h): “generating, within the field programmable gate array, a transmitter side clock signal derived from the second clock signal”

145. Altera discloses “generating, within the field programmable gate array, a transmitter side clock signal derived from the second clock signal.”

146. Altera explains: the “stabilized clock” generated by the Si5345 PLL “is used to feed the FPGA’s serial transmitter (TX) PLL in sync with the master Ethernet port’s clock frequency to realize a complete synchronization loop.” Ex. 1004, Altera at 3; *see also id.* at 4 (Si5345 PLL “suppl[ies] a clean reference clock for the serial transceivers’ TX PLLs synchronous to the RX recovered clock”). Figure 1 at Altera 2 shows that the same tx_ref_clk clock is transmitted to the fPLL (fractional PLL) within the FPGA, which then generates a transmitter-side clock signal

(sys_tx_master_clk), derived from the tx_ref_clk (the second clock signal) generated by the external PLL, as demonstrated with the orange lines.

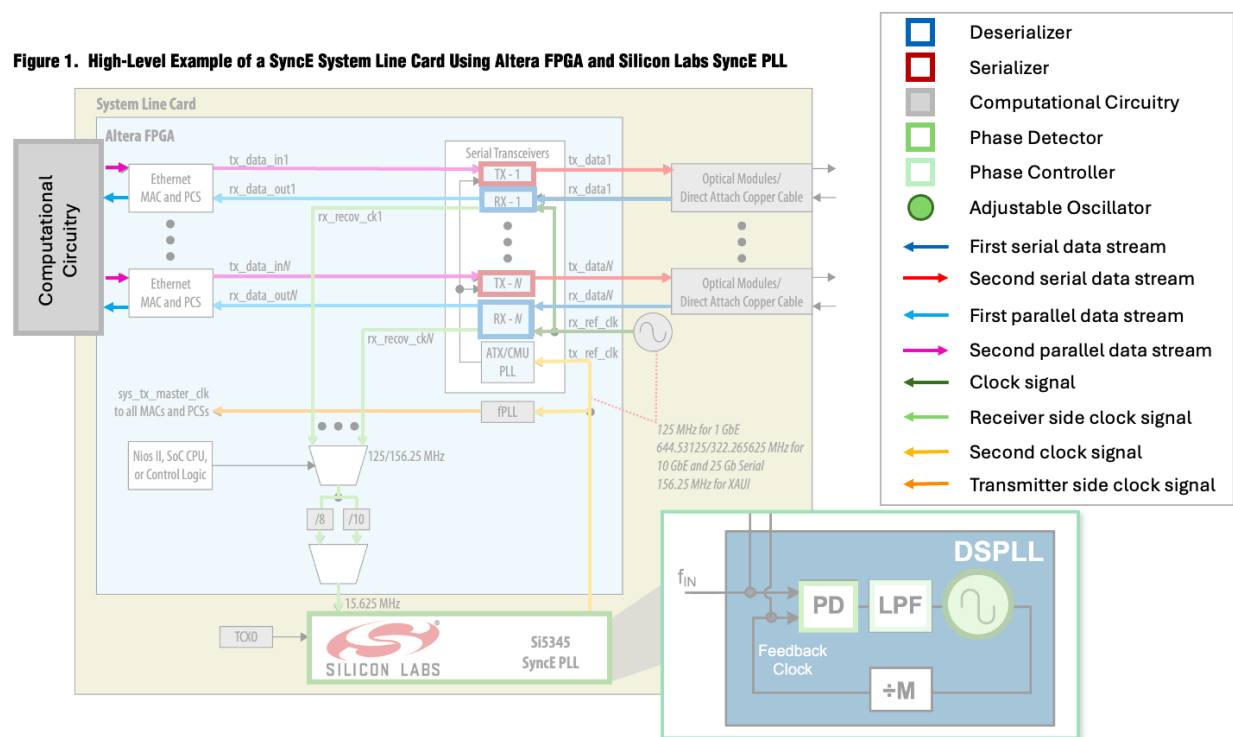


10. Claim 1(i): “performing, by the computational circuitry, a set of operations on at least a portion of the first plurality of parallel data streams to generate a second plurality of parallel data streams”

147. Altera discloses “performing, by the computational circuitry, a set of operations on at least a portion of the first plurality of parallel data streams to generate a second plurality of parallel data streams.”

148. The Stratix V FPGA used in Altera includes computational circuitry for performing operations on the first parallel data streams (the blue lines below) to generate the second parallel data streams (the pink lines below). Altera at 3

(discussing “FPGA core”). The Stratix Handbook describes “the features of the logic array block (LAB) in the Stratix® V core fabric.” Ex. 1005, Stratix Handbook at 256. That logic array block is the computation circuitry within Altera. The logic array block (LAB) “is composed of basic building blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.” *Id.*



149. A POSA would understand that the “computational circuitry” of an FPGA is also called: logic fabric, FPGA core, FPGA logic core (all terms used in the ’381 patent), or FPGA Fabric (a term used in Altera and Stratix Handbook). All of these terms describe the portion of the FPGA where computations are performed. For example, the ’381 patent states: “After deserialization, data is available for

computation in the FPGA logic core 106 where computation is completed. In embodiments, the computation of the data may be computed by receive-side computation 110a of the FPGA core 106. Similarly, data may be computed by the transmit-side computation 110b.” Ex. 1001, ’381 patent at 10:10-15. A POSA would understand that the claimed computational circuitry is called “FPGA Fabric” throughout Stratix Handbook.

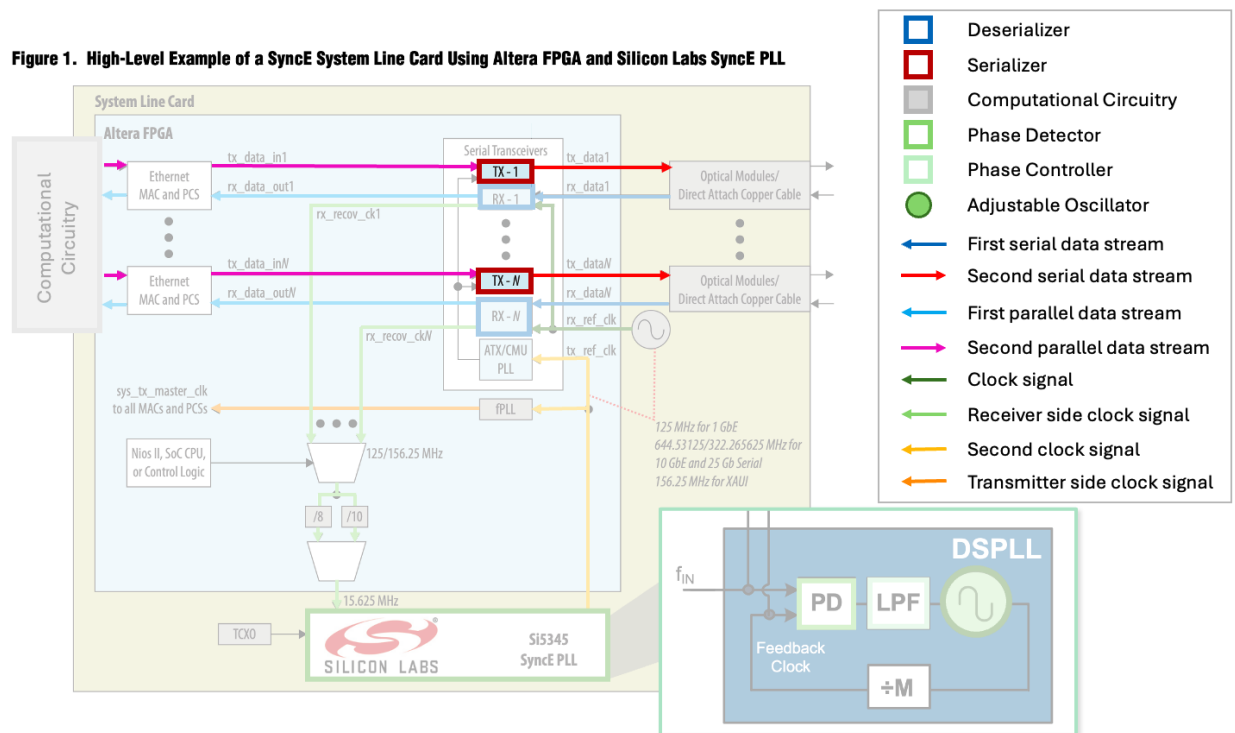
150. Altera explains that the FPGA-based solution was used for “voice, data, and video” applications. Ex. 1004, Altera at 1. Accordingly, a POSA would understand that the computational circuitry performs operations on information received as streams of parallel data words across one or a plurality of parallel data from the deserializer(s), and that such computational circuitry constructs, modifies, or triggers transmission of zero, one, or more packet(s) through each of one or more parallel interface(s) to the associated serializer(s) as a consequence. Altera at 2 describes “up to ten 1G/ 10Gbps Ethernet (1G / 10GbE) ports.”

151. *See also* Ex. 1013, Altera and IDT SyncE Solution at 2 (“In this system, the programmable logic in the FPGA is loaded with a design that includes one or more Ethernet cores, along with any other application logic needed on the card.”). A POSA would understand that this application logic performs operations on the received data streams.

11. Claim 1(j): “transmitting, from the field programmable gate array system, the second serial data stream, derived from the second plurality of parallel data streams”

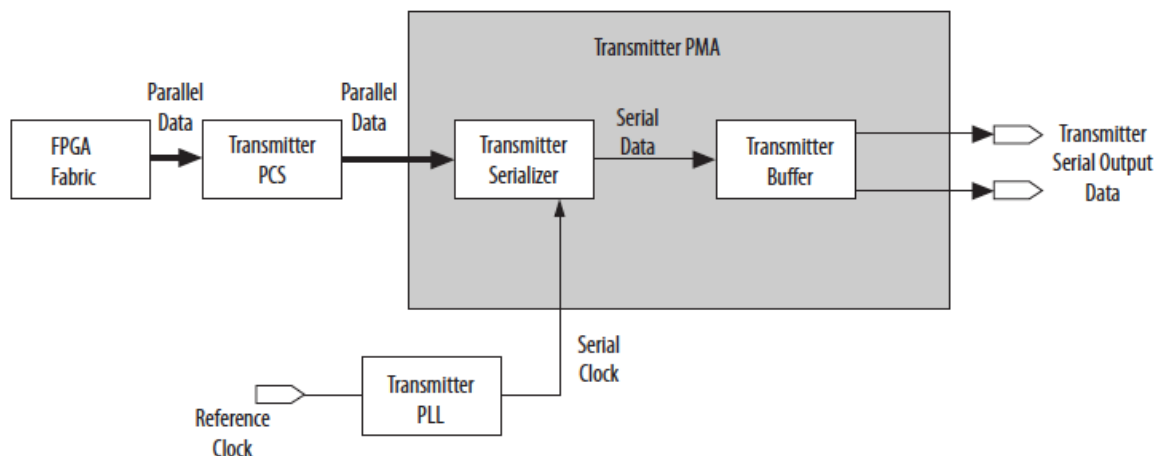
152. Altera discloses “transmitting, from the field programmable gate array system, the second serial data stream, derived from the second plurality of parallel data streams.”

153. As shown in Figure 1, the second plurality of parallel data streams, tx_data_in1 - tx_data_inN, are transmitted to the transmitter portion (TX-1 through TX-N) of the serial transceivers. The transmitters convert the parallel data (the pink lines below) streams to serial data streams (the red lines below). The serialized data, via tx_data1-tx_dataN, is then transmitted from the FPGA system to the optical modules/direct attach copper cable.



154. The Stratix Handbook confirms that Altera discloses this limitation. Figure 1-8 depicts parallel data being transmitted from the FPGA fabric into a serializer in the Transmitter PMA. The serializer generates serial data streams that are then transmitted off the FPGA. Ex. 1005, Stratix Handbook at 14 (“The PMA . . . transmits off-chip high-speed serial data streams.”), 26 (“The serializer converts the incoming low-speed parallel data from the transceiver PCS or FPGA fabric to high-speed serial data and sends the data to the transmitter buffer.”). This buffer is connected to the serial output data pins, *id.* at Figure 1-8 at 14.

Figure 1-8: Transmitter PMA Block Diagram



12. Claim 1 wherein clause: “wherein said method does not use clock domain crossing operations that delay processing of the first set of parallel data streams”

155. Altera discloses “wherein said method does not use clock domain crossing operations that delay processing of the first set of parallel data streams.” Altera does not teach any CDC operations.

156. In addition, a POSA would understand that in the system described in Altera, a clock domain crossing operation is not necessary in the computational circuitry where processing is performed (where the first set of parallel data streams are processed). Instead, Altera teaches that the transmitter side clock (sys_tx_master_clk, Altera at 2) is generated from the receiver side clock via the external PLL (and an internal PLL) and a POSA would understand that a clock domain crossing operation is not used. Thus, because a clock domain crossing operation is not used, a clock domain crossing cannot and does not delay processing of the first set of parallel data streams.

157. A clock domain crossing operation within the computational logic is necessary when asynchronous clock domains exist within the computational logic. Altera explains that its SyncE solution supports “[g]eneration of all system clock frequencies from a single recovered clock frequency,” with the ability for the “PLL to maintain stability and accuracy.” Altera at 3. The “single recovered clock” is the clock selected by the multiplexer among all potential receiver side clocks coming out of the deserializers where each is named rx_recov_clk[x] where [x] is 1-N, *id.*, Figure 1 of Altera at 2.

158. In the above quote, Altera uses the term “system clock frequencies” to refer to the clocks used by the computational logic that processes the parallel data

streams, all MACs, all PCSs and (while not relevant to this claim), the various serializers, as seen in Figure 1 of Altera at 2.

159. In other words, there are no clock domains to cross because the processing of the parallel, deserialized data uses a single clock; the clock from the selected deserializer.

C. Claim 4: “The method of claim 1, wherein the set of operations includes at least one of the following: (i) an arithmetic operation; (ii) a logical operation; (iii) a pipeline operation; and (iv) a memory access operation.”

160. Altera discloses “[t]he method of claim 1, wherein the set of operations includes at least one of the following: (i) an arithmetic operation; (ii) a logical operation; (iii) a pipeline operation; and (iv) a memory access operation.” In my discussion of 1(i) above I show that operations are anticipated by Altera. Here, I expand on the teachings to show certain specific operations.

161. A POSA would understand that any useful application of an FPGA that uses the serial transceivers must necessarily also use the computational circuitry. Altera explains that the FPGA-based solution was used for “voice, data, and video” applications. Altera at 1. Accordingly, a POSA would understand that the computational circuitry (i.e., FPGA fabric or FPGA core) performs operations on packets after they are received as streams of parallel data words from one or more deserializers, and that such operations would result in the transmission of zero, one, or more packet(s) by one or more parallel interface(s) to the associated serializer(s)

as a consequence. Altera at 2 describes “up to ten 1G/ 10Gbps Ethernet (1G / 10GbE) ports.” A POSA would understand that this network-connected FPGA would perform a variety of operations, including:

- an arithmetic operation, e.g. computation of a TCP/UDP/IP checksum
- a memory operation, e.g. looking up and potentially updating statistical information using information in the packet as the key
- a logical operation, e.g. deciding whether to process the packet further or drop it, using combinatorial logic (AND, OR, NOT, XOR, etc.)
- a pipeline operation, where at least one layer of flip flops between the receive and transmit interfaces creates a pipeline. A POSA would understand that pipelines are ubiquitous in synchronous digital designs and used, for example, when the parallel data words of a received packet (from the deserializer) are parsed, when a memory is read, and as fields of a packet are modified ahead of its transmission (serialization).

162. Arithmetic, logical, pipeline, and memory access operations are implemented in the FPGA, and a POSA would understand that these are the main facilities provided by an FPGA. For example, the Stratix Handbook describes “the features of the logic array block (LAB) in the Stratix® V core fabric.” Ex. 1005, Stratix Handbook at 256. The logic array block “is composed of basic building

blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions. You can use half of the available LABs in the Stratix V devices as a memory LAB (MLAB).” *Id.* The Stratix Handbook also includes a chapter that explains how to utilize embedded memory blocks in the FPGA when a design needs low-latency data storage that sits close to the programmable logic, and that the operation can be a memory access operation. *Id.* at 271. That chapter also refers to optional pipelining, which breaks long data paths into multiple stages and takes advantage of an FPGA’s parallel-processing abilities. *Id.* at 290. Accordingly, Altera discloses that the FPGA Fabric performs operations that are arithmetic, logical, pipeline, or memory access operations.

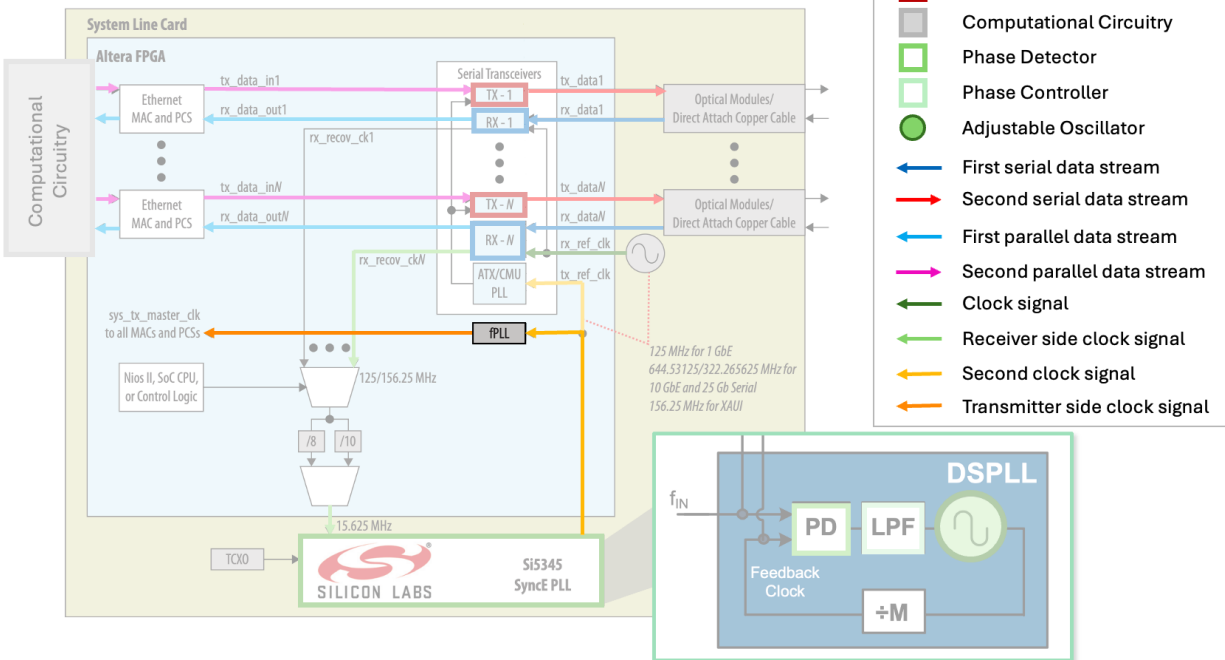
D. Claim 6: “The method of claim 1, wherein at least a portion of the set of operations performed in step (h) are performed after step (g).”

163. Altera discloses “[t]he method of claim 1, wherein at least a portion of the set of operations performed in step (h) are performed after step (g).”

164. Altera explains that a second clock signal is generated using the external PLL (g) before the transmitter-side clock signal derived from the second clock signal is generated within the FPGA (h). Altera explains that a “stabilized clock” is generated by the Si5345 PLL. Ex. 1004, Altera at 3. That “stabilized clock” is then “used to feed the FPGA’s serial transmitter (TX) PLL in sync with the master

Ethernet port's clock frequency to realize a complete synchronization loop.” *id.* at 3; *see also id.* at 4 (Si5345 PLL “suppl[ies] a clean reference clock for the serial transceivers’ TX PLLs synchronous to the RX recovered clock”). Figure 1 shows that the tx_ref_clk clock (the “second clock signal”) generated by the external PLL is passed into the FPGA and is passed to the fPLL, which generates the transmitter side clock (sys_tx_master_clk) that is provided to the MAC and PCS, and elsewhere, for use with parallel data. Thus, the generation of the transmitter side clock signal, derived from the second clock signal, is performed after the second clock signal is generated by the external PLL. A POSA would understand that a dependent clock cannot be generated before the clock on which it depends. Because the totality of step (g) is performed before step (h), it’s true that at least a portion of the steps are performed in the specified order.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL

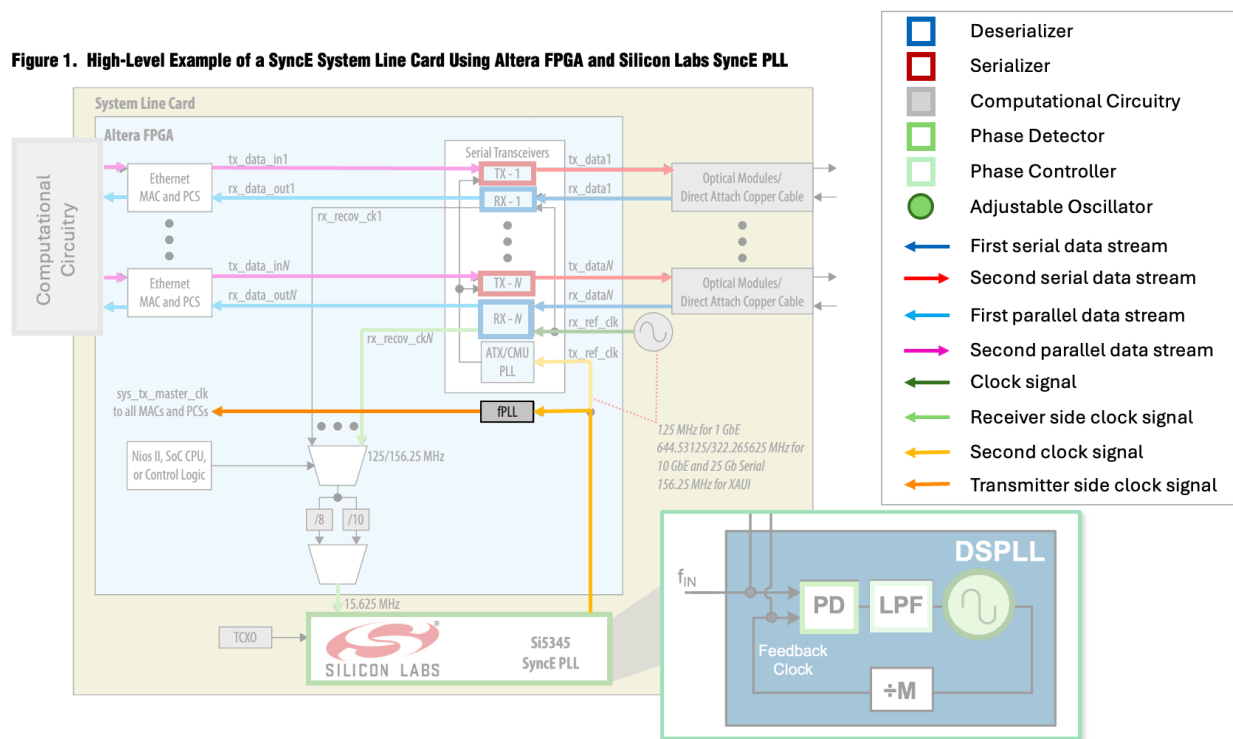


E. Claim 7: “The method of claim 1, wherein all of the set of operations performed in step (h) are performed after step (g).”

165. Altera discloses “[t]he method of claim 1, wherein all of the set of operations performed in step (h) are performed after step (g).”

166. Altera explains that a second clock signal is generated using the external PLL (g) before the transmitter-side clock signal derived from the second clock signal is generated within the FPGA (h). Altera explains that a “stabilized clock” is generated by the Si5345 PLL. Ex. 1004, Altera at 3. That “stabilized clock” is then “used to feed the FPGA’s serial transmitter (TX) PLL in sync with the master Ethernet port’s clock frequency to realize a complete synchronization loop.” Altera at 3; *see also id.* at 4 (Si5345 PLL “suppl[ies] a clean reference clock for the serial transceivers’ TX PLLs synchronous to the RX recovered clock”). Figure 1 shows

that the tx_ref_clk clock (the “second clock signal”) generated by the external PLL is passed into the FPGA and is passed to the fPLL, which generates the transmitter side clock (sys_tx_master_clk) that is provided to the MAC and PCS, and elsewhere, for use with parallel data. Thus, the generation of the transmitter side clock signal, composed of all operations necessary to derive it from the second clock signal, is performed after the second clock signal is generated by the external PLL.



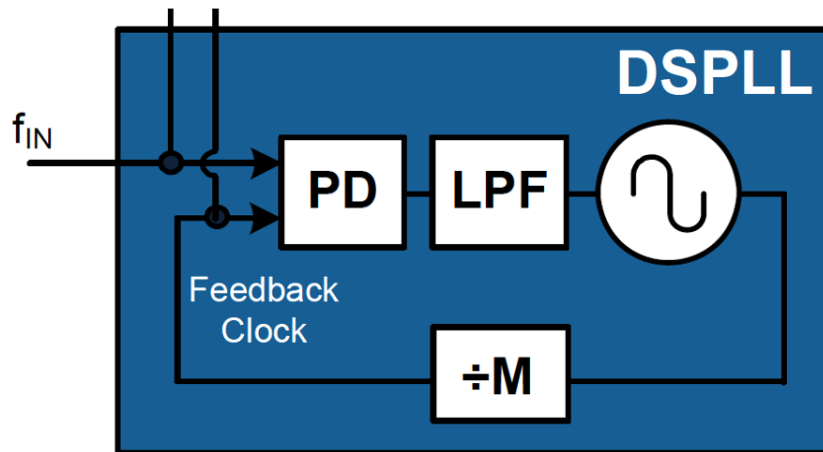
F. Claim 8(i): “The method of claim 1, wherein step (g) further comprises (i) generating, by the phase lock loop, a feedback clock signal associated with the transmitter side clock signal by performing the following steps until a first output of a phase detector of the field programmable gate array system is below a first threshold level.”

167. Altera discloses “[t]he method of claim 1, wherein step (g) further comprises (i) generating, by the phase lock loop, a feedback clock signal associated

with the transmitter side clock signal by performing the following steps until a first output of a phase detector of the field programmable gate array system is below a first threshold level.”

168. Step (g)—“generating, using the phase lock loop, a second clock signal”—takes place within the Si5345 SyncE PLL disclosed in Altera. Ex. 1004, Altera at 2. The limitations of claim 8 describe the normal operation of a PLL.

169. In Si5345, “[i]f at least one valid clock is available for synchronization, the DSPLL will automatically start the lock acquisition process. . . . During lock acquisition the outputs will generate a clock that follows the VCO frequency change as it pulls-in to the input clock frequency.” Ex. 1006, Si5345 Manual at 17. Figure 13 of Si5345 discloses that during lock acquisition, the PLL generates a “feedback clock” signal that is compared against the input clock signal. *Id.* at 27. The DSPLL continues in lock acquisition mode, following the steps set out in claim 8, until the output of the phase detector is below a first threshold level. “An example configuration where LOCK is indicated when there is less than 0.2 ppm frequency difference at the inputs of the phase detector.” *Id.* at 27.



Si5345/44/42 Figure 13 Excerpt

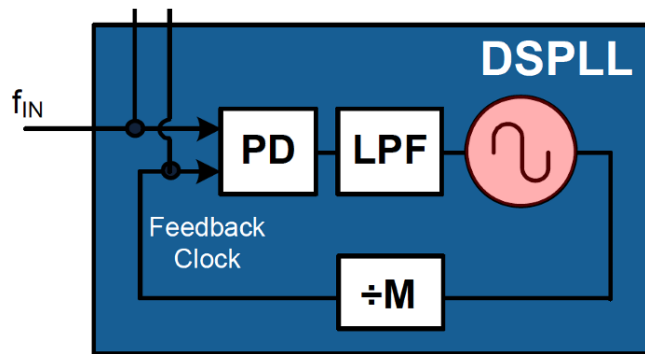
170. In Altera, the feedback clock is “associated” with the transmitter side clock signal—it is connected to and drives the tx_ref_clk signal of the FPGA. Ex. 1004, Altera at 2. The feedback clock is used by the external PLL in generating the second clock signal, and the second clock signal is in turn used to derive (and is thus associated with) the transmitter side clock signal as described in responses to claim 1(g) and 1(h) above.

G. Claim 8(i)(1): “generating, by an adjustable oscillator in the field programmable gate array system, the second clock signal”

171. Altera discloses “generating, by an adjustable oscillator in the field programmable gate array system, the second clock signal.”

172. The Si5345 includes an adjustable oscillator, illustrated with a sine wave symbol and highlighted in red. Ex. 1006, Si5345 Manual at 10, 11, 27. The adjustable oscillator generates the second clock signal. As shown in Figure 13, the second clock signal is transmitted to a divide-by-M, which generates the feedback

clock signal, as labeled in Figure 13. *Id.* at 27. Ultimately, as shown in Figure 2, “[o]nce locked, the DSPLL will generate output clocks that are both frequency and phase locked to its selected input clock,”—which is the second clock signal, generated by the adjustable oscillator. *Id.* at 17.



Si5345/44/42 Figure 13 Excerpt

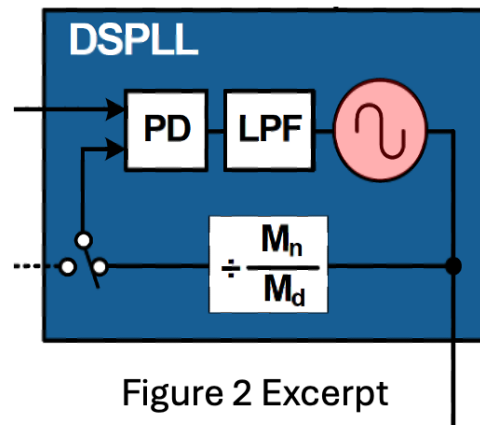
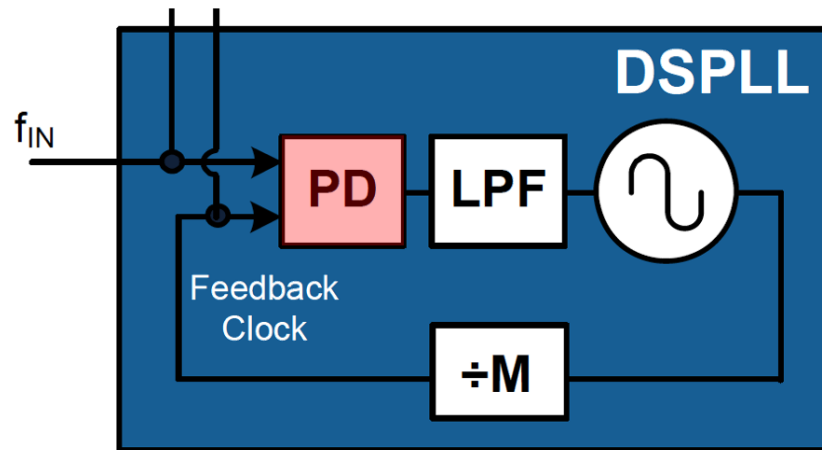


Figure 2 Excerpt

H. Claim 8(i)(2): “generating, by the phase detector, the first output based on a comparison of the receiver side clock signal and the feedback clock signal obtained from the second clock signal”

173. Altera discloses “generating, by the phase detector, the first output based on a comparison of the receiver side clock signal and the feedback clock signal obtained from the second clock signal.” The Si5345 includes a phase detector, abbreviated as “PD” and highlighted in red in Figure 13 below. Ex. 1006, Si5345 Manual at 10, 11, 27. The phase detector compares the input clock signal (f_{IN}) to the feedback clock signal. In Altera, the input clock signal is the receiver side clock signal and, as discussed in Claim 8(i)(1), the feedback clock signal is obtained from the second clock signal passing through the (programmable) divider. Based on that

comparison, the phase detector generates an output signal that is a function of the phase difference between the two clock signals.



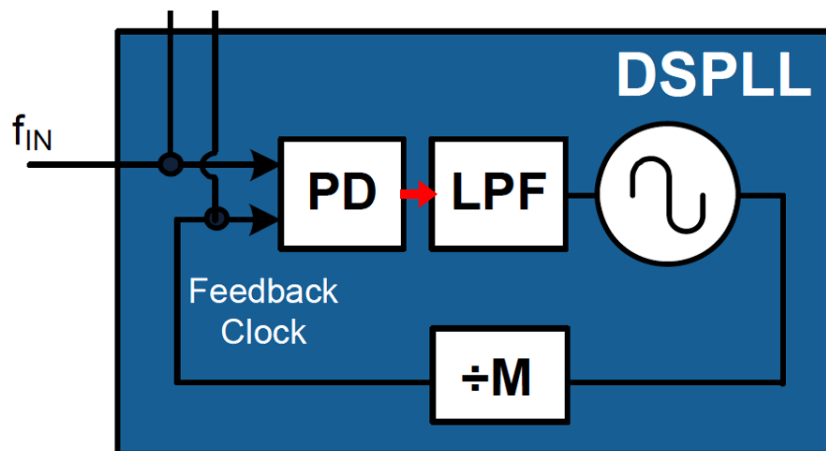
Si5345/44/42 Figure 13 Excerpt

I. Claim 8(i)(3): “transmitting, from the phase detector to a phase controller of the field programmable gate array system, the first output”

174. Altera discloses “transmitting, from the phase detector to a phase controller of the field programmable gate array system, the first output.”

175. In Si5345, the phase detector PD transmits the first output signal to the phase controller (also called a loop filter) and shown as a low-pass filter (LPF). Ex. 1006, Si5345 Manual at 10, 11, 27. A LPF is the phase controller in Altera 5: “A PLL loop bandwidth below 10 Hz is required for the PLL to adequately filter wander for SyncE.” In this case, a PLL loop bandwidth of 10 Hz is given, meaning that low frequencies below that frequency are largely passed through the filter (a LPF). The LPF receives the output of the phase detector and outputs phase control information.

176. A PLL is a feedback control system, and the role of the LPF is explained in “Design of Control Systems” by A. Frank D’Souza. Ex. 1030.¹⁹ “A transfer function that attenuates the amplitude of high-frequency inputs is called a filter of high frequencies or a low-pass filter.” *Id.* at page 173. “The bandwidth is defined as the spread between zero frequency and the cutoff frequency[.]” *Id.* at 177. The ’381 patent agrees that a LPF can serve as the loop filter, stating in the context of the phase controller: “The loop filter consists of a standard PLL loop filter with bandwidth between 400 Hz and 40 kHz . . . The bandwidth of the standard PLL loop filter, in some embodiments, may range between . . . , 0 Hz and 100 Hz, to name a few.” Ex. 1001, ’381 patent at 16:3-10.



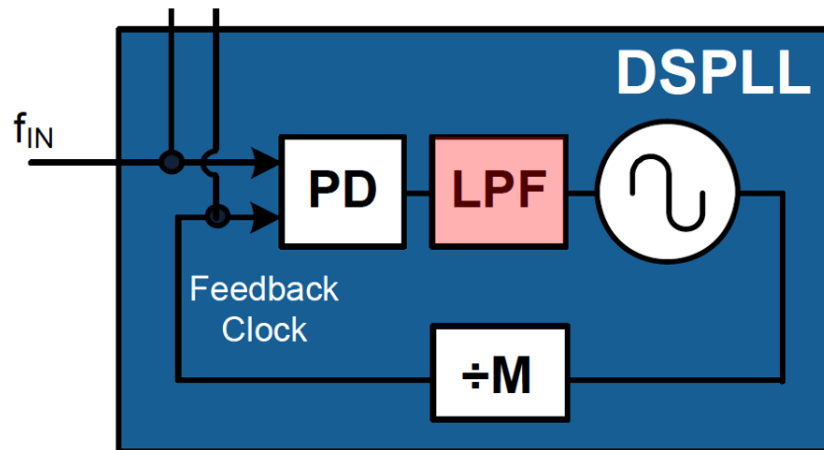
Si5345/44/42 Figure 13 Excerpt

¹⁹ “Design of Control Systems” by A. Frank D’Souza, published in 1988 Prentice Hall.

J. Claim 8(i)(4): “determining, by the phase controller, interim adjustment information based on the first output”

177. Altera discloses “determining, by the phase controller, interim adjustment information based on the first output.”

178. As described above, the LPF is a phase controller that uses the first output from the phase detector to determine how to adjust the clock signal by selectively passing only certain frequencies through toward the VCO. Ex. 1006, Si5345 Manual at 27. The result of the LPF operation, prior to being sent to the VCO, is interim adjustment information.

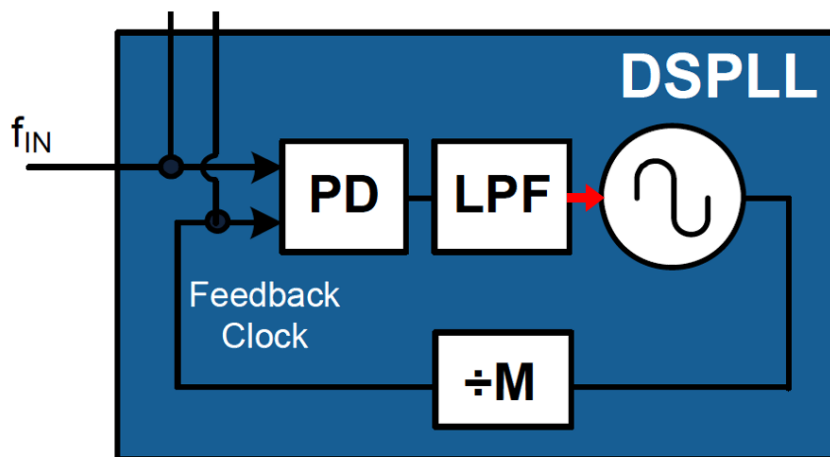


Si5345/44/42 Figure 13 Excerpt

K. Claim 8(i)(5): “transmitting, from the phase controller to the adjustable oscillator, the interim adjustment information”

179. Altera discloses “transmitting, from the phase controller to the adjustable oscillator, the interim adjustment information.”

180. In Si5345, the LPF transmits the interim adjustment information to the adjustable oscillator so the adjustable oscillator can adjust the clock signal. *Id.* at 10, 11, 27.



Si5345/44/42 Figure 13 Excerpt

L. Claim 8 wherein clauses

181. Altera discloses **“wherein, the adjustable oscillator adjusts the second clock signal based on the interim adjustment information.”** As discussed for claim 8(i)(1), the adjustable oscillator adjusts the second clock signal. As shown in Figure 13, the second clock signal is transmitted to an M divider, which generates the feedback clock signal to the PD. Ultimately, as shown in Figure 2, “[o]nce locked, the DSPLL will generate output clocks that are both frequency and phase locked to its selected input clock,” meaning the second clock signal will be output. Ex. 1006, Si5345 Manual at 17.

182. Altera also discloses “wherein steps (1) through (5) are repeated until the first output of the phase detector is below the first threshold level.” The DSPLL continues in lock acquisition mode, following the steps set out in claim 8, until the output of the phase detector is below a first threshold level, at which point it enters the LOCKED state. “An example configuration where LOCK is indicated when there is less than 0.2 ppm frequency difference at the inputs of the phase detector.” *Id.* at 27.

M. Claim 9: “The method of claim 1, wherein the first plurality of data streams has the same number of data streams as the second plurality of data streams.”

183. Altera discloses “[t]he method of claim 1, wherein the first plurality of data streams has the same number of data streams as the second plurality of data streams.”

184. A POSA would understand that there is a 1-to-1 correspondence of deserializers and serializers in Altera, each of which is associated with a data stream. Figure 1 shows that for every RX block there is also a TX block, totaling N in number. Ex. 1004, Altera at 2; *see also* Ex. 1005, Stratix Handbook at 7 (showing each transceiver channel has one receiver and one transmitter). And Altera explains that the “Altera® SyncE reference design implement[s] up to ten 1G/10 Gbps Ethernet (1G/10 GbE) ports,” indicating that in Altera there are up to 10 transceiver channels (e.g., 10 deserializers and 10 serializers). Ex. 1004, Altera at 2.

- N. Claim 10: “The method of claim 1, wherein the first plurality of data streams and the second plurality of data streams comprise one of the following: (i) eight (8) data streams; (ii) ten (10) data streams; (iii) sixteen (16) data streams; (iv) twenty (20) data streams; (v) thirty-two (32) data streams; (vi) forty (40) data streams; (vii) sixty-four (64) data streams; (viii) eighty (80) data streams; (ix) one hundred twenty-eight (128) data streams; and (x) one hundred sixty (160) data streams.”**

185. Altera discloses “[t]he method of claim 1, wherein the first plurality of data streams and the second plurality of data streams comprise one of the following: (i) eight (8) data streams; (ii) ten (10) data streams; (iii) sixteen (16) data streams; (iv) twenty (20) data streams; (v) thirty-two (32) data streams; (vi) forty (40) data streams; (vii) sixty-four (64) data streams; (viii) eighty (80) data streams; (ix) one hundred twenty-eight (128) data streams; and (x) one hundred sixty (160) data streams.”

186. As discussed above with respect to claim 9, Altera discloses a configuration with ten ports, meaning there are ten deserializers and ten serializers, and therefore ten data streams. *See* Ex. 1004, Altera at 2.

X. GROUND 1B: ALTERA, STRATIX, AND SI5345 RENDER CLAIMS 1, 4, AND 6-10 OBVIOUS

187. To the extent that a POSA would not have understood that the features of Stratix FPGA and Si5345 PLL are necessarily present in Altera, a POSA would have been motivated to combine the Stratix V Handbook and Si5345 Reference Manual using what Altera describes “a high-performance SyncE solution” that

consists of “Altera’s Stratix[®] V FPGA interoperating with Silicon Labs’ Si5345 jitter-attenuating phase-locked loop (PLL) to provide a complete SyncE clock recovery, clock selection, and synchronization sub-system.” Ex. 1004, Altera at 2. The Altera solution was developed by “Altera and Silicon Labs” and specifically used “the two companies’ products.” *Id.* at 2. And, as described above, a POSA would have been able to easily find and rely on the Stratix V Handbook and Si5345 Reference Manual, as they are among the specifications and data sheets that Altera and Silicon Labs made available for download from their respective websites. *See* Section IX.A above. The POSA would naturally be expected to follow links contained within these documents to gain additional information, as necessary.

188. Further, a POSA would have reasonably expected the combination of the Stratix FPGA and Si5345 PLL as set out in Altera to succeed, as Altera reports that its technique was successfully tested. Altera at 6. For example, “Altera and Silicon Labs have validated that the jitter performance of Silicon Labs’ Si5345 PLL meets the requirement of Altera’s Stratix V FPGA input reference clock and that the serial transceiver output jitter meets G.8262 requirements. The test includes Silicon Labs’ Si5345 PLL output jitter, phase noise tolerance at the PLL input, the clock synchronization loop’s phase noise filtering capability, and the ability of the loop to withstand short-term and long-term input interruptions and discontinuities.” Ex. 1004, Altera at 6. A POSA would have been confident in the success of the system

since Altera “demonstrated a high-performance and flexible solution for frequency synchronization of telecom and data communication networks based on the Sync standard using the Ethernet infrastructure.” *Id.*

189. Therefore, a POSA would have had reason to combine Stratix with Si5345 in precisely the same way that Altera says they should be combined. That combination of Altera, the Stratix Handbook, and the Si5345 Manual discloses all the limitations of claims 1, 4, and 6-10 for the same reasons as Altera anticipates those claims.

XI. GROUND 2: ALTERA AND LOCKWOOD RENDER CLAIMS 1-4 AND 6-12 OBVIOUS.

190. Altera (alone or in combination with the Stratix Handbook and Si5345 Manual) presents a solution that would have been attractive to a POSA to apply to low-latency trading applications as taught by Lockwood.

A. A Skilled Artisan Would Have Been Motivated to Combine Altera and Lockwood

191. It was common knowledge before the '381 patent that FPGA-based systems are useful for performing trading operations, as even the patent acknowledges. Ex. 1001, '381 patent at 1:42-44 (“FPGAs are used in the financial industry in high frequency trading where the rapid processing of the FPGA is desired.”). Lockwood demonstrates how a skilled artisan would have put Altera to use for trading applications.

192. Lockwood emphasizes “the race to minimize latency in high frequency trading.” Ex. 1007, Lockwood at 7. In low-latency trading, “[a] custom-designed datapath in an FPGA circuit offers the benefits of minimal, deterministic processing times down to clock cycles.” *Id.* at 8. In a regime like low-latency trading in which even a few clock cycles can make a difference, a skilled artisan would have recognized that Altera offers a solution that eliminates the latency cost associated with clock-domain crossing circuits. *See id.* at 2 (“[T]he first few players to execute orders may be the only ones able to profit from a given opportunity. . . . Every millisecond reduction in latency can improve arbitrage profitability by more than \$100M a year.”); Ex. 1014, Sahoo (“HFT is all about speed and minimizing latency: the faster you can run trading strategies and algorithms for analyzing minute price changes and executing trade orders, the higher the probability to win over competition.”)²⁰.

193. Additionally, Altera explains that its benefit is “high reliability,” resulting in “a high-performance and flexible solution for frequency synchronization of telecom and data networks based on the SyncE standard using the Ethernet infrastructure.” Ex. 1004, Altera at 2, 6. Lockwood explains that “Ethernet is currently more prevalent in the HFT ecosystem” and “will continue to dominate for

²⁰ Ex. 1014, <https://www.aldec.com/en/company/blog/169--the-race-to-zero-latency-for-high-frequency-trading>

the foreseeable future.” Ex. 1007, Lockwood at 2. The passage of time demonstrated that to be true. Thus, a POSA would have been motivated to use Altera in trading applications like Lockwood describes.

194. Lockwood provides several different examples that a POSA would recognize the Altera system could be incorporated in. For example, Lockwood describes an FPGA-based Smart Network Interface Controller (NICs), in which “some of the processing” is offloaded onto “a Field Programmable Gate Array (FPGA) on the network adapter.” *Id.* at 8. “Smart NICs typically bring together high-speed network interfaces, a PCIe host interface, memory and an FPGA. The FPGA implements the NIC controller, acting as the bridge between the host computer and the network and allows user-designed custom processing logic to be integrated directly into the data path.” *Id.* at 3. Altera—which touts its usefulness, for example, for in networking equipment—would be used as the FPGA in Lockwood Figure 2 and the computational circuitry of the Altera FPGA would be “function as a programmable trading platform.” *Id.*; *see also* Ex. 1004, Altera at 2.

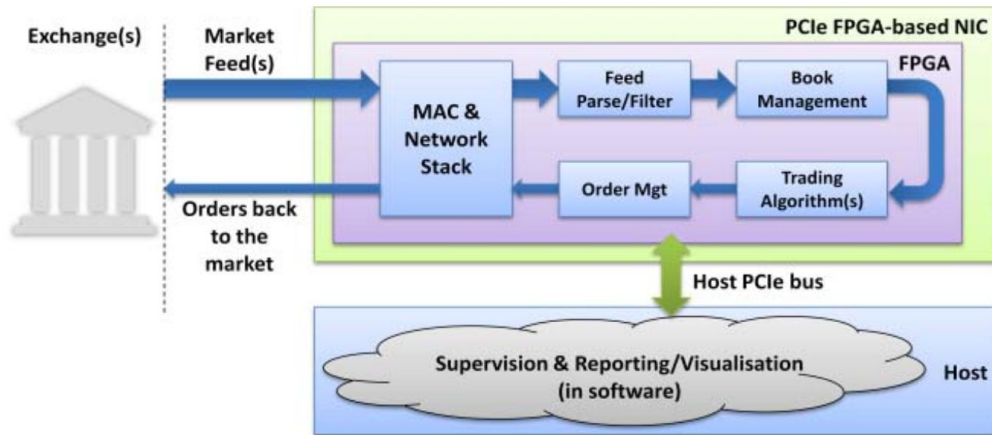


Figure 2 Typical Smart NIC in a supervised financial trading application

195. A skilled artisan would also recognize that Altera would be an attractive option to use for the FPGA shown in Figure 3 (bottom). Altera focuses on the physical and data link layers shown in Lockwood Figure 3 (bottom), leaving a skilled artisan free to use the programmable logic of the Altera FPGA for financial applications and other necessary functions.

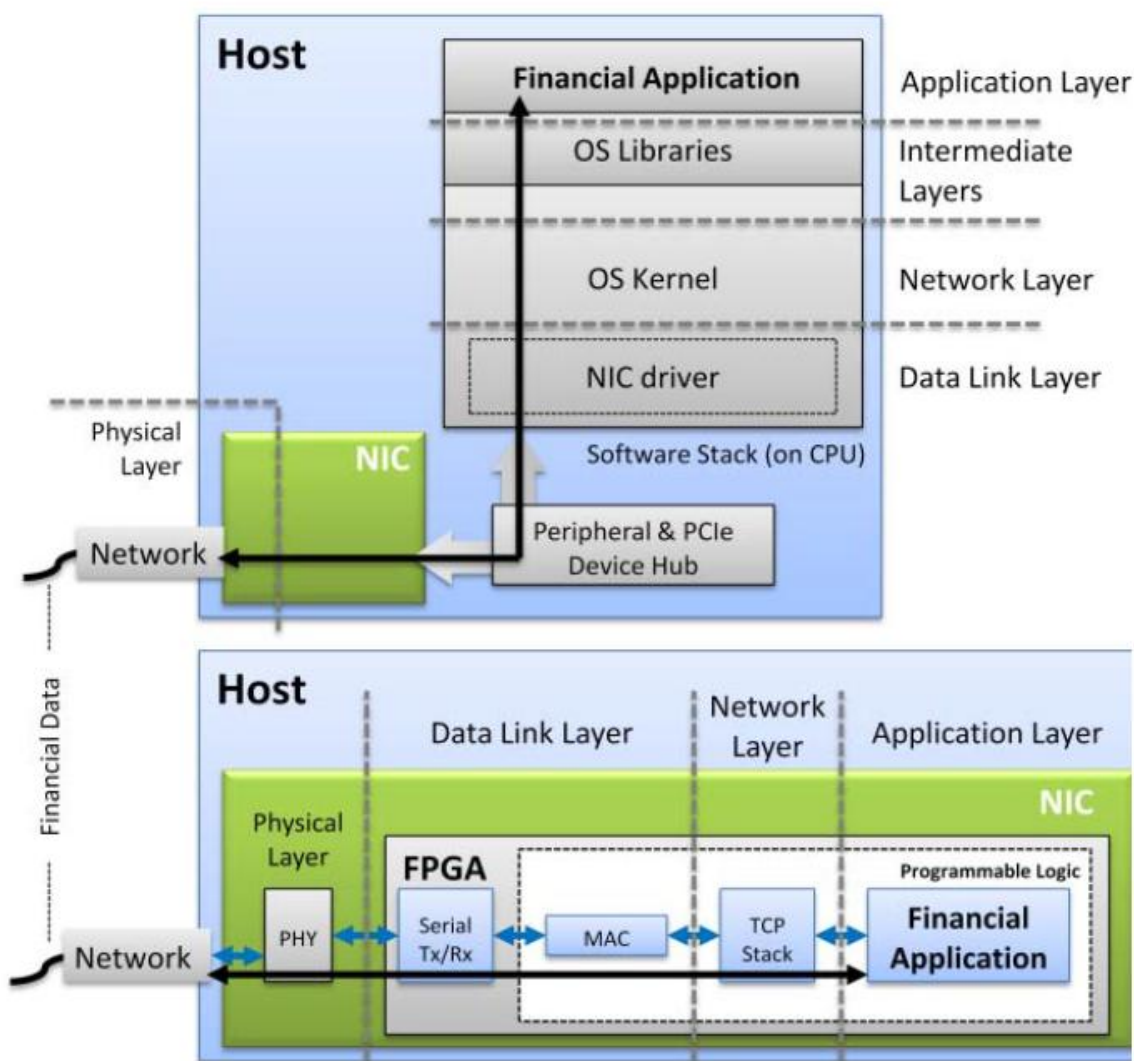


Figure 3 Software-based (top) and FPGA-based (bottom) trading platforms

196. A POSA seeking to implement the Altera FPGA for trading applications would have a reasonable expectation of success. An FPGA is programmable, meaning the computational circuitry can be programmed for whatever function the user desires. Ex. 1005, Stratix Handbook at 256 (The “logic array block (LAB) in the Stratix® V core fabric . . . is composed of basic building

blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.”); Ex. 1016, Aldridge at 23 (“an FPGA provides a blank slate of bitwise functional units that can be programmed to create any desired circuit or processor”).

197. To the extent that a POSA would not have understood that the features of Stratix FPGA and Si5345 PLL are necessarily present in Altera, a POSA would have been motivated to combine Altera, the Stratix Handbook, the Si5345 Manual, and Lockwood for all the same reasons. *See* Ground 1B above.

B. Claim 1(e), 1(i), and/or 4: It Would Have Been Obvious to Use the Computational Circuitry Present in Altera

198. To the extent that a POSA would have concluded that Altera does not describe using the computational circuitry of the Stratix FPGA—e.g., in claim limitations 1(e), 1(i), and/or 4—a POSA would have been motivated to use the Stratix computational circuitry with Altera.

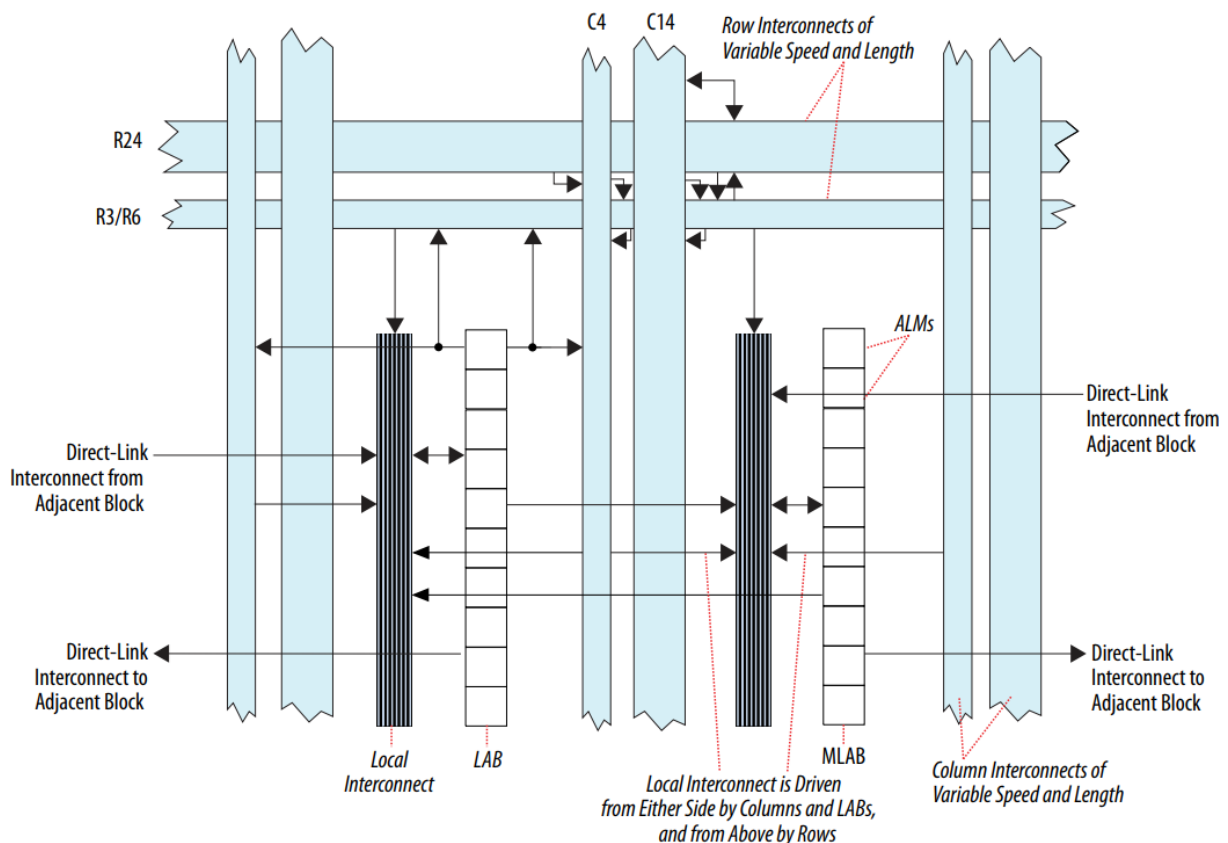
1. Claim 1(e) and 1(i)

199. Altera and Lockwood teach “transmitting, from the deserializer to computational circuitry in the field programmable gate array, the first plurality of parallel data streams” and “performing, by the computational circuitry, a set of operations on at least a portion of the first plurality of parallel data streams to generate a second plurality of parallel data streams.”

200. The Stratix FPGA contains a “logic array block (LAB) in the Stratix® V core fabric.” Ex. 1005, Stratix Handbook at 256. The logic array block “is composed of basic building blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.” *Id.* The Stratix FPGA also contains memory that is accessible to the FPGA’s computational logic, referred by the name “MLAB.” *Id.* at 257 (Figure 1-1).

Figure 1-1: LAB Structure and Interconnects Overview in Stratix V Devices

This figure shows an overview of the Stratix V LAB and MLAB structure with the LAB interconnects.



201. The POSA would understand that, together, these FPGA capabilities were available to perform arithmetic, logical, pipeline, and memory access operations.

202. It would have been obvious and beneficial for a POSA to put this computational circuitry to use in the Altera solution. The benefit of an FPGA relative to other custom hardware / integrated circuits is that an FPGA is programmable, meaning that a customer can configure the FPGA to carry out any such operations it needs. In the case of Altera, it would be obvious that a customer using the SyncE solution provided would use the FPGA computational circuitry to perform operations on the incoming voice, data, or video data streams. For example, another Altera SyncE white paper discloses that “[i]n this system, the programmable logic in the FPGA is loaded with a design that includes one or more Ethernet cores, along with any other application logic needed on the card.” Ex. 1013, Altera and IDT SyncE Solution at 2.

203. Comparing an FPGA to a CPU, Aldridge at 23 describes the core characteristic of an FPGA: “an FPGA provides a blank slate of bitwise functional units that can be programmed to create any desired circuit or processor. Some FPGAs contain a number of dedicated functional units, such as multipliers and memory blocks. Most of the area of an FPGA, however, is dedicated to the routing

infrastructure the run-time connectivity of the FPGA's functional units." *See* Ex. 1016, Aldridge at 23.

204. Lockwood reinforces that conclusion. Lockwood Figure 3 (bottom) depicts financial applications (e.g., trading algorithms) implemented in the programmable logic of the FPGA. Lockwood explains that "FPGAs offer the performance benefits of custom hardware while retaining programmability," such that "[t]rading algorithms can be continually improved." Ex. 1007, Lockwood at 4; *see also id.* at 2 ("Flexibility is also vital to enable trading platforms to adapt to changing market conditions and trading strategies."). Accordingly, it would have been obvious to perform operations using the computational circuitry of the Stratix V FPGA in Altera.

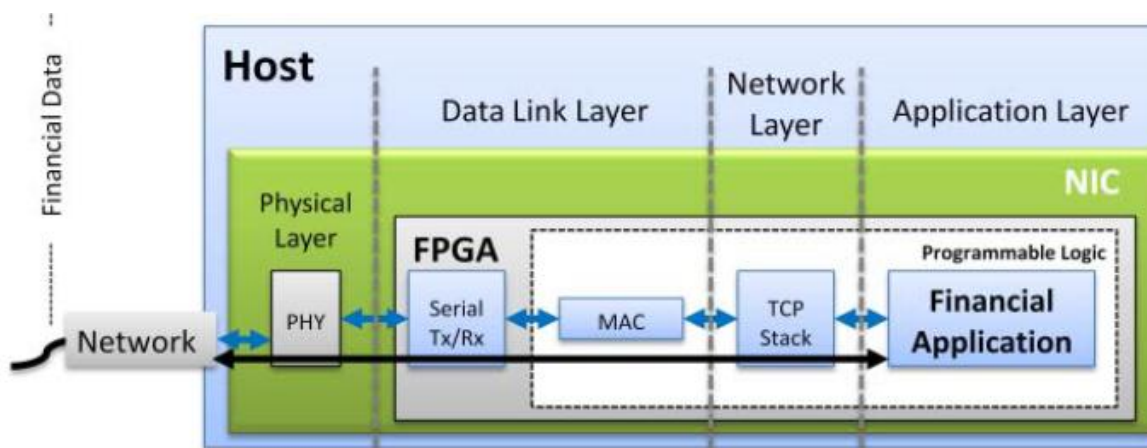


Figure 3 Software-based (top) and FPGA-based (bottom) trading platforms

205. Lockwood at 7 shows that "calculations" are performed on packets within the FPGA, paying particular attention to the latency through the FPGA.

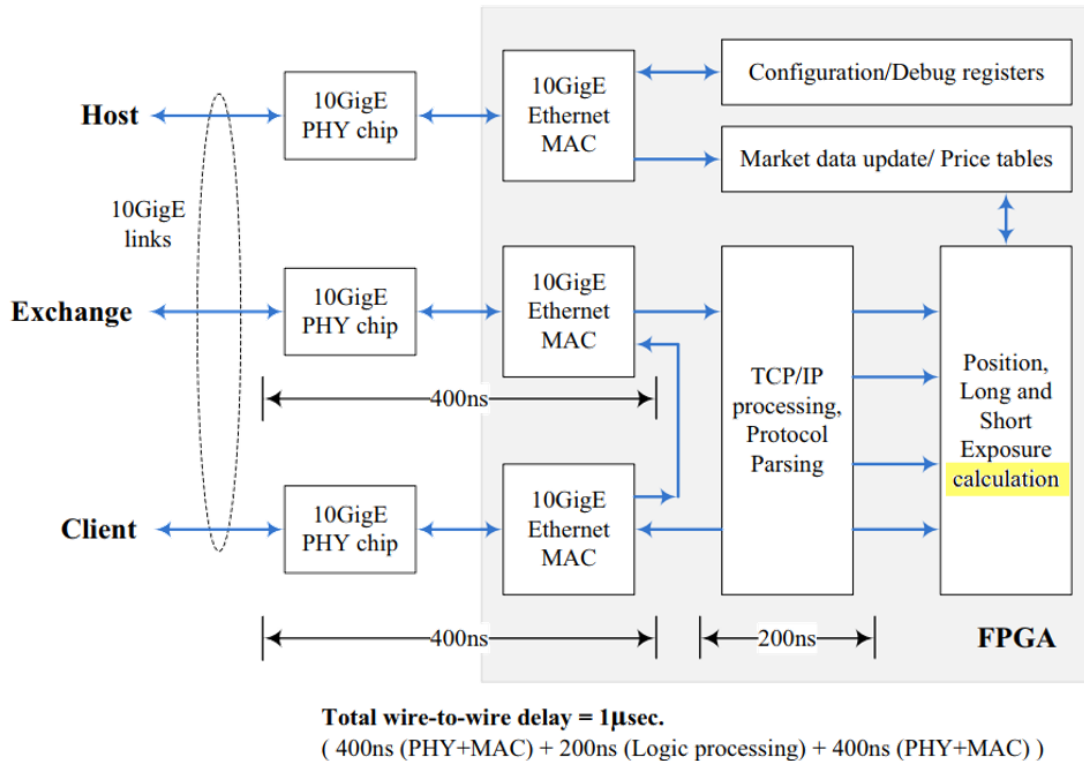


Figure 9 Exposure and Position-tracking Application built on Algo-Logic's HFT Platform

2. Claim 4

206. Altera and Lockwood teach “[t]he method of claim 1, wherein the set of operations includes at least one of the following: (i) an arithmetic operation; (ii) a logical operation; (iii) a pipeline operation; and (iv) a memory access operation.”

207. With respect to the types of operations described in claim 4 (arithmetic, logical, pipeline, or memory access), the Stratix V FPGA can be “configure[d] to implement logic functions, arithmetic functions, and register functions,” among others. Ex. 1005, Stratix Handbook at 256. A skilled artisan would understand that

trading operations like those described in Lockwood will include at least logical or arithmetic operations. Ex. 1007, Lockwood at 1 (“Traders employ a range of algorithms and heuristics to decide when and what to trade. These approaches combine current market data with historical data and outputs from computational models.”). Lockwood further describes the operations performed in more detail. *Id.* at 7.

- An arithmetic operation: “net position = (total long positions) – (total short positions).” *Id.* at 7.
- A logical operation: “If it is a Buy order . . . , then . . .” *Id.* at 7.
- A pipeline operation: “The breakup of 1μsec total delay is 400ns (PHY+MAC, high-speed serial receiver) + 200ns (parsing/calculations) + 400ns (PHY+MAC, high-speed serial transmitter).” *Id.* at 7. The 200ns of parsing / calculation includes pipeline operations at the clock rate of 156 MHz described at *id.* at 6.
- A memory access operation: “[D]ata is stored in fast on-chip memory and QDR-II directly attached to the FPGA. This memory is accessed . . . when an order enters [sic] FPGA and the table lookups and computation are performed directly in logic.” *Id.* at 7.

C. Claims 2 and 3: It Would Have Been Obvious to Use Altera for Trading Applications

208. Altera presents a solution that would have been attractive to a POSA to apply to trading applications as taught by Lockwood, rendering claims 2 and 3 obvious.

1. Claim 2

209. Altera and Lockwood teach “[t]he method of claim 1, wherein the first serial data stream comprises market data, the second serial data stream comprises order entry data, and the set of operations are associated with a trading algorithm.” As described in Ground 1, Altera discloses the first serial data received by the deserializer and the second serial data transmitted from the FPGA system. *See* Claims 1(a) and 1(j) above. A POSA would have been motivated by Lockwood to use Altera in trading applications. In a trading setting, a POSA would understand that the purpose of the FPGA system would be to supply data about the market such that an algorithm implemented in the FPGA can determine what trades to make, and when. Ex. 1007, Lockwood at 1 (“Traders receiver multiple high-datarate UDP/IP market data streams over the network from trading venues.”), 2 (“HFT strategies therefore require low-latency market access both to receive market data as well as to transmit new orders.”). In that application, the first serial data stream would include market data—e.g., price and trade-related data for a financial instrument reported by a trading venue, such as bid, asks, trades, and auction notifications. Ex. 1001, ’381

patent at 23:13, 27:25-27. For example, Lockwood Figure 2 shows the first serial data as “Market Feed(s)” entering the FPGA. Ex. 1007, Lockwood at 3; *see also id.* at Figure 3.

210. The second serial data would include order entry data—e.g., data related to entering orders to purchase or sell stocks, commodities, goods, and/or services. Ex. 1001, ’381 patent at 26:67-27:2. Lockwood Figure 2 shows the second serial data as “[o]rders back to the market” exiting the FPGA system. Ex. 1007, Lockwood at 3; *see also id.* at 1 (“The final step is to enter these orders into the exchange”), Figure 3.

211. A POSA would understand that the set of operations used on the market data to generate the order entry data would be a trading algorithm—e.g., a set of rules implemented in the FPGA to execute trades in financial markets. Lockwood in Figure 2 shows that “Trading Algorithm(s)” are implemented in the FPGA computational circuitry. *Id.* at 3; *see also id.* at 1 (“Traders employ a range of algorithms and heuristics to decide when and what to trade. These approaches combine current market data with historical data and outputs from computational models.”), Figure 3.

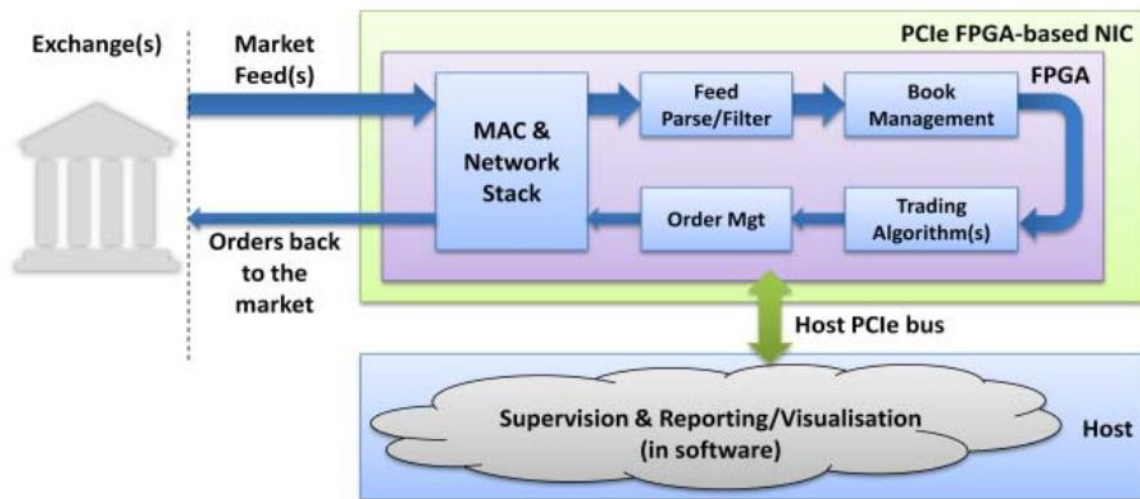


Figure 2 Typical Smart NIC in a supervised financial trading application

2. Claim 3

212. Claims 2 and 3 are very similar. Altera and Lockwood teach “[t]he method of claim 1, wherein the first serial data stream includes market data and the second serial data stream includes trading data.” For the reasons given for claim 2, in Altera and Lockwood “the first serial data stream includes market data.”

213. Claim 3 requires “trading data” instead of Claim 2’s “order entry data.” The second serial data would include trading data—e.g., data related to trading (e.g., purchasing or selling) stocks, commodities, goods, and/or services. Ex. 1001, ’381 patent at 26:67-27:2. Lockwood Figure 2 shows the second serial data as “[o]rders back to the market” exiting the FPGA system. Lockwood at 3; *see also id.* at 1 (“The final step is to enter these orders into the exchange”), Figure 3. See also a

configuration with three Ethernet ports in Figure 10, *id.* at 6. A POSA would understand that orders sent back to the exchange are a type of trading data.

D. Claim 11 and Claim 1 wherein: It Would Have Been Obvious to Modify Altera to Provide an External Feedback Clock into the Si5345 PLL

1. Claim 11

214. Based on Lockwood, a POSA would have been motivated to modify Altera such that “the transmitter side clock signal and the receiver side clock signal have the same frequency and phase.”

215. Here I expand on the discussion of the end of Claim 1. SyncE solutions require accurate “clock synchronization”—they consider both frequency (“ ± 4.6 ppm frequency accuracy”) and phase noise (“Limited phase wander”)—which the Altera solution is able to achieve. Ex. 1004, Altera at 2. In Altera the receiver side clock signal is generated when the receiver side “PHY IP cores recover the receiver (RX) clocks from their respective 1G/10 Gbps Ethernet ports.” *Id.* at 3. That receiver side clock is then transmitted to the Si5345 PLL. *Id.* The purpose of the Si5345 PLL is to “generate output clocks that are both *frequency* and *phase* locked to its selected input clock.” Ex. 1006, Si5345 Manual at 17 (emphasis added). “This stabilized clock is used to feed the FPGA’s serial transmitter (TX) PLL in sync with the master Ethernet port’s clock frequency to realize a complete synchronization loop.” Ex. 1004, Altera at 3.

216. Altera at 2 further shows a single clock from the fPLL (sys_tx_master_clk) that is used for both MAC and PCS functions. A POSA would understand that the fPLL shown is configured such that the resulting transmitter side clock signal frequency matches the receiver side clock signal frequency. However, certain system configuration choices impact the phases of the receiver-side clock signal (rx_recov_clk) and the transmitter side clock signal (sys_tx_master_clk) such that they would not automatically be phase-aligned, as further described in the discussion of Claim 12 below. A POSA would understand that aligning clock phases is a routine task that can be achieved in a variety of ways.

217. For example, a POSA would understand that the external Si5345 PLL supports External Feedback Mode, also described as “Zero Delay Mode.” Ex. 1006, Si5345 Manual at 44 (Figure 21). In “Zero Delay Mode,” rather than using the internal “feedback clock” discussed above with respect to claim 8 and shown in Figure 13 of the Si5345 Manual, the Si5345 PLL uses an alternate path for the clock in the feedback loop, and illustrates it with a thick gray line in Figure 21. As a result of this system configuration “the phase difference between the output, which is connected to the selected input will be nulled to zero” with respect to the other input. *Id.* at 20. This has the effect of zeroing the phase of the transmitter side clock with respect to the phase of the receiver side clock at the phase detector (PD).

Si5345-44-42-RM

7. Zero Delay Mode

A zero delay mode is available for applications that require fixed and consistent minimum delay between the selected input and outputs. The zero delay mode is configured by opening the internal feedback loop through software configuration and closing the loop externally as shown in Figure 21. This helps to cancel out the internal delay introduced by the dividers, the crosspoint, the input, and the output drivers. Any one of the outputs can be fed back to the FB_IN pins, although using the output driver that achieves the shortest trace length will help to minimize the input-to-output delay. The OUT9 and FB_IN pins are recommended for the external feedback connection in the Si5345. OUT3 and FB_IN pins are recommended for the external feedback connection in the Si5344. OUT1 or OUT2 are recommended with FB_IN in the Si5342. The FB_IN input pins must be terminated and ac-coupled when zero delay mode is used. A differential external feedback path connection is necessary for best performance.

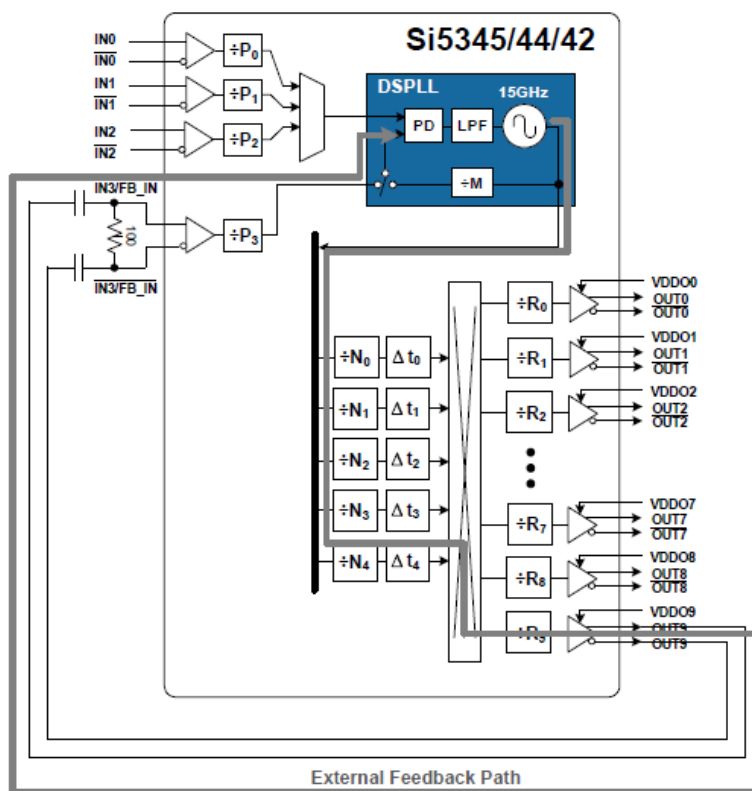
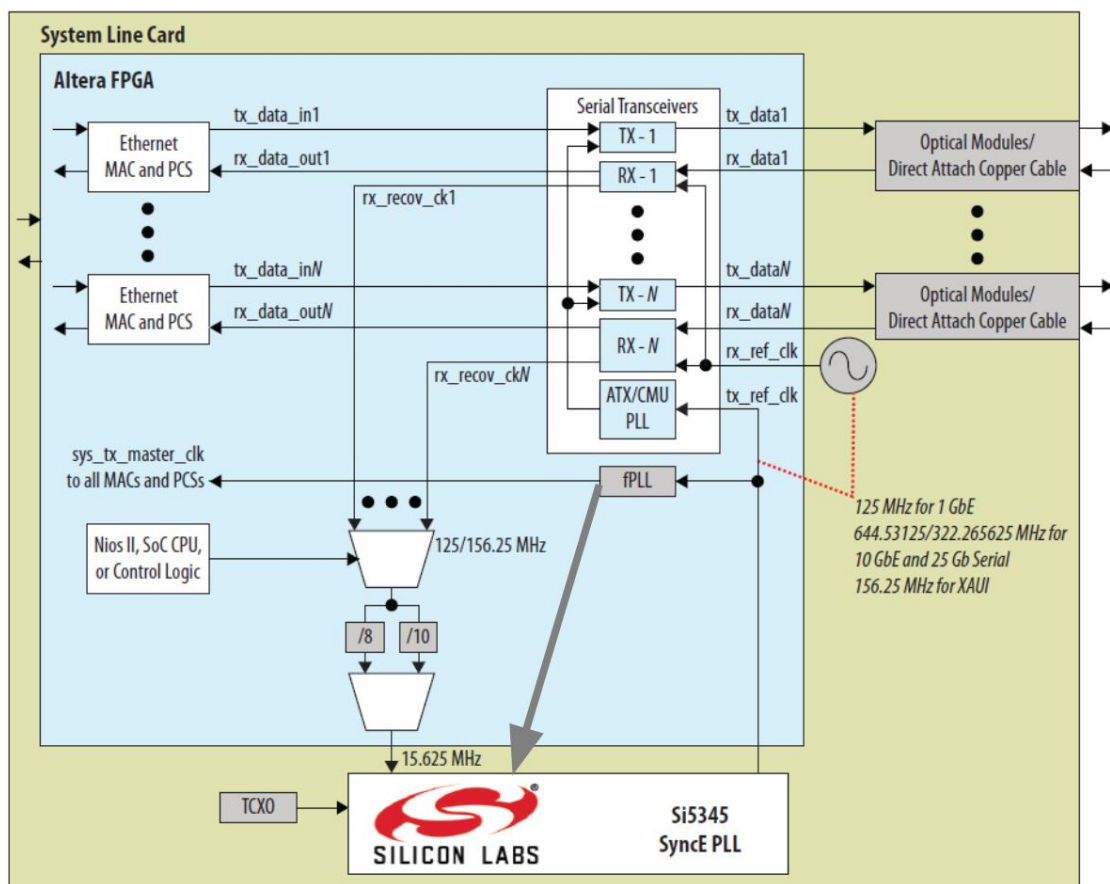


Figure 21. Si5345 Zero Delay Mode Set-up

218. A POSA would be motivated to phase align the receiver side and transmitter side clock signals for the purpose of reducing latency in the system. A POSA would understand that reducing latency can increase profitability. *See, e.g.,* Ex. 1007, Lockwood at 2 (“reduction in latency can improve arbitrage profitability”); *see also* Ex. 1014, Sahoo (“HFT is all about speed and minimizing

latency: the faster you can run trading strategies and algorithms for analyzing minute price changes and executing trade orders, the higher the probability to win over competition.”). A POSA would employ the above feature of the external Si5345 PLL (i.e. the “Optional External Feedback” input) to compensate for signal delays that could otherwise cause the phases of these clock signals to be different. This use of the alternate feedback path of the external PLL is illustrated with a gray line in Altera at 2.

Figure 1. High-Level Example of a SyncE System Line Card Using Altera FPGA and Silicon Labs SyncE PLL



219. A POSA would have a reasonable expectation that this minor modification to Altera would be successful. The modification utilizes capabilities already present in the Stratix FPGA and the Si5345 PLL and puts the Zero Delay Mode to precisely the use it is intended for. Ex. 1006, Si5345 Manual at 44 (“A zero delay mode is available for applications that require fixed and consistent minimum delay between the selected input and outputs.”).

220. A POSA would understand that the aforementioned PLL and Stratix V FPGA capabilities, and the documentation-recommended use of these capabilities, cause the receiver side and transmitter side clock signals to have the same frequency and phase.

2. Claim 1 wherein clause

221. To the extent that Altera does not anticipate the limitation in claim 1’s wherein clause, a POSA would understand that the modification described above with respect to claim 11 ensures that the method of claim 1 “does not use clock domain crossing operations that delay processing of the first set of parallel data streams.”

222. A POSA would have made this change to Altera for the same reasons discussed above with respect to claim 11. Specifically, a POSA would have been motivated by Lockwood to eliminate as much latency from the system as possible. Implementing the gray line in Altera to transmit the clock signal from the fPLL to

the external feedback input of the Si5345 PLL (IN3/FB_IN) eliminates the need for clock domain crossing operations in the path from receiving the first serial data to transmitting the second serial data. Notably, this exceeds the requirements of the claim 1 wherein clause, which does not require eliminating all CDC operations but only those “that delay processing of the first set of parallel data streams.”

223. As discussed above with respect to claim 11, a POSA would have had a reasonable expectation in the success of this modification, which utilizes capabilities already present in the Stratix FPGA and Si5345 PLL and puts the Zero Delay Mode to precisely the use it is intended for. Ex. 1006, Si5345 Manual at 44.

224. In fact, a POSA would recognize that there are a number of additional approaches / modifications that would satisfy the “wherein clause,” for example:

- a. Move the CDC out of the computational logic by employing the PCS Transmitter Phase Compensation FIFO. “The transmitter phase compensation FIFO interfaces with the FPGA fabric. The transmitter phase compensation FIFO compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock.” Ex. 1005, Stratix Handbook at 64. A CDC-induced delay outside of the computational logic does not “delay processing” in the computational logic.
- b. Implement a circuit within the programmable logic that adjusts the phase of the PLL until metastability is eliminated, similar to how the “dynamic

phase aligner (DPA) block” of the FPGA adjusts a clock phase clock to match the phase of the serial data. “The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if it is required.” Ex. 1005, Stratix Handbook at 437.

225. c. Employ a clock domain crossing within the computational logic that does not delay processing. E.g., fix the phase delay between clocks at 180 degrees via one or more of the PLLs such that some (e.g. half of the) computations can be performed between clock domains (such that processing is not delayed) and metastability is eliminated.

E. Claim 12: “The method of claim 1, wherein the transmitter side clock signal and the receiver side clock signal have the same frequency and different phases.”

226. Altera discloses “[t]he method of claim 1, wherein the transmitter side clock signal and the receiver side clock signal have the same frequency and different phases.”

227. SyncE solutions require accurate “clock synchronization”—they consider both frequency (“ ± 4.6 ppm frequency accuracy”) and phase noise (“Limited phase wander”)—which the Altera solution is able to achieve. Ex. 1004, Altera at 2. In Altera the receiver side clock signal is generated when the receiver side “PHY IP cores recover the receiver (RX) clocks from their respective 1G/10 Gbps Ethernet ports.” *Id.* at 3. That receiver side clock is then transmitted to the

Si5345 PLL. *Id.* The purpose of the Si5345 PLL is to “generate output clocks that are both *frequency* and *phase* locked to its selected input clock.” Ex. 1006, Si5345 Manual at 17 (emphasis added). “This stabilized clock is used to feed the FPGA’s serial transmitter (TX) PLL in sync with the master Ethernet port’s clock frequency to realize a complete synchronization loop.” Ex. 1004, Altera at 3. Altera further shows this stabilized clock feeding a Fraction PLL (fPLL), the output of which is used for MAC and PCS, labeling it also “sys_tx_master_clk.”

228. A phase offset accumulates between the output of the Si5345 and the output of the fPLL in Altera, due in part to the finite speed of light, and this phase difference would be unlikely to be an exact multiple of 360 degrees, and would result in receiver and transmitter side clocks that have the same frequency but different phases unless action was taken to circumvent it, e.g. as described previously.

229. A POSA would understand that the fPLL shown is configured such that the resulting transmitter side clock signal frequency matches the receiver side clock signal frequency through selection of the appropriate coefficients that, e.g., compensate for the division by 8 or 10 performed on the selected rx_recov_clk clock. A POSA would further understand that, if the phases of the receiver-side clock signal (rx_recov_clk) and the transmitter side clock signal (sys_tx_master_clk) were, after analysis, found to be the same—however unlikely that is—they are forced to be

different by programming a phase offset into the PLL such that desired non-zero phase offset is achieved.

230. For example, Altera includes an fPLL that generates the sys_tx_master_clk. That fPLL can be programmed to introduce a phase offset. Ex. 1005, Stratix Handbook at 358 (“The programmable phase shift feature allows the PLLs to generate output clocks with a fixed phase offset.”). Through phase adjustment in the fPLL, the transmitter and receiver side clocks have the same frequency and different phase.

231. “Fractional PLLs use divide counters and different voltage-controlled oscillator (VCO) taps to perform frequency synthesis *and phase shifts*. For example, you can reconfigure the counter settings and dynamically *phase-shift the fractional PLL output clock* in the PLLs of 28-nm devices. . . You can use these fractional PLL components to update the clock frequency, fractional PLL bandwidth, *and phase shift* in real time, without reconfiguring the entire FPGA.” Ex. 1017, *Implementing Fractional PLL Reconfiguration with Altera PLL and Altera PLL Reconfig Megafunctions*, Application Note AN-661-3.0, Altera Corporation, November 2013 at page 1 (emphasis added). Since the phase of the transmitter side clock is adjustable by the fPLL with respect to the receiver side clock, they can be forced to be different.

232. The Stratix Handbook explains this in detail, showing how the fPLL is programmed to add the desired phase offset. Ex. 1031, Stratix V Clocks and PLL, Table 1. Clock Networks and PLLs in Stratix V Devices, 2013, at 26.

4	Altera PLL IP Core Parameters - Clock Switchover Tab	UG-01087 2014.08.01
Parameter	Legal Value	Description
Desired Frequency ⁽¹⁾	—	Specifies the output clock frequency of the corresponding output clock port, <code>outclk[]</code> , in MHz. The default value is 100.0 MHz. The minimum and maximum values depend on the device used. The PLL only reads the numerals in the first six decimal places.
Actual Frequency	—	Specifies the actual value for the output clock frequency.
Phase Shift units	ps or degrees	Specifies the phase shift unit for the corresponding output clock port, <code>outclk[]</code> , in picoseconds (ps) or degrees.
Phase Shift	—	Specifies the requested value for the phase shift. The default value is 0 ps.
Actual Phase Shift	—	Specifies the actual value for the phase shift.

F. Claims 6-10

233. Altera and Lockwood would make obvious claims 6-10, which depend on claim 1, for the same reasons that Altera and Lockwood make claim 1 obvious as described above. *See also* Ground 1A (explaining how the additional limitations of claims 6-10 are taught by Altera).

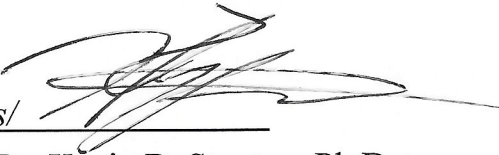
XII. CONCLUSION

234. In my opinion, claims 1-4 and 6-12 of the '381 patent are unpatentable for at least the reasons stated above.

235. I reserve the right to supplement my opinions in the future, to respond to any arguments raised by the owner of the '381 patent, and to take into account new information that becomes available to me.

236. I declare that all statements made herein are of my own knowledge, are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: November 24, 2025

s/ 
Dr. Kevin B. Stanton, Ph.D.