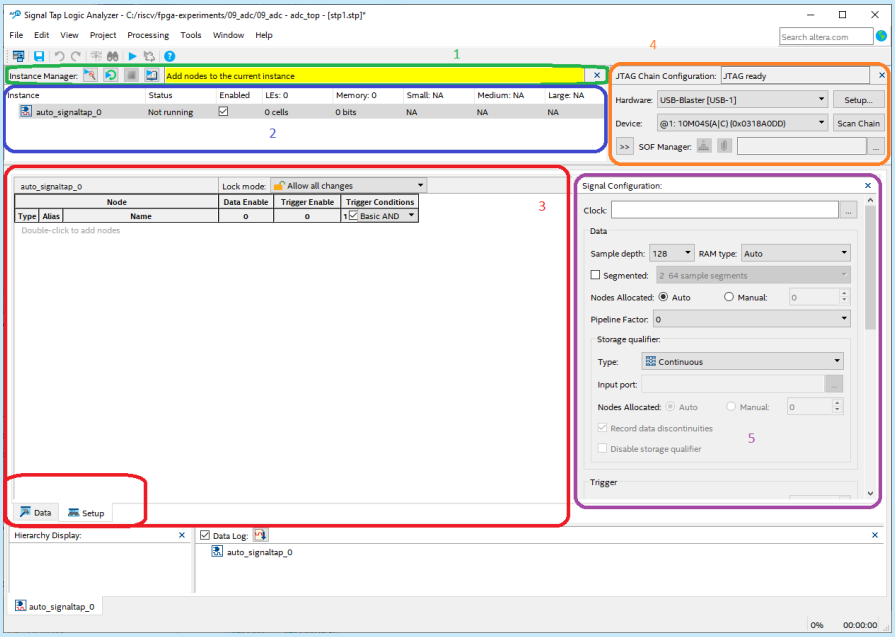
Eksperymenty z FPGA (9)

W tej części kursu przyjrzymy się przydatnej funkcji środowiska Quartus, czyli modułowi Signal Tap, pozwalającemu na podglądanie przebiegów zarejestrowanych wewnątrz układu FPGA. Dzięki niemu otrzymujemy wynik podobny do przebiegów z symulacji, ale wprost z „żywego" sprzętu.

FTP WSZYSTKIE POPRZEDNIE ODCINKI KURSU SĄ DOSTĘPNE NA STRONIE WWW.MEDIA.AVT.PL

Moduł odpowiedzialny za zbieranie i przechowywanie danych, zbudowany jest z tych samych zasobów co badana logika, przez co projekt wraz z dodatkowymi elementami powiększy się. Nie będzie to dokładnie nasz projekt, ponieważ po dołożeniu dodatkowych elementów, konieczna jest powtórna synteza, a co za tym idzie, rozłożenie logiki w strukturze układu FPGA zostanie wykonane ponownie. Może to skutkować zniknięciem pewnych amonali, które chcieliśmy wytropić, choć nie powinno się to często zdarzać. Pomimo niedogodności jest to przydatna metoda debugowania, pozwalająca przynajmniej częściowo zajrzeć do środka układu.



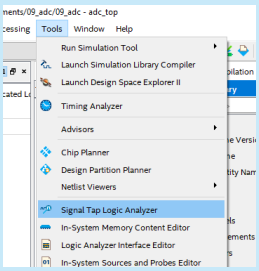
Rysunek 2. Główne okno *Signal Tap*

Signal Tap

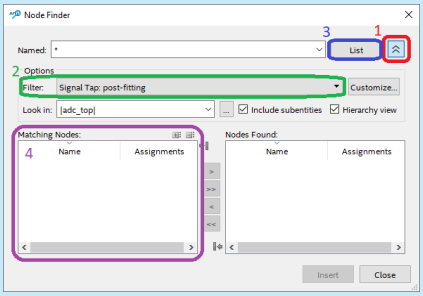
Do eksperymentów wykorzystamy projekt odczytu dwóch kanałów ADC, który stworzyliśmy w poprzedniej części kursu (można go pobrać z repozytorium [1]). Otwieramy środowisko Quartus i załadowujemy projekt *09_adc\09_adc.qpf*. Następnie wywołujemy jego budowę i czekamy, aż się zakończy. Warto zanotować poziom wykorzystania komponentów układu FPGA, aby po dodaniu modułu Signal Tap, porównać jak się zmienił.

Narzędzie włączamy wybierając opcję *Signal Tap Logic Analyzer* (**rysunek 1**), z menu *Tools*. Pojawi się główne okno programu, podobne do tego z **rysunku 2**. Można w nim wydzielić kilka grup opcji. Na samej górze znajduje się menu programu. Pasek oznaczony jako (1), pozwala na uruchomienie zbierania danych oraz prezentuje aktualny stan systemu. W tej chwili jego kolor jest żółty, co oznacza, że nie jest gotowy do pracy. Kolor czerwony sygnalizuje, że wprowadziliśmy duże zmiany, które wymagają powtórnego zbudowania projektu.

W sekcji (2) znajdziemy listę dostępnych instancji, czyli wersji naszego analizatora logicznego. Możemy przygotować kilka wariantów i aktywować ten, który akurat potrzebujemy. Na potrzeby projektu aktywujemy tylko jeden. Sekcja (3) to główne okno projektu składające się z dwóch kart, między którymi możemy się przełączać za pomocą przycisków, zlokalizowanych w lewym dolnym rogu. Na początku znajdujemy się w zakładce *Setup* (ustawienia). Na razie jest ona pusta, ale niebawem dodamy tu sygnały, które chcemy obserwować. W drugiej zakładce *Data* (dane) dostępne będą zebrane dane. Mniejszy panel (4) pozwala na obsługę interfejsu JTAG. Możemy tu wybrać programator, wykryć wersję układu FPGA oraz wgrać bitfile, dzięki czemu nie musimy przełączać się do okna programatora. Na końcu została nam zakładka (5) – *Signal Configuration* (konfiguracja sygnału), gdzie ustawiamy opcje takie jak zegar, który będzie taktował nasz moduł, oraz liczbę próbek jaką chcemy zapisać. Przyjrzyjmy się teraz krok po kroku co musimy skonfigurować, aby zebrać interesujące nas dane.



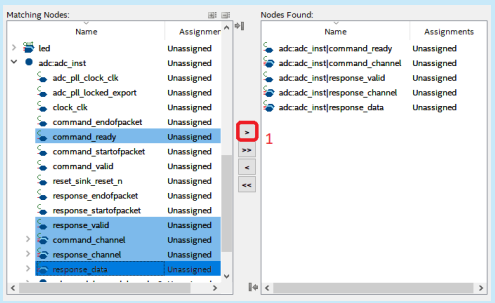
Rysunek 1. Włączamy program *Signal Tap*



Rysunek 3. Lista sygnałów, które możemy dodać do obserwacji

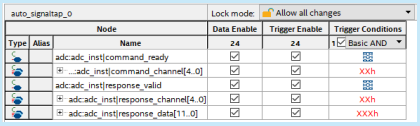
Konfiguracja sygnałów

Zacznijmy od dodania sygnałów, które chcemy obserwować. Klikamy na środku okna (3), obok napisu *Double-click to add nodes* (kliknij

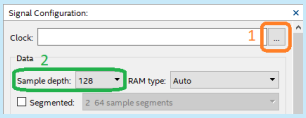


Rysunek 4. Wybieramy sygnały przypisane do instancji *adc_inst*

dwukrotnie aby dodać węzły). Otworzy się okno *Node Finder*, pokazane na **rysunku 3**. Najpierw naciskamy przycisk (1). Pojawią się dodatkowe opcje. Aby zobaczyć wszystkie dostępne sygnały z *Filter* (2) wybieramy *Design Entry* (all names). Następnie naciskamy *List* (3). Dostępne sygnały pojawiają się w oknie *Matching Nodes (4)*.



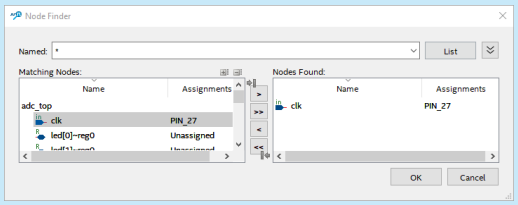
Rysunek 5. Lista sygnałów wybranych do obserwacji



Rysunek 6. Panel *Signal Configuration*

Na **rysunku 4** pokazano okno z już załadowaną listą połączeń. Teraz możemy wybrać sygnały, które nas interesują. Wybieramy wejściowy sygnał `command_channel`, który służy do wybrania kanału do przetworzenia, oraz `command_ready`, mówiący o gotowości przetwornika do kolejnego pomiaru. Decydujemy się także na trzy sygnały z interfejsu wyjściowego: `response_channel`, `response_data` oraz `response_valid`. Wybrane sygnały przenosimy do drugiej kolumny *Nodes Found* za pomocą przycisku (1). Na końcu zatwierdzamy wybór, naciskając znajdujący się w prawym dolnym rogu przycisk *Insert*, a następnie zamykamy okno klikając przycisk *Close*. Sygnały pojawiły się na głównym ekranie, co pokazuje **rysunek 5**.

Teraz przechodzimy do konfiguracji modułu, którą wykonamy w panelu *Signal Configuration* (po prawej stronie okna), pokazanego na **rysunku 6**. Zaczynamy od wyboru zegara. W linii *Clock* naciskamy przycisk (1). Znowu pojawi się okno *Node Finder*, pokazane na **rysunku 7**. Tutaj także naciskamy przycisk *List*, a następnie dodajemy linię `clk` z komponentu *adc_top*. Jest to nasze wejście zegarowe, gdzie dołączony jest zewnętrzny generator sygnału. Zatwierdzamy przyciskiem *OK*.

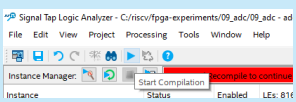


Rysunek 7. Wybieramy sygnał zegarowy w oknie *Node Finder*

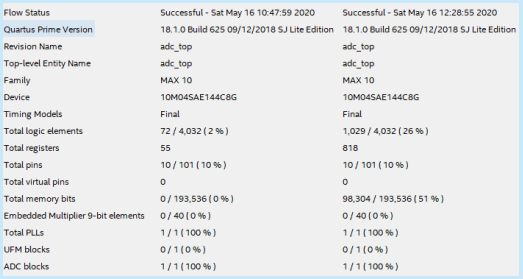
Wróćmy do rysunku 6 i punktu oznaczonego cyfrą (2). W zakładce *Data* (*dane*) w polu *Sample depth* (*liczba próbek*) konfigurujemy rozmiar pamięci, w której będą zapisywane wyniki pomiarów. Wybieramy 4 K, czyli 4096 tysiąca kolejnych wartości. Nie możemy wybrać dowolnie dużej wielkości, bo ogranicza nas rozmiar pamięci RAM dostępnej w układzie. Na przykład, jeśli spróbujemy wybrać 8 K, to podczas budowania projektu zostanie zwrócony błąd, spowodowany niewystarczającą liczbą zasobów.

Zbieramy dane

Podstawowa konfiguracja jest już gotowa. Teraz musimy powtórnie skompilować projekt, o czym informuje nas czerwony pasek na górze okna. Naciskamy niebieską strzałkę na pasku zadań (**rysunek 8**). Potwierdzamy, jeśli pojawi się pytane o dodanie do aktualnego projektu. Przed rozpoczęciem budowania, musimy też zapisać projekt naszego analizatora. Zostaniemy przeniesieni do środowiska Quartus. Kiedy budowa projektu się zakończy,



Rysunek 8. Uruchamiamy budowę projektu



Flow Status	Successful - Sat May 16 10:47:59 2020	Successful - Sat May 16 12:28:55 2020
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	adc_top	adc_top
Top-level Entity Name	adc_top	adc_top
Family	MAX 10	MAX 10
Device	10M04SAE144C8G	10M04SAE144C8G
Timing Models	Final	Final
Total logic elements	72 / 4,032 (2 %)	1,029 / 4,032 (26 %)
Total registers	55	818
Total pins	10 / 101 (10 %)	10 / 101 (10 %)
Total virtual pins	0	0
Total memory bits	0 / 193,536 (0 %)	98,304 / 193,536 (51 %)
Embedded Multiplier 9-bit elements	0 / 40 (0 %)	0 / 40 (0 %)
Total PLLs	1 / 1 (100 %)	1 / 1 (100 %)
UFM blocks	0 / 1 (0 %)	0 / 1 (0 %)
ADC blocks	1 / 1 (100 %)	1 / 1 (100 %)

Rysunek 9. Porównanie zużycia zasobów dla projektu bez oraz z modułem *SignalTap*

warto zwrócić uwagę na podsumowanie zużycia zasobów. Porównanie projektu bez i z *SignalTapem* pokazuje **rysunek 9**. Widzimy, że zużył on dość sporo komponentów, czyli około jednej czwartej zasobów logicznych oraz połowę dostępnej pamięci RAM.

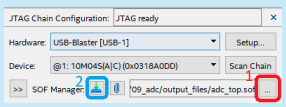
Wróćmy do ekranu *SignalTap*, a dokładniej do zlokalizowanej w prawym górnym rogu obsłudze programatora, którą znajdziemy na **rysunku 10**. Klikamy przycisk (1) i ładujemy plik ze zbudowanym projektem *output_files\adc_top.sof*. Następnie programujemy układ FPGA, klikając (2).

Kiedy proces zakończy się powodzeniem (co może zająć trochę czasu), pojawi się napis *Ready to acquire*. Zbieranie danych rozpoczynamy wyborem naszej instancji z widocznej na **rysunku 11** listy (1) oraz kliknięciem przycisku *Run Analysys* (2). Po chwili zostaniemy przełączeni z zakładki *Setup* do zakładki *Data*, widocznej na **rysunku 12**, gdzie zobaczymy uzyskane przebiegi. Za pomocą lewego i prawego przycisku myszy możemy powiększać lub pomniejszać wykres.

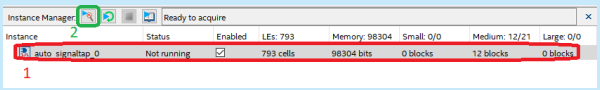
Aby zmienić sposób interpretowania danych, klikamy prawym przyciskiem myszy na nazwę sygnału. Z menu kontekstowego wybieramy ostatnią pozycję – *Bus Display Format* (format wyświetlania szyny), co pokazuje **rysunek 13**. Przykładowo, dla sygnału `response_data` wybiorę opcję *Unsigned Decimal*, czyli wartości dziesiętne bez znaku.

Możemy także dodać kursor, który pomoże nam zmierzyć czas pomiędzy zdarzeniami. W tym celu naciskamy na napis *click to insert time bar* (naciśnij aby dodać kursor czasowy) na belce nad przebiegiem – na rysunku 12 została ona oznaczona numerem (1). Pojawi się dodatkowe okienko, w którym widzimy zbliżenie na wybrany punkt w czasie – na **rysunku 14** zaznaczone jako (1). Kursor możemy przesuwać myszką, albo w sposób precyzyjny za pomocą strzałek (2). Sam kursor znajdziemy pod numerem (3).

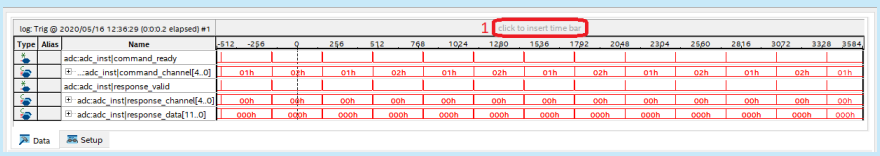
Kiedy naciśniemy po raz kolejny na górny pasek, dodany zostanie następny kursor, którego wartość podana będzie względem pierwszego (głównego) kursora. Na rysunku 14 dodane zostały dwa takie kursory – (4) i (5), umiejscowione w momentach, gdy pojawia się kolejna dana na wyjściu. Widzimy, że pomiędzy dwoma kolejnymi odczytami mija 320 cykli zegara, a ponownie obsłużenie kanału



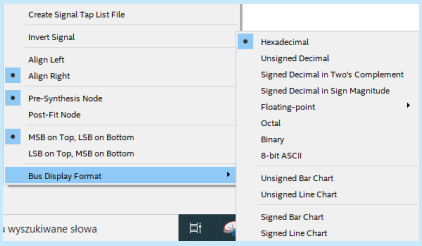
Rysunek 10. Obsługa programatora



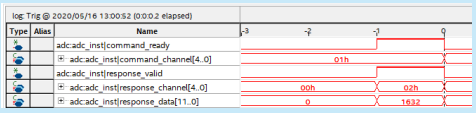
Rysunek 11. Uruchamiamy zbieranie danych



Rysunek 12. Wykres przedstawiający zebrane przebiegi



Rysunek 13. Zmiana formatu wyświetlania danych

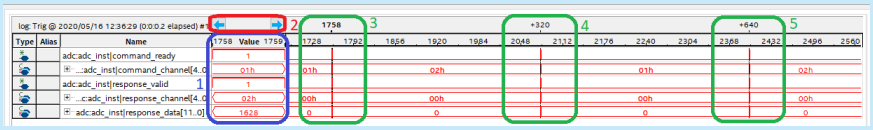


Rysunek 18. Próbka zerowa zgadza się z ustawieniem triggera

dane ciągle zbierane są w buforze cyklicznym, natomiast sam trigger decyduje tylko o momencie, kiedy zapis zostanie zatrzymany.

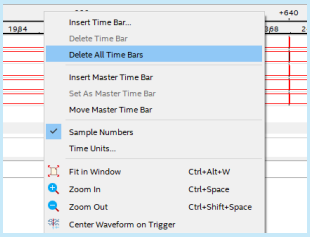
Zbieranie warunkowe

W tej chwili interesujące dane stanowią niewielki ułamek zawartości bufora, gdzie wartości pomiarów to tylko jedna próbka na 320 zapisanych. Możemy to zmienić, korzystając z opcji *Storage qualifier* z zakładki *Signal Configuration*. Domyślna opcja to *Continuous*, czyli „zapisuj wszystko”, przejdźmy jednak na opcję *Conditional* – warunkowo (**rysunek 20**). Kiedy wybierzemy opcję w oknie z sygnałami, pojawi się nowa kolumna – *Storage Qualifier*, co pokazuje **rysunkek 21**.



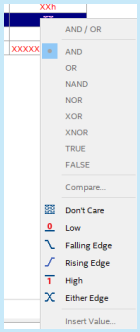
Rysunek 14. Dodanie kursora do przebiegów

pierwszego następuje po 620 cyklach. Ponieważ częstotliwość zegara to 8 MHz, odpowiada to kolejno 40 µs i 80 µs. Widzimy więc, że ADC dokonuje pomiaru z częstotliwością 25 kHz, czyli zgodnie z tym, co ustawiliśmy w konfiguracji. Aby usunąć wskaźniki, klikamy na pasek prawym przyciskiem myszy, a z menu kontekstowego wybieramy *Delete All Time Bars* (usuń wszystkie kursory czasu), jak na **rysunku 15**.



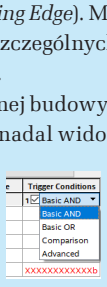
Rysunek 15. Usuwanie kursorów

Uruchomiliśmy narzędzie w trybie wyzwalanym przez nas. Ponieważ wybraliśmy bardzo duży rozmiar bufora, a interesujące nas zdarzenia (odczyt z ADC) występują często i w stałych odstępach czasu, to i tak złapaliśmy dość dużą ich liczbę. Jednak w przypadku zdarzeń występujących rzadziej, możemy posłużyć się wyzwalaniem (ang. *trigger*). Wróćmy do zakładki setup i w kolumnie trigger conditions (warunki wyzwalania) kliknijmy prawym przyciskiem myszy w wierszu `response_valid`. Pojawi się zaprezentowane na **rysunku 16** menu kontekstowe, gdzie możemy wybrać kiedy ma nastąpić wyzwolenie pomiaru. Domyślna opcja to *Don't Care* (nie przejmuj się). Możemy też wybrać reakcję na wartość, albo na zbocze. Ja wybiorę wyzwalanie na zbocze narastające (*Rising Edge*). Można także określić, jak będą traktowane kryteria dla poszczególnych sygnałów. Listę dostępnych opcji pokazuje **rysunek 17**.

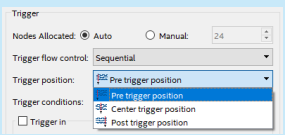


Rysunek 16. Ustawienia wyzwalania

Zmiana sposobu wyzwalania nie wymaga ponownej budowy projektu. Wiemy o tym, ponieważ w górnej części okna nadal widoczna jest informacja *Ready to acquire*. Wracamy więc do zakładki *Data* i wyzwalamy powtórnie pomiar przyciskiem *Run Analysis*. Kiedy dane zostaną załadowane, zobaczymy że zerowa próbka pokrywa się ze skonfigurowanym przez nas triggerem – zostało to pokazane na **rysunku 18**. W panelu *Signal Configuration* mamy możliwość dokonania bardziej zaawansowanej konfiguracji wyzwalania, gdzie między innymi możemy wybrać, czy chcemy aby moment wyzwolenia był na początku, w środku, czy na końcu danych (**rysunek 19**). Mamy taką możliwość, ponieważ

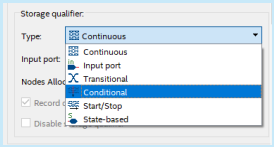


Rysunek 17. Konfiguracja wzajemnej zależności pomiędzy rożnymi wymuszeniami



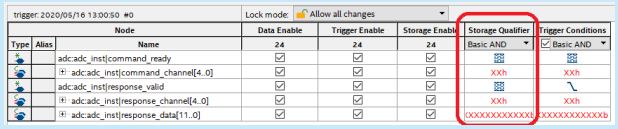
Rysunek 19. Konfiguracja momentu wyzwalania względem zbieranych danych

Ponieważ dokonujemy dużej modyfikacji, konieczna jest powtórna budowa projektu. Informuje nas o tym czerwony komunikat wyświetlony w górnej części okna (**rysunek 23**). Zgodnie z sugestią powtórnie uruchamiamy kompilację, a po jej zakończeniu ponownie programujemy układ FPGA. Kiedy będziemy gotowi, możemy znów rozpocząć zbieranie danych, a po jego zakończeniu, zobaczymy rezultat podobny jak na **rysunku 24**.

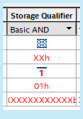


Rysunek 20. Dodajemy warunek, który musi zostać spełniony, aby dane zostały zapisane

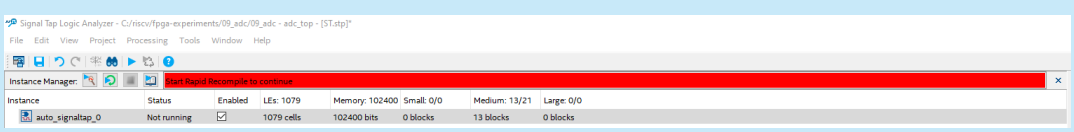
Zgodnie z oczekiwaniem, wartości w pierwszych czterech liniach nie ulegają zmianie, ponieważ są one ściśle związane z ustawionym przez nas warunkiem. Najciekawszy jest ostatni wiersz, zawierający wyniki pomiarów. Widzimy, że wartości zmieniają się bardzo często. Aby wynik był czytelniejszy, możemy zmienić tryb wyświetlania na graficzny. W tym celu, tak jak przy zmianie sposobu interpretacji danych, klikamy nazwę sygnału prawym przyciskiem myszy. Z menu kontekstowego, przedstawionego



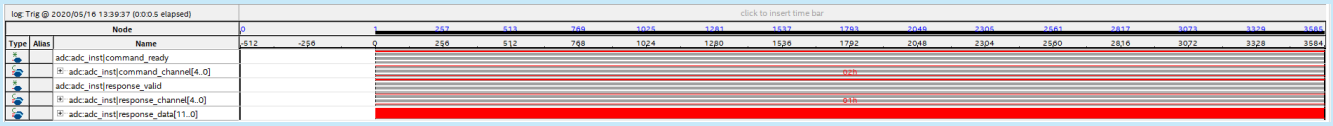
Rysunek 21. Nowa kolumna *Storage Qualifier*, która pozwala dodać warunek zapisu danych



Rysunek 22. Ustawiamy aby zapisane zostały jedynie poprawne próbki, zebrane z kanału 1

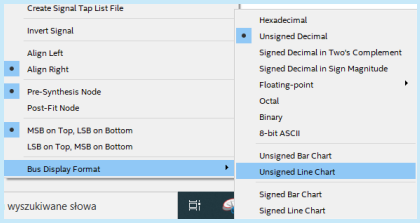


Rysunek 23. Informacja o konieczności przeprowadzenia ponownej budowy projektu



Rysunek 24. Dane zebrane przez *Signal Tap*

na **rysunku 25**, wybieramy opcję *Bus Display Format*, a następnie *Unsigned Line Chart* (wykres liniowy dla liczb bez znaku).



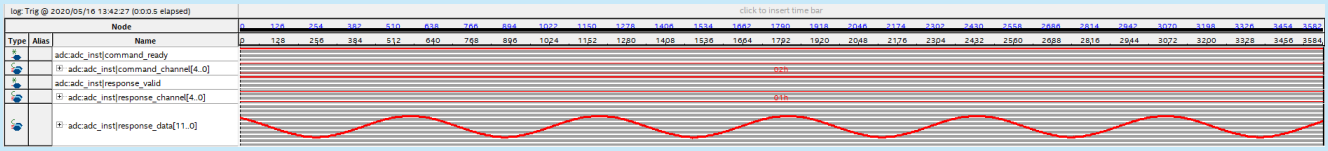
Rysunek 25. Wybór trybu graficznego

Rezultat pokazuje **rysunek 26**. Widzimy przebieg funkcji, ponieważ tym razem do wejścia, zamiast potencjometru, podłączyłem generator funkcji, ustawiony na sygnał sinusoidalny o częstotliwości 20 Hz, wartości międzyszczytowej 1 V i z offsetem 1,5 V. Jeśli przełączymy się na wykres słupkowy (*Unsigned Bar Chart*), to zobaczymy wynik podobny jak na **rysunku 27**. Tym razem, aby różnice między kolejnymi punktami były większe, sygnał wejściowy ma częstotliwość równą 1 kHz.

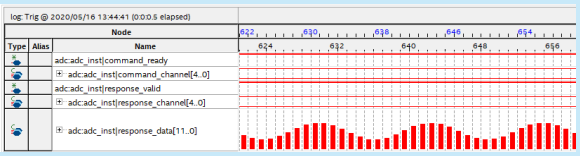
Podsumowanie

W tym odcinku zaznajomiliśmy się z bardzo przydatnym narzędziem podczas debugowania. *SignalTap* pozwala analizować przebiegi z wnętrza naszego układu FPGA. Opisane w artykule funkcje zostały także pokazane na filmie [2]. W kolejnej części kursu wrócimy do przetwornika ADC i zajmiemy się przetwarzaniem dźwięku.

Rafał Kozik
rafkozik@gmail.com



Rysunek 26. Dane z przetwornika ADC zaprezentowane w trybie graficznym



Rysunek 27. Wykres słupkowy (*Unsigned Bar Chart*)

Przypisy:
[1] ***https://bit.ly/30Bg2FI*** – repozytorium z przykładami
[2] ***https://bit.ly/2WHDjER*** – film demonstrujący konfigurację *Signal Tap*

Poprzednie części kursu i dodatkowe materiały dostępne są na stronie
www.media.avt.pl