AGH Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca magisterska

Michał Oczkowicz

kierunek studiów: **informatyka stosowana**

specjalność: **informatyka w nauce i technice**

Analiza działania szybkiego 10-bitowego przetwornika ADC zaprojektowanego w submikronowej technologii VLSI.

Opiekun: **dr hab. inż. Marek Idzik**

Kraków, grudzień 2010

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....
(czytelny podpis)

Kraków, grudzień 2010

Tematyka pracy magisterskiej i praktyki dyplomowej
Michała Oczkowicza, studenta V roku studiów kierunku informatyka stosowana, specjalności informatyka w nauce i technice

Temat pracy magisterskiej: **Analiza działania szybkiego 10-bitowego przetwornika ADC zaprojektowanego w submikronowej technologii VLSI.**

Opiekun pracy:	dr hab. inż. Marek Idzik
Recenzenci pracy:	dr hab. inż. .....
Miejsce praktyki dyplomowej:	WFiIS AGH, Kraków

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
   — zapoznanie się z ideą działania analogowo-cyfrowego przetwornika potokowego,
   — przygotowanie oprogramowania do symulacji projektowanego przetwornika,
   — dyskusja i analiza wyników symulacji,
   — sporządzenie sprawozdania z praktyki.
4. Kontynuacja prac związanych z tematem pracy magisterskiej.
5. Zebranie i opracowanie wyników symulacji.
6. Analiza wyników symulacji, ich omówienie i zatwierdzenie przez opiekuna.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie: grudzień 2010

....
(podpis kierownika katedry) (podpis opiekuna)

Merytoryczna ocena pracy przez opiekuna:

Końcowa ocena pracy przez opiekuna: .....

Data: ..... Podpis: .....

Merytoryczna ocena pracy przez recenzenta:

Końcowa ocena pracy przez recenzenta: .....

Data: ..... Podpis: .....

Spis treści

Wstęp

Projektowanie prototypowych układów scalonych o bardzo dużej skali integracji (ang. Very Large Scale Integration - VLSI), wiąże się bardzo często z dużym nakładem pracy i bardzo dużą liczbą testów koniecznych do przeprowadzenia. W Katedrze Oddziaływań i Detekcji Cząstek AGH od kilku lat trwają prace nad prototypowymi układami elektroniki odczytu do detektora świetlności, czyli detektora LumiCal (ang. Luminosity Calorimeter), dla liniowego zderzacza elektronów i pozytonów ILC (ang. International Linear Collider). Jednym z kluczowych elementów projektowanego toru elektroniki odczytu jest wielokanałowy przetwornik analogowo-cyfrowy (ang. Analog to Digital Converter – ADC), który umożliwiał będzie odpowiednio szybką i dokładną konwersję przetworzonych sygnałów z sensorów na dane cyfrowe.

Niniejsza praca poświęcona jest testom gotowych narzędzi oraz opisowi narzędzi stworzonych przez autora, służących do przyspieszenia i uproszczenia żmudnego procesu weryfikacji symulacyjnej projektów układów scalonych, w oparciu o projekt wspomnianego przetwornika analogowo-cyfrowego.
Praca podzielona została na pięć rozdziałów.

Pierwszy rozdział zawiera wprowadzenie teoretyczne na temat przetworników analogowo-cyfrowych. Przedstawione w nim zostały parametry jakimi opisujemy pracę przetworników oraz rodzaje i opisy działania różnych typów przetworników. Dokładniejszy opis poświęcony został przetwornikowi o architekturze potokowej, ze względu na fakt badania i testowania właśnie tej architektury w kolejnych rozdziałach.

W rozdziale drugim opisany został stworzony przez autora skrypt symulujący działanie idealnego potokowego przetwornika analogowo-cyfrowego oraz możliwości jakie udostępnia. W kolejnych podrozdziałach umieszczono również szereg analiz przeprowadzonych przy użyciu wspomnianego skryptu, dzięki którym możemy łatwo zbadać wpływ różnych parametrów na prace przetwornika. Parametry, których

wpływ został omówiony to między innymi amplituda sygnału wejściowego, temperatura, offset komparatorów, wielkość pojemności, wzmocnienie wzmacniacza.

Rozdział trzeci poświęcony został zbadaniu pracy symulatorów, używanych do testowania projektów układów scalonych. Pod uwagę wzięte zostały cztery symulatory firm *Cadence Design System* i *Synopsys*. W rozdziale zawarty został opis i porównanie dokładności wybranych symulatorów, jak również szybkości ich działania.

Czwarty rozdział zawiera opis, możliwości i przykłady działania skryptu stworzonego przez autora, automatyzującego proces uruchamiania symulacji i analizowania ich wyników .

W rozdziale piątym zebrane zostały dane na temat pierwszych prób uruchomienia symulacji projektowanego w Zespole Elektroniki Jądrowej układu 8 kanałowego przetwornika analogowo-cyfrowego. Zebrane dane zostały wzbogacone o kilka analiz sprawdzających poprawność i jakość wyników otrzymywanych z symulacji.

Praca kończy się podsumowaniem prac wykonanych przez autora w czasie jej tworzenia i końcowymi wnioskami.

Rozdział 1

Przetworniki ADC

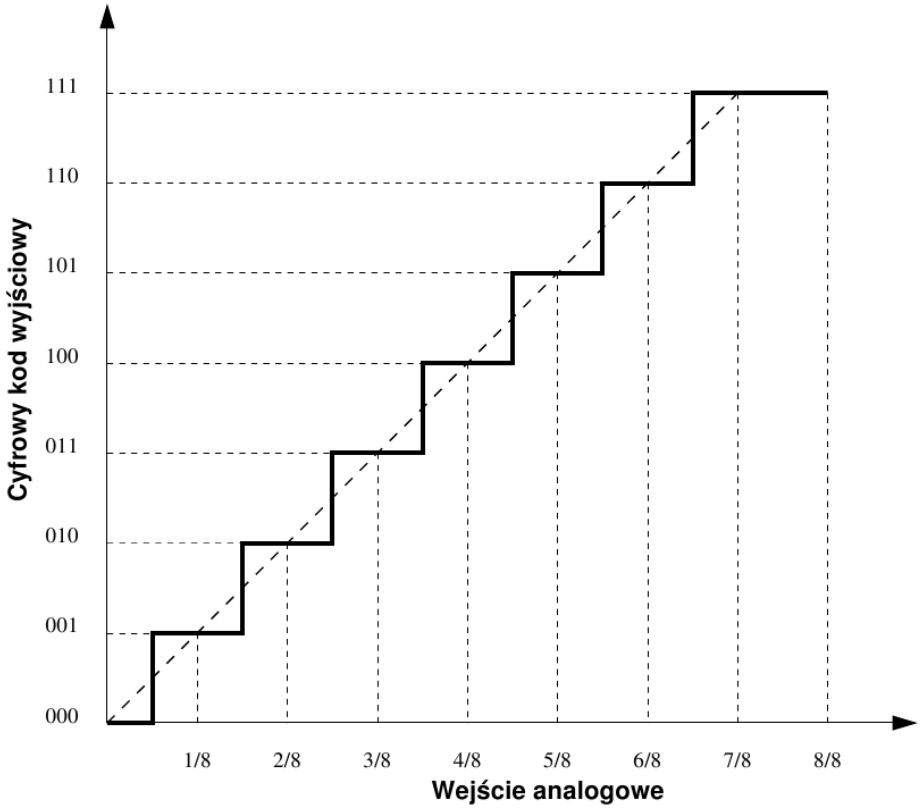
W rozdziale tym omówione zostały przetworniki analogowo-cyfrowe, ich podstawowe parametry statyczne i dynamiczne, jak również najbardziej znane rodzaje. Dokładniejszemu opisowi poddany został jeden rodzaj przetworników, a mianowicie przetworniki o architekturze potokowej, ze względu na fakt, iż dalsza część pracy opiera się głównie o przetworniki tego typu.

1.1. Parametry

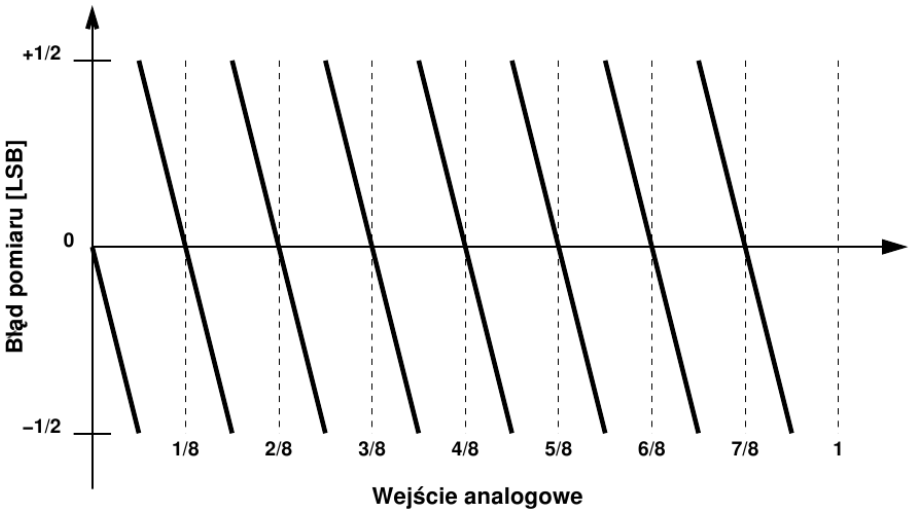
Podstawowym parametrem, który opisuje przetwornik analogowo-cyfrowy, jest jego rozdzielczość. Parametr ten mówi nam o tym ile poziomów kwantyzacji posiada nasz przetwornik, jest on zwykle podawany w bitach. Omawiany w tej pracy przetwornik jest 10-bitowy co oznacza, że posiada on $2^{10} = 1024$ dyskretnych wartości jakie może wytworzyć.

Zależność kodów wyjściowych przetwornika od napięcia podawanego na jego wejście nazywamy funkcją przenoszenia takiego przetwornika. Przykład funkcji przenoszenia dla idealnego 3-bitowego ADC został przedstawiony na rysunku 1.1. Jak możemy zauważyć, każdy kod wyjściowy przetwornika odpowiada jakiemuś zakresowi napięć wejściowych, który zmniejsza się wraz ze wzrostem rozdzielczości przetwornika. Zakres ten to błąd zamiany wartości analogowej na cyfrową związany z dyskretyzacją wartości analogowej. Błąd ten nazywa się błędem kwantyzacji (rys. 1.2) i posiada go każde, nawet idealne ADC [1].

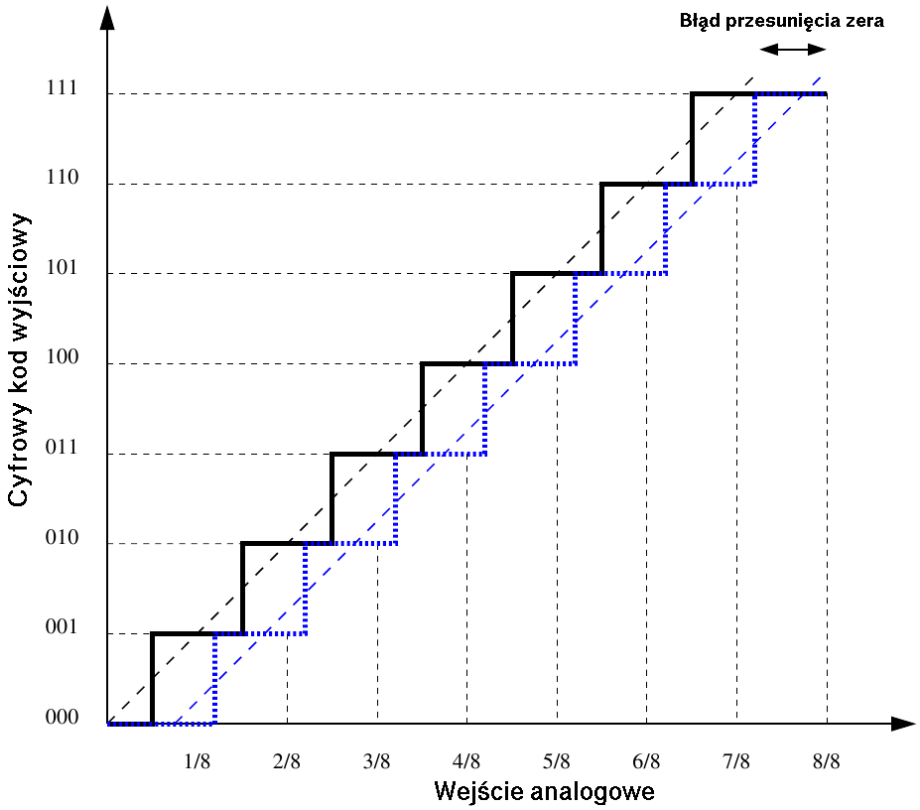
Ponadto każdy przetwornik możemy opisać przy pomocy wielu innych parametrów, które opisują jakość pracy takiego układu zarówno z sygnałami wolno zmiennymi (tzw. parametry statyczne), jak i sygnałami szybko zmiennymi (tzw. parametry dynamiczne). Poniżej omówione zostały najważniejsze z tych parametrów [2].



Rysunek 1.1. Funkcja przenoszenia idealnego 3-bitowego ADC.



Rysunek 1.2. Błąd kwantyzacji idealnego 3-bitowego ADC.



Rysunek 1.3. Błąd przesunięcia zera.

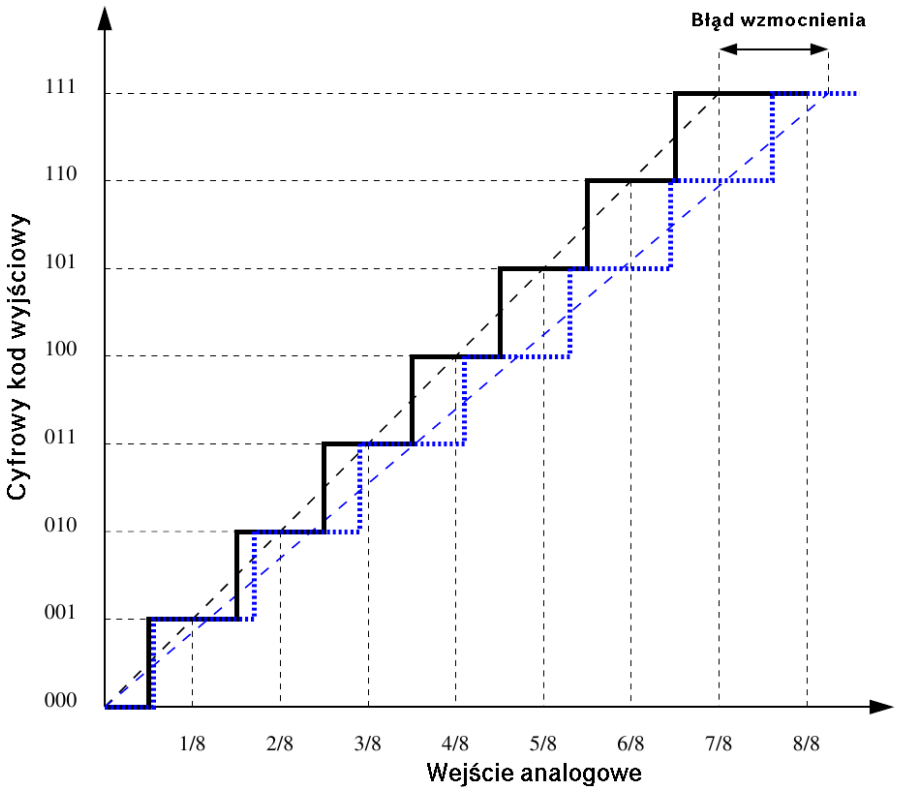
Błąd przesunięcia zera

Jest to przesunięcie funkcji przenoszenia przetwornika w stosunku do charakterystyki idealnej (rys. 1.3). Wartość ta mierzona jest jako przesunięcie w prawo lub lewo w stosunku do idealnej charakterystyki i podawana jest najczęściej, jak większość parametrów w stosunku do wartości napięcia najmniej znaczącego bitu (LSB) - równanie 1.1, np. $+\frac{2}{3}LSB$ [1].

$$LSB = \frac{V_{ref}}{2^M} \tag{1.1}$$

Błąd wzmocnienia

Błąd wzmocnienia to zmiana nachylenia funkcji przenoszenia w porównaniu z charakterystyką idealną (rys. 1.4). Wyznacza się ten parametr jako odchylenie rzeczywistej wartości napięcia wejściowego odpowiadającego największej wartości kodu wyjściowego od wartości idealnej, podczas gdy błąd przesunięcia zera wynosi zero [1].



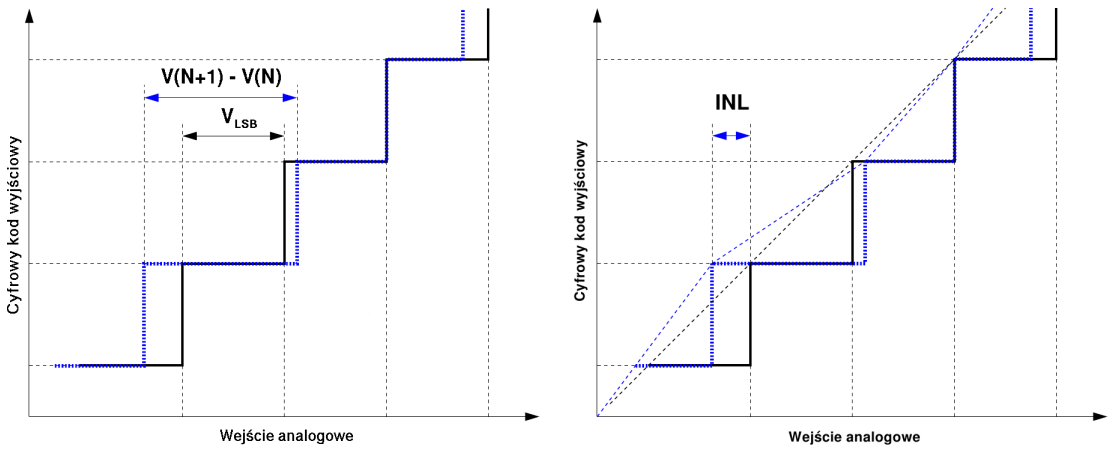
Rysunek 1.4. Błąd wzmocnienia.

Nieliniowość

W idealnym ADC każdy ze schodków funkcji przenoszenia ma taką samą szerokość, np. jak możemy zauważyć na rys. 1.1, każdy schodek reprezentuje dokładnie $\frac{1}{8}$ pełnej skali napięcia. Różnice pomiędzy idealną szerokością schodków a ich rzeczywistą szerokością nazywamy nieliniowością różniczkową (ang. Differential Non-Linearity – DNL) i obliczamy ją dla każdego z N schodków przy pomocy następującego równania:

$$DNL(N) = \frac{V(N+1) - V(N) - V_{LSB}}{V_{LSB}}, \tag{1.2}$$

gdzie $V(N+1)$ i $V(N)$ są progami przełączania kolejnych schodków, natomiast V_{LSB} oblicza się zgodnie z 1.1. Na rysunku 1.5(a) zobrazowany został sposób wyznaczania nieliniowości różniczkowej z funkcji przenoszenia przetwornika. Nieliniowość całkowa (ang. Integral Non-Linearity – INL) natomiast jest to odległość każdego ze schodków od prostej łączącej początek pierwszego i ostatniego schodka idealnej funkcji przenoszenia (rys. 1.5(b)). Istnieje zależność pomiędzy nieliniowością różniczkową



(a) różniczkowa (b) całkowa

Rysunek 1.5. Nieliniowości przetwornika analogowo-cyfrowego.

a całkową, a mianowicie nieliniowość całkową możemy wyznaczyć z równania:

$$INL(N) = \sum_{i=0}^{N} DNL(i). \tag{1.3}$$

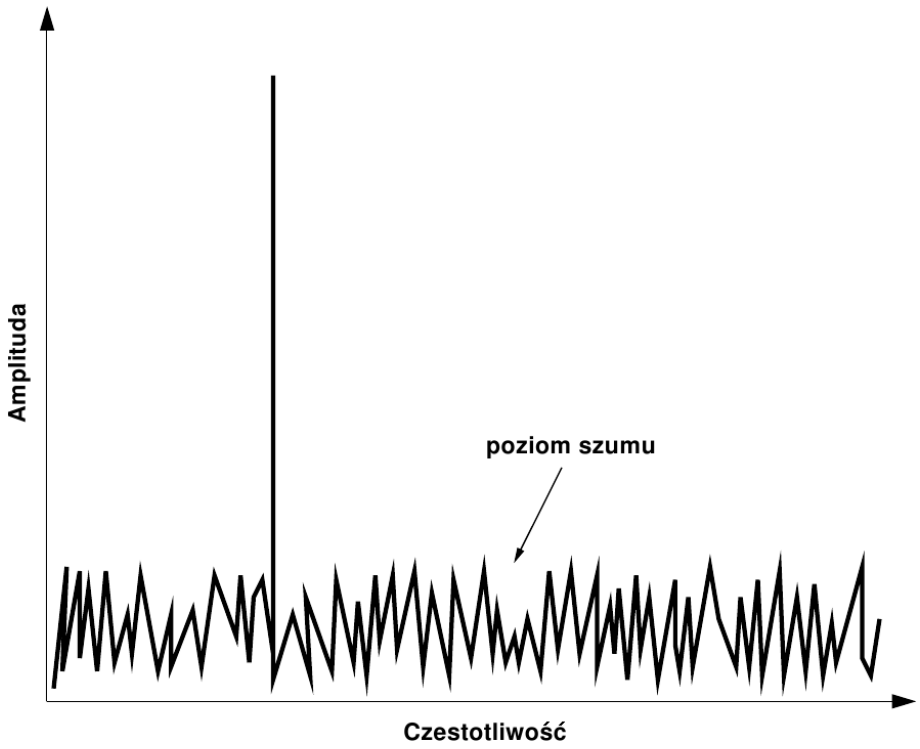
Omówione powyżej parametry opisywały jakość pracy ADC z sygnałami wolno zmiennymi, w dalszej części omówione zostały najważniejsze parametry dynamiczne jakie mają znaczenie przy opisie jakości pracy przetwornika [2, 1].

Do analizy pracy przetwornika z sygnałami szybkozmiennymi wykorzystuje się analizę częstotliwościową kodów wyjściowych przetwornika, na wejście którego podajemy sygnał sinusoidalny o odpowiedniej częstotliwości. Dokonywana ona jest przy pomocy szybkiej transformaty Fouriera (ang. Fast Fourier Transform – FFT). Na rysunku 1.6 zobaczyć możemy przykładowy wykres analizy częstotliwościowej przetwornika. Najwyższy prążek na wykresie to częstotliwość sygnału wejściowego badanego ADC, natomiast wszystkie pozostałe prążki to szum.

Na podstawie analizy FFT możemy wyznaczyć kilka ważnych parametrów którymi opisujemy jakość badanego przetwornika, parametry te zostały omówione poniżej [2].

Całkowite zniekształcenia harmoniczne

Całkowite zniekształcenia harmoniczne (ang. Total Harmonic Distortion – THD) to parametr odpowiadający stosunkowi zniekształceń wprowadzonych przez harmoniczne, które pojawiają się na skutek nieliniowości występujących na drodze sygnału analogowego w przetworniku ADC, do tego sygnału. Parametr ten określamy wzorem:



Rysunek 1.6. Przykład szybkiej transformaty Fouriera na kodach wyjściowych ADC.

$$THD = 20\log_{10}\left(\sqrt{\frac{\sum_{n=2}^{N_H+1} X_{avr}(f_{h[n]})^2}{X_{avr}\left(f_{sig}\right)^2}}\right), \tag{1.4}$$

gdzie $f_{h[n]}$ to częstotliwości wyznaczane jako kolejne całkowite wielokrotności częstotliwości f_{sig} zrzutowane na przedział częstotliwości objętych przez FFT, czyli od zera do f_s :

$$f_{h[n]} = (n \cdot f_{sig}) modulo(f_s). \tag{1.5}$$

Wartości $X_{avr}(i)$ są to średnie z modułów kilku pomiarów prążka i, wartość N_H to liczba harmonicznych uwzględnianych w obliczeniach, która standardowo wynosi 10. Parametr ten opisuje stosunek mocy niesionej przez harmoniczne sygnału do mocy niesionej przez sygnał. Jeżeli nie dobierzemy odpowiednio częstotliwości sygnału wejściowego, wówczas moc zgromadzona w prążku sygnałowym rozpłynie na pozostałe prążki, powodując błąd w obliczeniach [2].

Stosunek sygnału do szumu

Stosunek sygnału do szumu nie będącego zniekształceniami harmonicznymi (ang. Signal to Non-Harmonic Ratio – SNHR), to kolejny parametr opisujący jakość pracy

przetwornika. Parametr ten definiujemy jako stosunek mocy sygnału do mocy w pozostałej części pasma, nie wliczając częstotliwości harmonicznych:

$$SNHR = 20\log_{10}\left(\sqrt{\frac{X_{avr}\left(f_{sig}\right)^2}{\sum_{i=1}^{2^N-1, i\neq f_{h[n]}} X_{avr}(f_i)^2}}\right). \tag{1.6}$$

Wyznaczona wartość mówi o rzeczywistych szumach wynikających z efektów kwantyzacji, czy szumów elementów wewnętrznych (tranzystory, rezystancje) [2].

Stosunek sygnału do szumu i zniekształceń

Stosunek sygnału do szumu i zniekształceń (ang. Signal to Noise and Distortion – SINAD) to parametr określający całkowitą jakość pracy przetwornika. Zdefiniować go możemy jako stosunek energii niesionej przez sygnał do całkowitej energii szumów i zniekształceń [2]:

$$SINAD = 20\log_{10}\left(\sqrt{\frac{X_{avr}\left(f_{sig}\right)^2}{\sum_{i=1}^{2^N-1, i\neq f_{sig}} X_{avr}(f_i)^2}}\right). \tag{1.7}$$

Efektywna liczba bitów

Efektywna liczba bitów (ang. Effective Number of Bits) to parametr określający jakość pracy przetwornika. Jest odpowiednikiem parametru SINAD i określa rzeczywistą liczbę bitów przetwornika po uwzględnieniu szumów i zniekształceń:

$$ENOB = \frac{SINAD - 1.76}{6.02}. \tag{1.8}$$

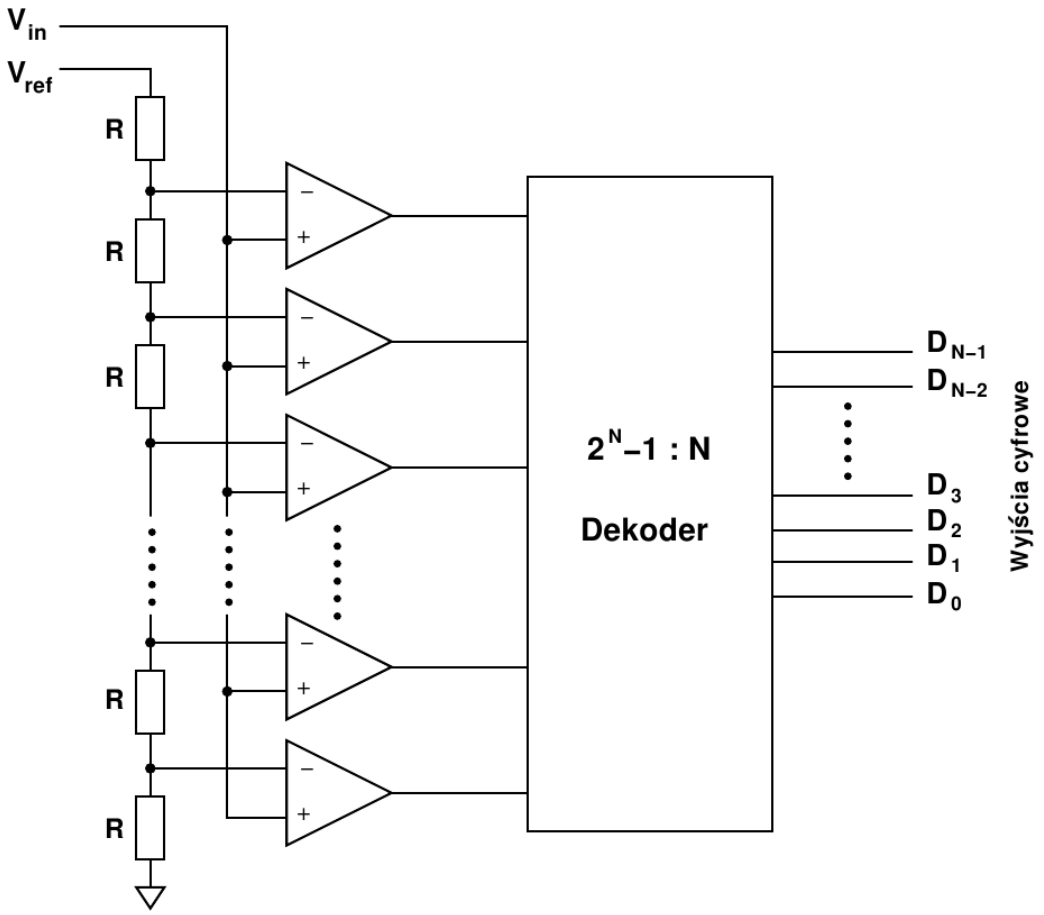
Zakres wolny od zniekształceń

Zakres wolny od zniekształceń (ang. Spurious Free Dynamic Range – SFDR) jest parametrem bardzo ważnym szczególnie w aplikacjach telekomunikacyjnych. Definiujemy parametr ten jako stosunek mocy sygnału do mocy największego zniekształcenia [2]:

$$SFDR = 20\log_{10}\left(\sqrt{\frac{X_{avr}\left(f_{sig}\right)^2}{\max_{i=1}^{2^N-1, i\neq f_{sig}}\left(X_{avr}(f_i)^2\right)}}\right). \tag{1.9}$$

1.2. Rodzaje przetworników

Istnieje wiele rodzajów przetworników analogowo-cyfrowych przystosowanych do różnych zastosowań. Wprowadzając klasyfikacje pod względem konstrukcji ADC możemy wyróżnić następujące typy: przetworniki o przetwarzaniu bezpośrednim, o architekturze potokowej, realizujące metodę kolejnych przybliżeń i przetworniki



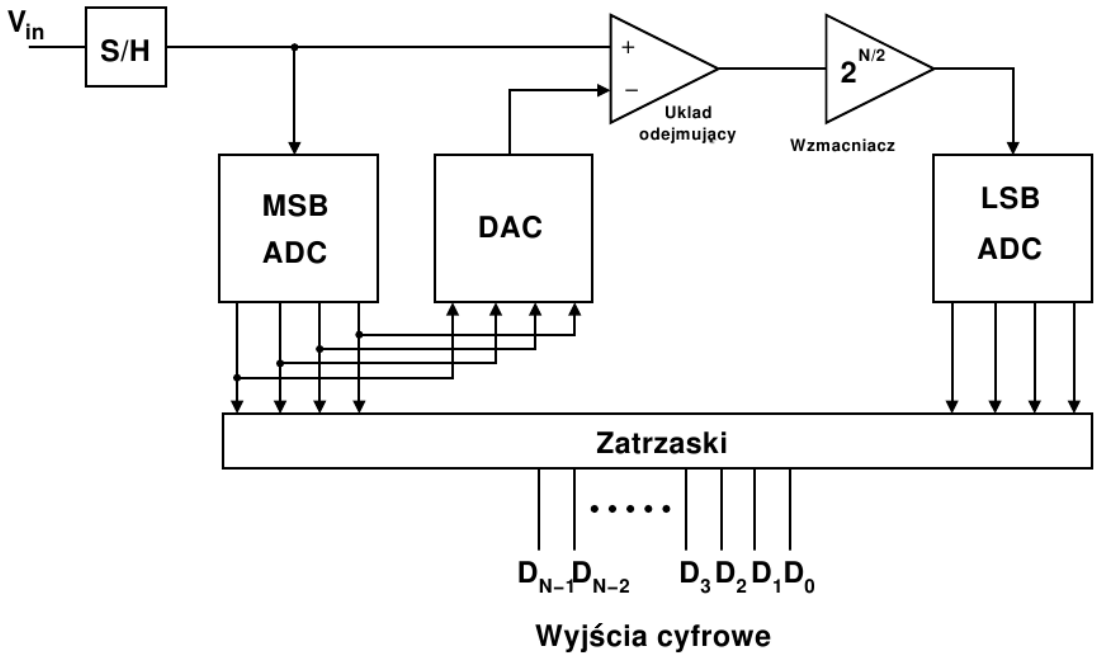
Rysunek 1.7. Schemat działania przetwornika równoległego - Flash [3].

z nadpróbkowaniem. Poniżej omówione zostaną pokrótce poszczególne typy przetworników, ich zasada działania, zalety i wady [3].

1.2.1. Przetworniki równoległe - Flash

Schemat działania tego typu przetworników został przedstawiony na rys. 1.7. Działanie tego przetwornika polega na jednoczesnym porównaniu sygnału wejściowego za pomocą $2^N - 1$ komparatorów z napięciami odniesienia uzyskiwanymi dzięki rezystancyjnemu dzielnikowi napięcia składającego się z 2^N rezystorów. Każdy z komparatorów porównuje sygnał wejściowy z napięciem odniesienia i w wyniku tego porównania, jeżeli sygnał wejściowy jest większy od sygnału odniesienia na wyjściu komparatora pojawia się 1, w pozostałych przypadkach pojawia się 0. Sygnały z wyjść komparatorów są konwertowane przy pomocy dekodera i na wyjściu przetwornika otrzymujemy słowo bitowe reprezentujące analogowe napięcie wejściowe.

Są to najszybsze ze wszystkich rodzajów przetworników, jednak kosztem szybkości jest powierzchnia zajmowana przez dużą liczbę komparatorów, jak również ich



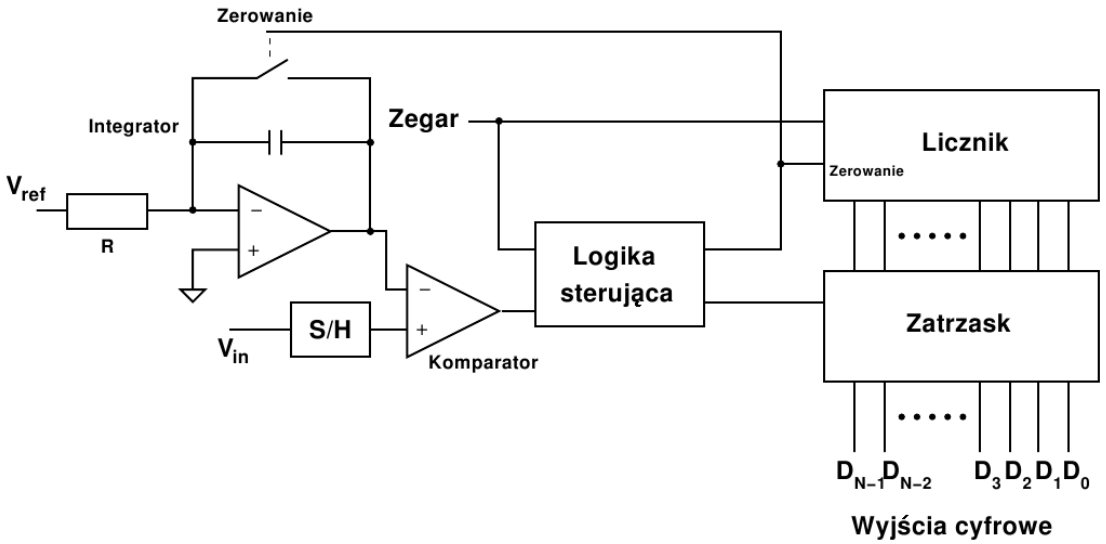
Rysunek 1.8. Schemat blokowy przetwornika szeregowo-równoległego [3].

zapotrzebowanie na moc, gdyż przy zwiększaniu dokładności przetwornika drastycznie rośnie liczba komparatorów potrzebnych do jego zbudowania (podwaja się przy wzroście dokładności o jeden bit) [3].

1.2.2. Przetworniki szeregowo-równoległe

Przetworniki szeregowo-równoległe to inny rodzaj przetworników Flash, nazywany również przetwornikiem dwu krokowym, co bardziej odzwierciedla zasadę jego działania. Schemat takiego układu przedstawiony jest na rys. 1.8, a jego zasada działania polega na rozdzieleniu konwersji sygnału wejściowego na dwa kroki. W pierwszym kroku zapamiętany w układzie próbkująco-pamiętającym (ang. sample-and-hold – S/H) sygnał zostaje przetworzony przez pierwszy z przetworników ADC, który generuje sygnał odpowiadający przybliżonej wartości napięcia wyjściowego. Kolejny krok polega na przetworzeniu otrzymanego z pierwszego przetwornika słowa cyfrowego przez przetwornik cyfrowo-analogowy, odjęciu sygnału wyjściowego od sygnału wejściowego, a następnie po wzmocnieniu $2^{N/2}$ razy podanie takiego sygnału na drugi przetwornik ADC. Po połączeniu kodów z obydwóch przetworników otrzymujemy kod wyjściowy przetwornika odpowiadający podanemu sygnałowi wejściowemu.

Niewątpliwą zaleta tego typu układów w porównaniu do przetworników typu Flash, jest zmniejszenie liczby komparatorów z $2^N - 1$ do $2(2^{N/2} - 1)$. Wadą na-



Rysunek 1.9. Schemat blokowy przetwornika z pojedynczym całkowaniem [3].

tomiast jest konieczność stosowania układu próbkująco-pamiętającego, jak również przetwornika cyfrowo-analogowego [3].

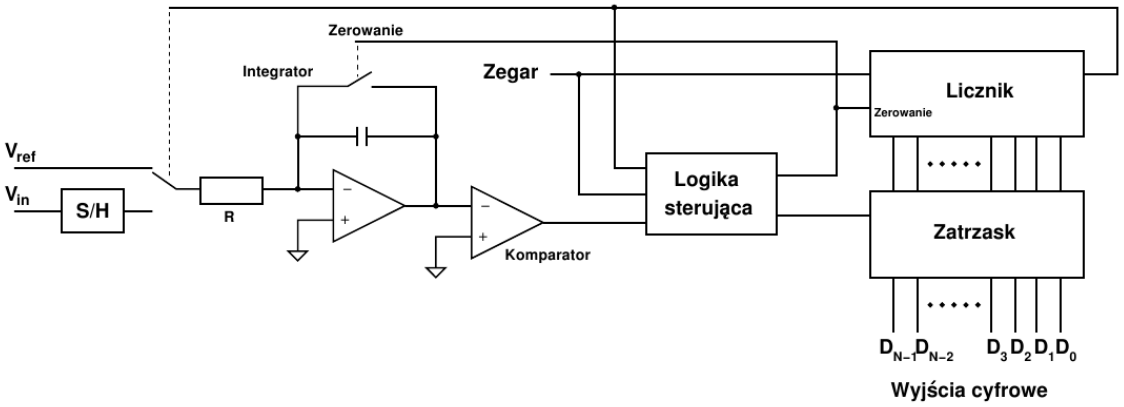
1.2.3. Przetworniki o architekturze potokowej

Ten typ przetworników został omówiony dokładniej w następnym punkcie, gdyż praca ta skupia się przede wszystkim na tego typu przetwornikach.

1.2.4. Przetworniki całkujące

Rozróżniamy dwa typy przetworników całkujących: z pojedynczym i podwójnym całkowaniem. Układy te całkują napięcie wejściowe, a następnie zliczają czas tego całkowania, na podstawie którego określany jest kod cyfrowy odpowiadający napięciu wejściowemu. Przetworniki całkujące są jednymi z najdokładniejszych przetworników analogowo-cyfrowych, są jednak stosunkowo wolne.

Na rys. 1.9 przedstawiony został schemat działania przetwornika ADC z pojedynczym całkowaniem, nazywanym również przetwornikiem z rozładowaniem liniowym (ang. single slope ADC). Układ ten działa w następujący sposób: w momencie rozpoczęcia konwersji sygnał wejściowy zostaje zapamiętany i przez cały czas porównywany jest z wartością sygnału zwracanego przez integrator - układ całkujący w którym całkowane jest napięcie odniesienia V_{ref}. W tym samym czasie licznik zlicza impulsy zegarowe, aż do momentu, gdy napięcie z integratora nie będzie równe napięciu wejściowemu. W momencie przekroczenia napięcia wejściowego przez napięcie



Rysunek 1.10. Schemat blokowy przetwornika z podwójnym całkowaniem [3].

na integratorze wyjście komparatora przełączy się zatrzymując przy tym zliczanie impulsów, następuje zapamiętanie stanu licznika i wyzerowanie układu w celu przygotowania do kolejnego przetwarzania. Liczba zliczonych impulsów zegarowych jest proporcjonalna do napięcia wejściowego. Czas konwersji takiego przetwornika można opisać równaniem:

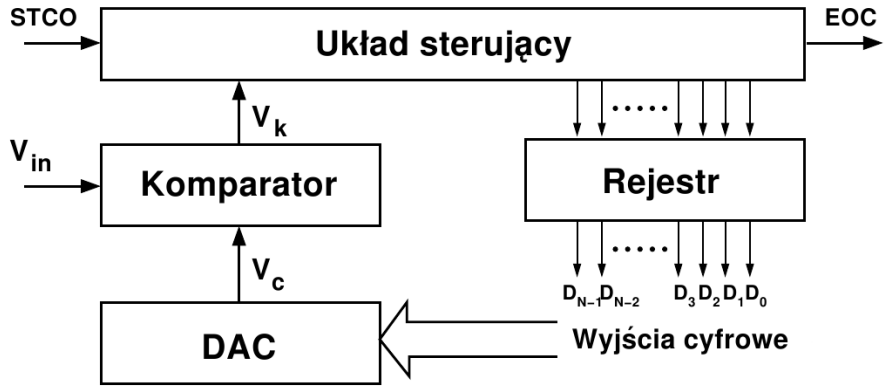
$$t_c = \frac{V_{in}}{V_{ref}} RC, \tag{1.10}$$

natomiast wyjściowy kod cyfrowy:

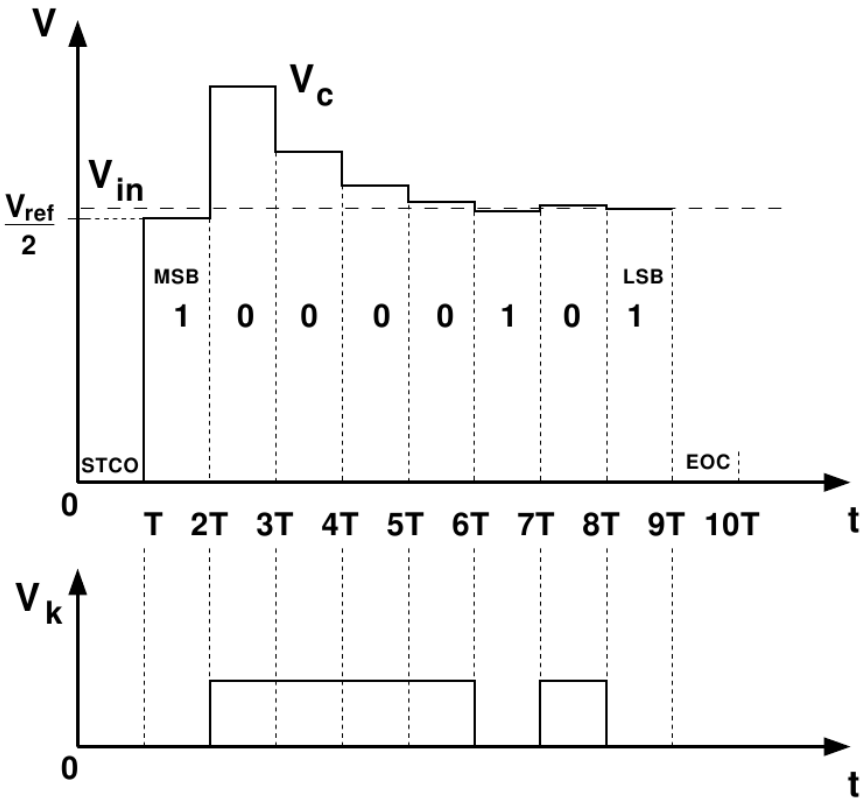
$$N = t_c \cdot f_c, \tag{1.11}$$

gdzie f_c to częstotliwość zegara wejściowego. Zaletą opisanego przetwornika jest prostota budowy i działania, wadami natomiast: duża czułość na zakłócenia w postaci tętnień przetwarzanego napięcia V_{in} oraz długi czas przetwarzania sygnału na liczbę. Pierwszej z tych wad pozbawiony jest przetwornik z podwójnym całkowaniem [3, 4].

W przetwornikach z podwójnym całkowaniem (rys. 1.10) całkowaniu podlega zarówno napięcie odniesienia jak również sygnał wejściowy. Przetwornik ten zamienia średnią wartość mierzonego napięcia na czas. W pierwszej fazie pracy układu do integratora podłączone jest mierzone napięcie V_{in}, przez co na wyjściu integratora otrzymujemy liniowo narastające napięcie. Czas narasta tego sygnału jest stały i odmierzany jest przez licznik lub układ sterujący. Licznik cały czas zlicza impulsy, a jego pojemność jest tak dobrana, że maksymalną liczbę impulsów zlicza w czasie całkowania sygnału V_{in}. Po zakończeniu całkowania napięcia V_{in}, do integratora doprowadzone zostaje napięcie wzorcowe V_{ref} o przeciwnej biegunowości. Kiedy napięcie wyjściowe z integratora osiągnie wartość zero następuje zakończenie zliczania impulsów [3, 4].



(a)



(b)

Rysunek 1.11. Przetwornik ADC z aproksymacją krokową: (a) schemat blokowy, (b) zasada działania [4].

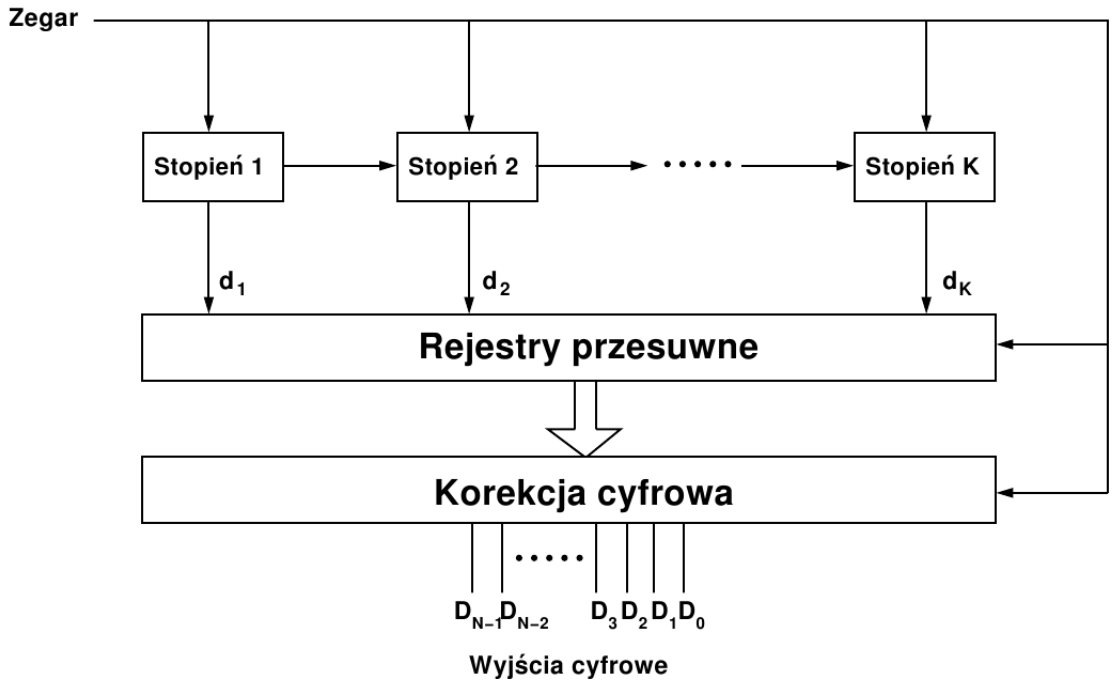
1.2.5. Przetworniki z aproksymacją krokową

Przetworniki z aproksymacją krokową, nazywane również przetwornikami realizującymi metodę kolejnych przybliżeń działają w sposób cykliczny. Cykl pracy N-bitowego przetwornika tego typu zajmuje N+1 taktów. Na rysunku 1.11 przedstawiony został schemat blokowy takiego przetwornika, jak również zasada jego działania na przykładzie 8-bitowego przetwornika. Przetwarzanie próbkowanego napięcia rozpoczyna się od odebrania przez układ sterujący sygnału rozpoczęcia przetwarzania STCO. Wartości poszczególnych bitów w rejestrze ustawiane są w kolejności od najbardziej (MSB), do najmniej znaczącego (LSB). W pierwszym kroku najbardziej znaczący bit w rejestrze ustawiany jest na 1, natomiast wszystkie pozostałe bity na 0. Następnie tak ustawione słowo wyjściowe, po konwersji w przetworniku cyfrowo-analogowym na odpowiadające mu napięcie V_c, porównywane jest z napięciem wejściowym V_{in} przy pomocy komparatora. W przypadku, gdy $V_c > V_{in}$, ustawiona w rejestrze jedynka przy przejściu do następnego bitu zostaje zamieniona na 0, w przeciwnym przypadku nie weryfikujemy zawartości rejestru. W analogiczny sposób postępujemy w każdym kolejnym kroku, aż ustawiony zostanie ostatni, najmniej znaczący bit. Otrzymane w ten sposób wyjściowe słowo jest reprezentacją cyfrową przetwarzanego napięcia wejściowego. W związku z tym, że konwersja taka ma iteracyjny charakter, częstotliwość próbkowania takich przetworników jest dużo mniejsza niż w przetwornikach o przetwarzaniu bezpośrednim i zależy w głównej mierze od rozdzielczości przetwornika - ilości kroków [4].

1.3. ADC o architekturze potokowej

Istnieje wiele różnych rodzajów przetworników analogowo-cyfrowych, które pokrótce opisane zostały w poprzednim podrozdziale. Wybór architektury, którą chcemy wykorzystać zależy między innymi od rozdzielczości wyjściowej, szybkości działania i zapotrzebowania na moc. Często kompromisem pomiędzy powyższymi parametrami jest architektura potokowa (ang. pipeline ADC). W tym rozdziale omówiona zostanie dokładniej zasada i sposób działania przetworników potokowych [2, 3].

Na rysunku 1.12 przedstawiony został schemat blokowy przetwornika potokowego, w skład którego wchodzi K stopni niskiej rozdzielczości, rejestry przesuwne i w przypadku stosowania korekcji cyfrowej, blok korekcji cyfrowej. Pojedynczy stopień przedstawiony został na rysunku 1.13, składa się on z przetwornika analogowo-cyfrowego niskiej rozdzielczości (ang. subADC) i jednostki arytmetycznej zwanej mnożącym przetwornikiem cyfrowo-analogowym (ang. Multiplying Digital

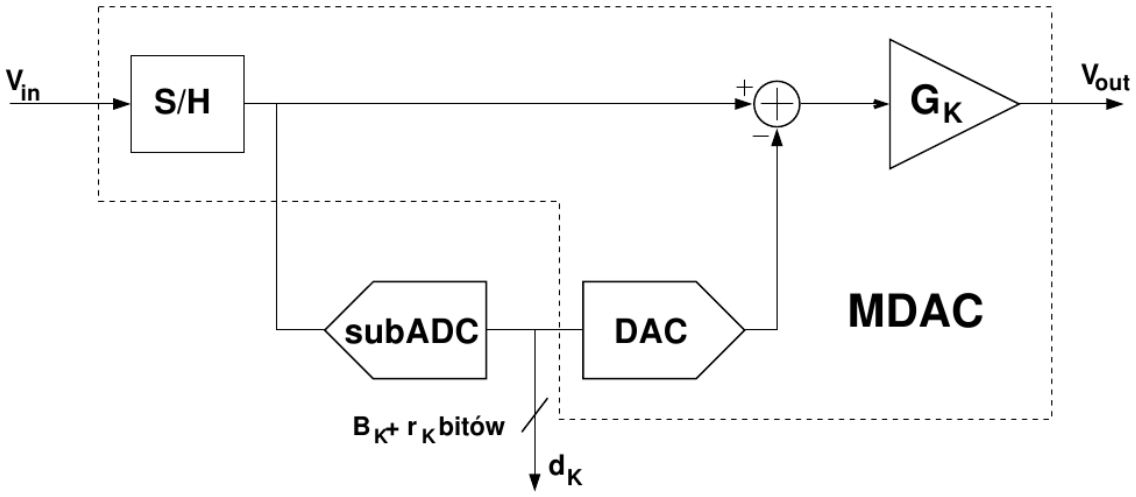


Rysunek 1.12. Schemat blokowy potokowego przetwornika analogowo-cyfrowego [2, 5].

to Analog Converter - MDAC), który składa się z układu próbkująco-pamiętającego (ang. sample and Hold - S/H), przetwornika cyfrowo-analogowego (ang. Digital to Analog Converter - DAC) oraz układu odejmująco-mnożącego. Każdy ze stopni ma wyjście cyfrowe i analogowe. Pierwsze z nich to d_K, przy pomocy którego zwracany jest cyfrowy kod o rozdzielczości $B_K + r_K$ bitów, który jest wynikiem konwersji sygnału wejściowego przez subADC. B_K oznacza efektywną liczbę bitów, natomiast r_K to ilość bitów redundantnych, które wykorzystywane są w algorytmie korekcji cyfrowej. W przypadku stosowania korekcji cyfrowej $r \geqslant 1$. Drugim wyjściem stopnia jest napięcie, którego wartość wyznaczona jest na podstawie odjęcia od zapamiętanego napięcia wejściowego wyniku konwersji cyfrowo-analogowej wyjścia d_K, z równoczesnym mnożeniem wyniku tej operacji przez $G_K = 2^{B_K}$. Napięcie to, odpowiadające błędowi kwantyzacji subADC, podawane jest na następny stopień przetwornika. Mnożenie dokonywane jest w celu przeskalowania różnicy na pełny zakres wejściowy kolejnego stopnia. Rozdzielczość przetwornika potokowego wyliczamy ze wzoru:

$$N = r_K + \sum_{i=1}^{K} B_i. \tag{1.12}$$

Wszystkie stopnie pracują w tym samym czasie, jednak przetwarzają rożne próbki. Podczas, gdy pierwszy stopień przetwarza najświeższą próbkę, stopień drugi przetwa-



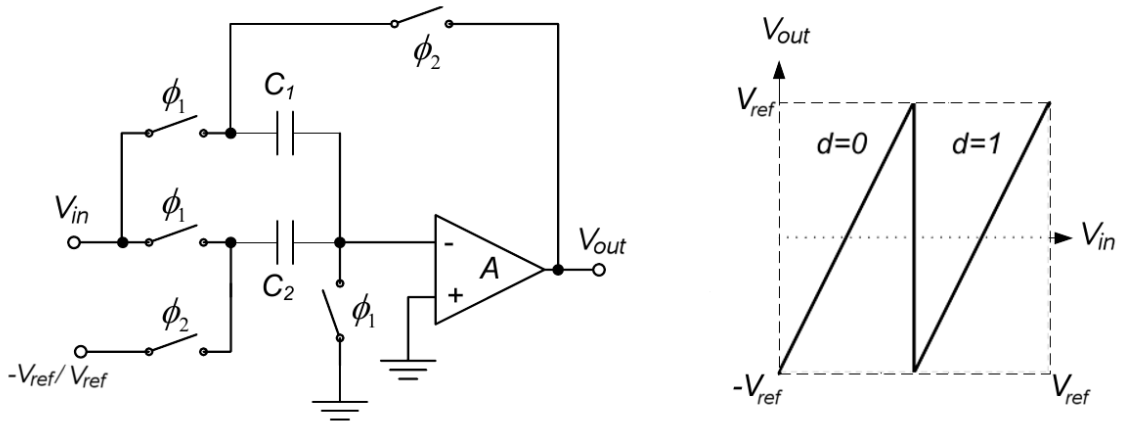
Rysunek 1.13. Schemat pojedynczego stopnia przetwornika potokowego [2].

rza już napięcie zwrócone ze stopnia pierwszego. Przetwarzanie takie jest koncepcją działania przetwornika potokowego. Dzięki przetwarzaniu potokowemu architektura ta jest dobrym kandydatem do szybkich przetworników o dużej rozdzielczości, gdyż pojedyncze stopnie mogą być niskiej rozdzielczości, a rozdzielczość całego przetwornika zależy tylko od ich ilości.

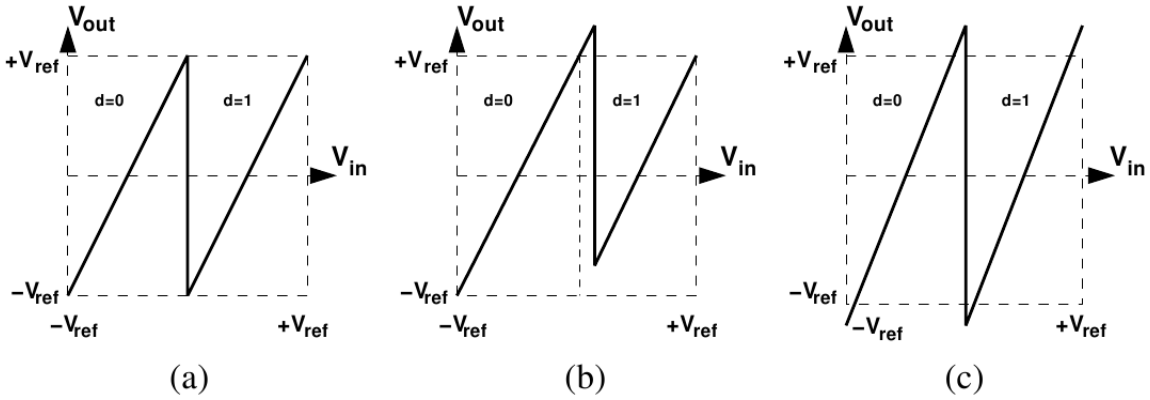
Architektura przetworników typu pipeline opiera się na tym, że przetwornik o dużej rozdzielczości możemy zastąpić kilkoma przetwornikami o niższej rozdzielczości połączonymi ze sobą, dzięki czemu zmniejszamy liczbę komparatorów potrzebnych do zbudowania przetwornika o takiej samej rozdzielczości, a co za tym idzie zmniejsza się przestrzeń i zapotrzebowanie na moc takiego układu [2, 5, 6, 3, 1].

1.3.1. Przetworniki ADC z całkowitą liczbą bitów na stopień

Stopnie n-bitowe to stopnie w których nie stosuje się bitów redundantnych. Najprostszym n-bitowym potokowym przetwornikiem analogowo-cyfrowym jest przetwornik złożony z jednobitowych stopni. Schemat układu mnożąco-odejmującego pojedynczego stopnia takiego przetwornika oraz jego funkcja przenoszenia, zostały zaprezentowane na rysunku 1.14. Przedstawiony układ pracuje w dwóch fazach. W pierwszej fazie ϕ_1 napięcie wejściowe V_{in} zapamiętane zostaje na kondensatorach C_1 i C_2. W fazie drugiej ϕ_2, w zależności od rezultatu konwersji subADC do zapamiętanego napięcia dodane lub odjęte zostaje napięcie V_{ref}, z równoczesnym mnożeniem wyniku przez $(1+C_1/C_2)$. Dla przetwornika 1-bitowego mnożenie powinno odbywać się przez 2, dlatego należy zadbać aby $C_1 = C_2$. Kod wyjściowy zwracany przez taki przetwornik dany jest wzorem:



Rysunek 1.14. Schemat układu mnożąco-odejmującego i funkcja przejścia 1-bitowego stopnia potokowego ADC [7].



Rysunek 1.15. Funkcje przejścia 1-bitowego stopnia potokowego ADC: (a) idealna, (b) offset komparatorów, (c) niedopasowanie kondensatorów [7].

$$D_o = d_1 \cdot 2^{N-1} + d_2 \cdot 2^{N-2} + \cdots + d_{N-1} \cdot 2^1 + d_N \cdot 2^0, \tag{1.13}$$

gdzie d_N to decyzja zwrócona przez N-ty stopień. Funkcja przenoszenia zawarta na tym rysunku jest odzwierciedleniem idealnej funkcji przenoszenia przez 1-bitowy stopień. W rzeczywistych układach funkcja ta ulega deformacjom spowodowanym nieidealnością takich układów. Na rysunku 1.15 zaprezentowane zostały deformacje funkcji przenoszenia w zależności od różnych czynników, takich jak: offset komparatorów, niedopasowanie pojemności. Na przedstawionym schemacie układu mnożąco-odejmującego zaobserwować możemy pojemności, których niedopasowanie powoduje zniekształcenia funkcji przenoszenia, jak również wzmacniacz, którego skończone wzmocnienie również powoduje powstawanie odchyleń w funkcji przenoszenia [7]. Wzór, w którym uwzględniono powyższe parametry, opisujący funkcję przenoszenia zapisać możemy w następującej formie [8, 9]:

$$V_{out} = \frac{V_{in} \cdot (C_1 + C_2) - (2 \cdot d - 1) \cdot V_{ref} \cdot C_2}{C_1 \cdot (1 + \frac{C_1 + C_2}{A \cdot C_1})}, \tag{1.14}$$

gdzie d jest decyzją zwróconą ze stopnia i wynosi 0 lub 1, A jest wzmocnieniem wzmacniacza operacyjnego, a V_{ref} napięciem odniesienia. Wzór ten wyprowadzamy analizując pracę układu 1.14 w fazie ϕ_1 i ϕ_2, co zaprezentowane zostało na rysunku 1.16. W pierwszej fazie ϕ_1 działania układu (rys. 1.16(a)), do pojemności C_1 i C_2 przyłożone zostaje napięcie V_{in}, czego skutkiem jest jego zapamiętanie w postaci zgromadzonych na okładkach kondensatorów ładunków, odpowiednio:

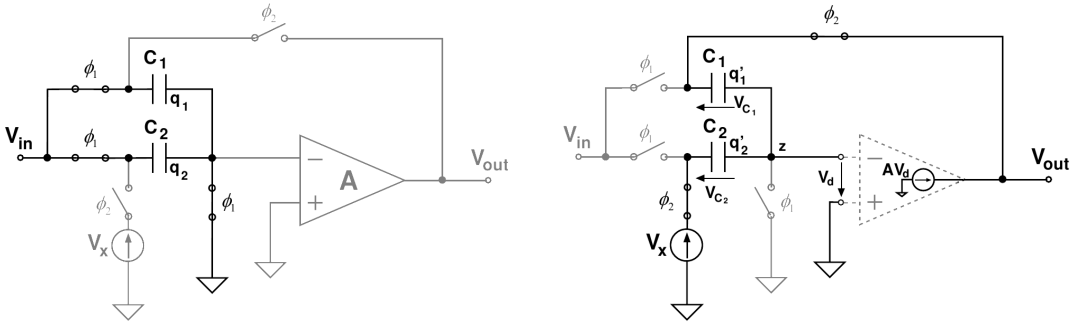
$$q_1 = V_{in} \cdot C_1, \qquad q_2 = V_{in} \cdot C_2. \tag{1.15}$$

Następnie kończy się faza próbkowania i układ przechodzi do fazy drugiej ϕ_2. W celu analizy pracy układu w fazie ϕ_2, na rysunku 1.16(b) umieszczony został uproszczony model wzmacniacza operacyjnego. W fazie drugiej wzmacniacz operacyjny umieszczony zostaje w pętli ujemnego sprzężenia zwrotnego, poprzez pojemność C_1, wymuszając w punkcie z napięcie $-V_d$. Ładunki zgromadzone na pojemnościach C_1 i C_2 wynoszą teraz odpowiednio:

$$q_1{}' = V_{C_1} \cdot C_1, \qquad q_2{}' = V_{C_2} \cdot C_2, \tag{1.16}$$

a ponieważ całkowity ładunek zgromadzony na pojemnościach C_1 i C_2 nie zmienił się, możemy zapisać następującą równość:

$$q_1 + q_2 \;=\; q_1{}' + q_2{}'. \tag{1.17}$$



(a) (b)

Rysunek 1.16. Fazy działania układu mnożąco-odejmującego (a) próbkowanie sygnału wejściowego, (b) odejmowanie od zapamiętanej wartości V_x i mnożenie wyniku przez $(1 + C_1/C_2)$.

Wiedząc ponadto, że:

$$V_{C_1} = V_{out} + V_d, \qquad V_{C_2} = V_d + V_x, \qquad V_{out} = A \cdot V_d, \tag{1.18}$$

oraz znając zależności 1.15, możemy równość 1.17 zapisać w następującej postaci:

$$V_{in} \cdot C_1 + V_{in} \cdot C_2 \;=\; (V_{out} + \frac{V_{out}}{A}) \cdot C_1 + (\frac{V_{out}}{A} + V_x) \cdot C_2, \tag{1.19}$$

z której po kilku prostych przekształceniach otrzymujemy:

$$V_{out} = \frac{V_{in} \cdot (C_1 + C_2) - V_x \cdot C_2}{C_1 \cdot (1 + \frac{C_1 + C_2}{A \cdot C_1})}. \tag{1.20}$$

Napięcie V_x zależne jest od decyzji subADC i dla stopnia 1-bitowego możemy jego wartość zapisać $V_x = (2 \cdot d - 1) \cdot V_{ref}$.

Przetwornik zbudowany z n-bitowych stopni nie potrzebuje bloku korekcji cyfrowej, gdyż wyjścia z każdego ze stopni nie podlegają żadnej korekcji w trakcie wyznaczania cyfrowej reprezentacji napięcia wejściowego. Konieczne jest jednak zastosowanie rejestrów przesuwnych, które zapamiętają wartości zwracane przez każdy ze stopni do czasu przetworzenia próbki przez wszystkie stopnie. Przetwornik taki ma jednak dużą wadę, gdyż aby był on dokładny, każdy stopień musi posiadać bardzo dokładne komparatory. Offset komparatorów w jednobitowych stopniach powodował będzie bowiem duże niedokładności, szczególnie jeżeli niedokładne będą komparatory w początkowych stopniach. Sposobem na zapobiegnięcie błędom powstałym przez niedokładność komparatorów, jest stosowanie stopni tzw. $n + 0.5$-bitowych [2, 5, 7].

Stopnie n-bitowe o większej rozdzielczości, buduje się w analogiczny sposób co 1-bitowe. Przy większej rozdzielczości zmienia się więc nie tylko rozdzielczość sub-ADC i DAC, ale również układ mnożąco-odejmujący. W n-bitowym stopniu układ mnożąco-odejmujący składa się z 2^n pojemności, jak również mnożenie odbywa się przez 2^n. Wzrost liczby pojemności zachodzi ze względu na większą liczbę poziomów kwantyzacji.

1.3.2. Przetworniki ADC z jednobitową redundancją

Stopnie zwane n+0.5-bitowymi, to stopnie które zwracają informacje na n+1 bitach, z tym że ostatni bit jest bitem redundantnym. W stopniach n+0.5-bitowych wykorzystuje się algorytm korekcji cyfrowej z jednobitową redundancją. Najprostszym stopniem n+0.5-bitowym jest stopień 1.5-bitowy. Używając 1.5-bitowych stopni do osiągnięcia M bitowej rozdzielczości potrzebujemy $M-1$ stopni, w których wzmocnienie reszty dla kolejnego stopnia wynosi 2. Napięcie wejściowe dla takiego stopnia powinno zawierać się w granicach $|V_{in}| \leq V_{ref}$, natomiast progi przełączania komparatorów powinny wynosić odpowiednio $-V_{ref}/4$ i $V_{ref}/4$. W praktyce progi te odbiegają od wartości idealnych ze względu na mniejsze lub większe niedokładności komparatorów [2, 5, 7].

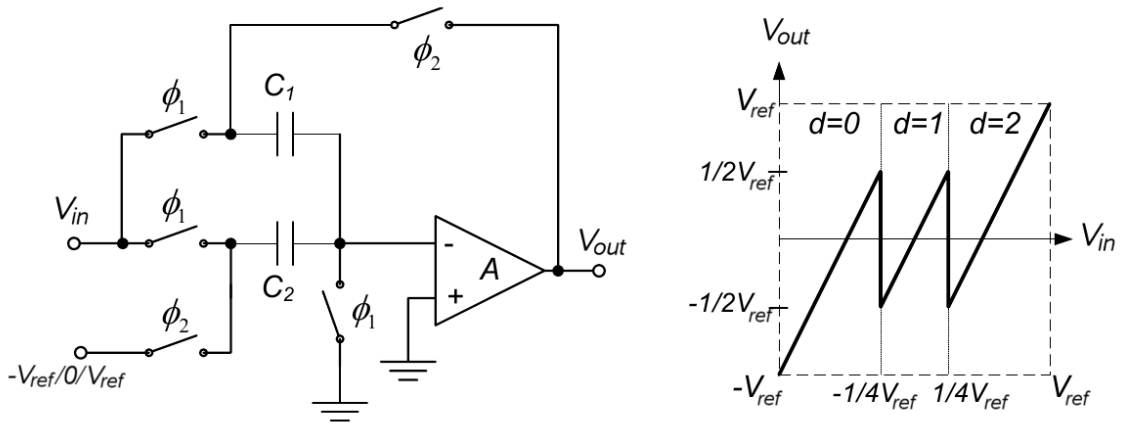
Funkcję przenoszenia dla pojedynczego stopnia 1.5-bitowego można opisać analitycznie w następujący sposób [2]:

$$V_{out} = 2V_{in} - V_R, V_R = \begin{cases} -V_{ref} & V_{in} \leqslant -1/4V_{ref} \\ 0 & -1/4V_{ref} < V_{in} < 1/4V_{ref} \\ V_{ref} & V_{in} \geqslant 1/4V_{ref} \end{cases}, \tag{1.21}$$

lub bardziej dokładnym wzorem, uwzględniającym niedopasowanie pojemności oraz wzmocnienie wzmacniacza [7, 8, 9]:

$$V_{out} = \frac{V_{in} \cdot (C_1 + C_2) - (d-1)\, V_{ref} \cdot C_2}{C_1 \cdot (1 + \frac{C_1 + C_2}{A \cdot C_1})}, \tag{1.22}$$

gdzie d jest decyzją zwróconą ze stopnia i wynosi 0, 1 lub 2, A jest wzmocnieniem wzmacniacza operacyjnego, a V_{ref} napięciem odniesienia. Na rysunku 1.17 przedstawiony został układ mnożąco-odejmujący pojedynczego 1.5-bitowego stopnia oraz funkcja przejścia takiego stopnia, na której przedstawione są decyzje zwracane przez stopień. Stopień 1.5-bitowy zwraca informacje na 2 bitach, jednak posiada tylko 3 stany wyjściowe. Stan 11 jest stanem zabronionym. W stopniach n+0.5-bitowych dodając jeden bit informacji więcej, rezygnujemy jednocześnie z jednego poziomu

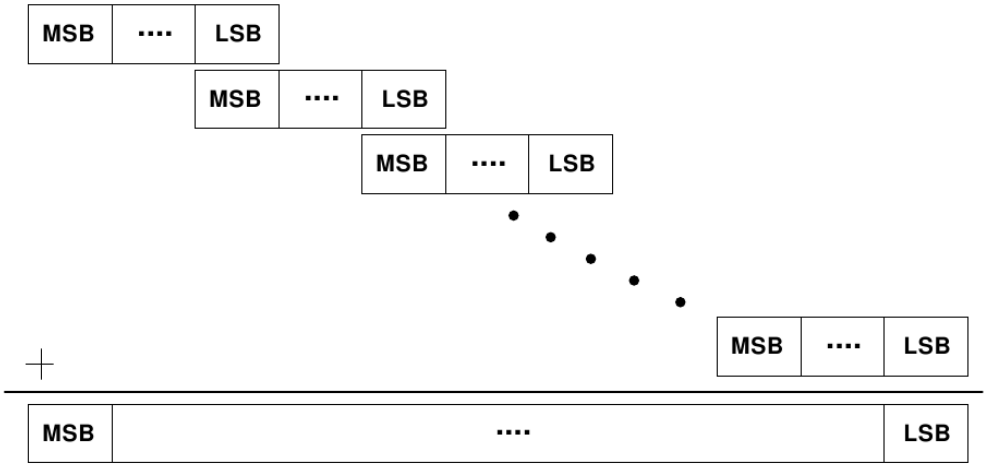


Rysunek 1.17. Schemat układu mnożąco-odejmującego i funkcja przejścia 1.5-bitowego stopnia potokowego ADC [7].

kwantyzacji. Dzięki takiemu zabiegowi, zwiększamy liczbę poziomów kwantyzacji o jeden, zachowując takie samo wzmocnienie oraz dla szerokiego zakresu napięć wejściowych, napięcie wyjściowe ograniczone jest do $|V_{out}| \leqslant V_{ref}/2$. Aspekt ten ma duże znaczenie, ze względu na dokładność komparatorów. W stopniach n-bitowych, aby przetwornik był dokładny konieczne jest zastosowanie bardzo dokładnych komparatorów, w stopniach n+0.5-bitowych natomiast nie tracimy dokładności przetwornika nawet przy niewielkich niedokładnościach komparatorów, gdyż nawet w przypadku offsetu powstały błąd może zostać skorygowany w procesie korekcji cyfrowej [2, 7].

Najbardziej popularnym algorytmem korekcji cyfrowej w przetwornikach potokowych jest algorytm redundantnego kodowania znaku z jednobitową redundancją. Korekcja ta jest dość prosta i polega na odpowiednim zsumowaniu wyjść ze wszystkich stopni. Wyjścia te oczywiście muszą być odpowiednio opóźnione, co wymaga zapamiętania informacji z poprzednich pomiarów. Sumowanie to zostało schematycznie przedstawione na rysunku 1.18 i polega ono na tym, ze do każdego (za wyjątkiem pierwszego) najbardziej znaczącego bitu słowa wyjściowego z pojedynczego stopnia dodajemy najmniej znaczący bit ze słowa wyjściowego wcześniejszego stopnia. Po takim zsumowaniu wyjść z wszystkich stopni otrzymujemy reprezentację cyfrową sygnału analogowego podanego na wejście przetwornika [2, 5].

Jak możemy zauważyć ostatni stopień nie podlega korekcji cyfrowej, nie powinien on więc wnosić informacji redundantnej. Najczęściej stopień ten zbudowany jest z samych komparatorów. Dla przetwornika zbudowanego z 1.5-bitowych stopni powinien być więc w pełni 2-bitowy, tzn. powinien mieć 4 możliwe stany kwantyzacji. Drugą możliwością jest zastosowanie w ostatnim stopniu przetwornika również



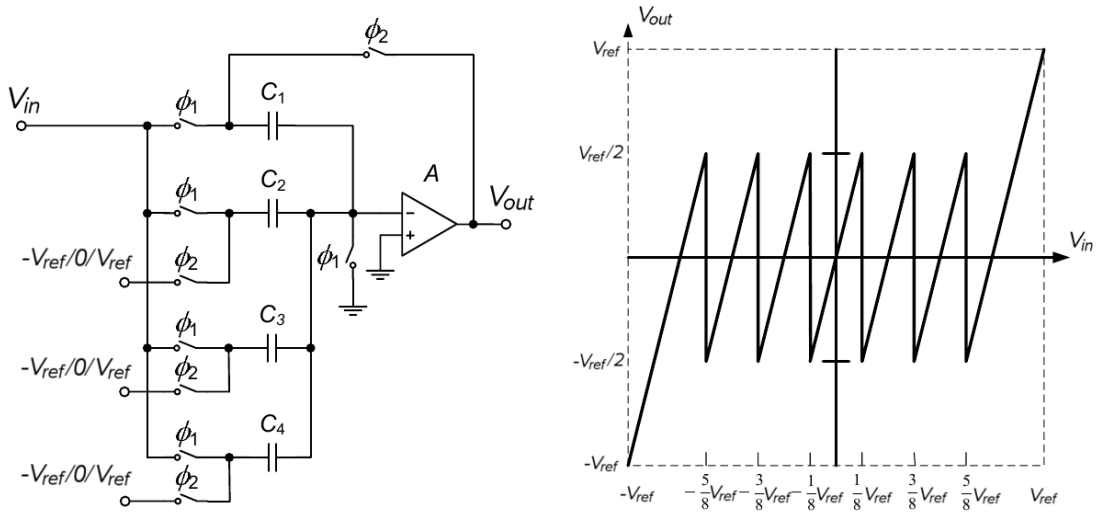
Rysunek 1.18. Korekcja cyfrowa potokowego przetwornika ADC zbudowanego z n+0.5-bitowych stopni, przy użyciu algorytmu z jednobitową redundancją [5].

n+0.5-bitowego stopnia, jednak w tym przypadku musimy liczyć się z tym, że pogorszą się nieznacznie parametry takiego przetwornika. Spowodowane jest to oczywiście zmniejszoną ilością poziomów kwantowania ostatniego stopnia.

Często stosowanymi stopniami są również stopnie 2.5-bitowe. Na rysunku 1.19 zamieszczony został schemat układu mnożąco-odejmującego 2.5-bitowego stopnia, jak również funkcja przejścia dla takiego stopnia. W stopniu 2.5-bitowym informacja zwracana jest na 3 bitach jednak dostępnych jest tylko 7 poziomów kwantyzacji (6 komparatorów). W tabeli 1.1 podano decyzje zwracane przez stopień w zależności od napięcia wejściowego. Funkcje przejścia dla 2.5-bitowego stopnia możemy zapisać przy pomocy następującego wzoru [7]:

$$\begin{aligned} V_{out} = & \frac{1}{C_1 \cdot (1 + \dfrac{C_1 + C_2 + C_3 + C_4}{A \cdot C_1})} \cdot \{V_{in} \cdot (C_1 + C_2 + C_3 + C_4) - \\ & - V_{ref} \cdot (C_2 \cdot a_1 + C_3 \cdot a_2 + C_4 \cdot a_3)\}, \end{aligned} \tag{1.23}$$

gdzie a_1, a_2, a_3 są decyzją zwróconą ze stopnia, A jest wzmocnieniem wzmacniacza operacyjnego, a V_{ref} napięciem odniesienia. Wzór ten uwzględnia niedopasowanie pojemności i wzmocnienie wzmacniacza. Aby zapewnić mnożenie przez 4 wszystkie pojemności powinny być jednakowe. Decyzje a_1, a_2 i a_3 mówią o tym, kiedy pojemności C_2, C_3 i C_4 powinny być połączone odpowiednio do $-V_{ref}$, 0 lub V_{ref}, podczas gdy ϕ_2 jest w stanie wysokim [7].



(a) (b)

Rysunek 1.19. (a) Schemat układu mnożąco-odejmującego i (b) funkcja przejścia 2.5-bitowego stopnia potokowego ADC [7].

Tablica 1.1. Decyzje zwracane przez stopnie 2.5-bitowe potokowego przetwornika ADC [7].

Wejście analogowe	Decyzje stopnia		
	a_1	a_2	a_3
$-V_{ref} \leqslant V_{in} < -5/8V_{ref}$	-1	-1	-1
$-5/8V_{ref} \leqslant V_{in} < -3/8V_{ref}$	-1	-1	0
$-3/8V_{ref} \leqslant V_{in} < -1/8V_{ref}$	-1	0	0
$-1/8V_{ref} \leqslant V_{in} < 1/8V_{ref}$	0	0	0
$1/8V_{ref} \leqslant V_{in} < 3/8V_{ref}$	1	0	0
$3/8V_{ref} \leqslant V_{in} < 5/8V_{ref}$	1	1	0
$5/8V_{ref} \leqslant V_{in} < V_{ref}$	1	1	1

Rozdział 2

Analiza działania idealnego ADC

W celu lepszego zrozumienia zasady działania potokowych przetworników analogowo-cyfrowych, napisany został program, który symuluje działanie takiego typu przetworników. W rozdziale tym opisane zostaną możliwości napisanego programu, jak również szereg analiz różnych konfiguracji potokowych przetworników i wpływ zmiany różnych parametrów na jakość ich pracy. Porównaniu poddana będzie ponadto praca idealnego przetwornika z pracą przetwornika w którym uwzględniono rzeczywiste efekty występujące w prawdziwych układach.

2.1. Projekt i możliwości idealnego przetwornika

Symulator idealnego przetwornika analogowo-cyfrowego powstał w języku skryptowym *Python*. Program miał na celu odzwierciedlenie działania przetworników analogowo-cyfrowych typu potokowego, o różnej liczbie bitów i zbudowanych z różnej rozdzielczości stopni. Ponadto stworzony przetwornik miał posiadać możliwość symulowania działania zarówno idealnego ADC, jak również efektów zachodzących w prawdziwych przetwornikach, w celu analizy wpływu nieidealności poszczególnych części rzeczywistego układu na jego działanie.

Projekt przetwornika powstał w oparciu o wiedzę zawartą w podrozdziale 1.3, na temat budowy i zasady działania pojedynczych stopni, jak również całego przetwornika. Idealny schemat przetwornika wzbogacony został jeszcze o parametry symulujące efekty pojawiające się w elementach na rzeczywistych układach, takich jak np. szumy, rozrzuty pojemności, offset komparatorów, skończone wzmocnienie wzmacniacza operacyjnego. Rozrzuty pojemności, szumy, jak również offset komparatorów symulowane są poprzez rozrzut Gaussowski wokół odpowiednio: pojemności ustawionej w programie, napięcia sygnału wejściowego dla każdego stopnia, idealnych progów przełączania. Wartości o jakie rozrzucone zostaną powyższe parametry zależą od ustawień w programie, a odchylenie standardowe napięcia szumu termicznego

związanego z przełączaniem pojemności zwanego również szumem kT/C wyliczamy ze wzoru:

$$v_n = \sqrt{\frac{k_B T}{C}}, \tag{2.1}$$

gdzie k_B to stała Boltzmanna, T to temperatura, a C wartość pojemności na której powstaje szum.

Napisany symulator przetwornika ADC udostępnia następujące opcje:

— różna liczba stopni - mamy możliwość budowania przetworników o różnej ilości stopni, co ma wpływ na jego rozdzielczość;

— stopnie różnego typu - stopnie przetwornika mogą być rożnego typu, do dyspozycji mamy 5 typów stopni: 1-bitowe, 1,5-bitowe, 2-bitowe, 2,5-bitowe, 3-bitowe, po wybraniu któregoś typu wszystkie stopnie przetwornika są tego typu, z wyjątkiem ostatniego, który ustawiany jest przy pomocy kolejnego parametru;

— typ ostatniego stopnia - typ ostatniego stopnia ustawiany jest niezależnie od typu pozostałych stopni, w tym wypadku również możemy ustawić 5 typów stopni: 1-bitowe, 1,5-bitowe, 2-bitowe, 2,5-bitowe, 3-bitowe;

— temperatura pracy - mamy możliwość ustalenia temperatury z jaką pracuje przetwornik, ustalona temperatura ma bezpośredni wpływ na szum kT/C w każdym stopniu przetwornika;

— wzmocnienie wzmacniacza - ustala wartość wzmocnienia wzmacniaczy w stopniu mnożąco-odejmującym każdego ze stopni;

— ustawianie pojemności - przy pomocy tego parametru ustawiamy wartość pojemności w układzie mnożąco-odejmującym przetwornika;

— skalowanie pojemności - parametr ten odpowiada za skalowanie pojemności ustawionej za pomocą poprzedniego parametru, w kolejnych stopniach przetwornika;

— offset komparatorów - dla każdego komparatora w stopniu mamy możliwość ustawienia pewnej wartości przesuwającej jego właściwy próg przełączania, z tym że bazując na tym, że wszystkie (z wyjątkiem ostatniego) stopnie są jednakowe, ustawione offsety we wszystkich komparatorach również są jednakowe;

— rozrzut pojemności - tym parametrem ustalamy procentowo wartość odchylenia standardowego rozkładu Gaussowskiego wartości pojemności w całym układzie;

— rozrzut offsetu - ustala wartość odchylenia standardowego z jakim rozrzucone Gaussowsko będą wartości progów przełączania komparatorów w każdym ze stopni.

2.2. Symulacje i wpływ parametrów układu na jego działanie

Pierwszym krokiem podczas testowania napisanego symulatora, było uruchomienie go z opcjami symulującymi działanie idealnego przetwornika analogowo-cyfrowego. Przetestowane zostały różne konfiguracje pojedynczych stopni składających się na 10-cio i 13-bitowe ADC typu potokowego. W tabeli 2.1 zebrane zostały parametry uzyskane w symulacjach dla poszczególnych konfiguracji. Symulacje te przeprowadzane były bez uwzględniania szumów i wszelkiego rodzaju niedokładności, dla amplitudy wejściowej sygnału wynoszącej 1V, co odpowiada pełnemu zakresowi sygnału wejściowego. Sygnałem podawanym na wejście symulowanego przetwornika była sinusoida o częstotliwości 14.6494140625Mhz, która próbkowana była przez przetwornik z częstotliwością 32MHz. Tak dobrane częstotliwości sygnału wejściowego i próbkowania pozwoliły na zebranie $2^{15} = 32768$ punktów z których wyliczana następnie była transformata Fouriera oraz przedstawione w tabeli parametry. Zebranie tak dużej ilości próbek ma za zadanie odzwierciedlić jednorodny rozkład błędu kwantyzacji, ma to duże znaczenie, gdyż nierównomierny rozkład punktów powodować będzie błędy w obliczanych wartościach parametrów przetwornika. Jak możemy zaobserwować w tabeli 2.1, przy symulowaniu idealnych przetworników, parametry osiągane z symulacji są bardzo zbliżone do idealnych, jednak nie osiągają swoich idealnych wartości. Spowodowane jest to omawianym wcześniej niejednorodnie rozłożonym błędem kwantyzacji. W dwóch przypadkach w tabeli 2.1 parametry odbiegają jednak od wartości idealnych bardziej niż w pozostałych przypadkach, a przypadki te to ADC zbudowane ze stopni n+0.5-bitowych. Pojawiające się nieidealności nie wynikają z budowy całego ADC, natomiast z budowy jego ostatnego stopnia, gdyż biorąc pod uwagę, że wszystkie stopnie działają w idealny sposób, nie ma w tej chwili znaczenia czy używamy stopni n-bitowych, czy n+0.5-bitowych, ma jednak to znaczenie w ostatnim stopniu. Stopień n-bitowy zwraca nam bowiem informacje na n bitach i mamy wówczas 2^n poziomów kwantyzacji, stopień n+0.5-bitowy zwraca nam natomiast informacje na $n+1$ bitach, lecz w tym przypadku mamy tylko $2^{n+1} - 1$ poziomów kwantyzacji. Zmniejszenie o jeden liczby poziomów kwantyzacji i nie skorygowanie powstałego błędu w procesie korekcji cyfrowej (dla ostatniego stopnia nie ma możliwości przeprowadzenia takiej korekcji), powoduje powstawanie zwiększonego błędu kwantowania. Wspomniane wcześniej zwiększone odstępstwa od wartości idealnych dla dwóch przypadków, związane są właśnie z zastosowaniem n+0.5-bitowego ostatniego stopnia.

Tablica 2.1. Porównanie parametrów osiąganych na symulatorze idealnego 10-cio i 13-bitowego ADC, składającego się z rożnych konfiguracji stopni (9 x 1.5b/s - dziewięć stopni 1.5-bitowych; 1.5b/s-2b/ost - ostatni stopień 2-bitowy, pozostałe 1.5-bitowe)

	10-bitowe ADC				13-bitowe ADC	
	10 x 1b/s	5 x 2b/s	9 x 1.5b/s	9 x 1.5b/s-2b/ost	6 x 2.5b/s	6 x 2.5b/s-3b/ost
THD [dB]	-83.59	-83.70	-74.02	-83.61	-101.10	-109.58
SINAD [dB]	61.90	61.90	61.18	61.90	79.72	80.01
SNHR [dB]	61.93	61.93	61.41	61.93	79.75	80.01
SFDR [dB]	84.29	84.34	79.76	84.27	105.48	105.44
ENOB	9.99	9.99	9.87	9.99	12.95	13.00

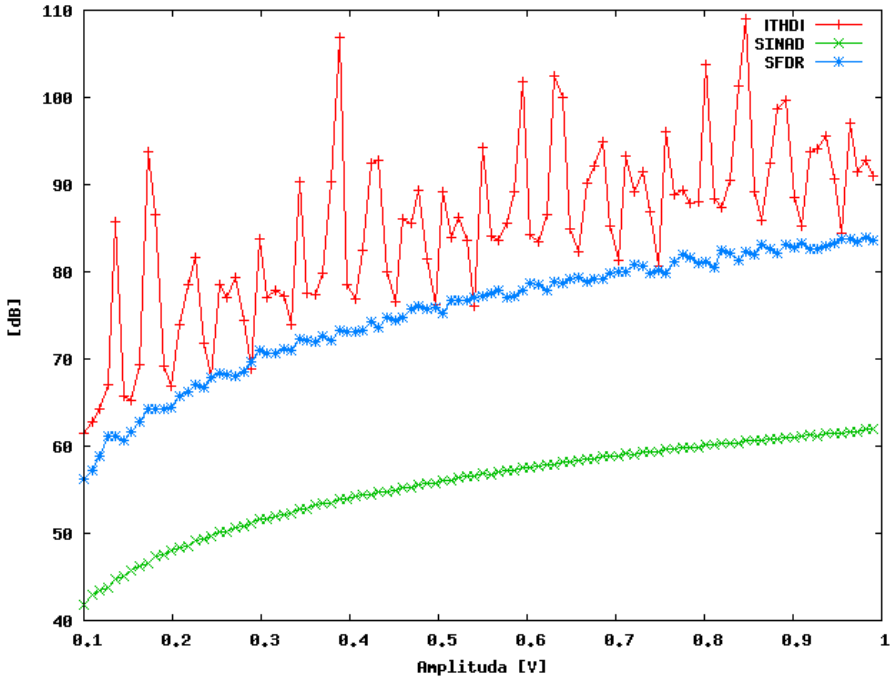
W rzeczywistych przetwornikach jednym z ważniejszych parametrów jest zależność jakości pracy od częstotliwości próbkowania, jak również częstotliwości sygnału wejściowego. W omawianym symulatorze nie uwzględniono jednak efektów związanych z szybkością pracy przetwornika, więc uzyskiwane wyniki nie są zależne od częstotliwości próbkowania i częstotliwości sygnału wejściowego.

2.2.1. Wpływ amplitudy sygnału wejściowego

Drugim z kolei testem było sprawdzenie zależności wpływu amplitudy sygnału wejściowego na parametry przetwornika. Na rysunku 2.1 zobaczyć możemy wykres obrazujący jak wpływa amplituda sygnału wejściowego na parametry 10-bitowego przetwornika ADC zbudowanego z 1-bitowych stopni. Jak można było się domyślać, zmniejszanie amplitudy powoduje pogorszenie parametrów przetwornika, gdyż szumy kwantyzacji pozostają na tym samym poziomie, natomiast zmniejsza się moc sygnału. Przedstawiony wykres został stworzony dla przetwornika zbudowanego z 1-bitowych stopni, jednak dla idealnych przetworników zbudowanych w innych konfiguracjach zależność ta jest taka sama, jedynie wartości mogą się nieznacznie różnić. Kolejne testy przeprowadzane były przy amplitudzie równej 1V, a wartości z tabeli 2.1 posłużyły jako idealne wartości, do których odnosił będę się rozpatrując model przetwornika uwzględniający szumy i nieidealności.

2.2.2. Wpływ wzmocnienia

Kolejną przeprowadzoną symulacją był wpływ wzmocnienia wzmacniacza, użytego w układzie mnożąco-odejmującym każdego ze stopni, na jakość pracy przetwornika. Idealny wzmacniacz operacyjny posiada nieskończone wzmocnienie napięciowe,

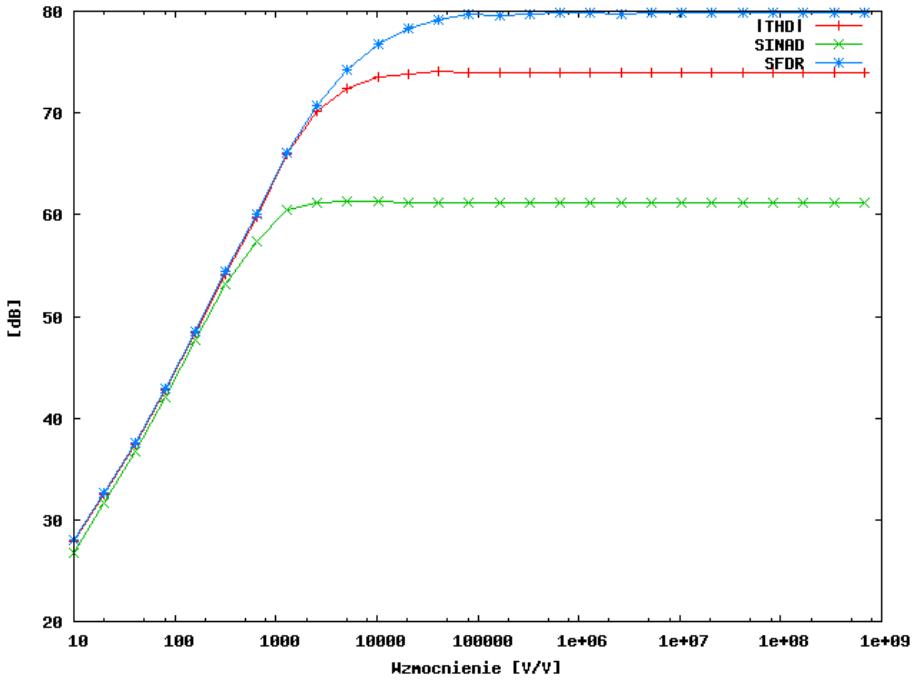


Rysunek 2.1. Wykres zależności parametrów idealnego 10-bitowego przetwornika ADC zbudowanego z 1-bitowych stopni od amplitudy sygnału wejściowego.

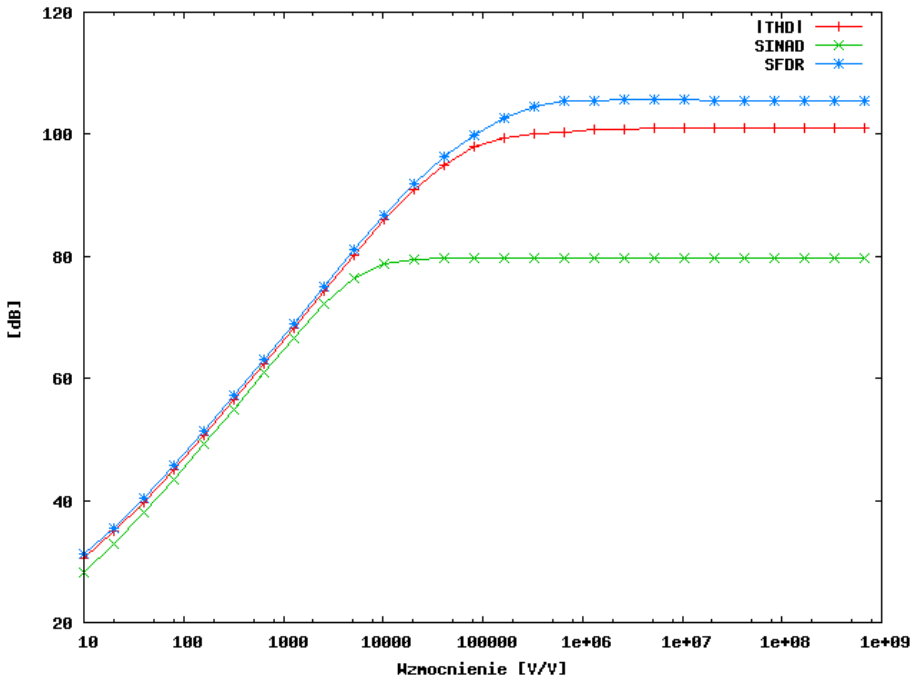
w rzeczywistych układach natomiast osiąga się wzmocnienia nawet do kilkudziesięciu milionów. Na rysunku 2.2 przedstawiony został wykres obrazujący zależność wartości parametrów osiąganych przez 10-bitowy przetwornik w zależności od wzmocnienia napięciowego wzmacniacza. Zaobserwować możemy, że przy wzmocnieniu mniejszym od około $3000V/V$ wartość parametrów jest tym niższa, im mniejsze jest wzmocnienie. Dla przetwornika 13-bitowego (rys. 2.3) natomiast, znaczne pogorszenie parametrów następuje poniżej wzmocnienia rzędu $30000V/V$. Z załączonych wykresów wynika więc, iż aby wzmocnienie wzmacniacza w potokowym przetworniku analogowo-cyfrowym nie wpływało na pogorszenie parametrów układu, musi być ono odpowiednio duże, a jego minimalna wielkość zależna jest od rozdzielczości przetwornika.

2.2.3. Wpływ pojemności

Następnym krokiem w badaniu wpływu nieidealności poszczególnych części przetwornika na jego działanie było sprawdzenie wpływu wielkości szumu kT/C na parametry osiągane przez przetwornik, poprzez zmianę pojemności w układzie mnożąco-odejmującym. Na rysunkach 2.4 i 2.5 zobaczyć możemy jak zmieniają się wartości najważniejszych parametrów przetwornika, przy zmianie pojemności w 10-bitowym



Rysunek 2.2. Wykres zależności parametrów idealnego 10-bitowego przetwornika ADC zbudowanego z 1.5-bitowych stopni od wzmocnienia wzmacniacza operacyjnego wbudowanego w każdy ze stopni.



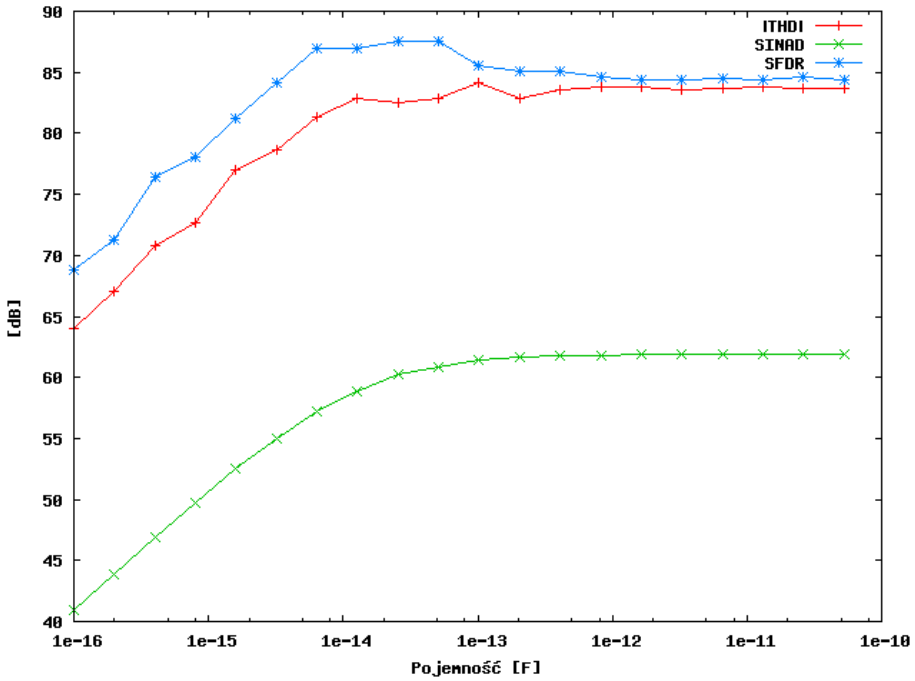
Rysunek 2.3. Wykres zależności parametrów idealnego 13-bitowego przetwornika ADC zbudowanego z 2.5-bitowych stopni od wzmocnienia wzmacniacza operacyjnego wbudowanego w każdy ze stopni.

Tablica 2.2. Zestawienie pojemności całkowitej C_c i parametru SINAD w układzie 10-bitowego potokowego ADC zbudowanego z 1.5-bitowych stopni, dla różnych czynników skalujących k i różnych pojemności $C_1 = C_2$ umieszczonych w układzie mnożąco-odejmującym pierwszego stopnia przetwornika

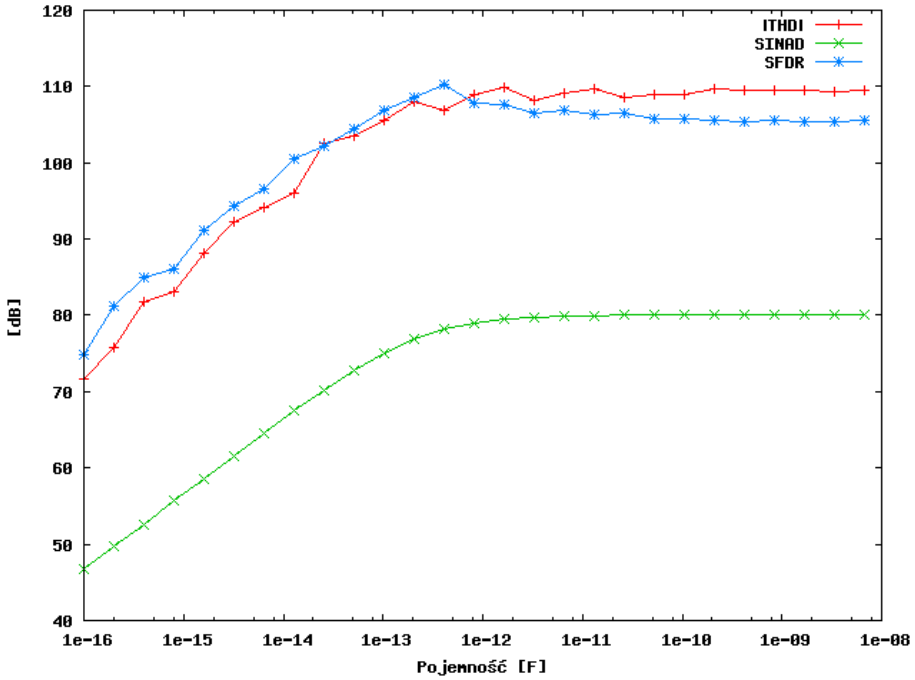
	$C_1 = C_2 = 0.05$pF		$C_1 = C_2 = 0.5$pF		$C_1 = C_2 = 5$pF	
k	C_c[pF]	SINAD[dB]	C_c[pF]	SINAD[dB]	C_c[pF]	SINAD[dB]
1	0.9	61.27	9	61.86	90	61.90
1/1.5	0.29	61.13	2.86	61.85	28.64	61.90
1/2	0.2	60.96	2	61.81	19.96	61.89
1/4	0.13	58.93	1.33	61.52	13.33	61.86
1/8	0.11	46.90	1.14	55.79	11.43	60.78

przetworniku zbudowanym z 1.5-bitowych stopni, jak również w 13-bitowym przetworniku zbudowanym z 2.5-bitowych stopni. Symulacje te przeprowadzane były dla temperatury $27°C$. Jak możemy zaobserwować, wraz ze wzrostem pojemności poprawiają się wartości wszystkich parametrów, zarówno w jednym jak i drugim przetworniku, co jest zgodne z równaniem 2.1. W rzeczywistych układach staramy się używać możliwie małych pojemności, ze względu na to, iż wraz ze wzrostem pojemności znacznie rośnie rozmiar układu, a co za tym idzie również jego cena. Wielkość pojemności wpływa na szybkość i czułość układu oraz pobieraną przez niego moc. Z omawianych wykresów odczytać możemy więc wartości pojemności, które będą kompromisem pomiędzy wielkością użytych pojemności, a dokładnością przetwornika. Dla 10-bitowego przetwornika przedstawionego na rysunku 2.4, kompromisem takim mogła by być pojemność wynosząca około $0.5pF$, gdyż dla pojemności większych od tej wartości, poprawa parametrów przetwornika jest bardzo nieznaczna lub nawet nie uzyskujemy żadnej poprawy. Stosowanie większych pojemności jest więc nie wskazane. Symulowany 13-bitowy układ wymaga stosowania nieco większych pojemności, aby nie pogarszały one jego parametrów. Najbardziej optymalną pojemnością dla takiego układu, powyżej której poprawa parametrów jest znikoma okazuje się pojemność w okolicy $5pF$. Zauważyć możemy, że wraz ze wzrostem rozdzielczości przetwornika istnieje konieczność stosowania większych pojemności, w celu utrzymania wymaganej dokładności układu.

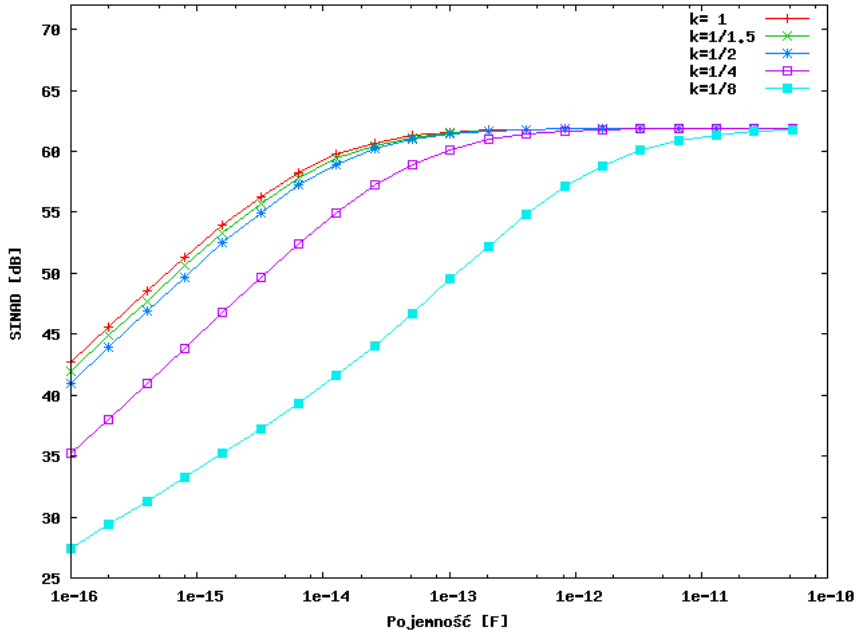
W celu zmniejszenia pojemności użytych w przetworniku, a co za tym idzie jego rozmiarów i zapotrzebowania na moc, stosuje się w przetwornikach potokowych skalowanie wielkości pojemności w kolejnych jego stopniach. Jak opisane było w poprzednim akapicie, zmniejszanie pojemności poniżej pewnej wartości, powoduje pogarsza-



Rysunek 2.4. Wykres zależności parametrów idealnego 10-bitowego przetwornika ADC zbudowanego z 1.5-bitowych stopni od wielkości pojemności użytych w układzie mnożąco-odejmującym, zakładając idealność wartości pojemności oraz temperaturę $27°C$.

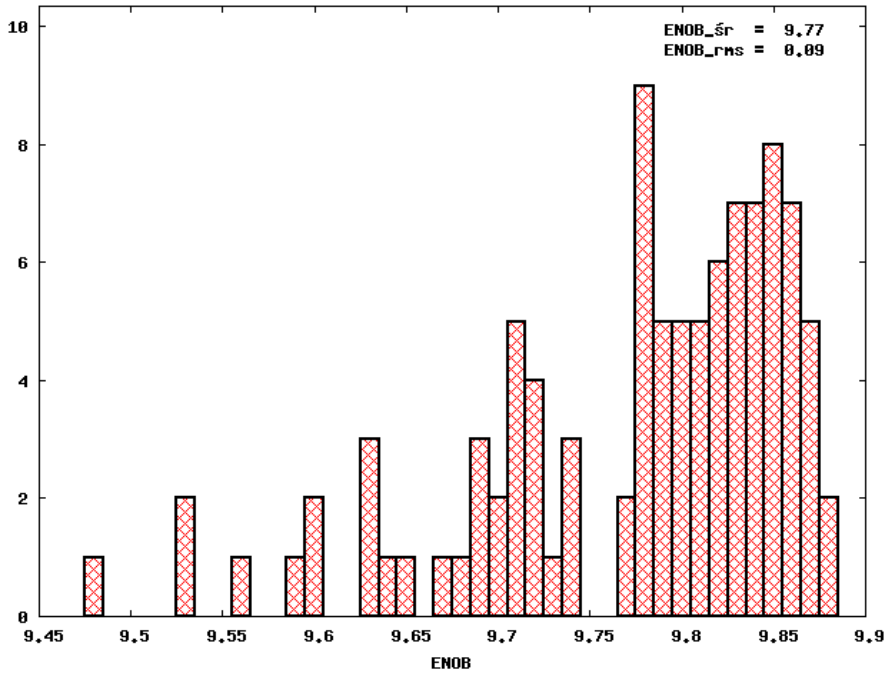


Rysunek 2.5. Wykres zależności parametrów idealnego 13-bitowego przetwornika ADC zbudowanego z 2.5-bitowych stopni od wielkości pojemności użytych w układzie mnożąco-odejmującym, zakładając idealność wartości pojemności oraz temperaturę $27°C$.



Rysunek 2.6. Wykres zależności parametru SINAD w 10-bitowym przetworniku o 1.5-bitowych stopniach, od wielkości pojemności $C_1 = C_2$ w układzie mnożąco-odejmującym pierwszego stopnia, dla różnych czynników skalujących k wielkość pojemności w kolejnych stopniach.

nie się parametrów przetwornika. Jeżeli jednak zmniejszania tego dokonujemy nie we wszystkich stopniach jednakowo, tylko w każdym kolejnym stopniu użyta pojemność jest coraz mniejsza, wówczas każdy kolejny stopień będzie pracował mniej dokładnie. Wpływ błędu z kolejnych stopni na prace całego przetwornika zmniejsza się jednak wraz ze wzrostem numeru stopnia. Dzięki takiej zależności zmniejszanie pojemności w mniej znaczących stopniach w nieznaczny sposób wpływa na dokładność przetwornika potokowego, natomiast zmniejsza jego rozmiary i pobór mocy. Na rysunku 2.6 przedstawione zostały wykresy obrazujące zależność parametru SINAD od wielkości pojemności użytych w pierwszym stopniu przetwornika, dla różnych czynników skalujących. Pojemność w każdym kolejnym stopniu, począwszy od drugiego, zmniejszana jest według wzoru $C_n = C_{n-1} \cdot k$ lub w innym zapisie $C_n = C_1 \cdot k^{n-1}$, gdzie n jest numerem stopnia przetwornika, a k czynnikiem skalującym. Pojemności w pierwszym stopniu nie są skalowane. Jak możemy zaobserwować na wykresie 2.6, symulacje potwierdziły pogarszanie parametrów przetwornika wraz ze wzrostem skalowania, pogarszanie to pojawia się jednak do osiągnięcia pewnej wartości pojemności, a następnie zanika. Możemy zauważyć, że przy małym skalowaniu, rzędu 1/1.5 lub 1/2 pojemność graniczna, po osiągnięciu której dalsze jej zwiększanie nie



Rysunek 2.7. Histogram 100 wartości ENOB wyliczonych w 10-bitowym przetworniku o 1.5-bitowych stopniach, w których rozrzut pojemności wynosił 0.1%.

poprawia pracy przetwornika, praktycznie pokrywa się z pojemnością przetwornika, w którym nie używano skalowania. Jeżeli użyjemy większego skalowania, rzędu np. 1/4 lub 1/8 wówczas po osiągnięciu pewnej pojemności, błąd wprowadzany przez skalowanie przestaje być istotny, wtedy jednak pojemność całkowita jest bardzo duża, czego staramy się uniknąć. Rozpatrzmy jednak całkowitą pojemność użytą w przetworniku dla różnych czynników skalujących i różnych pojemności (tab. 2.2). Jak możemy zobaczyć, nawet przy użyciu skalowania, zwiększanie pojemności w pierwszym stopniu, drastycznie zwiększa pojemność całkowitą C_c użytą w całym układzie. Dobierając jednak odpowiednio małą pojemność i odpowiednio mały współczynnik skalowania zmniejszamy znacznie pojemność całkowitą układu, bez pogarszania jego dokładności. W przeprowadzonych symulacjach optymalnymi parametrami jest wybór pojemności $C_1 = C_2 = 0.5$pF i współczynnika skalującego między 1/1.5 a 1/2. Dzięki takiemu wyborowi, kosztem minimalnych strat dokładności zmniejszamy od 3 do ponad 4 razy całkowitą pojemność przetwornika.

Kolejnym rozpatrywanym zjawiskiem mającym duży wpływ na prace przetwornika ADC, jest symulacja rozrzutu technologicznego pojemności. W symulacjach tych chodzi o zasymulowanie pojemności z rzeczywistego układu, gdyż wielkość pojemności w rzeczywistym układzie może się różnić od wielkości w projekcie o pewien

Tablica 2.3. Wpływ rozrzutu pojemności na parametr ENOB idealnych 10-cio i 13-bitowych ADC, składających się z rożnej rozdzielczości stopni (9 x 1.5b/s - dziewięć stopni 1.5-bitowych; 1.5b/s-2b/ost - ostatni stopień 2-bitowy, pozostałe 1.5-bitowe); wartości średnie (śr.) i odchylenie standardowe (rms) parametru ENOB wyliczane ze 100 symulacji

		10-bitowe ADC				13-bitowe ADC	
σ_C	ENOB	10 x 1b/s	5 x 2b/s	9 x 1.5b/s	9 x 1.5b/s-2b/ost	6 x 2.5b/s	6 x 2.5b/s-3b/ost
0.01%	śr.	9.97	9.98	9.85	9.97	12.66	12.70
	rms	0.00	0.01	0.01	0.00	0.02	0.02
0.1%	śr.	9.76	9.91	9.77	9.88	11.84	11.79
	rms	0.19	0.07	0.09	0.11	0.47	0.44
1%	śr.	7.48	8.34	8.13	8.15	8.70	8.71
	rms	0.72	0.58	0.62	0.60	0.61	0.55
10%	śr.	4.43	5.19	4.89	4.87	5.46	5.40
	rms	0.75	0.66	0.74	0.86	0.62	0.57

procent zależny od rozrzutów technologii. Sytuacja taka powoduje, że pojemności które powinny mieć dokładnie takie same wartości w rzeczywistości różnią się od siebie powodując błędy w działaniu przetwornika. Na rysunku 2.7 przedstawiono przykładowy histogram powstały ze 100 symulacji 10-bitowego przetwornika zbudowanego z 1.5-bitowych stopni, w których rozrzut pojemności wynosił 0.1%. Zauważyć możemy, że efektywna liczba bitów (ENOB), najczęściej wynosiła około 9.85, jednak średnia wartość wynosi 9.77, a odchylenie standardowe wynosi 0.09. Nawet przy tak małym rozrzucie pojemności mogą powstać tak duże błędy. W tabeli 2.3 zebrano wyniki symulacji analogicznych do powyższej, dla 10-cio i 13-bitowych przetworników o różnej konfiguracji oraz z różnym rozrzutem pojemności. Przy wzroście rozrzutu pojemności, wartość średnia efektywnej liczby bitów zmniejsza się coraz bardziej oraz zwiększa się odchylenie standardowe uzyskanych pomiarów, co świadczy o dużej rozpiętości uzyskiwanych wartości. Przy rozrzucie rzędu 0.01% wpływ niedopasowania pojemności na działanie przetworników jest dość mały, ze względu na małe różnice w wielkościach pojemności. Przy wzroście rozrzutu do 0.1% parametry przetwornika 10-bitowego pogarszają się, jednak w dość niewielkim stopniu, natomiast w przetworniku 13-bitowym pogorszenie parametrów jest kilkukrotnie większe niż w 10-bitowym. Jeżeli rozrzut pojemności przekracza jednak wartość 1%, wów-

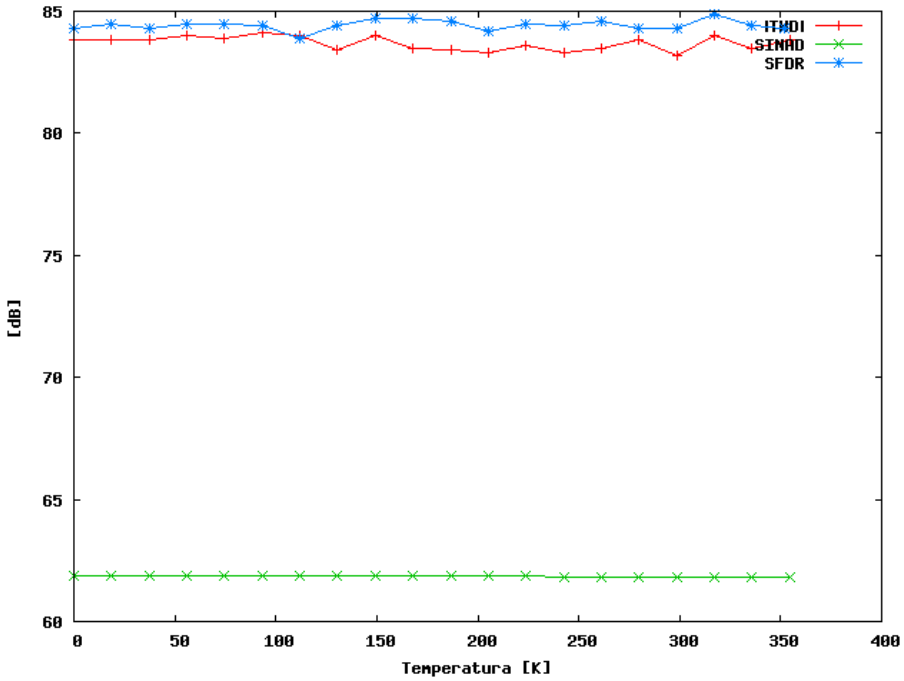
czas pogorszenie parametrów przetwornika zarówno 10-cio jak 13-bitowego jest duże i zwiększa się wraz ze wzrostem rozrzutu.

2.2.4. Wpływ temperatury

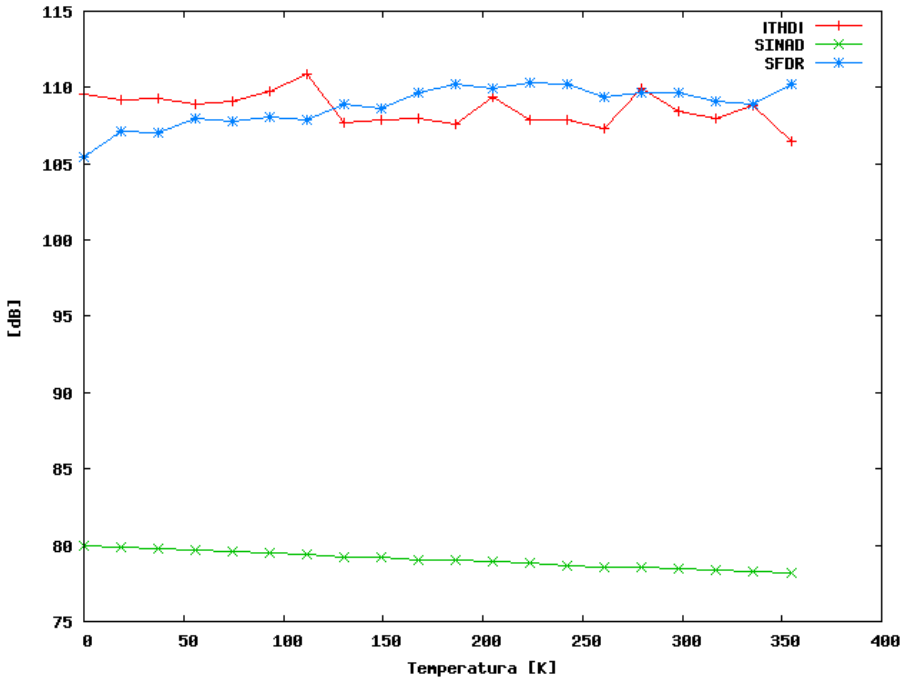
Na rysunkach 2.8 i 2.9 przedstawione zostały wykresy obrazujące zmiany parametrów 10-cio i 13-bitowego przetwornika od temperatury. Podczas symulacji wartość pojemności w układzie mnożąco-odejmującym wynosiła $0.5pF$ dla obydwu przetworników. Zależność temperatury, jak również wcześniej omawianej pojemności wiąże wzór 2.1, gdzie zauważyć możemy, iż wzrost temperatury wiąże się ze wzrostem szumu kT/C, co w efekcie powoduje pogorszenie parametrów przetwornika. Wielkość powstałego szumu zależna jest również od pojemności i wraz z jej wzrostem szum maleje. Na rysunkach 2.8 i 2.9 zauważyć możemy, iż dla 10-bitowego przetwornika zmiana temperatury od 0 do 350 K spowodowała niewielkie pogorszenie się jego parametru SINAD, pozostałe parametry oscylowały natomiast wokół swoich idealnych wartości. Dla 13-bitowego przetwornika pogorszenie parametru SINAD jest znacząco większe. Wartość szumu termicznego pojemności związana jest z pojemnością i temperaturą. Wiążąc rozważane wykresy z wykresami z rysunków 2.4 i 2.5 lub wzorem 2.1, zauważyć możemy, że gdyby w symulacjach użyta była większa pojemność, zmiana temperatury nie wpływałaby tak bardzo na działanie przetworników, gdyż wyższa temperatura skompensowana byłaby przez większą pojemność i szum kT/C pozostał by na tym samym poziomie. Ponadto zauważyć możemy, że użyta pojemność $0.5pF$, dla przetwornika 10-bitowego zmniejsza szum kT/C na tyle, że nie wpływa on już na pogorszenie parametrów tego przetwornika. Dla przetwornika 13-bitowego jednak ta sama wartość szumu kT/C jest na tyle duża, iż powoduje jeszcze lekkie pogorszenie jego parametrów, a zmniejszanie pojemności lub w tym przypadku zwiększanie temperatury powoduje wzrost szumu i spadek parametrów przetwornika.

2.2.5. Wpływ offsetu komparatorów

Bardzo duży wpływ na jakość pracy przetworników ADC ma również dokładność użytych w nich komparatorów. W tabeli 2.4 zebrane zostały wyniki symulacji parametru ENOB w zależności od wielkości offsetu komparatorów użytych w różnych stopniach 10-cio i 13-bitowych przetworników. Dla różnych przetworników wartość $\pm V_{th,of}$ była dodawana do idealnych wartości progów przełączania każdego z komparatorów w każdym ze stopni, a następnie na podstawie symulacji wyliczany był parametr ENOB takiego przetwornika. W symulacjach tych progi przełączania



Rysunek 2.8. Wykres zależności parametrów idealnego 10-bitowego przetwornika ADC zbudowanego z 1.5-bitowych stopni od temperatury.



Rysunek 2.9. Wykres zależności parametrów idealnego 13-bitowego przetwornika ADC zbudowanego z 2.5-bitowych stopni od temperatury.

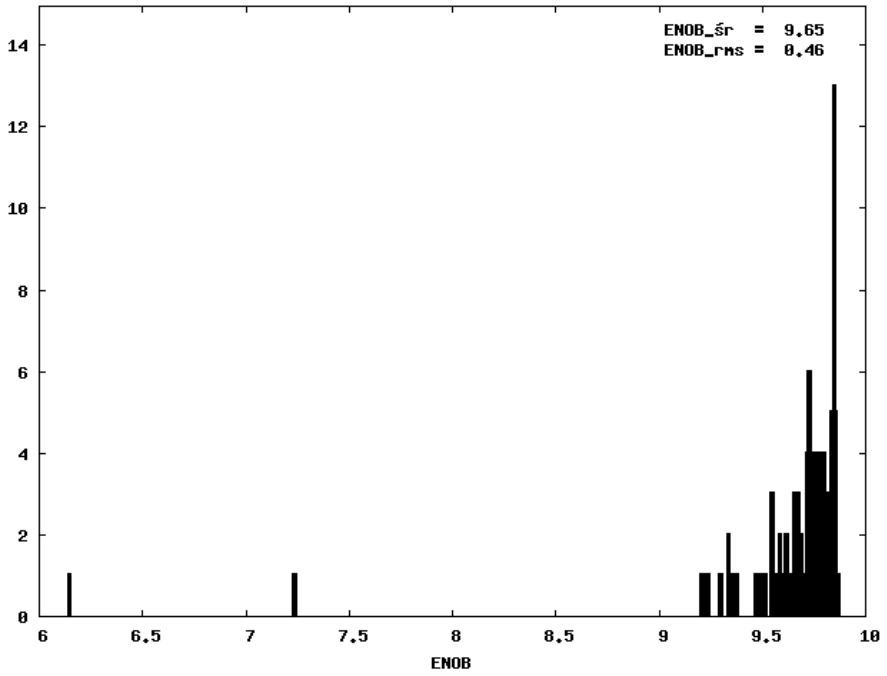
Tablica 2.4. Wpływ offsetu komparatorów na parametr ENOB idealnych 10-cio i 13-bitowych ADC, składających się z rożnej rozdzielczości stopni (9 x 1.5b/s - dziewięć stopni 1.5-bitowych; 1.5b/s-2b/ost - ostatni stopień 2-bitowy, pozostałe 1.5-bitowe)

	10-bitowe ADC				13-bitowe ADC	
$\pm V_{th,of}$ $[V]$	10 x 1b/s	5 x 2b/s	9 x 1.5b/s	9 x 1.5b/s-2b/ost	6 x 2.5b/s	6 x 2.5b/s-3b/ost
0	9.99	9.99	9.87	9.99	12.95	13.00
0.01	9.61	9.30	9.87	9.99	12.95	13.00
0.05	7.04	6.40	9.87	9.99	12.94	12.99
0.1	5.61	4.99	9.86	9.98	12.93	12.98
0.2	4.18	3.59	9.83	9.95	5.15	5.15
0.3	3.36	2.81	6.54	6.59	3.54	3.54

wszystkich komparatorów zostały więc przesunięte w górę lub w dół o pewną wartość. Przy omawianiu błędów związanych z offsetem komparatorów musimy powrócić do różnicy między przetwornikami potokowymi zbudowanymi z n i n+0.5-bitowych stopni. W stopniach n-bitowych dokładny komparator jest podstawą poprawnego działania przetwornika i nawet niewielkie jego niedokładności powodują powstawanie dużych błędów. W stopniach n+0.5-bitowych natomiast, niewielkie niedokładności komparatorów mogą zostać zniwelowane w procesie korekcji cyfrowej. Ważne jest, aby wartość offsetu w stopniach n+0.5-bitowych nie przekroczyła wartości:

$$\pm V_{th,of} = \frac{1}{2^{n+1}} \cdot V_{ref}, \tag{2.2}$$

gdyż poniżej tej wartości możliwe jest jej zniwelowanie, natomiast przekroczenie jej powoduje powstawanie błędów, których nie można zniwelować korekcją cyfrową. Jak możemy zauważyć wartość maksymalnego offsetu zależna jest tylko od rozdzielczości stopnia, a nie zależy w żaden sposób od rozdzielczości całego przetwornika. Analizując wartości zebrane w tabeli 2.4, zaobserwować możemy, że wyniki symulacji odzwierciedlają wiedzę teoretyczną. W przypadku stopni n-bitowych widzimy bowiem pogorszenie pracy przetwornika nawet przy bardzo małym offsecie komparatorów, a przy wzroście offsetu jakość pogarsza się w znaczący sposób. W przypadku stopni n+0.5-bitowych zauważyć możemy natomiast wyraźny próg, po przekroczeniu którego pogarsza się ich praca. Zgodnie ze wzorem 2.2, dla 1.5-bitowego stopnia i użytej w symulacji wartości $V_{ref} = 1V$, próg ten wynosi $\pm 0.25V$, natomiast dla stopnia 2.5-bitowego wynosi $\pm 0.125V$.



Rysunek 2.10. Histogram 100 wartości ENOB wyliczonych w 10-bitowym przetworniku o 1.5-bitowych stopniach, w których rozrzut offsetu komparatorów wynosił 0.1V.

Tablica 2.5. Wpływ rozrzutu offsetu komparatorów na parametr ENOB idealnych 10-cio i 13-bitowych ADC, składających się z rożnej rozdzielczości stopni (9 x 1.5b/s - dziewięć stopni 1.5-bitowych; 1.5b/s-2b/ost - ostatni stopień 2-bitowy, pozostałe 1.5-bitowe); wartości średnie (śr.) i odchylenie standardowe (rms) parametru ENOB wyliczane ze 100 symulacji

		10-bitowe ADC				13-bitowe ADC	
$\sigma_{V_{th,of}}$ $[V]$	ENOB	10 x 1b/s	5 x 2b/s	9 x 1.5b/s	9 x 1.5b/s-2b/ost	6 x 2.5b/s	6 x 2.5b/s-3b/ost
0.001	śr.	9.99	9.99	9.87	9.99	12.95	13.00
	rms	0.00	0.00	0.00	0.00	0.00	0.00
0.01	śr.	9.59	9.21	9.87	9.99	12.94	12.99
	rms	0.29	0.41	0.00	0.00	0.01	0.01
0.1	śr.	5.94	4.92	9.65	9.78	7.58	7.50
	rms	1.04	0.69	0.46	0.41	1.91	1.72
0.2	śr.	4.27	3.61	7.22	6.75	4.41	4.35
	rms	1.03	0.82	1.75	1.65	1.06	1.03

W pierwszym podejściu do zbadania zależności działania układu od offsetu komparatorów, offset dla wszystkich komparatorów był ustawiany na jakąś konkretną wartość, mniejszą lub większą i obserwowaliśmy jak wpływa to na parametry przetwornika. Drugim podejściem natomiast było przebadanie parametrów przetwornika, w którym offset komparatorów był różny dla każdego komparatora w każdym ze stopni. Wartość offsetu była losowana z rozkładu Gaussa o zadanym odchyleniu standardowym $\sigma_{V_{th,of}}$, dla każdego z komparatorów. Sytuacja taka powoduje jednak, iż poprawność działania przetwornika zależy tak naprawdę w bardzo dużym stopniu od losowania i największy wpływ na pracę przetwornika ma dokładność początkowych jego stopni. Na rysunku 2.10 przedstawiony został przykładowy histogram parametru ENOB, wyliczonego ze 100 symulacji 10-bitowego przetwornika potokowego zbudowanego z 1.5-bitowych stopni, w którym offsety komparatorów w każdej symulacji były losowane na nowo, a odchylenie standardowe losowanych offsetów wynosiło 0.1V. Jak widzimy zdecydowana większość wartości obliczanego parametru ENOB znajduje się w okolicach 9.65. Zdarzają się jednak wartości bardzo odsunięte od średniej, wynika to właśnie z faktu, iż przetwornik o tak kiepskim parametrze ENOB musiał mieć bardzo duży offset komparatorów w początkowych stopniach. W tabeli 2.5 zebrane zostały wyniki analogicznych symulacji, dla przetworników o różnej budowie i dla różnych wartości odchylenia standardowego offsetu. Analizując dane zebrane w tabeli, zaobserwować możemy, iż dla odchylenia rzędu 0.001V, żaden z przetworników nie pogorszył swojego działania. Przy wzroście odchylenia do 0.01V przetworniki n-bitowe pogorszyły już swoje działanie, natomiast n+0.5-bitowe nadal zachowują swoje parametry. Kolejne wzrosty odchylenia powodują, jak się można było spodziewać, jeszcze większe pogorszenie wyników.

Podsumowując przeprowadzone symulacje, zaobserwować możemy jak duży wpływ na poprawne działanie układu ma stworzenie projektu, który uwzględniał będzie wszystkie problemy pojawiające się w rzeczywistym układzie. Każda z przeprowadzonych symulacji rozpatrywała tylko jeden typ nieidealności układu, w rzeczywistości jednak wszystkie niedoskonałości układu wpływają jednocześnie na jego działanie. Symulacje miały jednak za zadanie zobrazować wpływ zmiany pojedynczych parametrów na działanie układu i zostało to zrealizowane. Dzięki takiemu symulatorowi mamy jednak możliwość wprowadzenia wszystkich powyższych nieidealności jednocześnie i zobaczyć wyniki symulacji takiego układu. Dzięki takiej możliwości, możemy do symulatora wprowadzić parametry rzeczywistego układu i porównywać ich działanie, jak również badać wpływ zmiany któregoś z parametrów na poprawę lub pogorszenie działania przetwornika. Napisany symulator jest więc pomocnym

narzędziem, pozwalającym w prosty sposób zobrazować zjawiska zachodzące w przetworniku analogowo-cyfrowym, jak również może okazać się pomocny, dla ilościowego określenia parametrów osiąganych przez przetwornik.

Rozdział 3

Porównanie oraz testy szybkości i dokładności symulatorów

W niniejszym rozdziale opisane zostały symulatory dostarczane przez firmy *Cadence Design System* i *Synopsys*, służące do symulowania działania analogowych układów elektronicznych. Na układzie 10-bitowego przetwornika ADC typu potokowego przedstawione zostały różnego rodzaju testy wykonywane na dostępnych symulatorach pod kątem szybkości działania, jak również dokładności wykonywanych przez nie obliczeń. Porównania i testy, miały na celu czytelne przedstawienie jakości i szybkości działania poszczególnych symulatorów, w celu łatwiejszego doboru symulatora do przeprowadzanej symulacji oraz wybranie najbardziej optymalnego do testowania wyżej wymienionego przetwornika ADC.

3.1. Przedstawienie i opis dostępnych symulatorów

W tej części opisane zostały testowane symulatory oraz ich wersje. Przedstawione zostały 4 symulatory:
— HSPICE należący do firmy *Synopsys*
— SPECTRE należący do firmy *Cadence Design Systems*
— APS należący do firmy *Cadence Design Systems*
— ULTRASIM należący do firmy *Cadence Design Systems*
Wszystkie symulatory udostępniają kilka rodzajów analiz możliwych do przeprowadzenia, tzn.: analizy czasowe, częstotliwościowe, stałoprądowe, szumowe. Wszystkie też symulują działanie układów analogowych, jednak posiadają szereg rożnych opcji, które pozwalają ustawiać jakość oraz szybkość ich pracy w wymaganym przez nas zakresie [10, 11, 12, 13].

3.1.1. Hspice

Pierwszym testowanym symulatorem był *HSPICE* firmy *SYNOPSYS* w wersji *C-2009.09-SP1 32-BIT* i *C-2009.09-SP1 64-BIT*. Podczas obliczeń symulator ten

używa równych kroków czasowych, których wartość zależna jest od użytych parametrów symulacji, a nie od dokładności obliczeń jak w przypadku innych symulatorów. Wielkość tych kroków ma znaczący wpływ na dokładność obliczeń, gdyż jeżeli krok okazuje się zbyt duży w stosunku do przeprowadzanej symulacji, wówczas powstają błędy obliczeń, które czynią symulacje niedokładną. Zmniejszanie kroków powoduje poprawę dokładności, jednak wzrasta zarazem liczba iteracji algorytmu, przez co wzrasta czas obliczeń [13]. Najbardziej znaczącymi parametrami wpływającymi na dokładność i zarazem długość analizy wykonywanej przez symulator, są:

— METHOD=GEAR|TRAP [PURETP]|BDF - ustawia algorytm numeryczny, przy pomocy którego wykonywane będą obliczenia; domyślna opcja to TRAP;
— RUNLVL= 0|1|2|3|4|5|6 - ustawia szybkość i dokładność symulacji; domyśla wartość to 3;
— KCLTEST=[0|1] - włącza sprawdzanie prawa Kirchhoffa, wydłużając przez to długość symulacji, zwiększamy jednak dzięki temu dokładność; domyślnie opcja wyłączona;
— INGOLD=[0|1|2]- ustawia format danych wyjściowych; domyślnie wartość 0 (format inżynierski, 1.23K, 123M);
— DELMAX=x - maksymalna długość kroku czasowego wykonywanego przez algorytm liczący; domyślnie wartość obliczana automatycznie;

Symulator ten udostępnia również parametr uruchomieniowy `-mt`, który umożliwia uruchomienie obliczeń na kilku procesorach, co znacznie skraca czas wykonywania symulacji. Do uruchomienia symulacji na wielu procesorach potrzebna jest jedna licencja na każde 2 procesory, wiec uruchamiając takie symulacje musimy wziąć pod uwagę ilość dostępnych licencji [13].

3.1.2. Spectre

Kolejnym testowanym symulatorem był *Cadence® Virtuoso® Spectre® Circuit Simulator* w wersji *7.1.1 32bit* i *64bit*. Kroki czasowe używane przez ten symulator, w przeciwieństwie do *HSPICE*, są zmieniane dynamicznie w trakcie symulacji, a dobór ich wielkość zależy od ustawionej tolerancji błędu obliczeń. Dzięki temu długość kroku w punktach czasowych, w których potrzebna jest większa dokładność jest mniejsza, natomiast w miejscach, gdzie nie potrzeba takiej dokładności kroki czasowe są dużo dłuższe. Zmniejsza to znacząco ilość iteracji algorytmu, a więc również czas obliczeń. Symulator ten udostępnia kilka opcji uruchomienia, umożliwiających zmianę dokładności obliczeń, a co za tym idzie ich szybkości. Umożliwia uruchomienie obliczeń na kilku procesorach, opcja analogiczna do *HSPICE* `+mt`, z tym że

w tym wypadku ilość procesorów ograniczona jest do 4 dla uruchomienia symulatora w normalnym trybie lub 8, kiedy uruchomimy tryb `+turbo`. „Multiprocesing" jest domyślnie wyłączony, z wyjątkiem opcji `+turbo`, gdyż po jej włączeniu zostaje również włączony „multiprocesing". Parametr `+turbo` przyspiesza działanie symulatora, a ponadto możemy ustawiać 3 różne jego tryby, dzięki którym zmieniamy dokładność i prędkość obliczeń:

— ***liberal*** - najszybszy a zarazem najmniej dokładny tryb, zalecany do schematów cyfrowych lub prostych analogowych. Dobrze sprawdza się we wstępnym testowaniu układów, a często daje wystarczającą dokładność, dla bardziej wymagających układów;

— ***moderate*** - dokładniejszy od poprzedniego, jest trybem domyślnym używanym przez symulator, powinien być używany jeżeli tryb *liberal* jest za mało dokładny;

— ***conservative*** - najbardziej dokładny a zarazem najwolniejszy z trybów dostępnych w symulatorze, przeznaczony dla bardzo czułych układów analogowych.

Wybór któregoś z powyższych trybów oznacza tak naprawdę ustawienie rożnych parametrów symulatora odpowiedzialnych za dokładność symulacji. Jeżeli tryb *conservative* nie daje nam wystarczającej dokładności możemy jeszcze ją zwiększyć, zmniejszając tolerancję błędu obliczeń, w naszym przypadku jest to parametr `reltol`. Symulacje układu powinno rozpoczynać się jednak od najszybszego trybu, sukcesywnie zwiększając dokładność w miarę potrzeby uzyskania dokładniejszych wyników.

Przy pomocy opcji `-64` przełączamy symulator na jego 64 bitową wersje, używanie jej jednak zalecane przez producenta jest jedynie dla bardzo dużych schematów (powyżej 200 tys. aktywnych elementów) lub dużych schematów z elementami pasożytniczymi.

Symulator ten posiada jeszcze jedną ciekawą opcję, która przydatna staje się podczas symulowania układów z elementami pasożytniczymi, a mianowicie opcja `+parasitics`. Parametr ten powoduje usunięcie części elementów pasożytniczych ze schematu, które w małym stopniu wpływają na jego działanie. Dzięki takiej operacji zmniejsza się wielkość układu, a co za tym idzie zmniejsza się czas jego symulacji [10, 14].

3.1.3. APS

Następny symulator, który poddany został testom to *Cadence® Virtuoso® Accelerated Parallel Simulator* w wersji *7.1.1 32bit* i *64bit*. Symulator ten jest w pełni kompatybilny ze *spectre*. Posiada takie same opcje uruchomienia, jak również parametry ustawiające dokładność obliczeń. Jest to jednak nowszy symulator, który

przeznaczony jest z założenia do bardzo dużych układów [10, 12]. Różnice jakie występują w odniesieniu do *spectre* to:

— nie posiada opcji `+turbo`, dokładność obliczeń ustawiamy za to analogicznie jak w *spectre*, tylko przy użyciu parametru `+errpreset` z wyborem trybu *liberal*, *moderate* lub *conservative*;
— domyślnie włączona jest obsługa wielu procesorów;
— liczba procesorów na ilu możemy wykonywać obliczenia wzrosła z 4 lub 8 dla *spectre* do 16 dla *APS*;
— w zależności od posiadanej licencji, używając jednej licencji możemy uruchomić obliczenia na 4 (`Virtuoso_Acceler_Parallel_L`) lub 8 (`Virtuoso_Acceler_Parallel_XL`) procesorach.

3.1.4. UltraSim

Ostatnim z dostępnych i testowanych symulatorów był *Cadence® Virtuoso® UltraSim® Full-Chip Simulator* w wersji *7.1.1 32bit* i *64bit*. Uruchomienie tego symulatora może odbyć się w trzech trybach zgodności. Pierwszy to tryb zgodności z *hspice*, wówczas symulator uruchamiamy z netlistą przygotowaną dla *hspice*. Drugi tryb to uruchomienie symulacji w trybie zgodności ze *spectre*. Musimy wówczas uruchomić symulator z opcją `-spectre` lub podawana jako parametr wejściowy netlista musi mieć rozszerzenie netlisty *spectre*, wtedy symulator sam wykrywa tryb zgodności. Trzeci tryb to możliwość uruchomienia symulacji z netlisty napisanej w języku Verilog. W tym przypadku, przed nazwą pliku z netlistą pojawić musi się opcja `-vlog`. Możliwe również jest uruchomienie symulacji w tak zwanym trybie mieszanym, to znaczy możemy w jednej netliście używać formatu zgodnego ze *spectre* i *hspice*, poprzedzonego odpowiednimi informacjami o tym jakiego języka aktualnie używamy, a symulator podczas wczytywania netlisty rozpozna język i wczyta odpowiednie dane potrzebne do symulacji. Uruchamianie tego symulatora w różnych trybach wiąże się jednak z różnymi czasami trwania takich symulacji. Tak jak poprzednio omawiane symulatory, tak i ten posiada wersje zarówno 32 jak i 64 bitową, jednak nie posiada on opcji pozwalającej uruchomić obliczenia na wielu procesorach. Jest to dość duży minus tego symulatora [11].

3.2. Symulowany układ

Układ poddawany symulacjom to 10-bitowy przetwornik analogowocyfrowy typu potokowego. Symulacje przeprowadzane były na samym schemacie układu, jak rów-

nież na jego schemacie z dołączonymi elementami pasożytniczymi. Układ bez elementów pasożytniczych składał się z około 7 tysięcy elementów, a plik netlisty dla tego schematu miał długość około 1000 linii. Po dołączeniu do schematu elementów pasożytniczych liczba jego elementów zwiększyła się do ok 150 tysięcy, natomiast plik netlisty rozrósł się do ponad 125 tysięcy linii.

Procesor na którym przeprowadzane były wszystkie symulacje to `Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz`. Komputer posiadał zainstalowany system operacyjny `Linux 2.6.26-2-amd64 (Debian2.6.26-21lenny4)`.

3.3. Przeprowadzone testy

Celem testów było przedstawienie dokładności i czasu przeprowadzania symulacji przez badane symulatory, jak również wpływu zmiany poszczególnych parametrów symulatorów na ich prace. W naszym przypadku symulatory były badane na tylko jednym schemacie, więc wyniki dokładności i czasu uzyskane w testach odnoszą się tak naprawdę tylko do tego konkretnego schematu. Ponieważ jednak testowane ADC typu potokowego jest bardzo dobrym przykładem zaawansowanego układu analogowo-cyfrowego, wyniki tych testów stanowią dobrą referencję dla całej klasy takich układów.

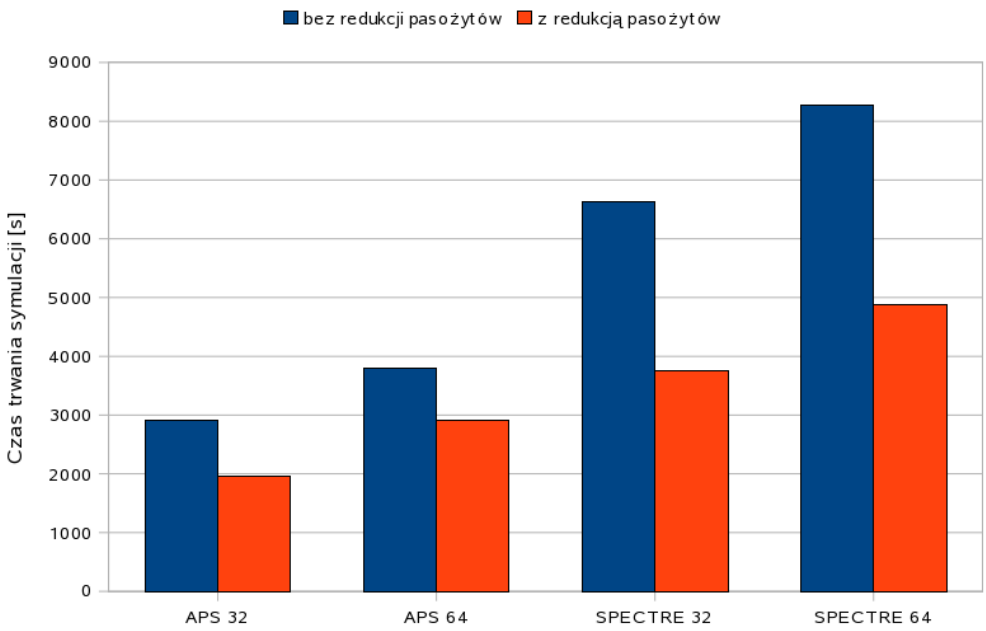
Testowanie wszystkich symulatorów polegało na przeprowadzaniu przy pomocy każdego z nich analiz czasowych schematu 10-bitowego przetwornika analogowo-cyfrowego oraz pomiarze czasu trwania każdej z symulacji. Na podstawie otrzymanych wyników określana była jakość pracy badanego przetwornika, poprzez wyliczenie podstawowych jego parametrów omówionych w podrozdziale 1.1. Głównym parametrem branym pod uwagę, był jednak stosunek sygnału do szumów i zniekształceń (SINAD), na podstawie którego określana była jakość pracy przetwornika. Na podstawie obliczanej wartości SINAD symulowanego przetwornika, określana była również dokładność działania symulatora. Odniesienie takie zastosowane zostało ze względu na to, iż do testowania symulatorów służył ten sam układ, a wzrost wartości SINAD przetwornika podczas zmiany parametrów symulatora mógł świadczyć o wzroście jego dokładności.

W pierwszym kroku w każdym z symulatorów testowane były opcje odpowiedzialne za dokładność i zarazem szybkość symulacji. Kolejnym krokiem było zbadanie szybkości działania symulatora na kilku procesorach. Następnie przetestowane zostały wersje 32 bitowe i 64 bitowe symulatorów. Ostatnim krokiem był wybór najlepszych opcji dla każdego z symulatorów i porównanie wszystkich ze sobą.

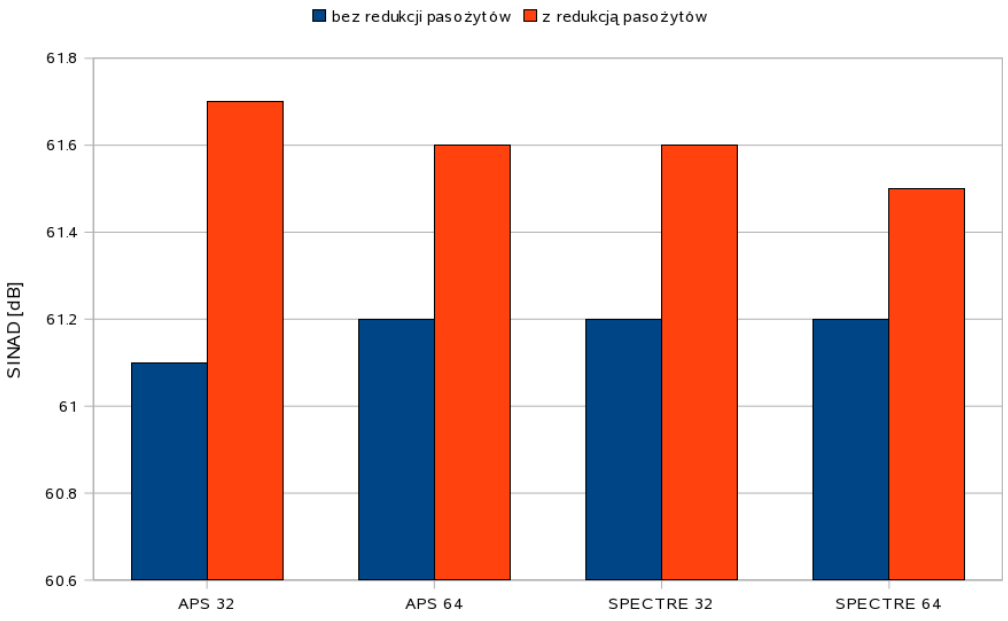
Przeprowadzone wstępne testy wykazały, że aby osiągnąć zadowalającą jakość wyników, opcje dostępne w pierwszym omawianym symulatorze, tzn. *hspice* ustawione musiały zostać na maksymalną dokładność tego symulatora, natomiast wielkość kroku czasowego musiała zostać ustawiona na wartość 100^{-12}. W celu poprawienia jeszcze dokładności obliczeń mamy możliwość jedynie zmniejszyć ustawiony krok czasowy. Przeprowadzone testy wykazały jednak, że zmniejszenie kroku do 10^{-12} wydłużyło czas obliczeń o niecałe 2 razy, natomiast zysk na dokładności był znikomy. Możemy na tej podstawie wnioskować, że zmniejszanie kroku czasowego ma sens jedynie do osiągnięcia wymaganej dokładności. Po jej osiągnięciu natomiast dalsze jego zmniejszanie faktycznie poprawia jeszcze otrzymywane wyniki, znacznie jednak wydłuża czas potrzebny na obliczenia, co jest ważnym czynnikiem podczas analizowania działania symulowanego układu.

Kolejne dwa omawiane symulatory: *spectre* i *APS*, ze względu na analogie dostępnych opcji i sposobu działania omówione zostaną razem. Symulatory te udostępniają 3 zdefiniowane tryby dokładności obliczeń. Podczas testowania dokładności tych symulatorów przetestowane zostały one wiec w tych 3 trybach. Jak się jednak okazało wyniki uzyskane w trybie najmniej dokładnym, nie poprawiały się w miarę zwiększania dokładności. Zwiększał się natomiast znacznie czas obliczeń. Tryb najmniej dokładny okazał się więc dla tych symulatorów najbardziej optymalny. Symulatory posiadają jeszcze opcję umożliwiającą redukcję elementów pasożytniczych. Wyniki symulacji przeprowadzonych na 4 procesorach, na układzie z elementami pasożytniczymi, z włączoną oraz wyłączoną opcją redukującą ich część, przedstawione są na rysunkach 3.1 i 3.2. Jak widzimy zysk czasowy z używania tej opcji wynosi od 1.3 do nawet 1.76 raza w stosunku do czasu symulacji bez redukcji elementów pasożytniczych. Jak jednak mogliśmy się spodziewać redukcja elementów pasożytniczych poprawia symulowane wyniki otrzymywane z przetwornika, co nie jest miarodajne, gdyż poprawa ta nie wynika z poprawy dokładności działania symulatora, a ze zmiany symulowanego schematu. Jeżeli jednak zależy nam na krótszym czasie symulacji, a dokładność symulacji jest zadowalająca, wówczas dobrze jest użyć powyższej opcji. Wziąć również pod uwagę możemy to, iż jak możemy zaobserwować na wykresie 3.6 SINAD wyliczony na podstawie wyników ze *spectre* bez redukcji pasożytów jest większy od obliczonego przez *hspice*, co sugeruje większą dokładność *spectre*.

Ostatni testowany symulator czyli *UltraSim*, możemy uruchamiać w 3 trybach zgodności oraz w trybie mieszanym. Przetestowane zostały tylko tryby zgodności ze *spectre* i *hspice* oraz tryb mieszany. Testy wykazały, że czas symulacji w trybie



Rysunek 3.1. Wykres porównujący czasy trwania symulacji schematu 10-bitowego ADC typu potokowego z elementami pasożytniczymi przez dwa symulatory z włączoną i wyłączoną opcją redukującą pasożyty. Symulatory uruchomione były na 4 procesorach.



Rysunek 3.2. Wykres porównujący SINAD schematu 10-bitowego ADC typu potokowego z elementami pasożytniczymi, obliczony z wyników otrzymanych przez dwa symulatory z włączoną i wyłączoną opcją redukującą pasożyty. Symulatory uruchomione były na 4 procesorach.

zgodności z *hspice* jest około dwukrotnie dłuższy niż symulacji w trybie zgodności ze *spectre*. Tryb mieszany natomiast jest zależny od sposobu w jaki będzie napisana netlista. W naszym przypadku długość symulacji w tym trybie bardzo nieznacznie różniła się od długości w trybie zgodności ze *spectre*.

Testowanym parametrem była również wersja symulatorów. Każdy z testowanych symulatorów dysponuje bowiem wersją zarówno 32 jak i 64 bitową. Wartość SINAD dla poszczególnych symulatorów w wersjach 32 i 64 bitowych nie różni się zbytnio, wiec z pewnym przybliżeniem możemy uznać wyniki otrzymywane z obydwu wersji za tak samo dokładne. Różnią się jednak zwykle czasy obliczeń pomiędzy poszczególnymi wersjami. Praktycznie zawsze zysk czasowy otrzymujemy korzystając z wersji 32 bitowej. Wyniki te zgadzają się z dokumentacją techniczną symulatorów, gdzie polecają stosowanie 64 bitowych wersji symulatorów tylko dla bardzo dużych schematów.

3.4. Porównanie

Porównanie omówionych w podrozdziale 3.1 symulatorów nie jest proste, gdyż porównujemy symulatory, które mają szereg różnych opcji zwiększających lub zmniejszających zarówno dokładność jak i czas obliczeń. W niniejszym zestawieniu, do porównania zostały wybrane symulatory z ich najbardziej optymalnymi ustawieniami, ze względu przede wszystkim na czas obliczeń, lecz przy zachowaniu rozsądnej ich dokładności określanej na podstawie przeprowadzanych testów. Poniżej omówione zostały opcje poszczególnych symulatorów z jakimi wykonywane był testy do porównania:

Hspice: najważniejsze parametry z jakimi został uruchamiany ten symulator, to: METHOD=GEAR (algorytm), RUNLVL=6 (maksymalna dokładność), KCLTEST=1 (włączone sprawdzanie prawa Kirchhoffa), INGOLD=2 (format danych wyjściowych), DELMAX ustawiony został na wartość 100^{-12};

Spectre: jak wynikało z testów, zwiększanie dokładności symulatora nie dawało żadnego odzwierciedlenia w otrzymywanych wynikach, więc do testów wybrane zostały najszybsze możliwe ustawienia, to znaczy włączone zostały opcje `+turbo=liberal`, jak również redukcja elementów pasożytniczych `+parasitics`;

APS: tak jak w *spectre* tak również tu użyte zostały najszybsze dostępne ustawienia, a mianowicie opcje: `+errpreset=liberal` i `+parasitics`;

UltraSim: uruchomiony został ze standardowymi opcjami, zwiększanie dokładno-

ści tego symulatora mogłoby przynieść jakieś efekty, jednak wydłużył by się na pewno czas symulacji.

Na wykresach 3.3 i 3.4 przedstawione jest porównanie czasu symulacji oraz otrzymanej wartości SINAD dla schematu bez elementów pasożytniczych przez 4 omawiane symulatory, w wersjach 32 i 64 bitowych na różnej liczbie procesorów. Dla wszystkich symulatorów, poza *UlraSim*, którego nie mamy możliwości uruchomić na kilku procesorach, wzrost liczby procesorów na których symulator wykonuje obliczenia implikuje spadek czasu obliczeń, z małym wyjątkiem, gdzie *spectre*, przy zwiększeniu liczby procesorów do 2 nieznacznie wydłużył obliczenia. Widzimy również różnice w czasach trwania symulacji przy użyciu poszczególnych symulatorów i jak łatwo zauważyć najwolniejszym okazał się *hspice* i był wolniejszy od pozostałych symulatorów od około 5 do ponad 9 razy. SINAD osiągany przez *hspice*, również był znacznie niższy niż osiągany przez *spectre* i *APS*. Sugeruje to, iż *hspice* oprócz znacznie wolniejszych symulacji daje również mniej dokładne wyniki. Najmniej dokładnym symulatorem okazał się jednak *UltraSim*. Pomimo czasu obliczeń krótszego od pozostałych symulatorów uruchamianych na jednym procesorze, wyniki jakie uzyskiwane były przy pomocy tego symulatora nie były wystarczająco dokładne, aby można było go używać do symulowania działania układów o podobnym stopniu skomplikowania co testowany. Pozostałe 2 symulatory, tzn. *spectre* i *APS* okazują się wiec najbardziej optymalnymi, z tym że między nimi również możemy zauważyć nieznaczne różnice. Jeżeli chodzi o dokładność to różnice między nimi są dość niewielkie, wiec możemy przyjąć, że ich dokładność jest na podobnym poziomie. Różnią się natomiast trochę czasy trwania symulacji, gdyż *APS* przyspiesza w stosunku do *spectre* od około 1.4, do około 1.6 razy, nie wliczając przypadku uruchomienia symulatorów na jednym procesorze, gdzie *spectre* okazuje się nieznacznie szybszy.

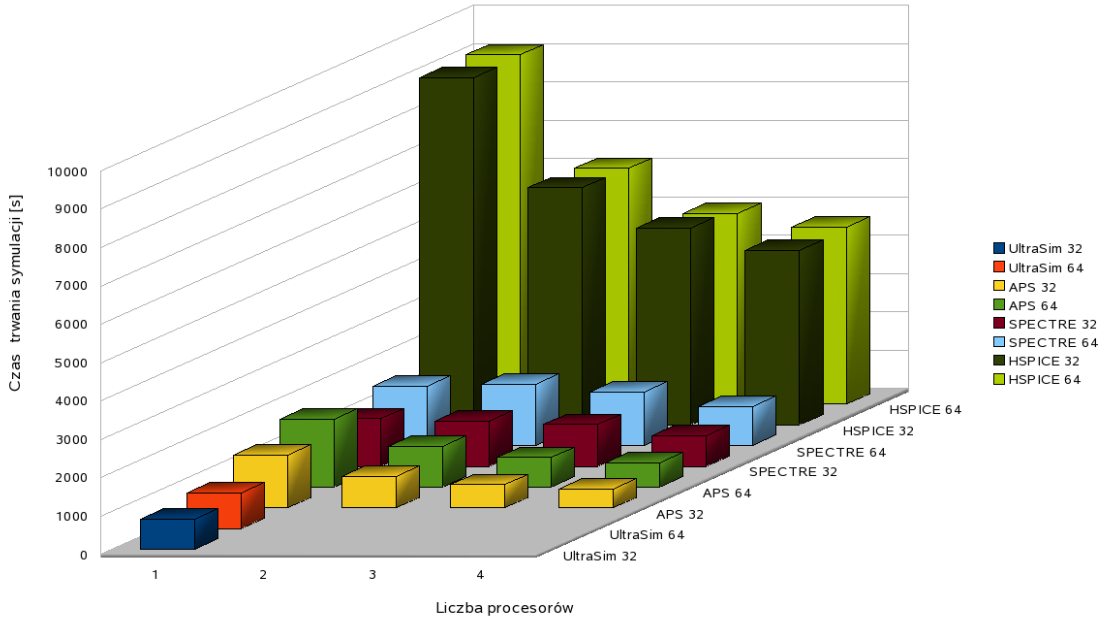
Testom poddany został również układ z pojemnościami pasożytniczymi. Podczas testów na układzie z elementami pasożytniczymi wszystkie symulatory za wyjątkiem *UltraSim* uruchamiane były na 4 procesorach. Wykresy przedstawiające zestawienie czasów symulacji jak również wartości SINAD dla takiego układu przedstawione zostały na rysunkach 3.5 i 3.6. W tym wypadku sytuacja wygląda podobnie jak przy układzie bez elementów pasożytniczych. Czas symulacji układu z elementami pasożytniczymi wzrasta jednak kilku lub nawet kilkunastu krotnie dla każdego z symulatorów. Tak jak w poprzednim przypadku, najwolniejszym symulatorem okazał się również *hspice*, a najszybszym *APS*. Wartości SINAD dla wszystkich symulatorów zmieniły nieznacznie wartości, jednak związane to jest z uwzględnieniem elementów pasożytniczych w układzie. Symulator *UltraSim* nie był dłużej studiowany, gdyż na-

Tablica 3.1. Zalecane użycie symulatorów APS i Spectre w zależności od wielkości symulowanego układu (1 pr. - symulator uruchomiony na jednym procesorze) [12]

Liczba elementów układu	Spectre	Spectre Turbo		APS				
		1 pr.	4 pr.	1 pr.	2 pr.	4 pr.	8 pr.	16 pr.
<100	✓	✓						
<5K		✓	✓					
<50K		✓	✓			✓	✓	✓
>50K				✓	✓	✓	✓	✓

wet jeżeli udało by się precyzyjniej ustawić opcje, kosztem czasu symulacji, to i tak nie jest możliwe użycie więcej niż jednego procesora. Początkowo pozornie najszybszy z symulatorów, jest zarazem najmniej dokładny i traci pozycje najszybszego, gdy porównamy go z pozostałymi symulatorami uruchomionymi na wielu procesorach.

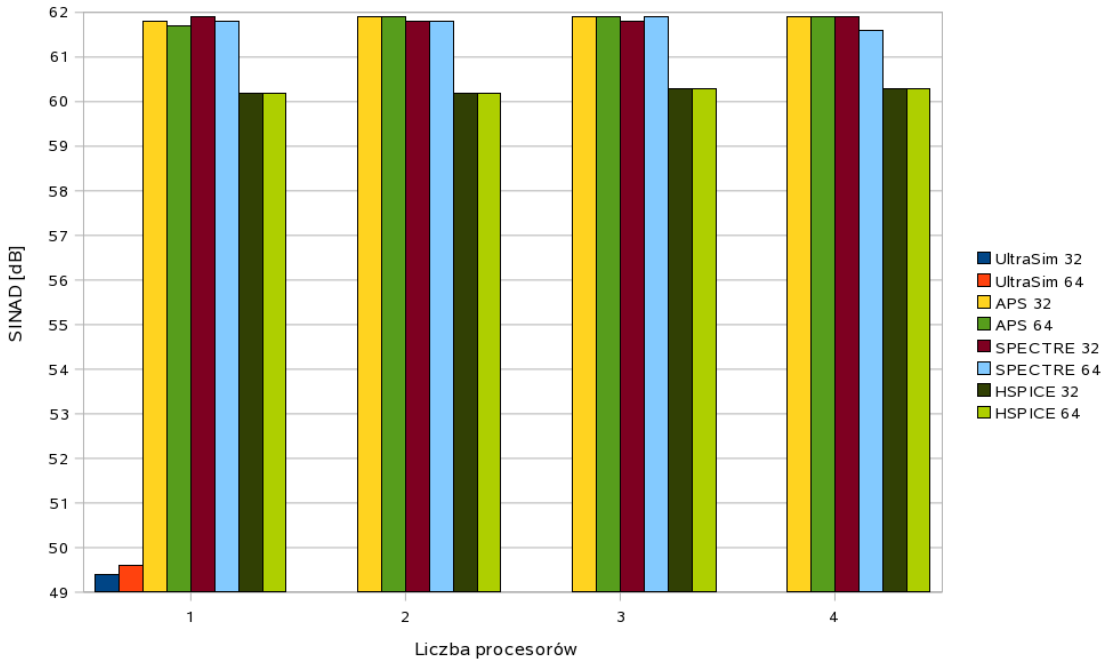
Podsumowując zebrane informacje, za najbardziej optymalny pod względem zarówno czasu jak i dokładności obliczeń, dla badanego układu przetwornika ADC okazał się *APS* w wersji 32 bitowej. Testy przy użyciu tego symulatora możemy uruchamiać aż na 16 procesorach jednocześnie, co przy bardzo dużych schematach bardzo skraca czas potrzebny na obliczenia. Używanie jednak zbyt wielu procesorów dla małych układów, jak również nieodpowiedniego symulatora, może nie skutkować przyspieszeniem symulacji, a jej spowolnieniem. W tabeli 3.1 przedstawione zostało rekomendowane przez producenta użycie symulatorów *APS* i *spectre*, czyli dwóch które najlepiej spisały się w powyższych testach, w zależności od wielkości symulowanego układu.



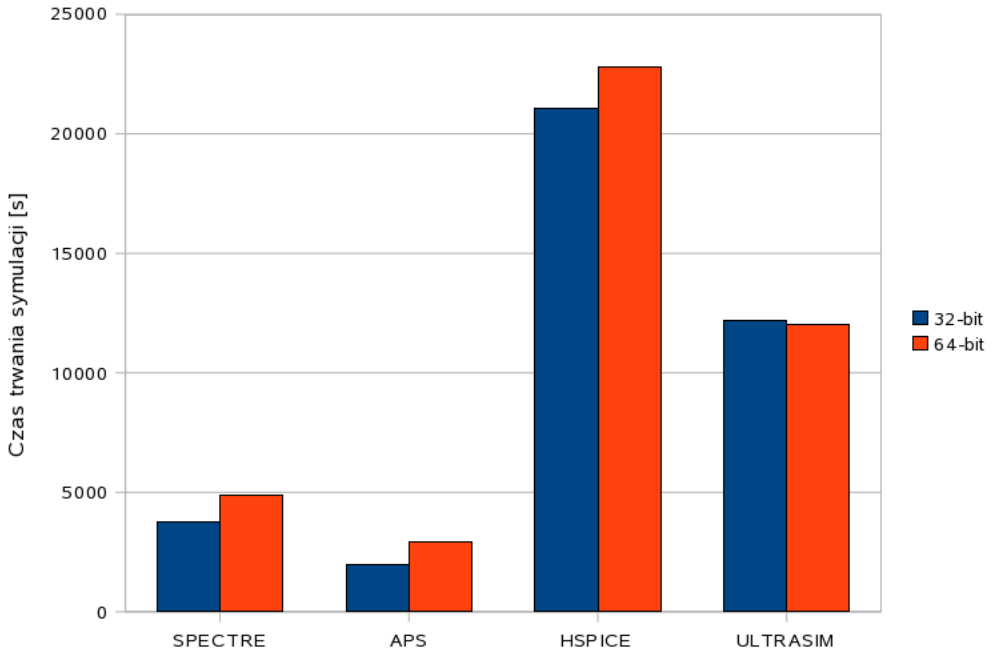
Rysunek 3.3. Wykres porównujący czasy trwania symulacji układu 10-bitowego ADC typu potokowego bez elementów pasożytniczych przez rożne symulatory.

Tablica 3.2. Porównanie czasów trwania symulacji układu 10-bitowego ADC typu potokowego bez elementów pasożytniczych przez rożne symulatory, czas wyrażony jest w sekundach

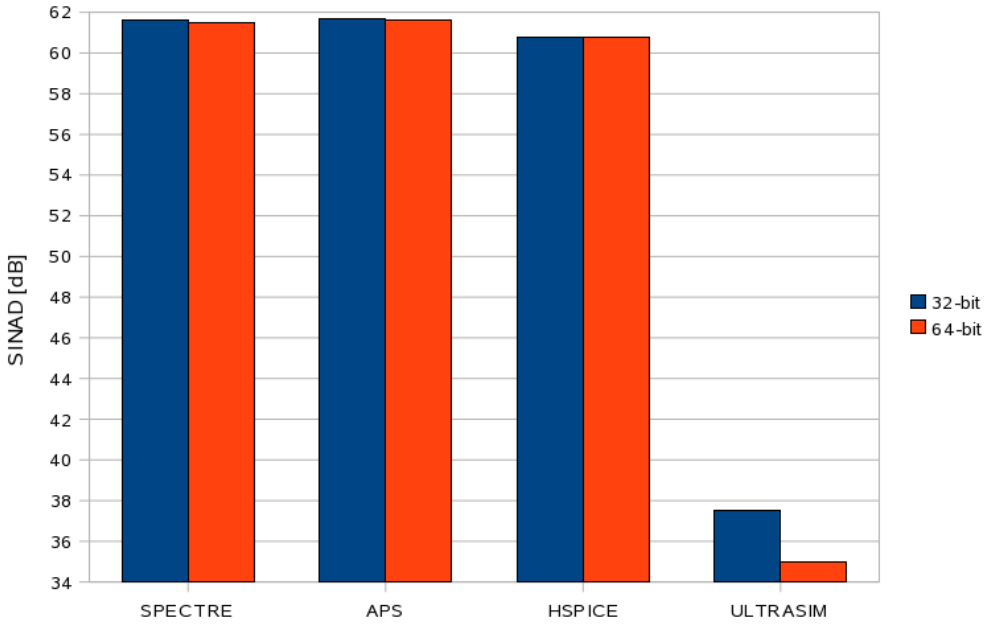
Symulatory	Liczba procesorów			
	1	2	3	4
UltraSim 32	794	-	-	-
UltraSim 64	922	-	-	-
APS 32	1370	817	601	495
APS 64	1770	1070	775	640
SPECTRE 32	1240	1180	1090	807
SPECTRE 64	1550	1600	1400	1020
HSPICE 32	9056	6184	5126	4544
HSPICE 64	9108	6152	4976	4624



Rysunek 3.4. Wykres porównujący SINAD układu 10-bitowego ADC typu potokowego bez elementów pasożytniczych, obliczony z wyników otrzymanych przez rożne symulatory.



Rysunek 3.5. Wykres porównujący czasy trwania symulacji układu 10-bitowego ADC typu potokowego z elementami pasożytniczymi przez rożne symulatory. Symulatory uruchomione były na 4 procesorach, z wyjątkiem ULTRASIM, który umożliwia wykorzystanie tylko jednego procesora.



Rysunek 3.6. Wykres porównujący SINAD układu 10-bitowego ADC typu potokowego z elementami pasożytniczymi, obliczony z wyników otrzymanych przez rożne symulatory. Symulatory uruchomione były na 4 procesorach, z wyjątkiem ULTRASIM, który umożliwia wykorzystanie tylko jednego procesora.

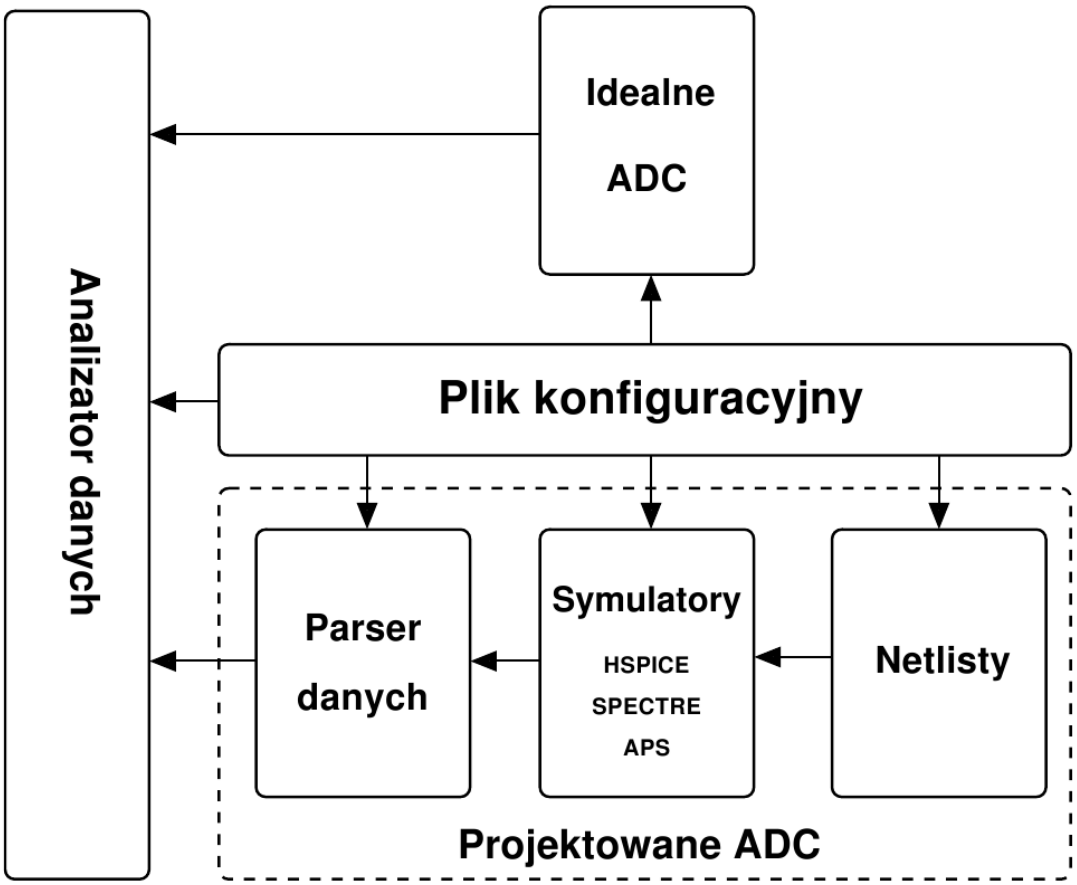
Tablica 3.3. Porównanie czasów trwania symulacji układu 10-bitowego ADC typu potokowego z elementami pasożytniczymi przez rożne symulatory, czas wyrażony jest w sekundach

Symulatory	Liczba procesorów	
	1	4
UltraSim 32	12212	-
UltraSim 64	12038	-
APS 32	-	1950
APS 64	-	2910
SPECTRE 32	-	3750
SPECTRE 64	-	4870
HSPICE 32	-	21064
HSPICE 64	-	22815

Rozdział 4

System do automatyzacji symulacji i przetwarzania ich wyników

W rozdziale tym opisany został stworzony przez autora system pozwalający na uproszczenie i zautomatyzowanie procesu symulowania projektu układu potokowego przetwornika analogowo-cyfrowego. Proces tworzenia projektu jakiegokolwiek układu scalonego składa się z kilku etapów, pierwszy z nich to stworzenie schematu układu i sprawdzenie poprawności jego działania. Jeżeli schemat jest poprawny wówczas tworzony jest dla tego układu „layout", czyli rozrysowywane jest rozłożenie wszystkich masek technologicznych na krzemie. Na podstawie „layoutu" wyliczane są efekty pasożytnicze powstałe w układzie, które dodajemy do schematu jako dodatkowe elementy, w celu zbadania pracy układu z tymi elementami. Jeżeli układ nadal pracuje poprawnie, wówczas wykonujemy analizę Monte Carlo, w której symulujemy rozrzuty technologiczne elementów układu, gdyż w zależności od technologii w jakiej wykonujemy układ, każdy element ma jakąś określoną dokładność. Ostatnim krokiem w testowaniu układu jest przeprowadzenie symulacji „kornerowych", czyli przetestowanie działania układu w najgorszych przypadkach (ang. Wost Case) niedokładności elementów w danej technologii. Dzięki takim symulacjom sprawdzamy czy w najgorszym możliwym przypadku produkcyjnym nasz układ nadal działałby poprawnie. Jeżeli układ pracuje poprawnie i parametry osiągane przez niego, zgodne są z założeniami projektowymi, wówczas układ wysyłany jest do produkcji, w przeciwnym przypadku musimy poprawić schemat lub „layout" i powtórzyć proces symulacji. Dzięki możliwości przeprowadzania wszystkich opisanych symulacji, zabezpieczamy się przed wysłaniem do produkcji nie działającego układu, a ze względu na to, iż proces produkcji jest drogi i czasochłonny, przeprowadzane symulacje są bardzo ważną częścią tworzenia układu. Dotychczas w celu przeprowadzenia symulacji działania układu trzeba było wykonać szereg operacji, które prowadziły do otrzymania potrzebnych nam danych mówiących o jakości pracy przetwornika. W czasie całego procesu wykonuje się wiele takich symulacji w których wykonujemy praktycznie te



Rysunek 4.1. Schemat blokowy systemu automatycznych symulacji.

same operacje za każdym razem. Dlatego właśnie powstał system upraszczający cały proces symulacji do jednego kroku.

System automatycznych symulacji został napisany w języku skryptowym *Python*, jako zbiór kilku skryptów, o różnej funkcjonalności. Ponieważ podczas przeprowadzania całego procesu symulacji istnieje czasem konieczność powtórzenia tylko kilku lub jednego z kroków procesu, dlatego właśnie cały system składa się z bloków, w których każdy spełnia określoną funkcję i może być używany niezależnie. Schemat blokowy całego systemu możemy zobaczyć na rysunku 4.1. System automatycznych symulacji składa się z 5 skryptów, do których dołączyć musimy jeszcze pliki z odpowiednio przygotowanymi netlistami, zawierającymi opis symulowanego układu w odpowiednim formacie zależnym od używanego symulatora. Wspomniane wcześniej skrypty to:

— **adc_config.py** - skrypt konfiguracyjny, najważniejszy zbiór, w którym ustawiane

są wszystkie parametry symulacji, takie jak rodzaj symulatora oraz rodzaje analizy do wykonania;

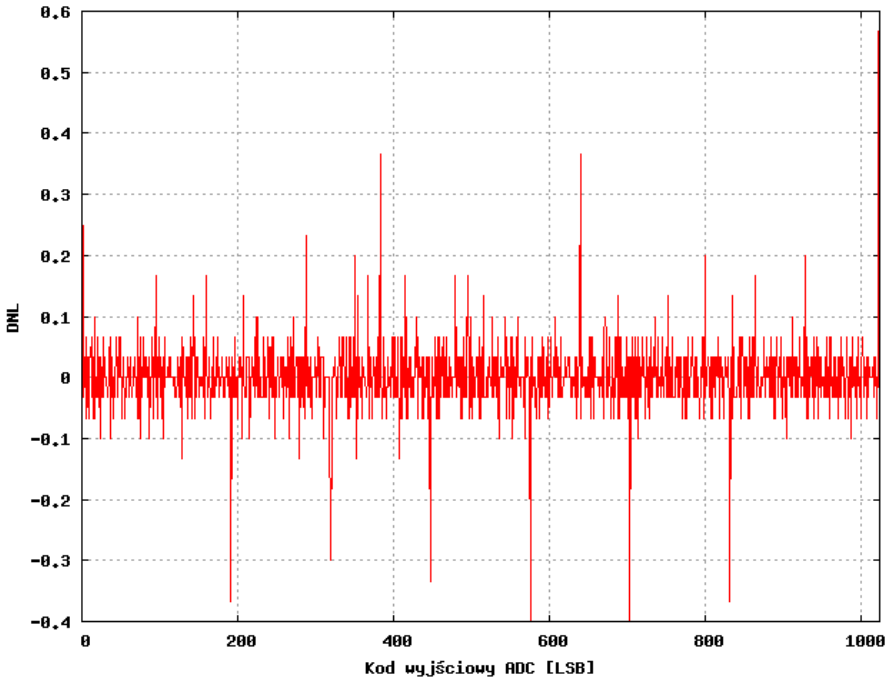
— **sym.py** - główny skrypt zarządzający całym procesem symulacji, przy jego pomocy wykonywane zostają wszystkie niezbędne w danej analizie kroki w celu uzyskania gotowego wyniku;

— **ideal_adc.py** - skrypt symulujący działanie idealnego potokowego przetwornika analogowo-cyfrowego (rozdział 2), z możliwością symulacji niektórych rzeczywistych efektów zachodzących w układzie;

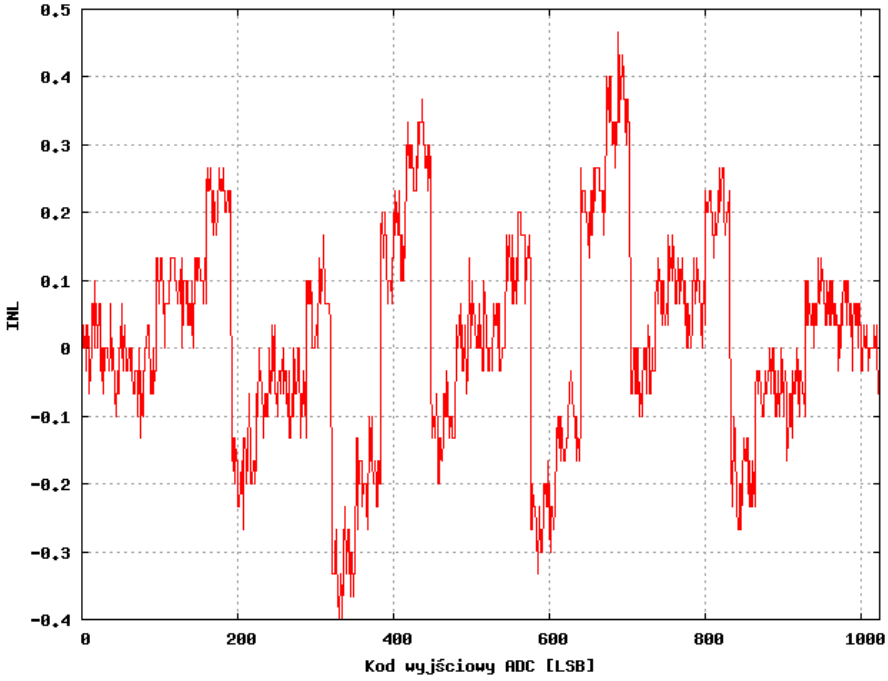
— **data_analysis.py** - skrypt ten służy do zanalizowania danych wyjściowych z przetwornika oraz wyliczenia jego parametrów, jak również zobrazowania wyników analiz na odpowiednich wykresach. W procesie analizy skrypt korzysta z szybkiej transformaty Fouriera;

— **data_parse.py** - zadaniem skryptu jest sparsowanie danych otrzymanych z symulatorów: *HSPICE*, *SPECTRE*, *APS*, aby możliwa była ich dalsza analiza;

Przy pomocy omawianego system, możemy przeprowadzać w prosty sposób szereg różnych symulacji. System oferuje dwie główne funkcjonalności, które możemy podzielić na: symulacje idealnego ADC oraz symulacje rzeczywistego projektu przetwornika. Podczas symulacji idealnego ADC mamy do dyspozycji przeprowadzenie analizy statycznej, jak również dynamicznej przetwornika, ponadto możemy zmieniać w łatwy sposób budowę przetwornika, jego rozdzielczość, jak również wiele parametrów, takich jak offset progów komparatorów, wartość pojemności, temperaturę pracy, wzmocnienie wzmacniacza, rozrzut pojemności i progów przełączania komparatorów. Wszystkie parametry dostępne w idealnym przetworniku omówione dokładniej zostały w rozdziale 2. W analizie statycznej mamy możliwość zmiany dokładności analizy poprzez zmianę liczby punktów pomiarowych. Wynikiem tej analizy są wykresy nieliniowości całkowej i różniczkowej (rysunek 4.2) badanego przetwornika. Wynikiem analizy dynamicznej jest wykres widma Fouriera sygnału próbkowanego przez badany przetwornik oraz wyliczone parametry dynamiczne przetwornika, umieszczone na wykresie. W przypadku analizy dynamicznej mamy ponadto możliwość przeprowadzenia dwóch dodatkowych serii pomiarów, jest to tak zwany „skan" i analiza Monte Carlo. Pierwsza seria pomiarów, czyli wspomniany „skan", to nic innego jak określona liczba analiz, w naszym przypadku dynamicznych, przetwornika w którym zmieniany jest jakiś z parametrów, dlatego właśnie „skan" przeprowadza się po jakimś parametrze. W przypadku omawianego skryptu analizę taką możemy wykonywać po każdym z dostępnych parametrów. Wynikiem jaki uzyskujemy z takiej analizy jest wykres zależności parametrów dynamicznych



(a) różniczkowa

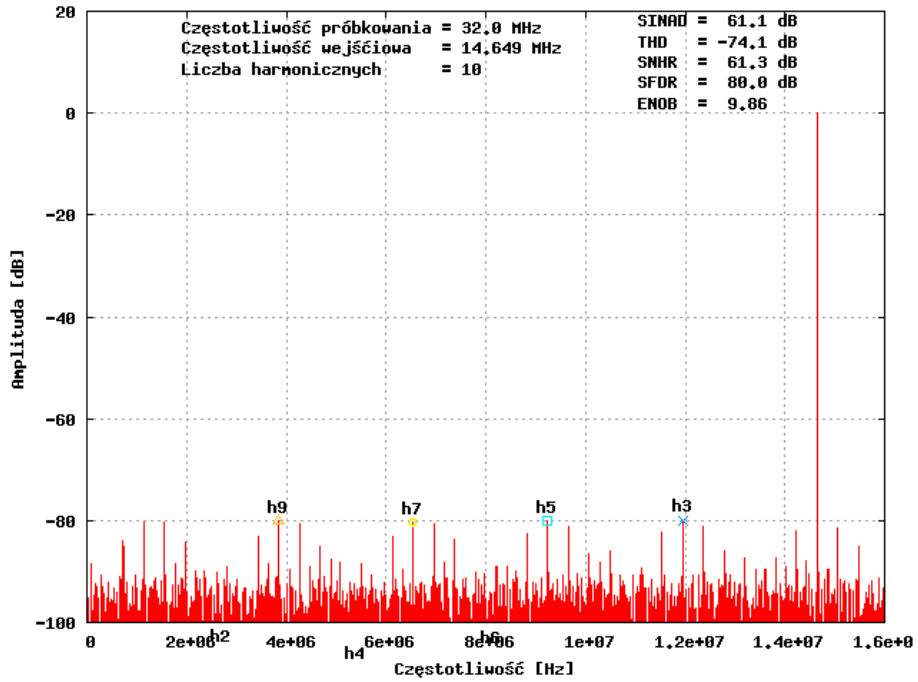


(b) całkowa

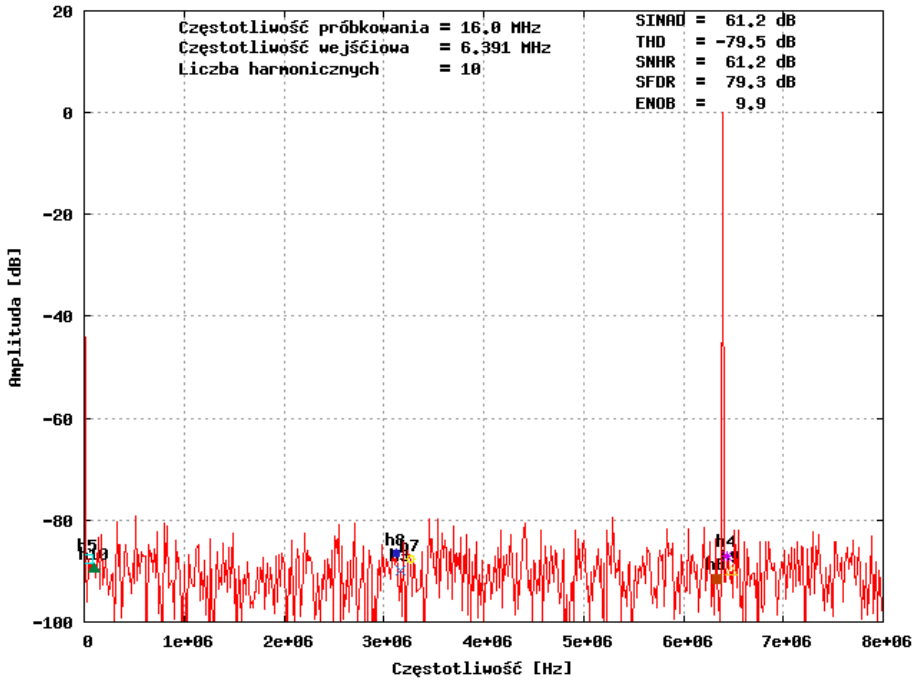
Rysunek 4.2. Przykładowe wykresy nieliniowości symulowanego 10-bitowego przetwornika analogowo-cyfrowego zbudowanego z 1.5-bitowych stopni.

przetwornika, od wartości zmienianego parametru. Analiza Monte Carlo to również określona liczba analiz, w naszym przypadku dynamicznych. W tej analizie nie zmieniamy wielkości parametrów, ponieważ w analizie tej chodzi o zbadanie rozkładu uzyskiwanych parametrów dynamicznych, które zmieniają się ze względu na rozrzuty technologiczne użytych elementów. Wynikiem uzyskiwanym z przeprowadzenia takiej analizy są histogramy parametrów dynamicznych przetwornika, z wyliczoną wartością średnią i odchyleniem standardowym każdego z parametrów.

Symulując rzeczywisty projekt przetwornika możemy wybrać jeden z trzech symulatorów: Hspice, Spectre lub APS. Testy prędkości i dokładności wymienionych symulatorów umieszczone zostały w rozdziale 3. Oprócz wyboru symulatora możemy wybrać również jego wersję 32 lub 64 bitową, jak również jego dokładność (dotyczy Spectre i APS). Wybierając jakiś symulator musimy dołączyć netlistę do konkretnego symulatora, w której znajduje się opis całego symulowanego układu. Netlista musi być odpowiednio przygotowana, aby mogła współpracować z napisanym skryptem, gdyż po jej przygotowaniu wszystkie parametry układu zmieniane są już z poziomu skryptu konfiguracyjnego. W obecnej wersji skryptu przy symulacji rzeczywistego projektu dostępna jest tylko analiza dynamiczna przetwornika, podobnie jednak jak w idealnym ADC mamy do dyspozycji przeprowadzenie kilku pomiarów różnego typu. Możemy skrypt uruchomić w trybie normalnym, gdzie uruchamiany jest jeden cykl pomiarowy, w wyniku czego otrzymujemy analizę Fouriera sygnału próbkowanego przez przetwornik i jego parametry dynamiczne. Drugą możliwością jest uruchomienie serii pomiarów („skan") po dowolnym parametrze przetwornika. Możemy również uruchomić serię pomiarów, w której nie zmieniamy parametrów, natomiast zmieniamy zdefiniowane przypadki „kornerów", czyli najgorszych możliwych modeli technologicznych elementów. W wyniku dwóch ostatnich analiz otrzymujemy serie wyników analiz dynamicznych z opisanymi parametrami które były zmieniane i ich wartościami. Przypadki „kornerów" możemy dowolnie definiować w skrypcie konfiguracyjnym. Czwartą i ostatnią możliwością uruchomienia symulacji dla rzeczywistego projektu, jest uruchomienie symulacji Monte Carlo. Tak jak opisane to było wcześniej w analizie Monte Carlo wykonywana jest określona liczba symulacji dynamicznych, w których uwzględniane są możliwości nieidealnego wykonania elementów, w związku z czym parametry przetwornika mogą się zmieniać. Wynikiem tej analizy jest histogram parametrów dynamicznych osiąganych przez symulowany przetwornik w każdej z symulacji.



(a)



(b)

Rysunek 4.3. Przykładowe wykresy: (a) idealnego, (b) rzeczywistego układu 10-bitowego przetwornika analogowo-cyfrowego zbudowanego z 1.5-bitowych stopni.

Rozdział 5

Symulacje układu 8 kanałowego ADC

W rozdziale tym opisane zostały symulacje przeprowadzone na układzie 8 kanałowego 10-bitowego potokowego przetwornika analogowo-cyfrowego projektowanego w Katedrze Oddziaływań i Detekcji Cząstek AGH. Dotychczas w procesie projektowania kolejnych prototypów powyższego układu, przed wysłaniem takiego prototypu do produkcji, nigdy nie symulowano całego układu, lecz tylko jego pojedyncze kanały. Spowodowane to było przede wszystkim wielkością takiego układu, jak również faktem wykorzystywania symulatorów, przy pomocy których wykonanie takich symulacji było zbyt czasochłonne lub wręcz nie było możliwe, ze względu na ograniczenia symulatora. Na podstawie badań różnych symulatorów przeprowadzonych w niniejszej pracy, podjęto jednak decyzję o próbie wykonania symulacji całego 8 kanałowego układu przy pomocy symulatora, który najlepiej wypadł w testach (rozdział 3), czyli APS. Po wstępnie przeprowadzonych testach udało się uruchomić symulację całego układu, zarówno w wersji bez elementów pasożytniczych, jak również po ich dodaniu do układu. Do przesymulowania działania wybrane zostały dwie ostatnie wersje prototypów układu 8 kanałowego ADC, nazywane dalej wersją pierwszą i wersją drugą. Układy różnią się między innymi sposobem rozprowadzenia sygnałów sterujących, jak również w wersji drugiej dodano możliwość sterowania większą liczbą parametrów układu. Wykonane symulacje przeprowadzone były bardziej pod kątem sprawdzenia możliwości symulowania tak dużego i złożonego układu, niż pod kątem przebadania wszystkich parametrów układu. Otrzymane wyniki natomiast posłużyć miały jako sprawdzenie poprawności symulacji, gdyż omawiane układy prototypowe zostały już wyprodukowane i wykonane zostały już pierwsze pomiary pierwszej wersji prototypu.

Jak już wspomniano wcześniej do testów posłużył symulator APS, w związku z tym konieczne było wygenerowanie z projektów dwóch testowanych układów netlist dla tego symulatora i odpowiednie ich przygotowanie do przeprowadzenia symulacji. Wygenerowane zostały 3 netlisty, a mianowicie:

Tablica 5.1. Parametry netlist układu 8 kanałowego 10-bitowego ADC

	1*	2**	3***
Ilość linii pliku netlisty	1'538'200	17'700	4'868'900
Wielkość pliku netlisty	81MB	1MB	250MB
Ilość elementów w układzie	1'208'932	99'543	2'611'065
Pamięć operacyjna (APS)	2.65GB	1.06GB	6.61GB
Redukcja rezystancji (APS)	0%	0%	39.4%
Redukcja pojemności (APS)	55.4%	0%	52.6%
* - pierwsza wersja układu z elementami pasożytniczymi ** - druga wersja układu bez elementów pasożytniczych *** - druga wersja układu z elementami pasożytniczymi			

1. pierwszej wersji układu z elementami pasożytniczymi,
2. drugiej wersji układu bez elementów pasożytniczych,
3. drugiej wersji układu z elementami pasożytniczymi.

Netlista pierwszej wersji układu bez elementów pasożytniczych również była generowana, jednak posłużyła tylko do testów uruchomienia symulacji, ze względu na to, iż wcześniejsze symulacje pojedynczych kanałów pierwszej wersji układu wykazały, że układ zaczyna poprawnie pracować dopiero po uwzględnieniu elementów pasożytniczych. W tabeli 5.1 zebrane zostały informacje na temat parametrów wygenerowanych netlist, takie jak ilość linii pliku netlisty, jego wielkość na dysku, jak również ilość elementów układu zdefiniowanych w netliście. W tabeli mamy również informacje związane już z samą symulacją i symulatorem APS, a mianowicie ilość pamięci operacyjnej wykorzystywanej przez symulator podczas pracy, jak również procentową redukcję elementów pasożytniczych układu. Porównując dane w tabeli możemy zobaczyć bardzo duże różnice w pierwszej wersji układu z elementami pasożytniczymi i drugiej wersji z elementami pasożytniczymi. Pomimo tego, że układy są bardzo podobne, nie licząc niewielkich zmian między wersjami, różnice które występują w parametrach netlist tych układów związane są z elementami pasożytniczymi. W netliście pierwszej wersji układu elementy pasożytnicze dodane są tylko dla części analogowej układu, natomiast w netliście drugiej wersji układu dodano elementy pasożytnicze dla całego układu, co jak można łatwo zauważyć zwiększa wielkość netlisty około 3 krotnie, natomiast ilość elementów ponad 2 krotnie. Porównując natomiast netlisty bez elementów pasożytniczych oraz po ich dodaniu, możemy zaobserwować jak dużo takich elementów dodawanych jest do układu. Jeżeli uwzględnimy elementy pasożytnicze w całym układzie, to wzrost wszystkich elementów wynosi ponad 26

Tablica 5.2. Czas trwania symulacji układu 8 kanałowego 10-bitowego ADC symulatorem APS, na 8 procesorach Intel(R) Xeon(R) L5640 @ 2.27GHz

f_{smp} [Hz]	Symulowany czas [s]	1* [s]	2** [s]	3*** [s]
1M	150μ	163k	45k	425k
5M	30μ	139k	42k	384k
10M	15μ	137k	35k	254k
16M	9.3μ	95k	26k	184k
20M	7.5μ	80k	28k	161k
25M	6μ	68k	21k	147k
32M	5μ	62k	19k	123k
40M	3.75μ	57k	14k	94k
50M	3μ	45k	12k	83k
* - pierwsza wersja układu z elementami pasożytniczymi				
** - druga wersja układu bez elementów pasożytniczych				
*** - druga wersja układu z elementami pasożytniczymi				

razy w stosunku do układu bez ich uwzględnienia, wielkość netlisty zwiększa się 250 razy, natomiast uwzględniając elementy pasożytnicze tylko w części analogowej, wówczas wzrost liczby elementów wynosi ponad 12 razy, a wzrost wielkości netlisty wynosi około 80 razy. Warto również zwrócić uwagę na ilość pamięci operacyjnej potrzebnej do pracy symulatora. Są to dość duże wielkości, a symulacje musimy wykonywać na komputerze, na którym mamy dostępną wymaganą ilość pamięci. Musimy również wziąć pod uwagę fakt, iż podczas symulacji drugiej wersji układu z elementami pasożytniczymi, symulator APS potrzebował aż 6.61GB pamięci operacyjnej, a aby zaadresować taką liczbę komórek pamięci musimy używać 64 bitowego zarówno systemu operacyjnego na jakim uruchamiany jest symulator, jak również sam symulator musi być w wersji 64 bitowej.

Kolejnym aspektem jaki interesuje nas podczas symulacji układu, jest czas działania symulatora jaki potrzebny jest na uzyskanie wymaganych wyników. W tabeli 5.2 zebrane zostały czasy działania symulatora APS podczas symulowania trzech układów 8 kanałowego ADC, z różnymi częstotliwościami próbkowania, a zarazem z różnymi symulowanymi czasami, z których uzyskano po 128 skonwertowanych przez układ próbek. Symulacje wykonywane były z wykorzystaniem 8 procesorów `Intel(R) Xeon(R) CPU L5640 @ 2.27GHz`, z dostępem do 16GB pamięci operacyjnej. Z analizy wyników zamieszczonych w tabelach 5.2 i 5.1 zauważyć możemy, iż istnieje bardzo duża zależność pomiędzy wielkością symulowanego układu, a długo-

ścią jego symulacji. Naturalnie, najszybciej wyliczone były wyniki symulacji układu bez elementów pasożytniczych. Wzrost czasu działania symulatora dla układów z elementami pasożytniczymi wynosił od około 3 razy dla pierwszej wersji układu, do 6, a nawet 9 razy dla drugiej wersji układu. Wzrost czasu obliczeń związany jest też z czasem jaki jest symulowany oraz symulowaną częstotliwością próbkowania przetwornika. Przy zmniejszaniu częstotliwości próbkowania jesteśmy zmuszeni wydłużyć symulowany czas, aby osiągnąć taką samą liczbę próbek. Zależność między symulowanym czasem, a częstotliwością próbkowania jest więc liniowa, długość obliczeń natomiast, jak możemy zauważyć narasta w sposób logarytmiczny w stosunku do symulowanego czasu. Wziąć pod uwagę należy również to, iż dla niskich częstotliwości próbkowania układów z elementami pasożytniczymi, czasy obliczeń 128 spróbkowanych punktów są już dość duże i wynoszą od około 2 dni do około 5 dni w zależności od wersji symulowanego układu. Przy symulacjach dłuższej pracy przetwornika, np. dla uzyskania 1024 punktów pomiarowych, czas symulacji oczywiście znacznie wzrasta, a więc musimy znacznie dłużej czekać na uzyskanie wyników symulacji, a w niektórych przypadkach czas oczekiwania na wyniki przestaje być rozsądny. Dlatego właśnie musimy mieć świadomość czasu trwania uruchamianej symulacji. Przy dłuższych symulacjach lub w przypadku konieczności posiadania danych z wielu parametrów symulowanego układu, powinniśmy wziąć pod uwagę wielkość pliku z danymi wynikowymi symulacji. Jeżeli istnieje możliwość, że plik ten przekraczał będzie 2GB, w celu możliwości zapisu takiego pliku musimy używać do symulacji 64 bitowej wersji symulatora APS, jak również system plików używany na komputerze, na którym wykonywana jest symulacja musi posiadać możliwość zapisu takiego pliku, w przeciwnym przypadku proces symulacji zakończy się niepowodzeniem.

W celu sprawdzenia poprawności przeprowadzonych symulacji, jak również porównania ich wyników z wynikami analizy już wyprodukowanych układów przeprowadzono dwie serie symulacji układów. Pierwsza z nich to sprawdzenie działania całego układu, poprzez przebadanie działania każdego kanału układu. Symulacja taka wykonana była dla każdego z omawianych wyżej układów, a mianowicie dla pierwszej wersji układu z elementami pasożytniczymi, jak i dla drugiej wersji układu, tak z elementami pasożytniczymi, jak i bez ich uwzględnienia. W tabeli 5.3 zebrane zostały parametry otrzymane z analizy sygnałów z 8 kanałów każdego z układów. Na każdy z kanałów podawany był sygnał sinusoidalny o innej częstotliwości f_{in}, jednak wszystkie sygnały próbkowane były z częstotliwością 25MHz przez czas potrzebny do uzyskania 128 próbek. Jak możemy zauważyć parame-

try uzyskiwane przez każdy z kanałów wszystkich 3 układów odpowiadają z dobrym przybliżeniem wartościom uzyskiwanym dla idealnego 10-bitowego przetwornika analogowo-cyfrowego oraz w pomiarach pierwszej wersji prototypu. Zauważalny jest jednak mały wyjątek, gdzie symulacja 5-go kanału pierwszej wersji układu daje znacząco gorsze wyniki. W chwili obecnej nie udało się ustalić przyczyny tak kiepskich wyników symulacji dla 5-go kanału pierwszej wersji układu i zważywszy na poprawne wyniki pozostałych kanałów oraz fakt nie zaobserwowania podobnego zjawiska w testach rzeczywistego układu prototypowego, najbardziej prawdopodobnym jest błąd numeryczny. Aby zweryfikować otrzymany wynik należałoby więc przeprowadzić kilka kolejnych testów dla tego konkretnego przypadku. Ze względu na czas wykonywania takich testów, na obecnym zaawansowaniu prac testy takie nie zostały jednak wykonane. Wspomniane przybliżenia wartości parametrów wynikają z faktu nie uwzględnienia w symulacjach rozrzutów technologicznych badanych układów, jak również małej liczby analizowanych punktów, co może prowadzić do fluktuacji w uzyskanych wartościach badanych parametrów, chociażby ze względu na niejednorodność rozłożenia otrzymanych punktów. W celu wstępnej oceny pracy układu otrzymane wyniki są jednak wystarczające i świadczą zarówno o poprawnej pracy układu, jak o poprawności przeprowadzonych symulacji. Druga seria symulacji polegała na przebadaniu działania pojedynczego kanału każdego z układów w zależności od częstotliwości próbkowania f_{smp} sygnału. W tabeli 5.4 zebrane zostały parametry wyliczone z analizy sygnałów f_{in} próbkowanych przez każdy z układów. Sygnały podawane były na 3-ci kanał każdego z układów. Mała liczba częstotliwości próbkowania wynika z ograniczeń wielkości zaokrągleń numerycznych, jakie musimy spełnić aby uzyskać poprawne wyniki symulacji, z analizy których wyznaczane są parametry symulowanego układu. Chodzi tutaj o zależność pomiędzy częstotliwością próbkowania, ilością symulowanych próbek, a sygnałem wejściowym. Dane umieszczone w tabeli są zgodne z danymi otrzymanymi z testów prototypu pierwszej wersji układu i w zadowalający sposób przybliżają jego działanie. Badany prototypowy układ, tak jak symulowane układy pracował bowiem poprawnie do częstotliwości próbkowania około 30 MHz, natomiast powyżej tej wartości pogarszały się znacznie jego parametry. Podczas symulacji na pojedynczym kanale nie stwierdzono ponadto żadnych przerzutów pomiędzy kanałami układów.

Tablica 5.3. Parametry układu 8 kanałowego 10-bitowego ADC próbkującego sygnały sinusoidalne o częstotliwościach f_{in} przez każdy z kanałów z częstotliwością 25MHz

Nr kanału	Wersja	1	2	3	4
$f_{in}[Hz]$	ADC	585'937.5	2'539'062.5	4'492'187.5	6'054'687.5
SINAD [dB]	1*	59.8	60.1	60.0	60.2
	2**	60.4	61.0	60.5	60.7
	3***	61.2	60.1	61.1	60.5
THD [dB]	1*	-68.3	-69.0	-65.8	-66.4
	2**	-69.2	-67.0	-69.6	-66.3
	3***	-68.6	-67.9	-70.9	-67.9
SNHR [dB]	1*	60.5	60.6	61.3	61.4
	2**	61.0	62.2	61.0	62.1
	3***	62.1	60.9	61.6	61.4
SFDR [dB]	1*	71.6	68.9	72.0	70.4
	2**	71.0	71.8	69.6	72.0
	3***	70.0	73.0	72.9	72.2
ENOB	1*	9.6	9.7	9.7	9.7
	2**	9.7	9.8	9.8	9.8
	3***	9.9	9.7	9.9	9.8

Nr kanału	Wersja	5	6	7	8
$f_{in}[Hz]$	ADC	8'398'437.5	9'960'937.5	11'132'812.5	11'914'062.4
SINAD [dB]	1*	33.8	59.8	59.8	59.6
	2**	60.6	60.2	60.7	60.0
	3***	60.5	59.8	60.2	59.8
THD [dB]	1*	-43.6	-65.6	-66.3	-66.3
	2**	-67.7	-61.8	-68.4	-64.8
	3***	-65.4	-65.5	-66.9	-66.6
SNHR [dB]	1*	34.2	61.1	60.9	60.7
	2**	61.6	61.1	61.5	61.8
	3***	62.2	61.2	61.3	60.9
SFDR [dB]	1*	41.7	69.0	71.8	71.4
	2**	71.6	70.6	71.8	68.9
	3***	69.3	69.3	70.0	68.7
ENOB	1*	5.3	9.6	9.6	9.6
	2**	9.8	9.7	9.8	9.7
	3***	9.8	9.6	9.7	9.6

* - pierwsza wersja układu z elementami pasożytniczymi

** - druga wersja układu bez elementów pasożytniczych

*** - druga wersja układu z elementami pasożytniczymi

Tablica 5.4. Parametry układu 8 kanałowego 10-bitowego ADC dla różnych częstotliwości próbkowania f_{smp} sygnału sinusoidalnego o częstotliwości f_{in}, podawanego na 3-ci kanał układu

$f_{smp}[Hz]$	Wersja	1M	5M	10M	16M	20M
$f_{in}[Hz]$	ADC	54'687.5	273'437.5	543'875	875'000	1'093'750
SINAD [dB]	1*	60.4	60.2	60.0	60.2	60.1
	2**	60.9	61.3	60.9	60.9	61.3
	3***	60.7	60.6	60.6	61.1	60.7
THD [dB]	1*	-66.5	-69.7	-67.2	-67.0	-66.9
	2**	-69.7	-68.4	-69.6	-71.0	-68.8
	3***	-71.5	-71.9	-67.7	-70.8	-68.5
SNHR [dB]	1*	61.7	60.7	60.9	61.2	61.1
	2**	61.5	62.2	61.5	61.3	62.2
	3***	61.1	60.9	61.6	61.6	61.5
SFDR [dB]	1*	71.9	71.8	71.4	71.0	69.1
	2**	68.1	71.2	71.5	71.1	70.7
	3***	69.6	70.0	71.6	72.2	72.7
ENOB	1*	9.7	9.7	9.7	9.7	9.7
	2**	9.8	9.9	9.8	9.8	9.9
	3***	9.8	9.8	9.8	9.9	9.8

$f_{smp}[Hz]$	Wersja	25M	32M	40M	50M
$f_{in}[Hz]$	ADC	1'367'187.5	1'750'000	2'187'500	2'734'375
SINAD [dB]	1*	60.1	8.3	47.5	36.6
	2**	60.7	60.5	60.5	0.0
	3***	60.4	0.0	0.0	0.0
THD [dB]	1*	-69.7	-9.3	-50.0	-38.4
	2**	-67.1	-67.0	-65.7	0.0
	3***	-67.8	0.0	0.0	0.0
SNHR [dB]	1*	60.6	15.2	51.0	41.3
	2**	61.8	61.6	62.0	0.0
	3***	61.3	0.0	0.0	0.0
SFDR [dB]	1*	71.9	17.3	53.4	41.3
	2**	71.5	69.2	71.6	0.0
	3***	72.2	0.0	0.0	0.0
ENOB	1*	9.7	1.1	7.6	5.8
	2**	9.8	9.8	9.8	0.0
	3***	9.7	0.0	0.0	0.0

* - pierwsza wersja układu z elementami pasożytniczymi
** - druga wersja układu bez elementów pasożytniczych
*** - druga wersja układu z elementami pasożytniczymi

Rozdział 6

Podsumowanie

Celem niniejszej pracy była analiza działania przetworników analogowo-cyfrowych typu potokowego, w oparciu o projektowany w Zespole Elektroniki Jądrowej układ 8 kanałowego, 10-bitowego przetwornika ADC. W zakres pracy wchodziło przeanalizowanie procesu symulacji wspomnianego układu oraz stworzenie narzędzi ułatwiających przeprowadzanie symulacji, jak również przetestowanie dostępnych komercyjnych symulatorów, dzięki którym możliwa jest weryfikacja projektu przed oddaniem go do produkcji. Aby osiągnąć wyznaczone cele konieczne było wykonanie prac, które podzielić możemy na cztery główne części:

- analiza działania idealnego potokowego przetwornika analogowo-cyfrowego,
- testy i porównania symulatorów układów elektronicznych,
- stworzenie systemu do automatyzacji symulacji i analizy ich wyników,
- przeprowadzenie symulacji testowych projektowanego układu wielokanałowego przetwornika analogowo-cyfrowego.

Cel pracy został osiągnięty, czego potwierdzeniem są rozdziały od drugiego do piątego, których zawartość w całości (lub prawie) została opracowana przez autora. Poniżej zebrano natomiast opis najważniejszych prac wykonanych w ramach opracowywania wspomnianych rozdziałów, jak również wnioski płynące z ich wykonania.

W ramach prac nad idealnym przetwornikiem analogowo-cyfrowym wykonano następujące czynności:

— zaprojektowano i napisano skrypt w języku `Python`, dzięki któremu mamy możliwość symulacji pracy idealnego przetwornika analogowo-cyfrowego, ponadto mamy możliwość wprowadzania do przetwornika głównych efektów pogarszających pracę przetwornika, a występujących w rzeczywistych układach,

— wykonano szereg symulacji pracy potokowego przetwornika analogowo-cyfrowego, przy użyciu stworzonego skryptu,

— na podstawie przeprowadzonych symulacji przeanalizowano wpływ poszczegól-

nych parametrów przetwornika na jego pracę. Analizowane parametry to amplituda sygnału wejściowego, temperatura, offset komparatorów, wzmocnienie, wielkość pojemności.

Dzięki stworzeniu skryptu symulującego działanie idealnego potokowego przetwornika analogowo-cyfrowego, mamy możliwość badania w prosty i szybki sposób wpływu poszczególnych parametrów przetwornika na osiągane przez niego parametry. Wyliczone parametry układu, po wprowadzeniu do skryptu danych analogicznych jak występujące w rzeczywistym układzie, mogą posłużyć jako dobre porównanie dla parametrów osiąganych przez rzeczywisty układ.

Podczas pracy nad drugą częścią pracy, a mianowicie testami symulatorów układów elektronicznych wykonano:

— analizę działania czterech dostępnych symulatorów: `Hspice`, `Spectre`, `APS` i `UltraSim`, sposoby ich uruchamiania, dostępne opcje zmieniające szybkość i dokładność obliczeń,
— testy prędkości i dokładności każdego z symulatorów, z użyciem różnych opcji,
— porównanie wyników testów oraz wybór symulatora optymalnego pod względem szybkości i dokładności obliczeń, którym okazał się symulator `APS`.

Przeprowadzone testy wykazały znaczną przewagę zarówno czasową jak i dokładnościową symulatora APS nad pozostałymi testowanymi symulatorami. Dzięki przeprowadzeniu takich testów, używany dotychczas najczęściej w testowaniu poprawności projektu układu przetwornika ADC symulator `Hspice` został zastąpiony przez `APS`, skracając znacznie czas obliczeń, przy zachowaniu takiej samej lub większej dokładności.

Trzecia część pracy, czyli system automatycznych symulacji i przetwarzania ich wyników wymagała wykonania, co następuje:

— zebrania informacji na temat obecnego systemu wykonywania symulacji, jak również narzędzi wykorzystywanych podczas ich wykonywania,
— dostosowania skryptów używanych obecnie w procesie symulacji, ze względu na używanie nowych symulatorów,
— zaprojektowania i napisania skryptu automatyzującego i upraszczającego cały proces symulacji,
— wykonania instrukcji korzystania z napisanego skryptu.

Stworzony system automatycznego uruchamiania symulacji i analizy ich wyników okazał się bardzo pomocnym narzędziem podczas przeprowadzania testów układu przetwornika analogowo-cyfrowego. Dzięki użyciu stworzonego systemu, po ustawie-

niu wszystkich parametrów i uruchomieniu skryptu, otrzymujemy w wyniku jego działania gotowy wynik analizy, jaką chcieliśmy wykonać. Z osoby która zajmuje się uruchamianiem kolejnych symulacji zdjęty zostaje przez to ciężar pamiętania ustawień wszystkich uruchomionych symulacji i konieczności „ręcznego" analizowania każdej z symulacji.

W ramach czwartej części wykonano następujące prace:

— zapoznano się z działaniem układu 8 kanałowego 10-bitowego przetwornika analogowo-cyfrowego, jak również konieczne było przystosowanie parametrów sterujących układ w celu możliwości poprawnego jego symulowania,

— przeprowadzono pierwsze udane próby uruchomienia symulacji całego układu wielokanałowego przetwornika, gdyż do tej pory symulacjom poddawane były jedynie pojedyncze kanały układu,

— zebrano i omówiono wszelkie problemy jakie pojawiły się podczas prób pierwszych symulacji całego układu, jak również omówiono czasy trwania takich symulacji dla różnych częstotliwości próbkowania (więc również różnych symulowanych czasów), na przykładzie dwóch ostatnich wersji układu,

— przeprowadzono pierwsze analizy wyników wykonanych symulacji, w celu potwierdzenia poprawności zarówno przeprowadzonych symulacji jak również pracy symulowanego układu.

Dzięki przeprowadzeniu z powodzeniem pierwszych symulacji całego układu wielokanałowego przetwornika analogowo-cyfrowego, będzie teraz możliwe przeprowadzanie takich symulacji na kolejnych wersjach prototypów układu, przed wysłaniem go do produkcji. Istotność przeprowadzania takich symulacji polega na tym, że w razie jakiegoś błędu w nie przetestowanym projekcie, oddajemy do produkcji nie działający układ, marnując przez to dużo czasu i pieniędzy. Jest to więc ważny element w procesie projektowania układów elektronicznych. Dotychczas testowane było działanie jedynie pojedynczego kanału, natomiast działanie całego układu nie było testowane ze względu na problemy z uruchomieniem symulacji tak dużego i skomplikowanego układu. Dzięki przeprowadzonym testom symulatorów i zastosowaniu symulatora `APS` udało się uruchomić takie symulacje i wykonać je w rozsądnym czasie. Waga przeprowadzania takich symulacji polega na tym, że możemy wyeliminować większość potencjalnych błędów, przed oddaniem układu do produkcji. Bez wykonania takich symulacji, w razie jakiegoś błędu w projekcie, oddajemy do produkcji nie działający układ, marnując przez to dużo czasu i pieniędzy.

Dodatek A

Instrukcja obsługi systemu automatycznych symulacji

W załączniku tym zamieszczona została instrukcja obsługi systemu automatycznych symulacji, w której opisane zostało krok po kroku, w jaki sposób powinny zostać ustawione parametry w pliku konfiguracyjny, aby uruchomić poszczególne typy symulacji.

A.1. Skrypt konfiguracyjny systemu automatycznych symulacji

```
1  ######################################################################
2  #config file for ideal_adc.py & data_analysis.py & symulacje.py#
3  ######################################################################
4
5  #symulator 1-idealne ADC, 2-hspice, 3-spectre, 4-aps
6  sim=4
7
8  #katalog z~wynikami symulacji (np.: "wyniki/" lub "")
9  result_dir="wyniki2/"
10
11 #ilosc stopni ADC
12 Stages=9
13
14 #okres probkowania
15 Tclk=31.25e-9
16
17 #czestotliwosc sygnalu wejsciowego
18 signalFreq=12.75e6
19
20 #ilosc danych na wyjsciu (2^n)
21 n=128
22
23 #amplituda sygnalu
24 amp=0.95
25
26 #uruchomienie skryptu bez symulatora True/False lub 1/0
27 no_sim_run=0
28
```

```
29 ################################
30 #idealne ADC
31 ################################
32
33 #symulacja dynamiczna/statyczna True/False
34 dynamic_simulation= True
35
36 #szerokosc bitu przy analizie statycznej
37 bit_width=30
38
39 #rozrzut progow komparatorow o podana wartosc
40 Vth_gauss_sigma=0.0
41
42 #offset komparatorow dla wszystkich stopni(wart. poprawna + dVth)
43 dVth=[0.05,0.05,0.05,0.05,0.05,0.05,0.05]
44
45 #rozrzut technologiczny pojemnosci (%/100)
46 C_gauss_sigma=0.000
47
48 #czynnik skalujacy pojemnosci w~kazdym kolejnym stopniu
49 C_k=1.
50
51 #wartosc pojemnosci dla 1 stopnia C1,C2,....,C8
52 C=[0.5e-12,0.5e-12,0.5e-12,0.5e-12,0.5e-12,0.5e-12,0.5e-12,0.5e-12]
53
54 #typ pojedynczego stopnia potokowego ADC: "1"-1bitowy, "1.5"-1.5bitowy,
55 #                          "2"-2 bitowy, "2.5"-2.5bitowy, "3"-3bitowy
56 Stage_type="2.5"
57
58 #typ pojedynczego stopnia potokowego ADC: "1"-1bitowy, "1.5"-1.5bitowy,
59 #                          "2"-2 bitowy, "2.5"-2.5bitowy, "3"-3bitowy
60 Last_Stage_type="2.5"
61
62 #temperatura w~st.C
63 T=27
64
65 #wzmocnienie Wzmacniacza
66 A=100000000
67
68 #scan po:
69 #""-wylaczony,     "A"-wzmocnieniu(A), "T"-temperaturze,     "amp"-amplitudzie,
70 #"C[0]"-C1,        "C[1]"-C2,          "C[2]"-C3, ....., "C[7]"-C8,
71 #"dVth[0]"-dVth1, "dVth[1]"-dVth2,    "dVth[2]"-dVth3,....., "dVth[6]"-dVth7
72 scan=""
73 scan_min=10
74 scan_max=1000000000
75 scan_krok="log"
76
77 #Monte Carlo (rozrzuty technologiczne C i progi komparatorow):
78 #0-wylaczone, inna liczba oznacza liczbe iteracji MC
79 mc=0
80
81 ################################
82 ################################
83
```

```
84  ################################
85  #symulowane ADC
86  ################################
87
88  #netlista odpowiednia dla wybranego symulatora
89  #(np.: "hspice.sp", "spectre.scs", "aps.scs")
90  netlist="spectre.scs"
91
92  #wersja symulatora 32bit/64bit
93  sim_bit=32
94
95  #liczba procesorow uzywanych do symulacji
96  #mt=-1 => automatyczne wybranie max. liczby dostepnych na maszynie
97  mt=4
98
99  #dokladnosc symulatorow - spectra & aps => "liberal"|"moderate"|"conservative"
100 errpreset="liberal"
101
102 #parasitics on(1)/off(0) (tylko spectra i aps)
103 parasitics=1
104
105 #offset czasu przy analizie pierwszej probki - data_parse.py
106 time_offset=25e-9
107
108 #scan po parametrach lub cornerach [wartosci parametrow lub numery cornerow]
109 corner_scan=[]          #np. corner_scan=[1,2,6,8]
110 param_scan={}           #np. param_scan={"temperature":[0,25,27,85]}
111
112 #symulacja MonteCarlo True/False 1/0 (spectre & aps)
113 montecarlo=0
114
115 #liczba symulacji dla MonteCarlo
116 mc_numruns=2
117
118 #ziarno dla gausa w~sym MonteCarlo, jak (-1) to automatyczne
119 mc_seed=1234
120
121 #punkty pomijane na poczatku dzialania adc
122 dummy_point=25
123
124 #Parametry symulacji (nazwy zgodne z~netlista):
125 netlist_param={}
126 netlist_param["vref_in"]=1
127 netlist_param["vcm"]=1.6
128 netlist_param["vup"]=1.2
129 netlist_param["vdown"]=1.9
130 netlist_param["vdel1p"]=0
131 netlist_param["vdel12"]=0
132 netlist_param["fpulse"]=0.48
133 netlist_param["ibias0"]=50e-6
134 netlist_param["ibiasSH"]=40e-6
135 netlist_param["kpn"]=2
136 netlist_param["nb"]=3
137 netlist_param["nbn"]=2.5
138 netlist_param["nbnp"]=4.5
```

```
139 netlist_param["nbp"]=1
140 netlist_param["tdel"]=10e-9
141 netlist_param["vdel"]=0
142 netlist_param["nbSH"]=3.5
143 netlist_param["nbnSH"]=3.5
144 netlist_param["nbnpSH"]=5.5
145 netlist_param["nbpSH"]=2
146 netlist_param["vsup_a"]=3.3
147 netlist_param["vsup_d"]=3.3
148 netlist_param["tem"]=27
149 netlist_param["temnom"]=27
150
151 ####################################kornery#####################################
152
153 #sciezka do cornerow
154 corner_path="/cad/ams_v4.00"
155
156 #parametry do MC (spectre & aps)
157 mc_param='include "%s/spectre/c35/mcparams.scs"'%(corner_path)
158
159 #definicje schematow cornerow
160 corner={}
161 #corner no.MOS(0)RES(1) CAP(2)  BIP(3)  IND(4)  Temp(5) Supply(6)NAZWA(7)
162 corner[0]=["mc","mc",   "mc",   "mc",   "mc",   25,     3.3,    "MC"]
163 corner[1]=["tm","tm",   "tm",   "tm",   "tm",   25,     3.3,    "Typ"]
164 corner[2]=["wp","wp",   "wp",   "hs",   "tm",   0,      3.6,    "WP"]
165 corner[3]=["ws","ws",   "ws",   "hb",   "tm",   85,     3.0,    "WS1"]
166 corner[4]=["ws","ws",   "ws",   "lb",   "tm",   0,      3.0,    "WS2"]
167 corner[5]=["wo","wp",   "wp",   "hs",   "tm",   0,      3.6,    "WOWP"]
168 corner[6]=["wo","ws",   "ws",   "hb",   "tm",   85,     3.0,    "WOWS"]
169 corner[7]=["wz","wp",   "wp",   "hs",   "tm",   0,      3.6,    "WZWP"]
170 corner[8]=["wz","ws",   "ws",   "hb",   "tm",   85,     3.0,    "WZWS"]
171
172 #definiowanie ktore parametry odpowiadaja Temp i Supply z~cornerow
173 temp_supply_def={5:["tem", "temnom"], 6:["vsup_a","vsup_d"]}
174
175 #wybrany schemat cornerow
176 corner_schem=1
177
178 #corner include dla SPECTRE & APS
179 corner_include_s=[]
180 corner_include_s.append('include "%s/spectre/c35/cmos53.scs" section=cmos'%(corner_path))
181 corner_include_s.append('include "%s/spectre/c35/res.scs" section=res'%(corner_path))
182 corner_include_s.append('include "%s/spectre/c35/cap.scs" section=cap'%(corner_path))
183 corner_include_s.append('include "%s/spectre/c35/bip.scs" section=bip'%(corner_path))
184 corner_include_s.append('include "%s/spectre/c35/ind.scs" section=ind'%(corner_path))
185
186 #corner include dla HSPICE
187 corner_include_h=[]
188 corner_include_h.append('.LIB "%s/hspiceS/c35/cmos49.lib" cmos'%(corner_path))
189 corner_include_h.append('.LIB "%s/hspiceS/c35/res.lib" res'%(corner_path))
190 corner_include_h.append('.LIB "%s/hspiceS/c35/cap.lib" cap'%(corner_path))
191 corner_include_h.append('.LIB "%s/hspiceS/c35/bip.lib" bip'%(corner_path))
192 ##################################
193 ##################################
```

```
194 if __name__ == "__main__":
195     main()
```

A.2. Symulacje idealnego ADC

Jedną z możliwości systemu automatycznych symulacji jest symulowanie działania idealnego potokowego przetwornika analogowo-cyfrowego. Aby przełączyć się w tryb symulacji idealnego ADC, w pliku konfiguracyjnym, nadajemy wartość 1 parametrowi `sim=1`. Następnie w pierwszej sekcji pliku konfiguracyjnego ustawiamy parametry próbkowanego sygnału (częstotliwość - `signalFreq`, amplitudę - `amp`), jak również okres jego próbkowania (`Tclk`) i ilość próbek które chcemy uzyskać (`n`). Po dokonaniu tych ustawień konfigurujemy z jakiego typu stopni składa się symulowany przez nas przetwornik (`Stage_type` i `Last_stage_type`) oraz z ilu takich stopni się składa (`Stages`), przez co automatycznie definiujemy rozdzielczość naszego przetwornika. Po zdefiniowaniu przetwornika, możemy wprowadzić jeszcze kilka parametrów służących symulowaniu rzeczywistych efektów w przetworniku, takich jak temperatura pracy (`T`), wzmocnienie (`A`), wielkość pojemności w stopniach (`C`), ich rozrzut (`C_gauss_sigma`) i czynnik skalujący pojemności w kolejnych stopniach (`C_k`), offset każdego komparatora w stopniu (`dVth`) oraz rozrzut progów przełączania komparatorów (Vth_gauss_sigma). Po dokonaniu ustawień wszystkich omówionych wyżej parametrów możemy wybrać jedną z opisanych niżej analiz. Każdą analizę po jej ustawieniu uruchamiamy skryptem `./symylacje.py`

A.2.1. Pojedyncza analiza

Podczas symulacji idealnego ADC mamy do dyspozycji dwie analizy, statyczną i dynamiczną. Aby ustawić interesująca nas analizę zmieniamy odpowiednio wartość parametru `dynamic_symulation` na *True* lub *False*. Przy wyborze analizy dynamicznej na wejście przetwornika podawany będzie sygnał o parametrach wcześniej ustawionych, natomiast w analizie statycznej na wejściu pojawi się sygnał liniowy zmieniający się od wartości minimalnej, do maksymalnej. Szybkość narastania sygnału zależna jest od okresu próbkowania przetwornika (`Tclk`), jak również ilości punktów próbkowanych na każdy bit przetwornika (`bit_width`). Należy jeszcze pamiętać, że aby uruchomić pojedynczą analizę należy ustawić parametr `scan=‘‘’’` i `mc=0`.

A.2.2. Skan

Symulując idealne ADC, możemy przeprowadzać serie symulacji przetwornika, w której zmieniamy tylko wartość jakiegoś jego parametru. Takiej analizy możemy dokonywać po każdym ze zdefiniowanych parametrów. Parametr po którym chcemy przeprowadzić symulacje podajemy w zmiennej `scan` skryptu konfiguracyjnego. Zdefiniować musimy również wartość minimalną (`scan_min`) i maksymalną (`scan_max`) analizowanego parametru, jak również krok o jaki będzie sie zmieniał parametr (`scan_krok`). Jeżeli ustawimy `scan_krok=‘‘log’’`, wówczas wartość parametru będzie zmieniana logarytmicznie. W celu uruchomienia skanu należy ustawić parametr `mc=0`.

A.2.3. Monte Carlo

Ostatnią analizą możliwą do przeprowadzenia w idealny ADC jest analiza Monte Carlo. W celu ustawienia tej analizy należy przypisać parametrowi `mc` odpowiednią liczbę, która odpowiadać będzie, za liczbę przeprowadzonych symulacji. Warto również odpowiednio ustawić parametry odpowiadające za rzeczywiste efekty w przetworniku, opisane wcześniej. W analizie Monte Carlo wyłączony musi być parametr `scan=‘‘’’`.

A.3. Symulacje rzeczywistego ADC

Drugą z możliwości systemu automatycznych symulacji jest przeprowadzanie symulacji rzeczywistego układu przetwornika analogowo-cyfrowego. Do tego celu skrypt wykorzystuje jeden z trzech symulatorów układów cyfrowo- analogowych: Hspice, Spectre lub APS. Aby przełączyć się w tryb symulowania rzeczywistego układu, a zarazem wybrać symulator, który użyty będzie do symulacji ustawiamy parametr `sym` na odpowiednią wartość (Hspice - `2`, Spectre - `3`, APS - `4`). Następnie ustawiamy parametry próbkowanego sygnału (częstotliwość - `signalFreq`, amplitudę - `amp`), jak również okres jego próbkowania (`Tclk`) i ilość próbek które chcemy uzyskać (`n`). Definiujemy teraz ilość stopni przetwornika (`Stages`), a następnie przechodzimy do sekcji skryptu konfiguracyjnego, odpowiedzialnej za rzeczywiste ADC. Po wybraniu któregoś z symulatorów podajemy ścieżkę do netlisty przygotowanej dla tego symulatora w parametrze `netlist`. Plik netlisty musi być odpowiednio przygotowany, a parametry w nim zawarte muszą zgadzać się z parametrami (`netlist_param`) w pliku konfiguracyjnym. Następnie ustawiamy wersję symulatora (`sim_bit`), 32 lub 64 bitową, liczbę procesorów na których wykonywane będą obliczenia, jak również (dla Spectre

i APS) możemy ustawić dokładność symulatorów i opcję redukującą ilość elementów pasożytniczych w schemacie. Po ustawieniu wszystkich powyższych parametrów ustawiamy jedną z poniższych analiz i uruchamiamy skrypt `./symylacje.py`.

A.3.1. Pojedyncza analiza

W celu uruchomienia pojedynczej analizy dynamicznej rzeczywistego układu przetwornika ADC należy ustawić parametr `dynamic_symulation` na wartość *True*. Przetwornik będzie próbkował sygnał (`signalFreq`) ustawiony w pierwszej części skryptu konfiguracyjnego z zadanym tam okresem próbkowania (`Tclk`), przez czas potrzebny do zebrania zdeklarowanej liczby próbek (`n`). Zadbać należy również o to, aby odpowiednio ustawić trzy inne parametry, a mianowicie: `montecarlo=0`, `param_scan={}`, `corner_scan=[]`.

A.3.2. Skan

W procesie symulacji układu rzeczywistego przetwornika analogowo-cyfrowego możemy uruchomić serie pomiarów po jakimś parametrze lub po kilku parametrach. Parametry po jakich chcemy aby były wykonane symulacje oraz wartości tych parametrów definiujemy w zmiennej `param_scan`. Musimy pamiętać, aby ustawić parametry `montecarlo=0` i `corner_scan=[]`.

A.3.3. Monte Carlo

Kolejną serią pomiarów jaką możemy uruchomić symulując rzeczywisty przetwornik ADC jest analiza Monte Carlo. Aby uruchomić tą analizę musimy ustawić parametr `montecarlo=1`. Ponadto w parametrze `mc_numruns` ustawiamy liczbę symulacji które mają zostać wykonane. Możemy również podać ziarno do generatora liczb pseudolosowych w parametrze `mc_seed`. Jeżeli parametrowi `mc_seed` przypiszemy wartość -1 wówczas ziarno zostanie wylosowane. Na koniec sprawdzamy ustawienie parametrów `param_scan={}` i `corner_scan=[]`.

A.3.4. Najgorsze modele technologiczne

Ostatnią analizą możliwą do przeprowadzenia w układzie rzeczywistego przetwornika ADC jest analiza „kornerowa”, czyli analiza najgorszych modeli technologicznych elementów. Przypadki „kornerów” możemy definiować w parametrze `corner`. W parametrze `corner_path` ustawiamy natomiast ścieżkę dostępu do plików z danymi o „kornerach”. Po zdefiniowaniu jakiegoś przypadku „kornerów” ustawiamy go

przy pomocy parametru `corner_schem` lub definiujemy parametr `corner_scan` i podajemy numery zdefiniowanych „kornerów" w celu przeprowadzenia serii pomiarów z różnymi przypadkami „kornerów". Przy serii pomiarów „kornerowych" należny pamiętać, aby ustawić parametry `montecarlo=0` i `param_scan={}`.

Bibliografia

[1] Len Staller, *Understanding analog to digital converter specifications*, `http://www.embedded.com`, 2005.

[2] Szymon Kulis, *Projekt elektroniki front-end dla kalorymetru LumiCal dla ILC*, praca magisterska, Wydział Fizyki i Informatyki Stosowanej, AGH, 2008.

[3] Adam Drózd, *Architektura przetworników A/C*.

[4] Andrzej Marusaka, *Urządzenia elektroniki*, 1982.

[5] Kannan Sockalingam, *Error compensation in pipeline A/D converters*, B.S. University of Mine, 2000.

[6] Kannan Sockalingam, Rick Thibodeau *10-Bit 5MHz Pipeline A/D Converter*, 2002.

[7] Cheongyuen William Tsang, *Calibrated Analog-to-Digital Converters in Deep Sub-micron CMOS*, 2008.

[8] Y. Chiu, *An adaptive filtering platform for digitall calibrated A/D conversion*, 2004.

[9] Andrew Masami Abo, *Design for Reliability of Low-voltage, Switched-capacitor Circuits*, 1992

[10] Cadence Design Systems, *Virtuoso Spectre Circuit Simulator User Guide, Product Version 7.1.1*, 2009.

[11] Cadence Design Systems, *Virtuoso UltraSim Simulator User Guide, Product Version 7.1.1*, 2009.

[12] Cadence Design Systems, *Virtuoso Accelerated Parallel Simulator User Guide, Product Version 7.1.1*, 2009.

[13] Synopsys, *HSPICE User Guide: Simulation and Analysis*, 2008.

[14] Irshad Alam, *Spectre Turbo Circuit Simulator*, Cadence Design Systems, India.

[15] Krzysztof Świentek, *Lumi-Multi-ADC*, 2010.

Spis rysunków

Spis tablic