

**WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**

KATEDRA ELEKTRONIKI

**PRACA DYPLOMOWA MAGISTERSKA**

Scalony system sterowania windą

ASIC for lift control

Autor: Mateusz Dyrdół

Kierunek studiów: Elektronika i Telekomunikacja

Opiekun pracy: prof. dr hab. inż. Andrzej Kos

Kraków, 2019

**OŚWIADCZENIE**

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz.631 z późn. zm.):„ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.

……………………………….*podpis dyplomanta*

**Spis treści**

[Wstęp 5](#_Toc12485426)

[Cel pracy 6](#_Toc12485427)

[Rozdział 1 Technologia dźwigów osobowych 7](#_Toc12485428)

[1.1 Pierwsze systemy dźwigowe 7](#_Toc12485429)

[1.2 Strategie sterowania windą 8](#_Toc12485430)

[1.2.1 Strategia zbiorowej kontroli 8](#_Toc12485431)

[1.2.2 Strategia strefowa 10](#_Toc12485432)

[1.2.3 Strategia oparta na wyszukiwaniu 12](#_Toc12485433)

[1.3 Cechy standardowej windy 13](#_Toc12485434)

[1.3.1 Zależności czasowe 15](#_Toc12485435)

[1.3.2 Interfejs użytkownika w standardowej windzie 16](#_Toc12485436)

[1.4 Układy elektroniczne sterujące windą 17](#_Toc12485437)

[1.5 Przepisy prawne i normy 18](#_Toc12485438)

[1.6 System przyszłości 20](#_Toc12485439)

[1.6.1 Algorytm wysyłki docelowej 20](#_Toc12485440)

[1.6.2 Kierunki rozwoju technologii 23](#_Toc12485441)

[Rozdział 2 Opis algorytmu w języku Verilog 25](#_Toc12485442)

[2.1 Tworzenie modułów z wykorzystaniem języka Verilog 25](#_Toc12485443)

[2.2 Rozwój oprogramowania w oparciu o cechy użytkowe 26](#_Toc12485444)

[2.3 Architektura systemu 28](#_Toc12485445)

[2.3.1 Moduł pełnego piętra 30](#_Toc12485446)

[2.3.2 Moduł piętra przejściowego 31](#_Toc12485447)

[2.3.3 Interfejs użytkownika 32](#_Toc12485448)

[2.4 Weryfikacja kodu 36](#_Toc12485449)

[2.4.1 Przypadki testowe 37](#_Toc12485450)

[Rozdział 3 Projektowanie układu scalonego 39](#_Toc12485451)

[3.1 Synteza układu logicznego 39](#_Toc12485452)

[3.2 Symulacj apo syntezie? 41](#_Toc12485453)

[3.3 Struktura topografii 42](#_Toc12485454)

[3.3.1 Generacja układu 43](#_Toc12485455)

[3.3.2 Planowanie podłoża 45](#_Toc12485456)

[3.3.3 Układanie warstw zasilania 46](#_Toc12485457)

[3.3.4 Łączenie bramek 47](#_Toc12485458)

[3.3.5 Optymalizacja 47](#_Toc12485459)

[3.3.6 Weryfikacja 48](#_Toc12485460)

[Podsumowanie i wnioski 49](#_Toc12485461)

[Streszczenie 50](#_Toc12485462)

[Summary 51](#_Toc12485463)

[Spis stosowanych skrótów 52](#_Toc12485464)

[Słowa kluczowe 53](#_Toc12485465)

[Keywords 54](#_Toc12485466)

[Bibliografia 55](#_Toc12485467)

[Dodatek A. 57](#_Toc12485468)

[Dodatek B. Spis zawartości dołączonej płyty CD 58](#_Toc12485469)

[Spis ilustracji 59](#_Toc12485470)

[Spis tabel 60](#_Toc12485471)

# Wstęp

We współczesnym świecie windy stały się integralną częścią budynków komercyjnych i publicznych. Ułatwiają szybsze przemieszczanie osób i bagaży między piętrami. System sterowania windą jest jednym z najbardziej popularnych modułów sterujących w elektronice, który przez lata był usprawniany. Zwykle windy są zaprojektowane dla określonych celów budynku z uwzględnieniem głównych czynników, takich jak wysokość budynku, liczba osób podróżujących do każdego piętra i przewidywane okresy działania. Tradycyjne systemy sterowania windami opierają się głównie na logice przekaźników, sterownikach PLC i mikrokontrolerze, ale główną wadą tych systemów jest to, że mają zmniejszoną liczbę wejść i wyjść. Ponadto w systemach przekaźnikowych układy sterowania mają wysoką awaryjność, która wynika głównie z licznych połączeń i złożoności obwodu elektrycznego.

Poziom ludzkiego życia rośnie bardzo szybko, dlatego systemy sterownia windami stale się rozwijają. Na świecie powstają coraz większe budowle do który trzeba dopasować odpowiednie podejście. Zwiększenie poziom niezawodności, komfortu, bezpieczeństwa i optymalizacji zużycia energii windy to priorytetowe zadania. Wybór odpowiedniej strategii przez projektanta świadczy o wysokiej jakości usługi i jest opłacalną inwestycja. W wysokich budynkach z dużym przepływem pasażerów, zwłaszcza w biurach, strategia dla systemu wind ma kluczowe znaczenie.

Przyszłość jest otwarta na nowe rozwiązania. Producenci wind wciąż pracują by poprawić wydajność systemów wind. Nowe algorytmy wykorzystują sztuczną inteligencję do planowania przyszłych tras w czasie obsługi innych poleceń. Projektanci łączą powstałe w XX wieku strategie w jedną, w celu dopasowania do nowych budynków. Niezależnie od potrzeb, wszystkim twórcom nowych systemów sterowania windą przyświeca jeden cel: zminimalizowanie czasu podróży pasażera.

# Cel pracy

Praca magisterska ma na celu przedstawienie algorytmu sterowania windą i jego realizację jako układ scalony w technice od ogółu do szczegółu (*top-down)*. Opis behawioralny układu cyfrowego został w języku opisu sprzętu Verilog. Sterownik bazuje na automacie skończonym, gdzie poszczególne stany są instrukcjami wykonywanymi przez windę.

Podstawowym kwestią do rozważenia w pierwszym etapie pracy jest wybranie optymalnego rozwiązania ruchu dźwigu. Główną inspiracją do opisania algorytmu jest obserwacja istniejących rozwiązań oraz usprawnienie ich działania. System musi być przyjazny dla użytkownika, sterowanie zgodnie ze strategią zbiorowej kontroli będzie odbywać się za pomocą przycisków wewnątrz i na zewnątrz windy. Algorytm opisany w tej pracy został stworzony dla ośmiopiętrowego budynku. Analogiczna funkcjonalność pięter skłoniła autora pracy do modułowej budowy architektury systemu co ułatwia łatwą rozbudowę o nowe piętra. Dodatkową cechą zaproponowanego systemu są przyciski wewnątrz windy, których funkcja resetowalności pozwala na korektę wybranej destynacji. Przed stworzeniem układu scalonego, cała funkcjonalność została przetestowana bazując na opisywanych w pracy metodach rozwoju i weryfikacji oprogramowania. Kolejnym ważnym aspektem przy projektowaniu systemu sterowania windą jest energooszczędność. Szacuje się ze w niskich budynkach mieszkalnych winda jeździ sporadycznie - poniżej **5% czasu dobowego**[[1]](#footnote-1). To rozwiązanie zostało również dołączone w projekcie jako stan bezczynności, w którym system nie wykonuje żadnych działań i odpytuje o polecenia w dłuższych odstępach czasu niż zwykle.

Układ scalony został zrealizowany przy użyciu pakietu Cadence. Komórki dostępnych bibliotek standardowych pozwalają na mapowanie modułu do fizycznych układów logicznych. Wygenerowany model został umiejscowiony na podłożu krzemowym. Ostatecznie porty wejścia i wyjścia zostały połączone z obudową. Zakończony projekt jest gotowy do fabrykacji lub do zaprogramowania na układzie FPGA.

# Technologia dźwigów osobowych

## Pierwsze systemy dźwigowe

Prymitywne windy były używane już w III wieku pne i były obsługiwane przez ludzi, zwierzęta lub koła wodne. W 1743 r. zbudowano dla króla Ludwika XV przeciwwagę, napędzaną przez człowieka, osobistą windę, łączącą jego mieszkanie w Wersalu z mieszkaniem jego kochanki, Madame de Chateauroux, której kwatera znajdowała się piętro wyżej od króla Ludwika. Pierwsze nowoczesne windy pasażerskie nie więcej niż 150 lat temu. Windy parowe i hydrauliczne zostały już wprowadzone przez 1852, kiedy Elisha Otis stworzył jedno z najważniejszych wynalazków wind, sprzęgło, które uniemożliwiło upadek windy. Następnie w 1857 roku zainstalowano pierwszą windę pasażerską w sklepie E. Haughwout & Company, w Nowym Jorku. Rozwój technologii windy był bardzo szybki ze względu na potrzeby firm przemysłowych.

Pierwsze windy były obsługiwane przez proste urządzenia mechaniczne, takie jak sterowanie „liną ręczną” (Strakosch 1967). Pasażer mógł przywołać windę, naciskając przycisk dostępny po obu stronach kabiny. Szyby windowe nie były całkowicie zamknięte co powodowało, że eksploatacja wind była dość niebezpieczna.

Nowoczesne windy zostały opracowane w XIX wieku. Powolne dźwigi powoli ewoluowały od napędzanej parą do mocy hydraulicznej. Pierwsze hydrauliczne dźwigi zostały zaprojektowane z wykorzystaniem ciśnienia wody jako źródła energii. Technologia silnika i metody sterowania ewoluowały szybko, a elektryczność szybko stała się akceptowanym źródłem energii. Pierwsza elektryczna winda została zbudowana przez niemieckiego wynalazcę Wenera Von Siemensa w 1880 roku. W dzisiejszych czasach nowoczesne budynki komercyjne zwykle mają wiele wind z ujednoliconym systemem sterowania. Ze względu na przeznaczenie windy dzielimy na:

* dźwigi osobowe napędzane hydraulicznie bądź elektrycznie, które służą do przenoszenia osób na różne poziomy budynków. Zwykle przemieszczają się one dość szybko i sztywno w pozycji pionowej,
* dźwigi osobowo-towarowe to z kolei urządzenie, które nadaje się zarówno do przewozu osób, jak i towaru,
* dźwigi szpitalne, zwykle przeznaczony jest do przewozu noszy, a także aparatury szpitalnej oraz pacjentów danego oddziału, czy też personelu medycznego,
* dźwigi budowlane przeznaczone do przewozu grupy robotników oraz materiałów budowlanych, które zwykle przewożone są na wysokie elewacje.

Niezależnie od przeznaczenia, najważniejszą kwestią jest sposób sterowania dźwigiem. Od systemu korbowego sterowanego przez dźwigowego aż po systemy sterowane dyspozycjami, windy stały się szybsze, bardziej zaawansowane i niezawodne w poruszaniu się między piętrami.

Współczesne systemy sterowania windą są głównie w oparciu o logikę przekaźnikową, układy PLC lub mikrokontrolery. Windę można uznać jako złożony system reaktywny, który wymaga równoległego przetwarzania zdarzeń z wieloma wejściami i wyjściami.

## Strategie sterowania windą

Przez lata opracowano wiele strategii sterowania winda. Pierwsze z nich były to bardzo proste strategie a dziś powszechne są algorytmy wykorzystujące sztuczną inteligencję i uczenie maszynowe w celu poprawienia ruchów wind. W wysokich budynkach z dużym przepływem pasażerów, zwłaszcza w biurach, strategia dla system wind ma wielkie znaczenie. Głównym zadaniem jest zminimalizowanie czasu oczekiwania pasażerów oraz czas dostawy na piętro, poprawiając tym samym komfort i wydajność systemu. Istnieje wiele różnych wind w wielu odmianach, które nie są używane tylko do przewozu ludzi czy towarów. Są takie, które mogą być wywoływane i kontrolowane tylko przez operatora, przesuwające się poziomo lub w wielu kierunkach. Strategia powinna zoptymalizować ważne i unikalne dla budynku cechy, które są określone dla każdego z nich w projekcie. Powoduje to dużą liczbę strategii, której liczba rośnie wraz z rozwojem technologii.

### Strategia zbiorowej kontroli

Do jednej z najbardziej popularnych algorytmów należy strategia zbiorowej kontroli. Ten rodzaj systemu jest uznawany za standardowy algorytm kontroli windy. Polega na tym, że winda jedzie w jednym, ustalonym kierunku zabierając pasażerów jadących w tym kierunku. Kiedy po drodze nie ma więcej żądań w kierunku jazdy windy, następuje zmiana kierunku. W przeciwnym razie wózek zatrzymuje się i przechodzi w stan bezczynności, do czasu, gdy upuści ją ostatni pasażer.[5] Jedyną wadą tej strategii jest zjawisko zwane *grupowaniem,* gdzie kilka wózków odbiera to samo żądanie z piętra i przyjeżdża w podobnym czasie, zwiększając tym samym zarówno czas oczekiwania dla pozostałych pasażerów w systemie, jak i odległość podróży wózka windy.

Popularność tej strategii spowodowała użycie jej w innej technologii. Podobnie jest w sterowaniu ramieniem dysku twardego. Dostęp do danych na dysków odbywa się w ruchu uporządkowanym. Nadejście nowego żądania, gdy napęd jest w stanie bezczynności, zapoczątkowuje ruch ramienia w kierunku cylindra, w którym przechowywane są dane, zarówno w, jak i na zewnątrz. Dodatkowe polecenia są obsługiwane tylko w bieżącym kierunku ruchu ramienia, aż ramię osiągnie krawędź dysku. Kierunek ramienia odwraca się, a żądania pozostające w przeciwnym kierunku są obsługiwane. Cykl jest powtarzany, aż wszystkie aktywnych polecenia są wykonane. Jest to opis algorytmu SCAN nazywanego również *algorytmem windy.*

**

*Rysunek 1‑1 Ruch głowicy dysku twardego zgodny z algorytmem SCAN*

***Źródło: [7]***

Dysk twardy znacząco różni się od dźwigu osobowego. Ważne jest by zestaw tych instrukcji odnieść do działania projektowanego algorytmu. Najważniejszą cechą jest utrzymanie kierunku.

### Strategia strefowa

Pierwsze opisy strategii strefowej została opisana przez Georga Strakoscha i Roberta Caporale w pracy *The Vertical Transportation Handbook* z 1983 roku. Sterowanie strefą jest alternatywą dla sterowania zbiorowego i jest preferowane w budynkach z systemem wielu wind. W przypadku korzystania ze strategii strefowej budynek powinien zostać rozdzielony strefy o rozmiarach zależnych od ruchu każdej strefy. Każda winda ma swoje miejsce stopu i tylko odbiera pasażerów z pięter wewnątrz strefy. W przypadku strategii strefy winda ignoruje wszystkie żądania nawet poza jej strefą podczas podróży. Niektóre windy mogą mieć te same piętra w strefach w zależności od przepływu pasażerów i wykorzystywać do poprawy podstawowego założenia strategii [4]. Wózek zatrzymuje się w strefie, gdy jest bezczynna.

Ta strategia ma na celu utrzymanie samochodów w oddzieleniu i uniknięciu zjawiska grupowania. System ten jest odpowiedni dużego natężenia ruchu, gdy połączenia hali są rozłożone na wszystkie strony budynek, ale jednocześnie traci dużą elastyczność [6] od samochodów nie mogą się nawzajem pokrywać. Do stworzenia optymalnej strategii, należy wybrać rozkład stref ostrożnie. Podejmując decyzję o podziale budynku należy wziąć kilka zmiennych do rozważenia. Możesz go podzielić w zależności na populacji pięter lub jeśli jest ważne piętro, takie jak piętro kierownicze. Ogólnym pojęciem jest posiadanie tylu stref, ile jest dostępnych szybów windowych [7].



*Rysunek 1‑2 System strefowy w budynku Shanghai Tower*

***Źródło: [8]***

Istniejący budynkiem, w którym działa systemie strefowym jest największy wieżowiec w Chinach – Shanghai Tower. Budynek został podzielony na dziewięć stref, każdy z nich oznaczona innym kolorem na rysunku 1-2. Połączenia między strefami znajdują się w rożnych miejscach piętra. Ogromna ilość 128 pięter jest skomunikowana przez 108 wind. Dodatkowo windy poruszające się w tym budynku osiągają największe prędkości sięgające do 20.5m/s.



*Rysunek 1‑3 Przykład połączenia dwóch stref*

***Źródło: [8]***

System został zrealizowany przez firmę Mitsubishi. Dostęp do hotelu odbywa się przez piąty hol na wysokości 101/102 piętra. Lokalne strefy są obsługiwane przez windy jednopokładowe w całej wieży, a taras widokowy na szczycie wieży jest obsługiwany przez trzy windy wahadłowe o bardzo dużej prędkości, które poruszają się z prędkością 18 metrów na sekundę. Te trzy windy wahadłowe są uzupełnione trzema windami strażackimi, które znacznie zwiększają przepustowość gości na taras widokowy w szczytowych okresach użytkowania. W przypadku pożaru lub innej sytuacji awaryjnej, windy wahadłowe są zaprojektowane do ewakuacji pasażerów ze specjalny pięter schronienia, rozmieszczonych w regularnych odstępach na całej wysokości wieży [9].

### Strategia oparta na wyszukiwaniu

W przeciwieństwie do opisanych powyżej algorytmów, strategia oparta na wyszukiwaniu opiera się na wybraniu windy o najkrótszym czasie oczekiwania. Optymalizacja odbywa się w dwóch trybach zachłannym i nie-zachłannym. Różnica między tymi strategiami polega na tym, że zachłanne strategie wyszukiwania wykonują natychmiastowe przydzielanie połączeń, czyli przypisują połączenie z wózkiem windy po ich pierwszej rejestracji i nigdy nie rozważają ponownie tych zadań. Chciwe algorytmy rezygnują z pewnej miary wydajności ze względu na brak elastyczności, ale także wymagają mniej czasu obliczeniowego. Przeciwny algorytm jest elastyczny i może ponownie ocenić przydziały połączeń w świetle nowych informacji ciągłych z systemu windy. Nie-zachłanne algorytmy odkładają swoje zadania lub rozważają je w świetle zaktualizowanych informacje, które mogą otrzymać w dodatkowych połączeń lub ilości miejsc dla pasażerów. Ten typ algorytmu zajmie więcej czasu, aby zdecydować, które połączenie powinno zostać przypisane co skutkuje zwiększeniem średniego czasu oczekiwania, ale ogólny wynik może być ostatecznie lepszy [5].

Niezależnie od podejścia, system wybiera kabinę, która minimalizuje czas oczekiwania, czas podróży i liczbę pasażerów. System również wybiera współczynniki i szacowanie funkcji. Symulacje przed uruchomienie systemu służą do weryfikacji ich skuteczności. Po każdym zdarzeniu kontroler szuka najlepszego przypisania połączeń do wózka windy. Słabą stroną tego podejścia jest jego wymaganie obliczeniowe [6].

## Cechy standardowej windy

Każdy projektant windy jest głównie zainteresowany niecierpliwością pasażerów podczas oczekiwania oraz czasem podróży. Podczas gdy pasażerowie czekają na piętrze pośrednim, ich niecierpliwość rośnie. W środowisku komercyjnym z reguły pracownicy są mniej tolerancyjni w oczekiwaniu niż ludzie w środowisku mieszkalnym. Badania wskazują, że pasażerowie stają się niecierpliwym po odczekaniu około 30 sekund w budynku handlowym i około 60 sekund w budynku mieszkalnym [11].

Dobrą analogią jest porównanie transportu wertykalnego do przepływu ciągłego wody. System ciągłego przepływu transportuje wodę ze zbiornika strumieniem lub wężem, która jest przenoszony do miejsca przeznaczenia. System wsadowy przenosi zmierzone ilości do zbiornika, gdzie się gromadzą się, dopóki kolejna partia nie zostanie przeniesiona, zwykle w wiadrze. Można porównać windę do takiego przenośnika wsadowego. Przybycie ludzi do budynku jest w ciągłym przepływie, a system windy to przenośnik przenoszący ludzi ze zbiornika (lobby) do ich docelowych miejsc. Idealnym rozwiązaniem dla wind jest posiadanie wielu wind do przybliżenia procesu ciągłego przepływu, tak aby lobby (zbiornik) nigdy nie było wypełnione nadwyżką ilości osób, którą przewiezie jedna winda.

Z tych obserwacji można wywnioskować pierwsze wymagane dla dobrej obsługi windy: system musi zapewniać wystarczającą ilości usług windy dla maksymalna stawka przylotu lub odlotu pasażera oczekiwana w szczytowym okresie ruchu. To jest możliwe do osiągnięcia albo przez platformę o wystarczającej powierzchni, aby pomieścić wszystkie osoby czekając na przejażdżkę lub alternatywnie, wystarczającą liczbę mniejszych platform. Alternatywa większej ilości platform jest zazwyczaj preferowana, ze względów bezpieczeństwa. Drugim wymaganiem jest to by zaprojektować system tak, aby zapewnić średni czas oczekiwania poniżej 30 sekund dla budynku komercyjnego i mniej niż 60 sekund w budynkach mieszkalnych.

Kolejnym podobnym przykładem systemu zbliżonego do windy są schody ruchome. Platformy są dostępne przy minimalnym czasie oczekiwania, aby osoba miała natychmiastowy dostęp do schodów. Każda z platform jest wystarczająco duża, aby pomieścić tylko jedną lub dwie osoby, jeśli więcej osób czeka na platformę w tej samej chwili, ktoś musi czekać. Potencjalni pasażerowie nie zniecierpliwiają się, ponieważ widzą schody w ruchu, których czas oczekiwania jest bardzo krótki. W przypadku windy, osoba czekająca na wyższym piętrze może nie być w stanie sprawdzić, czy wózek windy jest w środku wykonywania żądania co może powodować niecierpliwość podczas oczekiwania.

Ponadto, gdy pasażerowie wchodzą na ruchome schodów, wiedzą, że zostaną zawiezieni na następne piętro w stosunkowo krótkim czasie i - poza ekstremalnymi przypadkami w niektórych stacjach metra - widzą koniec schodów. Pasażerowie windy często nie wiedzą, jak długo będą podróżować na wybrane piętro. Jeśli obsługiwane jest wiele pięter w ruchliwym budynku i liczba wind jest ograniczona, osoba może być w windzie przez długi okres czasu. Badania wykazały, że jazda około 100 sekund staje się granicą tolerancji, gdy winda robi kilka przystanków, na których wysiada jedna osoba. Tolerancja wydłuża się do około 150 sekund, jeśli kilka osób jest obsługiwanych na każdym przystanku; przeciętna osoba "czuje się bardziej spokojna”. Ostatecznie, jeżeli monotonia jest zakłócona przez zmieniającą się scenerię (np. widok z wieżowca), pasażer może tolerować jazdę nawet przez 180 sekund. Te czynniki czasowe są przybliżone, ponieważ tolerancja jednostki może różnic się w zależności od celu użycia windy.

### Zależności czasowe

Bazując na poprzednich można zauważyć jak bardzo są ważne parametry czasowe dla systemu windy. Do obliczeń całkowitej podróży windą, praktyczną procedurą jest rozbicie podróży na pojedyncze składniki. Podstawowy modelem jest winda dwuprzystankowa. Przypuśćmy, że istnieje budynek, gdzie ​​dwa kolejne przystanki są oddalone od siebie o 3 metry. Gdy pasażerowie przybywają na i obsługują przycisk wezwania windy, zaczynając. Kiedy opuszczają windę na drugim podeście, ich podróż jest zakończona. Po zarejestrowaniu połączenia winda obsługuje pasażera i godzinę czynniki będą następujące.



*Rysunek 1‑4 Model windy dwuprzystankowej*

***Źródło: [5]***

Odnosząc się do rysunku 1.4, jeśli wózek znajduje się na dolnym podeście, drzwi windy muszą się otworzyć, gdy pasażer naciśnie przycisk wywołania (a). Drzwi otwierają się około dwie sekundy, w zależności od szerokości i rodzaju drzwi. Kolejne dwie sekundy trwa wejście pasażera do środka i naciśnięcie przycisk (b). Drzwi zamykają się (około 3 sekundy), a samochód musi przejechać 3 metry do następnego piętra (około 7,5s)(c). Kolejno drzwi otwierają się, co zajmuje 2 sekundy, a kolejne 2 sekundy pasażer opuszcza windę (d). Całkowity czas spędzony przez tego pasażera wynosi około 18,5 sekund. Zanim inna osoba będzie mogła skorzystać z usługi, musi upłynąć więcej czasu. Drzwi musi zamknąć się ponownie (3s), a wózek windy musi wrócić na początkowe piętro (7,5s). W tym punkcie cykl się powtarza. Całkowity czas podróży w obie strony ma około 30 sekund. Tak więc 30 sekund to przybliżony czas osoby, która właśnie przeoczyła windę na pierwszym. Ten czas się nazywa „interwałem” między obsługą windy na parterze. Jeśli uznamy to jako proces ciągły ze strumieniem pasażerów poruszających się w jednym kierunku, przeciętny pasażer może oczekiwać średnio połowę czasu podróży windą w obie strony. Niektórym pasażerom uda się przybyć tuż przed przyjazdem windy i nie będą musieli czekać, podczas gdy inni mogą przeoczyć windę i ich czas oczekiwania będzie wynosił cały interwał, co skutkuje średnim czasem oczekiwania piętnastu sekund. Jeśli dwie windy są obok siebie, każda z nich obsługuje dwa przystanki i działa zgodnie z opisem (każdy z czasem podróży w obie strony 30 sekund), odstęp będzie wynosił połowę czasu podróży w obie strony, czyli 15 sekund, a średni czas oczekiwania wyniesie około 7,5 sekundy.

Dwustopniowa winda to najprostszy model systemu wind. Oczekiwane jest, że pasażer, który wsiądzie na jednym piętrze wysiądzie na drugim. Czas transferu jest zminimalizowany i nie ma mowy o prawdopodobnych zatrzymaniach, ponieważ jest tylko jeden możliwy przystanek. Planowanie jest proste: jeśli ruch jest dwukierunkowy, jeden wózek powinien znajdować się na górze, a drugi na dole; w ruchu jednokierunkowym windy powinny być skoncentrowane u góry lub u dołu. Obliczenia windy stają się bardziej złożone z trzema lub większa ilością przystanków.

### Interfejs użytkownika w standardowej windzie

W nowoczesnych budynkach ludzie są przyzwyczajeni do przycisku przywołania windy, następnie przyjazdu, po którym otwierają się drzwi, aby system wykonał swoje zadanie. Zazwyczaj pasażerowie zauważą zapaloną lampkę pokazującą kierunek podróży windy i wybierają szyb odpowiadający żądanemu kierunkowi. W bardziej ruchliwych budynkach istnieje duża możliwość zatrzymania dwóch wózków jednocześnie na piętrze. Gdy pasażerowie wsiądą do kabiny windy, oczekuje się, że wybiorą przycisk na docelowe piętro. Niezastosowanie się do tej reguły może spowodować, że winda nie pojedzie w wybrane miejsce. Odmianą powyższych reguł jest wprowadzenie docelowych systemów operacyjnych które są szczegółowo opisane w rozdziale 1.6. Ogólnie dotyczy to grup wind, którym system przydziela kabinę pasażerom, która zabiera ich na docelowe miejsce. Szczególna winda odpowiadając na to wezwanie, pokazuje kierunkową lampkę z odczytem wskazującym na przypisaną windę. Pasażer wchodzi i zostaje zabrany do miejsca przeznaczenia. Od pasażerów wymaga się innych niż poprzednio działań. W środku kabiny może być więcej niż jeden oczekujący pasażer przez co winda może zatrzymywać się na wielu piętrach. Każdy pasażer musi zwracać uwagę i wsiąść do wózka przeznaczonego dla jego celu podróży. Nie ma panelu obsługi w środku windy więc pasażer popełniający błąd musi wysiąść na losowym piętrze i zacząć od nowa. Zaletą systemu jest to, że ma tendencję do zmniejszania liczby zatrzymuje windę i poprawia obsługę. Ta różnica w obsłudze pasażerów odniosła wielki sukces w budynkach biurowych, w których ludzie zapoznają się z podejściem, ale ograniczono użycie tego w hotelach, w których nieznane systemu powodują trudności.

## Układy elektroniczne sterujące windą

Główną częścią systemu windy nazywana jest kontrolerem. jest urządzenie logiczne, które monitoruje różne funkcje bezpieczeństwa, zapewnia środki do sterowania sygnały i zapewnia wyjścia do sterowania sygnałami i urządzeniami. Pierwsze elektroniczne układy sterowania wind były oparte na logice przekaźnikowej. W nowoczesnych windach logika sterowania jest głównie oparta na sterowniku PLC, prawdopodobnie dlatego, że przekaźniki są droższe niż elementy elektroniczne. Poza tym PLC ułatwia proste połączenia elektryczne i zmniejsza ryzyko awarii. W starszych wariantach wind kierunek samochodu jest kontrolowany przez selektory zamontowane w szybie, po jednym na każdym piętrze. Na nowoczesnych windach może być tylko jeden przełącznik impulsowy, zamontowany na górnej części wózka windy.

Kontroler może mieć różne formy, kształty, rozmiary i może zapewniać funkcję za pomocą przekaźników, dyskretnych elementów logicznych, dedykowanych systemów mikroprocesorowych lub programowalnych sterowników logicznych. Funkcjonalny układ logiczny może znajdować się w skrzynce sterowniczej w maszynowni lub może być zintegrowany z kabiną. Sterowanie ruchem głównie stanowi integralną część systemu lub wysyła sygnały do ​​niezależnego systemu sterowania ruchem. Architektura sterownika jest skończoną maszyna stanu lub wiele maszyn stanów które odczytują dane wejściowe systemu i, w oparciu o aktualny stan systemu, wysyłają wyjścia w ustalonym stanie.

W obecnej chwili sterowniki PLC są powszechnie stosowane w sterowaniu i automatyzacji przemysłu, w którym proste algorytmy są wystarczające. Wraz z rozwojem technologii pojawiły się potrzeby do przetwarzania większej ilości danych a także przyśpieszenie operacji logicznych przez przetwarzanie równoległego. Układy macierzy programowalnych bramek (tzw. FPGA) stanowią lepsze rozwiązanie jako kontroler windy z dodatkowymi zaletami jak konfigurowalność, mniejsze zużycie energii, krótki czas odpowiedzi i elastyczność w rozbudowie projektów. Jedyną wadą tego podejścia jest czas implementacji danego algorytmu, choć ten słaby punkt jest powoli redukowany przez rozwój narzędzi kompilujących kod z języka wysokiego poziomu (Java, Python, Ruby) do języków opisu sprzętu. Biorąc pod uwagę kierunek rozwoju elektroniki oraz wpływ sztucznej inteligencji na nowe systemy, przewidywalne jest przejście układów sterujących do tego typu rozwiązań. Do celów na wysoką skalę produkcji układy FPGA zastępuje się układami o specjalizowanej funkcjonalności (tzw. ASIC). Te układy są znacznie bardziej wydajny niż FPGA ze względu moduły specjalnie projektowane do określonego zadania. Zużycie energii przez układy ASIC można bardzo dokładnie kontrolować i optymalizować. Ponadto w wyspecjalizowanych układach można projektować analogowe peryferia, na przykład nadajnik-odbiornik WiFi, na tym samym waflu krzemowym co rdzenie mikroprocesora. Jest to zaleta, której brakuje FPGA. Ze względu na brak możliwości zmiany funkcjonalności układu wyspecjalizowanego, wersje prototypowe wykonuje się na FPGA aż osiągnie się pełną walidację. Układy scalone w fazie testowania mają możliwość dostosowania konfiguracji do budynku, w którym zainstalowano windę. Zminimalizowane układy pobierają mniej energii, wykonują więcej operacji w ciągu sekundy i zajmują znacznie mniejszą powierzchnie niż odpowiadające ich układy PLC. Celem sterowania windą za pomocą układów ASIC jest zwiększenie niezawodności, redukcja wielkości jednostek sterujących windą oraz zmniejszenie poboru energii.

## Przepisy prawne i normy

Instalacja oraz funkcjonowanie dźwigów osobowych w Polsce jest regulowane przez przepisy unijne. Największa zmiana w ostatnim czasie dotycząca norm dźwigowych w Polsce miała miejsce w sierpniu 2017 roku. Obecnie **krajach Unii Europejskiej wszystkie instalowane windy podlegają tym samym dyrektywom i rozporządzeniom.**

**Dyrektywa w sprawie dźwigów 2014/33 / UE zezwala na swobodny obrót dźwigami i elementami bezpieczeństwa do dźwigów na wewnętrznym rynku UE i zapewnia wysoki poziom bezpieczeństwa użytkownikom dźwigów i personelowi obsługi technicznej. To zharmonizowane prawodawstwo UE reguluje projektowanie, produkcję i instalację dźwigów. Dotyczy to głównie instalatorów dźwigów i producentów podzespołów, ale ma również istotne konsekwencje dla właścicieli i użytkowników dźwigów. Obecna dyrektywa jest dostosowany do nowej polityki ram prawnych i ma zastosowanie od 20 kwietnia 2016 r., Zastępując poprzednią dyrektywę 95/16 / WE [12].**

**Bezpieczeństwo istniejących dźwigów (zainstalowanych przed wejściem w życie dyrektywy 95/16 / WE w sprawie dźwigów) podlega przepisom krajowym. Zalecenie dotyczące poprawy bezpieczeństwa istniejących dźwigów zachęca kraje UE do podjęcia wszelkich niezbędnych działań w celu zapewnienia zadowalającego poziomu utrzymania istniejących dźwigów i poprawy bezpieczeństwa tych dźwigów. Zalecenie nie jest prawnie wiążące i jest wdrażane przez kraje UE w świetle sytuacji i przepisów istniejących na szczeblu krajowym [13].**

**W odniesieniu dostępu dla osób niepełnosprawnych do kabin wind, kraje UE zachęca się do podejmowania wszelkich środków krajowych niezbędnych do zapewnienia, że ​​wszystkie poziomy istniejących budynków, jak również te w budowie, są dostępne dla osób niepełnosprawnych, w szczególności dla osób korzystających z wózków inwalidzkich. Zaleca się, aby we wszystkich nowych budynkach zapewnić co najmniej jedną windę dostępną dla osób niepełnosprawnych na wózkach inwalidzkich. Ponadto winda musi spełniać wszystkie wymogi regulacyjne (w zakresie wymiarów, położenia elementów sterujących itp.) [12].**

**Standaryzacja wind i schodów ruchomych nie kończy się na granicach Unii Europejskiej. Rynek wind i schodów ruchomych jest naprawdę globalny, nawet w porównaniu z innymi sektorami przemysłu: ponad 75% wszystkich nowych instalacji rocznych (63% w samych Chinach), region Azji i Pacyfiku dominuje obecnie na globalnym rynku wind i schodów ruchomych i jest ma wzrosnąć jeszcze bardziej w latach 2016-2023. Europejski przemysł dźwigowy ze ściśle zintegrowanymi łańcuchami produkcyjnymi i dostawczymi na całym świecie, ma znaczący udział w globalnym rynku. W tej sytuacji niezwykle ważne jest, aby normy europejskie były otwarte na świat: dzięki uznawaniu norm także poza rynkiem europejskim firmy, które je przyjmują, nie muszą obawiać się, że zostaną odcięte od rozwijających się rynków Azji i Pacyfiku z powodu niezgodności. Pozytywne jest zatem, że większość krajów uznała wartość norm CEN / TC 10, zwłaszcza EN 81-20 / 50 i EN 115-1, i postanowiła wdrożyć je w swoim systemie regulacyjnym [14].**

## System przyszłości

### Algorytm wysyłki docelowej

W dzisiejszych miastach, gdzie dominują wieżowce, windy stały się nieodzownym środkiem transportu w życiu codziennym. Jednak coraz więcej pięter i użytkowników sprawiło, że tradycyjny system windy nie jest w stanie skutecznie rozprowadzać i przenosić użytkowników do miejsca przeznaczenia, co powoduje problemy z wydajnością transportu w budynku, takie jak długi czas oczekiwania w holu windy czy wydłużony czas podróży.

Z tego powodu w większości budynków zastosowano metody mające na celu zwiększenie wydajności transportu budynków, takie jak wdrożenie strefowego systemu windy a nawet zastosowanie dwupoziomowych wind. Wraz z rozwojem technologii doprowadziło to do stworzenia bardziej optymalnego systemu dystrybucji wind, znanego jako „wysyłka docelowa”, który w ostatnich latach stał się bardziej atrakcyjną technologią dla wind, przerywając stosowanie tradycyjnych systemów wind i metod dystrybucji.

Tradycyjne systemy wind mają przyciski wezwania na zewnątrz windy do wywoływania windy. Po tym, jak pasażer wejdzie do windy, wciska przycisk żądanego piętra. System nie może przewidzieć pięter, do których wszyscy pasażerowie muszą się udać, więc pasażerowie są zmuszeni do jazdy na różne piętra. Zwiększa to zatem czas jazdy windy, a nawet powoduje, że wszystkie windy podnoszą się i wracają do głównego piętra w tym samym czasie, bezpośrednio wpływając na czas oczekiwania i wydajność windy. Dzięki docelowemu systemowi wysyłkowemu wyeliminowano tradycyjne przyciski wywołania. Zamiast tego pasażerowie wjeżdżają do miejsca docelowego przez wybranie docelowego piętra przed szybem windy, na urządzeniu takim jak klawiatura dziesiętna lub ekran dotykowy.



*Rysunek 1‑5 Wybór piętra w systemie z wysyłką docelową*

***Źródło: []***

System wysyłania miejsc docelowych polega na grupowaniu pasażerów według tego samego miejsca docelowego, gdy tylko znajdą się w miejscu przeznaczenia i przypisuje ich do tych samych wagonów windowych. Kieruje poszczególnymi wagonami windy w grupie, aby obsługiwały tylko określone piętra. Dlatego liczba przystanków jest zmniejszona, a ponieważ winda zatrzymuje się na kilku przystankach, czas podróży jest krótszy niż w konwencjonalnym systemie wind. Zasadą tego systemu jest doprowadzenie pasażerów do miejsca docelowego w możliwie najkrótszym czasie z mniejszym skupiskiem i komfortem.



*Rysunek 1‑6 Przyciski żądań w windzie z wysyłką docelową i klasycznej windzie*

***Źródło: []***

Metodę alokacji można zmienić w zależności od sytuacji. Na przykład, przejście do wyższych, niższych lub sąsiednich pięter zostanie przypisane do tej samej windy, aby uniknąć nieefektywności, takich jak zatrzymanie się na piętrze gdzie nie użytkownik nacisnął zły przycisk kierunku ruchu. Niektóre systemy wind dwupokładowych, w połączeniu z tym systemem, mogą sprawić, że winda będzie bardziej elastyczna, dopasowana do potrzeb budynku.

Windy wyposażone w docelowy system wysyłkowy mają co najmniej dwie klawiatury numeryczne lub ekrany dotykowe w każdym holu, które zastępują tradycyjne przyciski żądań. Jeśli windy mają ekrany dotykowe, zazwyczaj zawierają listę dostępnych pięter obsługiwanych przez windy. Istnieją również tabliczki identyfikacyjne windy zawierające literę do identyfikacji wind (np. A, B, C itd.).



*Rysunek 1‑7 Identyfikacja wind w systemie wysyłkowym*

***Źródło: []***

Pasażerowie są przypisywani do danych szybów i nie mają potrzeby wybierania pięter w środku wagonów. Pozostawiając tylko przyciski otwierania i zamykania drzwi oraz przyciski alarmowe. Przyciski pięter są ukryte za panelem i zwykle nie są używane w normalnych warunkach. Niektóre windy mogą nadal posiadać przyciski podłogowe, ale nie można ich nacisnąć; wskazują tylko piętra na których winda się zatrzymuje. Istnieją również systemy z konfiguracją hybrydową, gdzie przyciski można naciskać. W takim układzie panel sterowania z wysyłką docelową znajduje się tylko na niektórych określonych piętrach a na pozostałych piętrach obowiązuje układ konwencjonalnej windy.

### Kierunki rozwoju technologii

Pomimo tak wyrafinowanych systemów operacyjnych wind, wciąż istnieją ogromne możliwości dla rozwoju i ulepszeń. Producenci wind pracowali ciężko by poprawić wydajność windy zarówno z perspektywy operacyjnej w systemach dyspozytorskich oraz z perspektywy efektywności energetycznej z ulepszoną konstrukcją silnika i napędu. Praktyczne, wydajne przekładnie zębate i przekładnie do zastosowań niskopoziomowych zastępują mniej wydajne konstrukcje hydrauliczne.



*Rysunek 1‑8 Nowa generacja mechanizmów wind GEN2 LIFT firmy OTIS*

*Źródło: [16]*

Brak maszynowni pozwala architektom na większą elastyczność projektowania. Konstruktorzy korzystają z kontrolowanego, usprawnionego procesu instalacji i minimalnej ingerencji w inne transakcje. Dla właścicieli budynków system Gen2 przekłada się na niższe koszty budowy i znaczny wzrost powierzchni do wynajęcia. Modułowy kontroler, zaprojektowany dla grup składających się z maksymalnie trzech samochodów, zawiera nową generację płytek drukowanych i oprogramowania, aby zapewnić optymalny czas reakcji. Napęd VF z cyfrową pętlą zamkniętą, z technologią sterowania wektorowego, dodatkowo zwiększa wydajność i dokładność, a cyfrowy enkoder prędkości zapewnia prawidłową prędkość i pozycję samochodu. Rezultatem jest system wyjątkowej niezawodności.

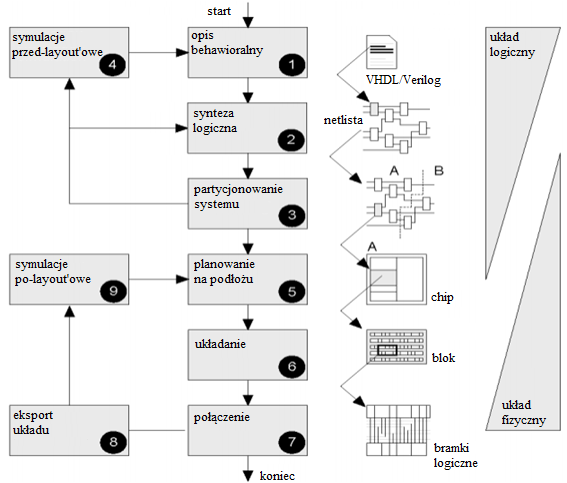
Trend będzie kontynuowany przy użyciu lżejszych materiałów, bardziej wydajnych napędów regeneracyjnych. Ulepszenia w zdalnym monitorowaniu, niezawodności systemu i konserwacji opartej na użytkowaniu będą kontynuowane przez bardziej zaawansowaną diagnostykę kontrolerów oraz wyższy poziom integracji z narzędziami zarządzania utrzymaniem ruchu. Poprawiono poziom niezawodności, co skutkuje mniejszą ilością zbudowanych wind w budynku o określonej powierzchni. Niezawodność będzie miarą tego, czy takie systemy będą praktyczne w budynkach o dużym natężeniu ruchu. Z związku z rozwojem technologii następuje ciągłe ulepszanie interfejsów użytkownika i urządzeń bezpieczeństwa dla wind.

# Opis algorytmu w języku Verilog

## Tworzenie modułów z wykorzystaniem języka Verilog

Język Verilog HDL zawiera funkcje opisujące charakter projektu, przepływ danych, skład strukturalny, opóźnienia i mechanizm generowania przebiegów, w tym aspekty monitorowania odpowiedzi i weryfikacji, wszystkie modelowane przy użyciu jednego języka. Ponadto język zapewnia interfejs języka programowania, dzięki któremu można uzyskać dostęp do elementów wewnętrznych projektu podczas symulacji, w tym do sterowania przebiegiem symulacji. Język nie tylko definiuje składnię, ale także definiuje bardzo jasną semantykę symulacji dla każdego konstruktu językowego. Dlatego modele napisane w tym języku można zweryfikować za pomocą symulatora Verilog. Język dziedziczy wiele symboli operatora i konstruktów z języka programowania C. Verilog HDL zapewnia szeroki zakres możliwości modelowania, z których niektóre są dość trudne do zrozumienia na początku. Jednak podstawowy podzbiór języka jest łatwy do opanowania i użycia. Jest to wystarczające do modelowania większości aplikacji.

Przy tworzeniu kontrolera windy został wykorzystany program Icarus Verilog (iverilog) z pakietem GTKWave oraz zintegrowane środowisko programistyczne Aldec Active-HDL w wersji studenckiej. Iverilog jet doskonałym narzędziem się do kompilacji i symulacji małych modułów. Z kolei firma Aldec dostarcza całkowite środowisko do rozwoju architektury systemu, który po symulacji można przygotować do zaprogramowania układów FPGA. Interfejs programu Active jest bardziej przyjazny dla użytkownika. Bezpośredni dostęp do wewnętrznych sygnałów pozwala na szybką implementacje oraz weryfikacje projektowanego algorytmu. Wersja studencka, mimo swoich ograniczeń, umożliwia projektowanie układu logicznych w celach edukacyjnych na profesjonalnym poziomie. Produkt firmy Icarus jest dystrybuowany na licencji *Creative Commons Attribution 3.0 Unported License,* która pozwala na użycie oprogramowania dla własnych celów.



*Rysunek 2‑1 Schemat projektowania układu ASIC*

*Źródło: [15]*

## Rozwój oprogramowania w oparciu o cechy użytkowe

Wybraną metodologia tworzenia oprogramowania jest podejście iteracyjne z nastawieniem na implementacje cech produktu (ang. *Feature-Driven Development)*. Jest to zwinna metoda tworzenia oprogramowania. Ten proces łączy wiele uznanych, najlepszych praktyk w branży programistów w spójną całość. Praktyki te są sterowane z perspektywy funkcjonalności, które są najważniejsze dla klienta. Jego głównym celem jest dostarczanie namacalnego, działającego oprogramowania wielokrotnie w odpowiednim czasie, zgodnie z zasadami stojącymi za manifestem Agile.

Głównym elementem tej metodyki jest wyznaczenie cech produktu do stworzenia i zaplanowaniu ich implementacji w krótkim, określonym czasie. Duża funkcjonalność jest dzielona na mniejsze co pozwala na kontrolę nad funkcjonalnością produktu. Po każdej iteracji nowe cechy są integrowane z dotychczasową wersją oprogramowania i testowane. W przypadku wystąpienia błędu, kod jest analizowany i poprawiany. Produkt jest dostarczany okresowo wraz z nowo wprowadzonymi zmianami. Zaletą tego podejścia jest rozbicie projektu na mniejsze części, które są łatwe do kontrolowania. Podzielenie architektury oprogramowania na małe obszary umożliwia pokrycie w dużym stopniu przez testy modułowe i zapewnienie wysokiej jakości produktu. Kolejną zaletą jest możliwość testowania wraz z rozwojem oprogramowania co nie powoduje przerwy na testowanie po zakończeniu pracy programistów. Klasyczna metoda FDD składa się z pięciu faz:

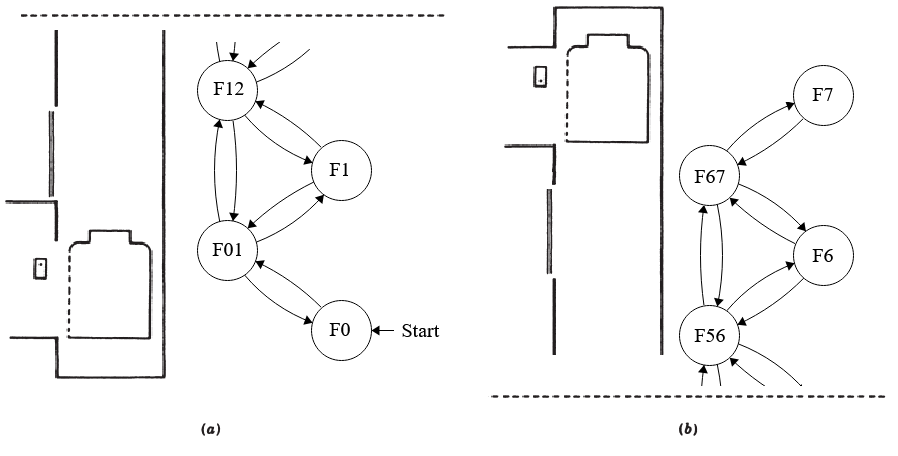
1. Opracowanie ogólnej architektury produktu
2. Budowanie listy cech produktu
3. Zaplanowanie w czasie implementacji cech
4. Projektowanie każdej z listy planowych cech
5. Budowanie oprogramowania w nastawieniu na cechy

Ostatnie dwa punkty są powtarzane aż do zakończenia implementacji wszystkich zaplanowanych cech. W przypadku rozbudowy produktu, klient może zaproponować nowe funkcje, które zostaną być dołączone w nowych iteracjach. Takie podejście do tworzenia oprogramowania jest bardzo adaptacyjne i ułatwia rozbudowę kodu. Charakterystyczne cechy tej oraz podobnych metod zostały określone jako zwinne (ang. agile)

Przy wyborze metodologii Agile wszystko zależy od wymagań projektu. Na przykład, w przypadku małych projektów, które nie są złożone, można łatwo przejść do programowania ekstremalnego, czyli takiego, gdzie nie wiadomo do końca, co się tak naprawdę robi i jak to prawidłowo zrobić, głównie opartego na obserwacji innych projektów, które odniosły sukces. Metodyki zwinne takie jak Scrum czy FDD są zalecane, jeśli chodzi o projekty oprogramowania, które są bardziej złożone i większe. Odnosząc się do tematu pracy, metodologia z nastawieniem na implementacje cech produktu była słuszna do tego projektu, gdyż pierwsze próby opisu kontrolera polegały na cechach modelu obserwowanych systemów. W początkowej fazie kluczowe było określenie cechy windy, jak będzie wyglądać kontrola jazdy wózka bazując na istniejących rozwiązaniach standardowej windy. Te wymagania sprawiły, że mimo małej złożoności projektu i jednoosobowym zespole użycie tej metodologii miało sens i w dużym stopniu pozwoliło na sukcesywną implementacje algorytmu sterowania windą.

## Architektura systemu

Bazując na cechach modelu strategii zbiorowej kontroli został stworzony algorytm kontrolera windy. Główną częścią systemie jest maszyna stanów obsługująca poruszanie się po piętrach oraz sterowanie drzwiami windy. Podstawowym założeniem dla projektu jest zaproponowanie systemu dla ośmiu pięter. Wertykalny ruch odbywa się przez przełączanie między piętrami na których winda się zatrzymuje i obsługuje pasażerów oraz na piętrach, gdzie drzwi są zamknięte i wózek windy porusza się w uporządkowanym kierunku. Na podstawie tych obserwacji zaproponowano algorytm składający się z: 8 stanów „pełnych” pięter, 7 stanów przejścia między piętrami oraz z pozostałych 4 stanów do obsługi drzwi windy, oczekiwania i bezczynności.

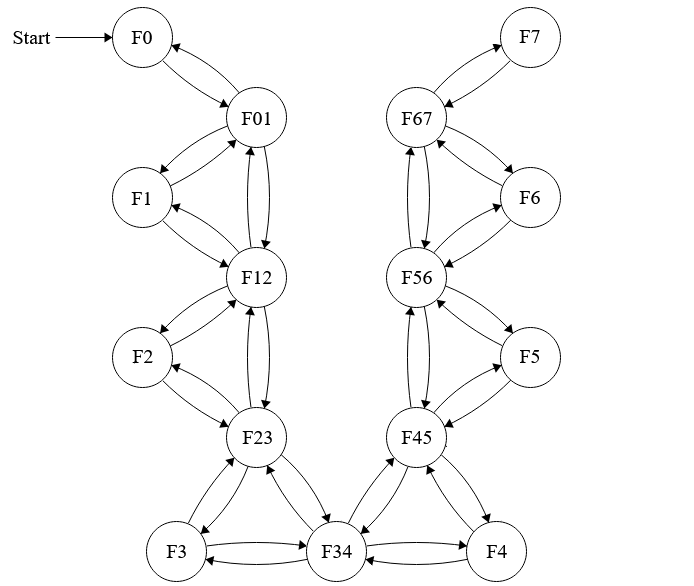


*Rysunek 2‑2 Ruch wertykalny na końcowych platformach*

***Źródło: Opracowanie własne***

Na rysunku 2-2a została przedstawiona początkowy etap działania systemu. Pierwsze uruchomienie jest wymagane na najniższym piętrze. Litera „F” pochodzi od słowa „floor” oznaczającego piętro. Jedna cyfra wskazuje na pełne piętro a dwie są skrótowym zapisem poruszania się między piętrami. Rysunku 2-2b pokazuje najwyższy przystanek wraz z poprzedzającymi stanami. Środkowa części algorytmu jest zbudowana z podobnych modułów, różniących się numerem pietra oraz innymi rejestrami przycisków do sprawdzenia. Więcej na temat algorytmu dla stanów zostanie w paragrafie 2.3.1 oraz 2.3.2.

Kierując się metodyką programowania nastawioną na cechy, pierwsze zadania polegały na stworzeniu ruchu po końcowych piętrach. Kolejnym aspektem było zatrzymywanie się na piętrach pośrednich w czasie ruchu od początku do końca linii budynku. Ostatnie dwie cechy polegały na umiejętności przewidzenia jednokrotnej lub wielokrotnej zmiany kierunku ruchu w środkowych piętrach przed dotarciem na piętro, aby uniknąć niepotrzebnej jazdy na ostatnie piętro. Z każdą iteracją maszyna stanów była modyfikowana, która ostatecznie przybrała formę przedstawioną na poniższym rysunku.



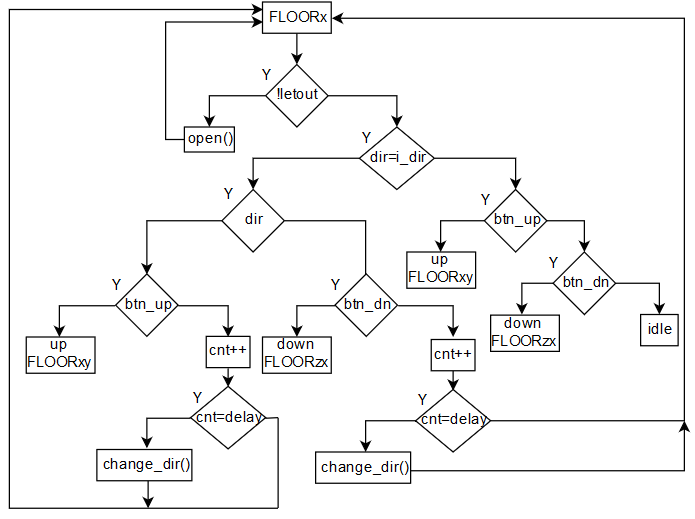
*Rysunek 2‑3 Algorytm ruchu wertykalnego dla ośmiopiętrowej windy*

***Źródło: Opracowanie własne***

Mimo skomplikowanego wyglądu, ruch wertykalny windy jest bardzo naturalny. Wózek windy porusza się po piętrach przejściowych po całej długości linii zatrzymując się na odpowiednich piętrach. Kierunek ruchu jest kontrolowany w każdym stanie by uniknąć niepotrzebnych przystanków.

### Moduł pełnego piętra

Podstawową funkcją windy jest obsługa poleceń pasażerów. Pierwszym etapem po dotarciu na platformę jest otworzenie drzwi. Realizacja odbywa się przez ustawienie flagi „*letout*” przy otwieraniu po pierwszej iteracji algorytmu. Podczas wykonywania funkcji „*open*”, następuje zmiana stanu, drzwi są otwierane z odpowiednim czasem pozwalającym na opuszczenie kabiny. System wraca do stanu FLOORx i sprawdza istniejące żądania. Początkowo sprawdzane jest, czy kierunek będzie kontynuowany warunkiem „*dir==i\_dir*”, gdzie „*dir”* to kierunek w ustalonym w tym stanie a „*i\_dir”* to kierunek w poprzednim stanie. W zależności od kierunku, sprawdzana jest kombinacja przycisków do żądania w górę (btn\_up) lub w dół (btn\_dn). Jeżeli istnieją takie polecenia to ustalany jest kolejny stan a drzwi się zamykają. W przypadku, kiedy nie istnieją poszczególne żądania, zwiększany jest licznik, który po przepełnieniu zmieni kierunek w tym stanie. System zakłada 5 sekund na wejście pasażera i naciśniecie przycisku. Dodatkowo przed zmianą kierunku odliczane jest 500 milisekund.



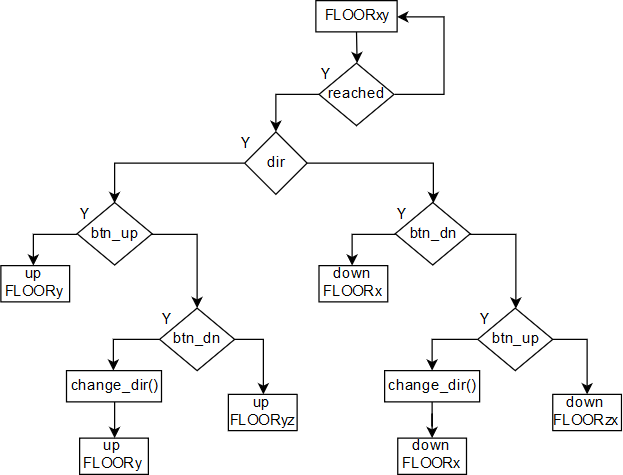
*Rysunek 2‑4 Algorytm na pełnym piętrze*

***Źródło: Opracowanie własne***

Po zmianie kierunku wewnątrz stanu, algorytm kieruje się na prawą stronę drzewa. Pierwsza sprawdzana jest kombinacja przycisków w górę choć w tej sytuacji kolejność może być dowolna, gdyż zakładamy, że kontynowanie kierunku już zostało zaniechane. Ostatecznie przy braku jakichkolwiek polecenia system przechodzi w stan oszczędzania energii. W przypadku pietra pierwszego lub ostatniego, algorytm jest upraszczany z powodu braku kolejnych pięter. Warto zauważyć, że stan bezczynności jest tylko możliwy dla pełnego piętra i następuje po 5,5 sekundach, jeśli nie został naciśniety żaden z przycisków.

### Moduł piętra przejściowego

Stan przejścia między piętrami zależy głownie od pozycji kabiny względem platformy na piętrze. Zgodnie z rysunkiem 2-3 decyzja o kolejnym stanie nie zapada, dopóki sygnał *„reached”* nie został ustawiony. Oznacza to, że winda dojechała do kolejnego piętra. Każdy cykl sekwencji następuje z narastającym zboczem zegara. Biorąc pod uwagę taktowanie 1kHz oraz prędkość standardowej winda równą 1m/s oznacza, że winda pokona tylko jeden milimetr i system podejmie decyzję czy zatrzymać się czy kontynuować ruch.



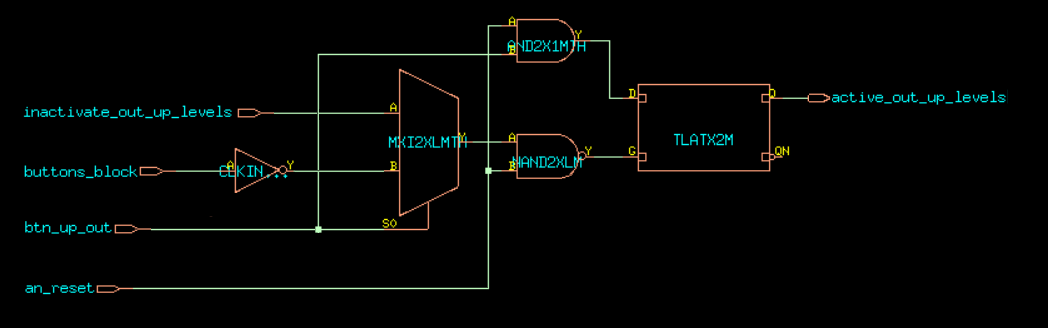
*Rysunek 2‑5 Algorytm przejazdu między piętrami*

***Źródło: Opracowanie własne***

Na rysunku 2-4 nazwą FLOORxy zostały nazwane piętra przejściowe, na przykład FLOOR45 to przejazd między czwartym a piątymi piętrem. Zapis nie oznacza kierunku jazdy, może to być ruch w górę z czwartego na piąte piętro lub odwrotnie. Po otrzymaniu od czujników informacji o dotarciu na piętro, sprawdzany jest kierunek ruchu. Jeżeli wartość wynosi „1” to winda porusza się w górę, jeżeli jest „0” to kieruje się w dół. Funkcja change\_dir() oznacza zmianę kierunku. Taka sytuacja może nastąpić, gdy powyżej danego pietra nie ma już żądań a istnieją żądania z dołu oraz pasażer na tym piętrze przywołał windę z żądaniem na dół. W tym przypadku wózek windy na kolejnej platformie będzie rozpatrywał przywołania priorytetowo z kierunkiem przeciwnym. Ostatecznie, jeśli nie ma żadnych przywołań na danym piętrze to winda kontynuuje ruch bez zatrzymania i kieruje się do kolejnego stanu przejściowego (w górę FLOOR56, w dół FLOOR34). Rozwiązanie odpytywania przez algorytm o stan przycisków przy ostatecznej fazie jazdy umożliwia elastyczną zmianę stanu przycisków wewnątrz kabiny. W kolejnym paragrafie zostanie przybliżona funkcjonalność tych resetowalnych przycisków, które mogą być zmieniane aż do dotarcia na piętro. To działanie umożliwia korektę destynacji przez pasażera i usunięcie niepotrzebnych przystanków na piętrach. Warto również wspomnieć, że przy odpytywaniu o żądania, brany jest pod uwagę czujnik tensometryczny. Zgodnie z wymagania norm, jeżeli pomiar ciężkości przekracza 90% dopuszczalnej wagi, winda nie powinna przyjmować nowych pasażerów. Realizacja polega na sprawdzeniu czy istnieje polecenie ruchu w dotychczasowym kierunku bez chęci wyjścia z pasażerów podczas przeciążonej windy.

### Interfejs użytkownika

Zgodnie z wybranym modelem strategii zbiorowej kontroli wysyłanie żądań do systemu windy odbywa się przez przyciski na zewnątrz szybu oraz wewnątrz kabiny. Na każdym piętrze znajdują się dwa przyciski sygnalizujące kontrolerowi, w którym kierunku pasażer chce się poruszać. Na pierwszym możliwym piętrze jest tylko przycisk żądania „w górę” i analogicznie na ostatnim jest tylko „w dół”.



*Rysunek 2‑6 Układ cyfrowy do rejestru przycisku zewnętrznego*

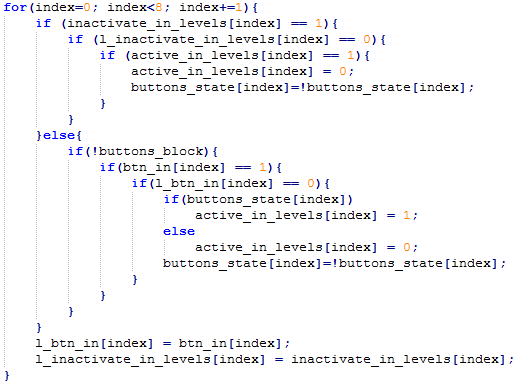
***Źródło: Opracowanie własne***

Kontroler przechowuje informacje o naciśnięciu przycisku w rejestrze. Przyciski zewnętrzne są niezależne od sygnału zegarowego i zmieniają się tylko pod wpływem czterech sygnałów wejściowych: an\_reset, btn\_up\_out, buttons\_block i inactive\_out\_up\_levels. Wartość jest przechowywana w rejestrze active\_out\_up\_levels, który jest brany pod uwagę podczas sprawdzania żądań. Sygnały wejściowe na rysunku 2-6 oznaczają:

* inactive\_out\_up\_levels – sygnał do ustawiania niskiego stanu na rejestrze active\_out\_up\_levels po wykonaniu żądania przez windę,
* buttons\_block – sygnał blokujący przyciski aktywny wysokim stanem,
* btn\_up\_out – sygnał wejściowy dołączony do fizycznego przycisku,
* an\_reset – asynchroniczny reset aktywny opadającym zboczem

W całej architekturze występuje czternaście instancji tego układu; siedem dla przycisku żądania „w górę” i siedem dla żądania „w dół”. Układ kombinacyjny jest asynchroniczny co może skłonić do pytań odnośnie zjawiska hazardu. Ten problem jest rozwiązany na poziomie syntezy logicznej w czasie optymalizacji układu.

W przeciwieństwie do przycisków na zewnątrz windy, wewnętrzne przyciski zależą od sygnału zegarowego. Dodatkową cechą jest możliwość resetowania wyboru piętra. Polega na ponownym kliknięciu przycisku, który został naciśniety omyłkowo. Algorytm tych przycisków jest bardziej skomplikowany i wymagana wyjaśnienia. Poniższy kod opisuje zachowanie resetowalnych przycisków.



*Rysunek 2‑7 Kod dla resetowalnych przycisków*

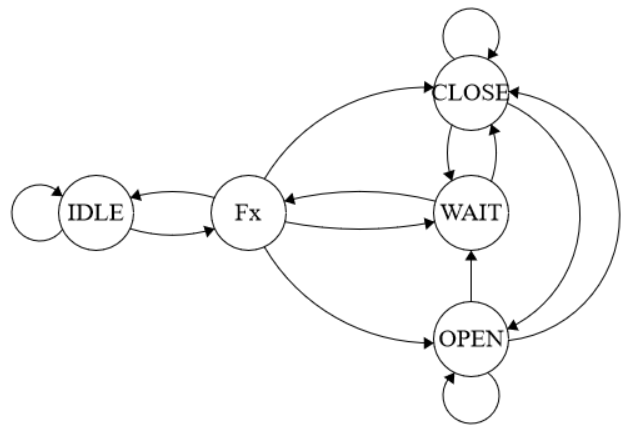
***Źródło: Opracowanie własne***

Nazwy zmiennych oznaczają na rysunku 2-7:

* active\_out\_up\_levels – rejestr zapisujący żądanie jazdy na dane piętro
* buttons\_block – sygnał blokujący przyciski aktywny wysokim stanem,
* buttons\_state – rejestr zapisujący
* btn\_in – sygnał wejściowy dołączony do fizycznego przycisku,
* l\_btn\_in – rejestr zapisujący stan wejścia btn\_in w poprzednim takcie zegarowym,
* inactivate\_in\_levels – sygnał do ustawiania niskiego stanu na rejestrze active\_in\_levels po wykonaniu żądania przez windę,
* l\_inactivate\_in\_levels – rejestr zapisujący stan sygnału inactivate\_in\_levels w poprzednim takcie zegarowym,

Do stworzenia modułu resetowalnych przycisków najważniejszy jest rejestr, który będzie zapamiętywał przyciśniecie, w tym przypadku buttons\_state. W zależności od tej wartości, przyciśnięcie aktywuje lub dezaktywuje polecenie. Przytrzymanie przycisku, nie zmienia wartości, bowiem zmiana jest tylko ważna, gdy w poprzednim takcie przycisk nie był wciśnięty. W przypadku gdy wózek windy jest w ruchu a polecenie zostało usunięte, zgodnie z algorytmem stanu przejściowego, winda będzie kontynuować kierunek aż do krańcowego piętra.

Pozostałe przyciski w kabinie służą otwierania, zamykania drzwi oraz wzywania pomocy. Działanie przycisków drzwi jest możliwe tylko w stanie pełnego piętra, gdy wózek windy się nie porusza.



*Rysunek 2‑8 Maszyna stanów obsługi drzwi*

***Źródło: Opracowanie własne***

Przejścia w algorytmie można podzielić na kilka sytuacji:

* Przejście do stanu IDLE następuje tylko w przypadku, gdy nie ma żadnych aktywnych poleceń, stan trwa w pętli przez 5 sekund,
* Przejście z piętra do stanu OPEN i kolejno WAIT następuje przy otworzeniu drzwi po dotarciu na platformę,
* Przejście z piętra do stanu CLOSE następuje przy zamknięciu drzwi przed wyruszeniem na kolejne piętro,
* Przejścia między stanami OPEN i CLOSE oznaczają sytuację, gdy użytkownik korzystając z przycisków obsługi drzwi może zamknąć lub otworzyć drzwi,
* Pętle w stanach OPEN i CLOSE oznaczają sterowanie drzwiami, aż dane polecenie zostanie wykonane,
* Przejście między stanami WAIT i CLOSE następuje przy zamykaniu drzwi, dodatkowy czas pozwala na wejście spóźnionej osoby

## Weryfikacja kodu

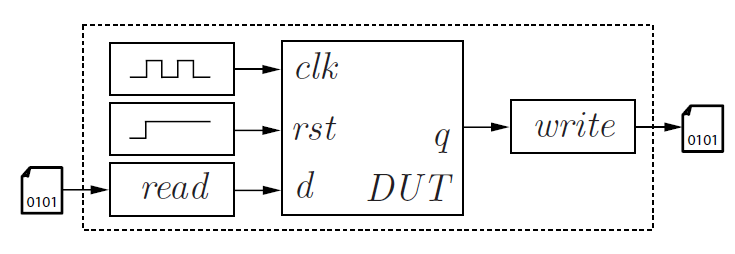
Zgodnie z metodologią tworzenia oprogramowania w oparciu o cechy, równocześnie z fazą implementacji, nowe funkcje są testowane. Przetestowanie jest kluczową sprawą przy projektowaniu układu ASIC. Układy te nie mają możliwości zmiany struktury po fabrykacji, dlatego zapewnienie pokrycia funkcjonalności przez scenariusze testowe jest sprawą kluczową. Najlepszą metodą przy testowaniu algorytmu, którego kod jest znany są testy strukturalne (ang. *white box*). Trzy wykorzystane techniki w testowaniu:

* Pokrycie linii kodu *(ang. statement coverage)* – technika testowania sprawdzająca wszystkie linijki implementowanego kodu, znane również jako pokrycie segmentu. Określa, które fragmenty programu pokrywają się ze sobą, czyli zostały wywołane przez zestaw testowy oraz wskazuje na te, których nie udało się wywołać. *Statement coverage* pozwala w prosty sposób zidentyfikować obszar niewytestowanego kod oraz znaleźć „kod martwy”, który nigdy nie zostanie wykonany.
* Pokrycie gałęzi *(ang. branch coverage)* – w języku programowania jest ściśle związana z instrukcją IF, która posiada dwa „odgałęzienia”: Prawdę i Fałsz (*True and False*). Ta metoda polega na sprawdzeniu, czy każda z gałęzi decyzji po instrukcji IF jest wykonywana co najmniej raz. Generalnie będą występowały dwa warunki, jedna gałąź do sprawdzenia, gdy warunek jest spełniony i pozostała, gdzie warunek jest fałszywy. Jeżeli dla drugiej sytuacji nie istnieje żadna instrukcja do wykonania to sprawdzana jest tylko jedna gałąź. Takie podejście zapewnia, że kod z decyzyjnego punktu widzenia jest poprawnie wykonywany.
* Analiza statyczna kodu (ang. static analysis) - metoda znajdowania błędów programu komputerowego, która jest wykonywana przez badanie linijek kodu bez wykonywania programu. Proces ten powstaje przez zrozumienie struktury kodu i może pomóc w zapewnieniu zgodności ze standardami branżowymi. Zautomatyzowane narzędzia mogą pomóc programistom w przeprowadzeniu analizy statycznej. Proces sprawdzania kodu wyłącznie przez kontrolę wizualną (na przykład przez oglądanie wydruku), bez pomocy automatycznych narzędzi, jest czasami nazywany zrozumieniem programu. Główną zaletą analizy statycznej jest fakt, że może ona ujawniać błędy, które nie pojawiają się, dopóki nie nastąpi wyjątkowy scenariusz po pewnym czasie. Niemniej jednak analiza statyczna jest tylko pierwszym krokiem w kompleksowym systemie kontroli jakości oprogramowania. Po przeprowadzeniu analizy statycznej często przeprowadza się analizę dynamiczną w celu wykrycia subtelnych defektów lub luk. Analiza dynamiczna obejmuje testowanie i ocenę programu opartego na wykonaniu. Analizy statyczne i dynamiczne, rozpatrywane razem, są czasami nazywane testami „na szkle”.

Połączenie tych trzech technik pozwala sprawdzenie funkcjonalności zaimplementowanego algorytmu w minimalnej ilości testów przy pokryciu wszystkich linijek kodu.

### Przypadki testowe

W poprzednim paragrafie wspomniano jak bardzo ważne podczas pisania kodu jest sprawdzenie jego funkcjonalności. Najbardziej powszechną metodą weryfikacji kodu w języku opisu sprzętu jest utworzenie testbench’a, tj. utworzenie instancji DUT (ang. Device Under Test), wygenerowania wektorów testowych (zestawu sygnałów wejściowych) i monitorowanie wyjścia, tak jak pokazano na rysunku 2-9.



*Rysunek 2‑9 Przykładowy testbench*

***Źródło: Opracowanie własne***

Do przetestowania przy użyciu wcześniej wspomniany metody zostało utworzono 20 scenariuszy testowych. Pierwsze dwa polegały na sprawdzeniu przycisku resetu w stanie, gdy winda była w ruchu oraz gdy winda się nie poruszała. Kolejne testy dotyczyły przycisków zewnętrznych jak i wewnętrznych, które zostały sprawdzone osobno w dwóch osobnych testach dla każdego z trzech typu\ów przycisków. Sterowanie drzwiami przy pomocy przycisków zostało sprawdzone przez dwa testy, rozdzielone na otwieranie i zamykanie. Testy czujników drzwi, czujnika ciężaru, wysłania sygnału alarmowego oraz blokowania przycisków w czasie zagrożenia zostały wykonane w czterech osobnych scenariuszach. Cechy utrzymania kierunku założone na początku implementacji zostały osobno przetestowane w trzech testach. Zachowanie windy podczas korzystania z resetowalnych przycisków zostało sprawdzone w jednym teście. Ostatecznie cały system został sprawdzony pod kątem całej funkcjonalności w dwóch rozbudowanych testach, bazując na pozostały scenariuszach testowych.

Działanie systemu zależy od pozycji drzwi, ruchu kabiny oraz czujników. W tym celu został stworzony moduł, który symuluje jazdę wózka windy. Głównym celem tej instancji było wysyłanie zwrotnych informacji po otrzymaniu poleceń takich jak zamknięte drzwi lub dotarcie do platformy.

# Projektowanie układu scalonego

## Synteza układu logicznego

Synteza to proces przyjmowania projektu zapisanego w języku opisu sprzętu, takiego jak Verilog i skompilowanie go w netlistę połączonych bramek, które są ułożone z biblioteki dostarczonej przez użytkownika. Innymi słowami, zachowanie opisane w HDL[[2]](#footnote-2) jest syntezowane do układów cyfrowych które zachowują się w ten sam sposób. Projekt po syntezie można umieścić i połączyć z innymi modułami do stworzenia całkowitego układu scalonego. Istnieją trzy typy syntezy: synteza logiczna, która mapuje projekty poziomów bramek do biblioteki logicznej, syntezy RTL[[3]](#footnote-3), tworząc poziom bramki netlist z zachowania RTL, synteza behawioralna (wysokiego poziomu), która tworzy RTL opis z reprezentacji algorytmicznej. Wszystkie z nich używają plików typu *Liberty* do opisania zachowania bramek. Obsługuje szeroki zakres płaskich i hierarchicznych stylów projektowania a także służy do optymalizacji prędkości, powierzchnię, i moc układów kombinacyjnych, jak i sekwencyjnych.

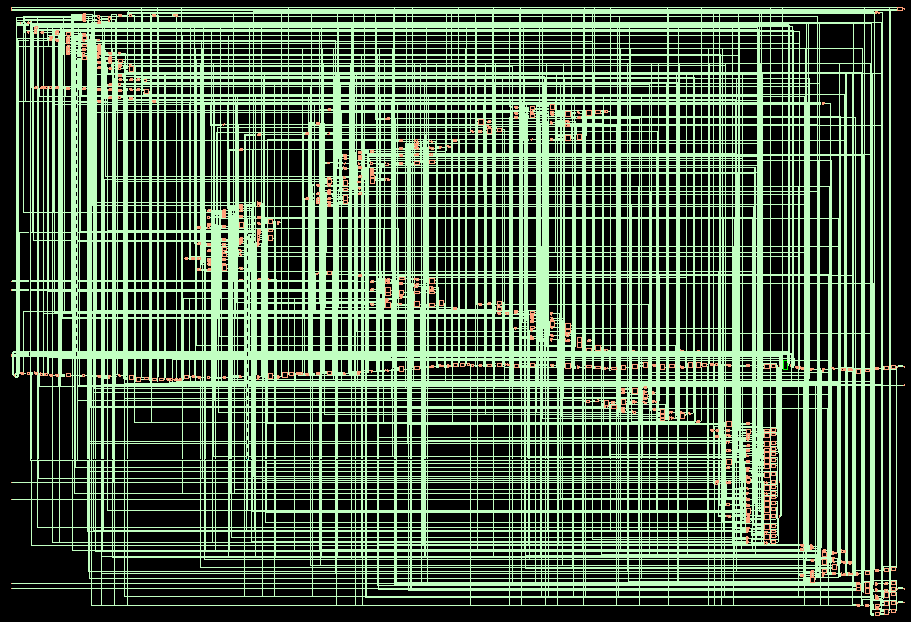
Korzystając z oprogramowania Cadence Genus, dobrą praktyką przy tworzeniu dużych projektów jest uruchamianie syntezy ze skryptu. Pozwala to na zaoszczędzenie czasu oraz wykonanie po kolei wszystkich operacji za projektanta. Typowy skrypt syntezy układu logicznego do układu strukturalnego składa się z następujących poleceń:

1. Ustawienie zmiennych i parametrów, które będą używane do syntezowania. Najczęściej są to ścieżki do plików z modułami do syntezy oraz biblioteki, które zawierają opisy strukturalne.
2. Odczyt plików, analiza pod kątem składni językowej oraz translacja.
3. Ustawienie ograniczeń układu w związku z taktowaniem układu oraz opóźnienia wejścia i wyjścia.
4. Mapowanie opisu w strukturalny układ cyfrowy.
5. Utworzenie raportów o ilości komórek, powierzchni oraz
6. Zapisanie plików wynikowych.

Skrypt został dołączony w załączniku A z odpowiednimi komentarzami. Po ustawieniu zmiennych skrypt zapisuje ścieżki do bibliotek. Najważniejsze są dwa parametry **init\_lib\_search\_path** oraz **init\_hdl\_search\_path**, które zależą od modułu do syntezy oraz docelowej technologii. Zanim kompilator projektu może zsyntetyzować projekt, konieczne jest podanie, które biblioteki powinny być używane do tworzenia. Oprócz informacji o komórce, dodawane są także modele połączeń opisujące warstwy komórek oraz i opóźnienia czasowe.

Kolejną ważną rzeczą jest ustawienie ograniczeń nałożonych przez taktowanie sygnału zegarowe. Sterowanie układami mechaniki jak w windzie nie wymaga dużych częstotliwości, sygnał 1kHz jest porównywalnie wolny w stosunku do taktowań dzisiejszych mikroprocesorów. W skrypcie taktowanie jest ustawanie na 10kHz, gdyż mniejsze częstotliwości nie są już możliwe do ustawienia. Genus przyjmuje wartości okresu w pikosekundach. W przypadku tego projektu analiza czasowa jest dodatkową informacją o projekcie. Przy wspomnianej wcześniej częstotliwości 1kHz, spodziewany jest wynik pozytywny po sprawdzeniu zachowania granic

Po ustawieniu wszystkich pozostałych zmiennych, skrypt czyta moduł zapisany w Verilogu i uruchamia proces translacji. Głównym zadaniem w tym etapie jest przetłumaczenie projektu napisanego w języku behawioralnym w technologii niezależne listy bramek logicznych. Występuje sprawdzanie składni, a następnie buduje projekt przy użyciu komponentów ogólnych. Kolejno wszystkie ustawienia związane z sygnałem zegarowym są dodawane do projektu i sprawdzane jest czy wszystkie moduły zostały połączone bez problemów. Ewentualne błędy lub ostrzeżenia przed mapowaniem do układu strukturalne są *check\_design -unresolved.* Ostatecznie komenda *synthesize -to\_mapped* wykonuje właściwą syntezę. Dodatkowy argument -to\_mapped wskazuje syntezerowi by używał bibliotek dodanych przez użytkowania niż korzystał z wbudowanych bibliotek dostępnych w oprogramowaniu Genus. Końcowy rezultat jest widoczny na rysunku 3-1. Po przybliżeniu widoku zauważalne są struktury, które zostaną wyeksportowane w pliku wynikowym. Schemat w złożonych układach scalonych niestety nie ma praktycznego zastosowania oprócz funkcji poglądowej. Po zakończeniu działania na strukturze układu, skrypt wykonuje zapisy trzech raportów o pokryciu ograniczeń czasowych, poborze mocy przez układ oraz ilości bramek logicznych w strukturze. Ostatnim krokiem jest zapisane pliku strukturalnego w języku Verilog, ograniczeń czasowych projektu oraz plików opisujących komórki w programie Innovus. Są to kluczowe pliki w kolejnych etapach projektowania całego układu do fabrykacji.



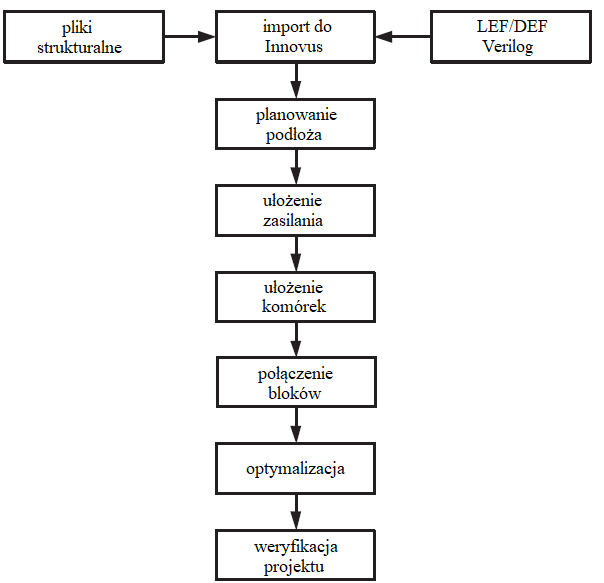
*Rysunek 3‑1 Struktura układu po syntezie*

***Źródło: Opracowanie własne***

## Symulacja po syntezie

## Struktura topografii

Proces polegający na pobraniu plików z opisem strukturalnym i stworzenie z niego fizycznego układu nazywa się określeniem układania i połączenia (ang. *place&route*). Jak sugeruje nazwa, ten etap składa się z dwóch kroków. Pierwszy to umieszczenie, obejmuje podjęcie decyzji, gdzie umieścić wszystkie komórki lub układy komórek w ograniczonej przestrzeni. Następnie następuje połączenie tych układów, w którym decyduje się o dokładnym rozmieszczeniu wszystkich przewodów potrzebnych do podłączenia umieszczonych komponentów. Najczęściej elementy łączy się przez warstwy metali i przelotki. Ten krok musi zaimplementować wszystkie pożądane połączenia, przestrzegając zasad i ograniczeń procesu produkcyjnego.



*Rysunek 3‑2*

***Źródło: Opracowanie własne***

Cały projekt topografii jest wykonany w programie Cadence Innovus. Proces ma więcej ważnych zmian wizualnych niż synteza, dlatego warto kolejne punkty w wykonywać nadzorując zmiany w widoku.

### Generacja układu

Do stworzenia topografii, pliki LEF muszą zostać zaimportowane w celu utworzenia biblioteki fizycznej i logicznej. Te pliki zawierają informacje o celu technologii, komórkach i klocki. Jeśli używane są komórki pamięci, odpowiedni LEF les muszą być również czytane. Import można wykonać za pomocą GUI lub za pomocą skryptu. Dołączone pliki LEF muszą być zgodne z użytymi do syntezy, aby zachować jednolitość komórek. Podstawowe pliki, które należy przygotować przed przystąpienie do importu struktury do programu Innovus to:

* Pliki informacji technologicznych dla komórek i makr w formacie wymiany bibliotek (LEF)
* Netlista po syntezie w formacie Verilog
* Ograniczenia projektowe dla netlisty struktury w formacie Standard Design Constraint (SDC)
* Standardowe informacje o taktowaniu komórki w formacie biblioteki czasowej (TLF)

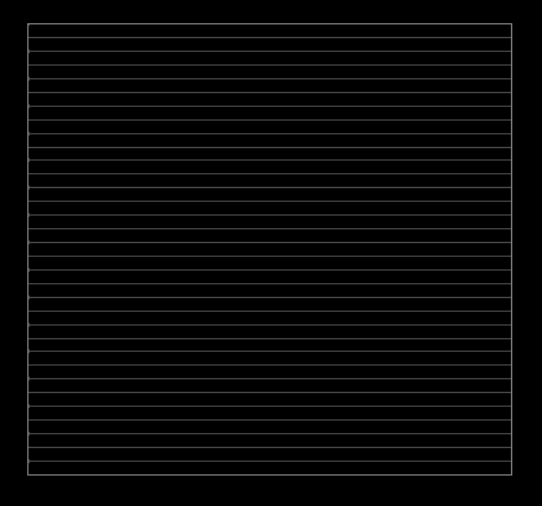
Posługując się skryptem z poprzedniego rozdziału o syntezie w programie Genus, wszystkie wymienione wyżej pliki są generowanie automatycznie. Dwa ostanie pliki nie mają swojego miejsca do zadeklarowanie podczas importowania. Dołączenie plików odbywa się plik MMMC w formacie mmode.tcl. Plik ten wywołuje komendy dołączające oba pliki do zaimportowanego projektu.

Pliki LEF wymagane dla konkretnego projektu zależą od zastosowanej technologii i obecności wszelkich twardych makr w projekcie. Wszystkie informacje o procesie fizycznym są zawarte w pliku - dokładniej TSMC 130 nm, 8-warstwowa informacja o procesie metalowym jest podana w plik tsmc13fsg\_8lm\_tech.lef. Podobnie jest z informacją makro standardowej komórki TSMC 130 nm dostarczone w tsmc13nvt \_macros.lef. Dodatkowo, wszelkie twarde makra obecne w projekcie muszą mają również odpowiedni plik LEF, ponieważ nie można ich zbudować z komórek liści obecnych w standardowa biblioteka makra LEF.

Netlista projektu jest zwykle niezmodyfikowanym plikiem wynikowym z takiego narzędzia syntezy jako kompilator projektu. Jedną z potencjalnych kwestii jest to, że wszystkie typy komórek instancji na liście musi być unikalne. W dołączonym skrypcie syntezy, generacja netlisty odbywa się komendą write\_hdl.

Plik SDC zawiera ograniczenia, takie jak warunki pracy, modele obciążenia połączeń, czasy narastania i opadania sygnału zegarowego i opóźnienia wejścia / wyjścia. Chociaż nie jest to konieczne jako plik wejściowy do, wyniki są poprawione, jeśli ten plik jest dołączony. Analogicznie jak netlistę, można go utworzyć z poziomu kompilatora projektu za pomocą komendy write\_sdc po syntezie.

Informacje o czasie dla standardowych komórek są dostarczane przez plik TLF. Zazwyczaj jest to zapewnione przez dostawcę technologii i nie ma potrzeby generacji tych informacji. Po połączeniu tych wszystkich plików zostaje stworzony pogląd w trybie layout z



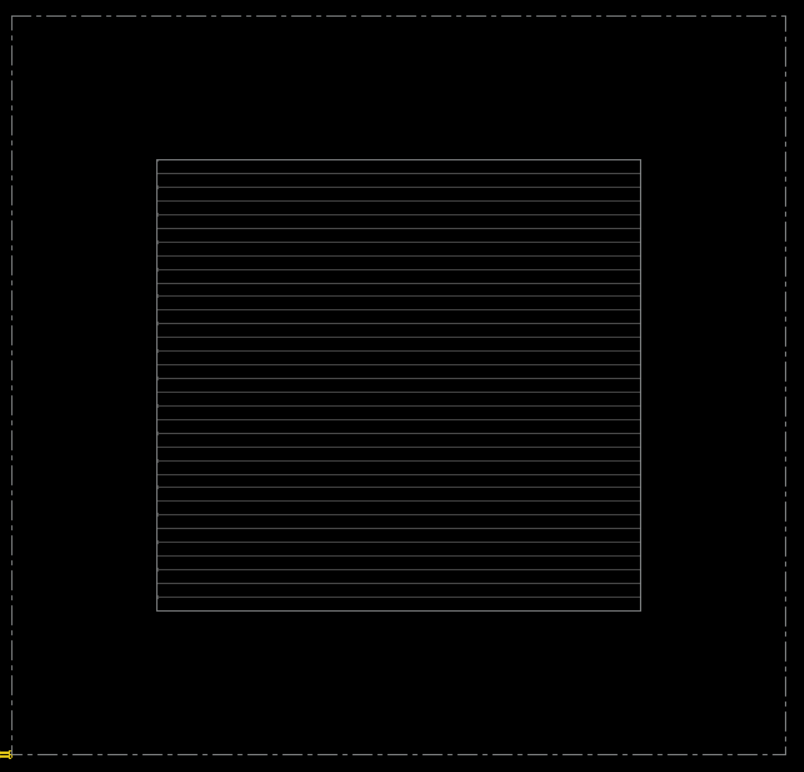
*Rysunek 3‑3 Komórki standardowe po imporcie do programu Innovus*

***Źródło: Opracowanie własne***

Przy importowaniu struktury wymagane jest podanie nazw sygnału zasilania i masy. Dobra praktyką jest sprawdzenie w pliku LEF nazw stosowanych przy opisach strukturalnych komórek standardowych. Producenci bibliotek z reguły stosują dwa różne opisy: vdd! i gnd! oraz VDD i VSS. Nieprawidłowe nazwy sygnału mogą powodować problemy z nałożeniem warstwa zasilających wokół rdzenia układu scalonego.

### Planowanie podłoża

Planowanie podłoża projektu jest pierwszym krokiem po zaimportowaniu plików technologicznych na poziomie netlisty bramek. W tym kroku plan piętra musi być przygotowany na zasilanie, rozmieszczenie klocków wejścia-wyjścia, twarde makra (np. pamięć RAM) i standardowe komórki. Projektant jest zobowiązany do określenia rozmiaru matrycy przez rozmieszczenie IO[[4]](#footnote-4) na ramie podkładki i ramie podkładki do odległości rdzenia. Co więcej, rdzenie wiersze muszą zostać przerwane, gdy powinny zostać umieszczone twarde makra. W przypadku tego projektu, Istnieje tylko jedna instancja, która bardzo ułatwia planowanie podłoża. Jedyna rzeczą jest zostawienie miejsca na warstwy zasilania. Zaplanowany obszar został oznaczony przerywana linią na poniższym rysunku.

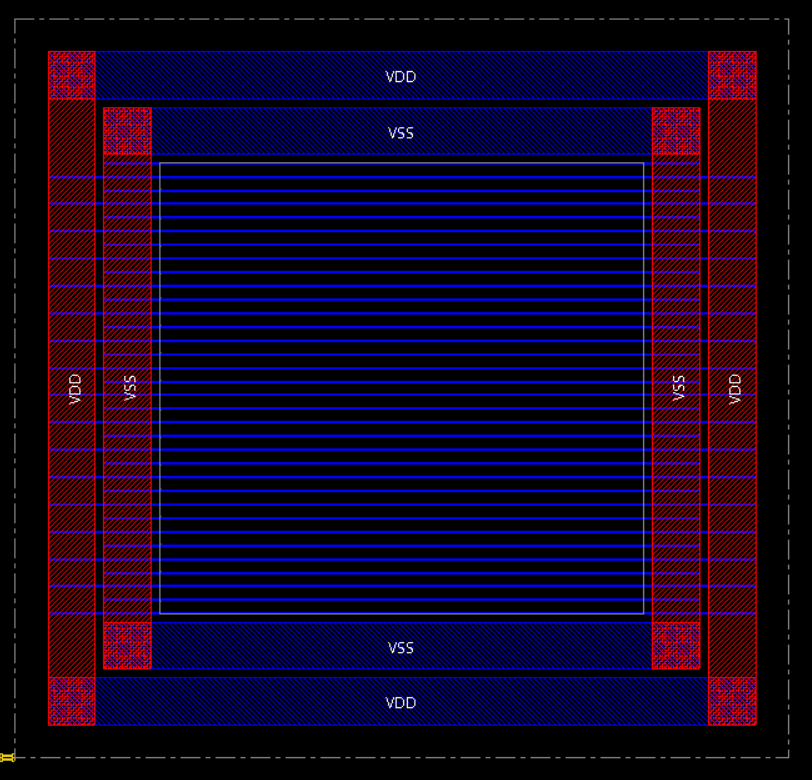


*Rysunek 3‑4 Układ po planowaniu podłoża*

***Źródło: Opracowanie własne***

### Układanie warstw zasilania

Planowanie i układanie warstw zasilnia jest niezbędne do dostarczenia zasilania do standardowych komórek i makr. Pierwszy krokiem jest dodanie pierścieni, które są umieszczone na obwodzie rdzenia i zasilania do pasków, które przenoszą moc przez chip. Oddzielny pierścień służy do zasilania (VDD) i masy (VSS). Do poprowadzenia horyzontalnych ścieżek zasilania jest używana warstwa METAL1 a do części wertykalnych jest wykorzystana warstwa METAL2. Paski mocy są dodane na warstwie METAL1 i są umieszczone w odległości 100 μm; ta wartość jest kompromisem między zapewnieniem wystarczającego rozkładu mocy linii przy minimalizacji blokowania trasowania spowodowanego przestrzenią zajmowaną przez linie zasilające.[17] Przy tworzeniu warstw zasilania dla mało skomplikowanych układów, wystarczaje jest stworzeni pierścieni zasilania wokół pierścienia bez potrzeby dodawania pasków mocy. Paski dodajemy przy użyciu instrukcji *special route.*

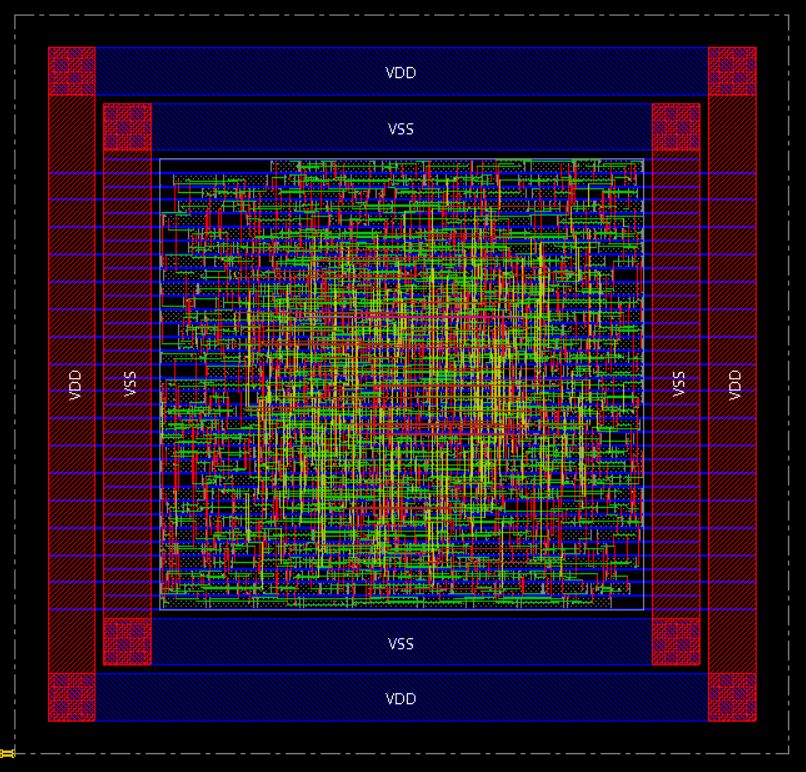


*Rysunek 3‑5 Pierscienie i paski zasilania dołączone do układu scalonego*

***Źródło: Opracowanie własne***

### Łączenie komórek standardowych

Po zaplanowaniu podłoża układu oraz dołączeniu warstw zasilających standardowe rozmieszczenie komórek i makr. Po zakończeniu standardowe komórki i bloki makr, które implementują funkcjonalność netlist, powinny być umieszczone w rzędach topografii.

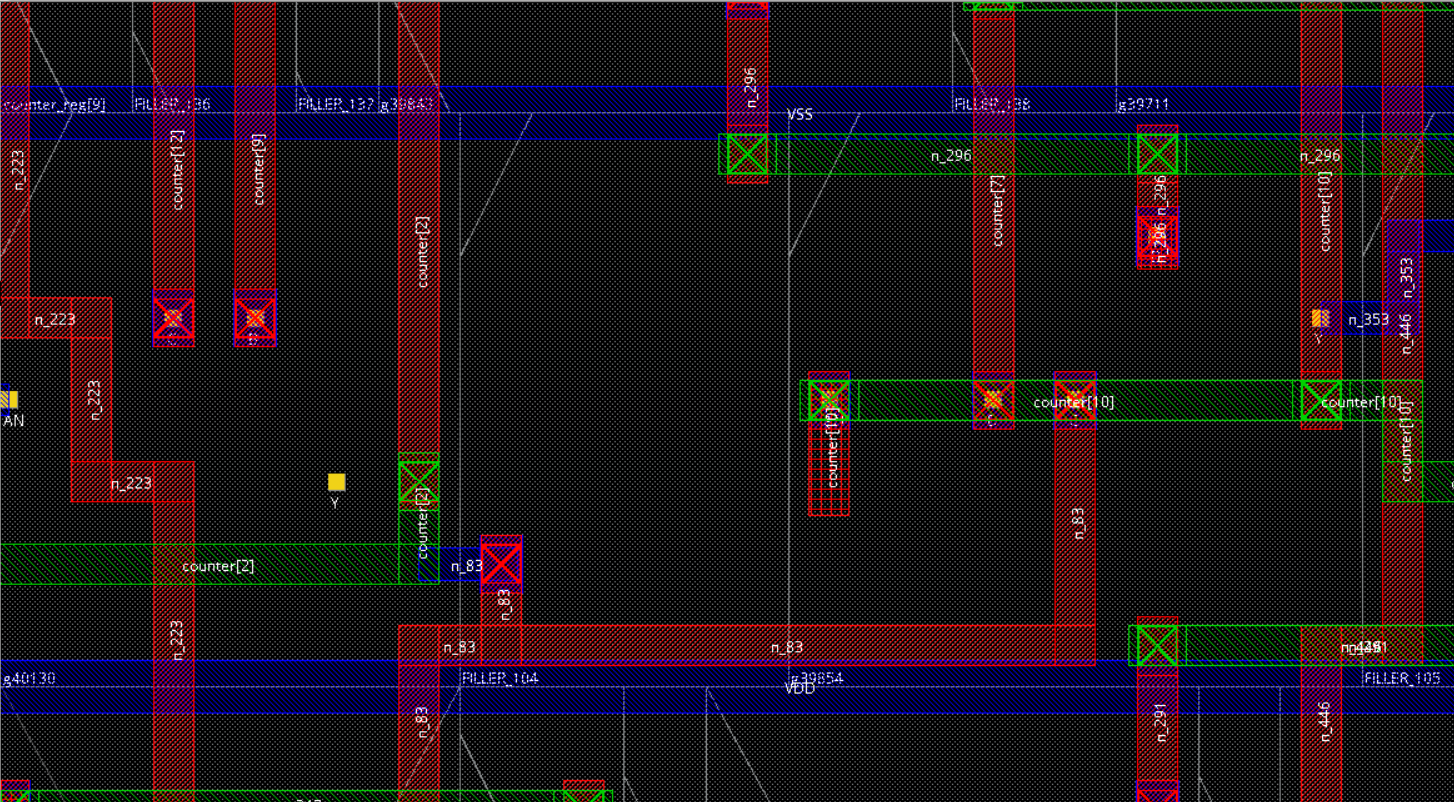


*Rysunek 3‑6 Widok fizyczny po ułożeniu komórek standardowych*

***Źródło: Opracowanie własne***

Pierwsze ułożenie struktury i połączenie warstwami metali jest nieoptymalne i wymaga dalszego procesowania. Im większy projekt tym większa szansa, że duża ilość połączeń może zostać usunięta. Podczas projektowania ważne jest doprowadzanie sygnału zegarowego, który po optymalizacji zmniejsza swoje opóźnienie dla „najgorszej ścieżki”.

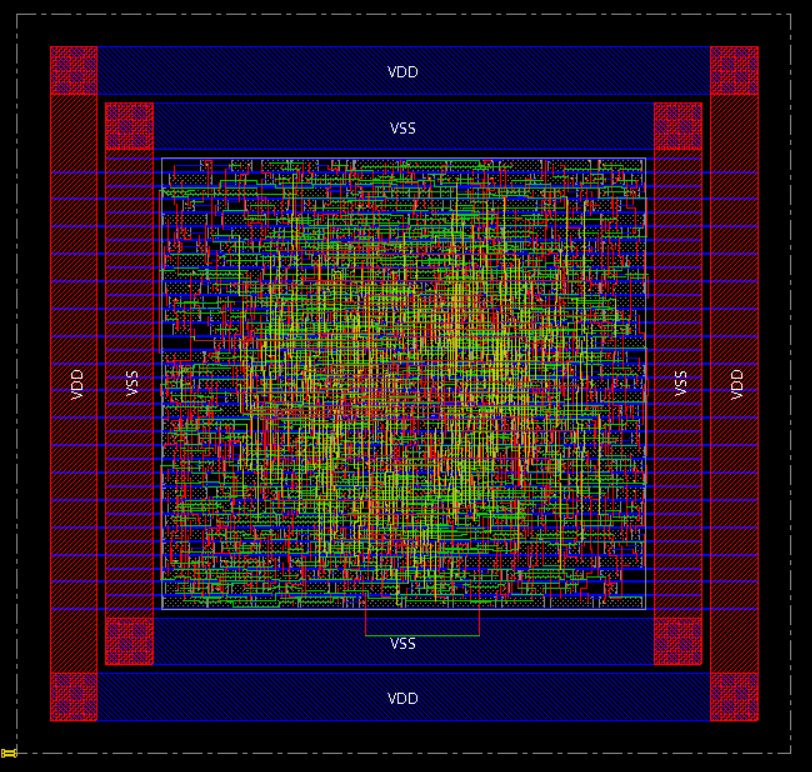
### Optymalizacja



*Rysunek 3‑7 Komórka wypełniająca przerwę między komórkami standardowymi*

***Źródło: Opracowanie własne***

Zaawansowany narzedzie do łączenia NanoRoute obsługuje wszystkie wyzwania routingu zarówno na poziomie bloków, jak i pełnych układów. Łączy on w sobie charakterystykę routera opartego na sieci z elastycznością poza siecią, a jednocześnie ocenia i optymalizuje topologię połączeń w oparciu o efekty 3D dotyczące czasu, obszaru, mocy, zdolności produkcyjnej i wydajności. Zasilany przez super-wątkową płytę montażową rozwiązanie NanoRoute kończy miliony sieci łączności na godzinę. Zapewnia najwyższą jakość wyników w ułamku czasu zajmowanego przez inne routery na rynku. Technologia NanoRoute jest również w pełni wyposażona, aby sprostać wymaganiom projektowym 20/16 / 14nm, w tym podwójnemu modelowaniu i obsłudze FinFET. Wykorzystując podejście poprawne według konstrukcji, router NanoRoute rozwiązuje potencjalne konflikty podwójnego wzorca w locie dla topologii routingu, która jest nie tylko podwójnym wzorcowaniem i zaawansowaną poprawką DRC za pierwszym razem, ale jest również bardziej wydajna obszarowo.



*Rysunek 3‑8 koncowa topografia*

***Źródło: Opracowanie własne***

Analiza mocy i wydajności energetycznej na wysokich poziomach abstrakcji, takich jak RTL lub gatelevel, zapewnia wyniki rzędu wielkości szybsze niż analizy na poziomie fizycznym. Jednak dokładność cierpi z powodu utraty szczegółów i założeń przy wyższych poziomach abstrakcji. Aby uzyskać dokładne szacunki zużycia energii i energii, ważne jest przeprowadzenie analizy niższych poziomów w celu porównania z wynikami wyższego poziomu, co pozwoli na ustalenie błędu i poprawienie wyników analizy z powrotem do analiz wyższych poziomów. Ponadto bardziej dokładne dane dotyczące powierzchni i czasu można uzyskać z projektu po układzie, który może być również wykorzystany do poprawy dokładności szacowania poziomu RTL. W tym rozdziale opisano zarówno kroki wymagane do wykonania fizycznego układu (w tym planowanie pięter), jak i miejsce i trasa na koprocesorze na poziomie bramy wraz z analizą mocy koprocesora poziomu fizycznego.

### Weryfikacja

Connectivty

geometrys

Drc

# Podsumowanie i wnioski

# Streszczenie

# Summary

# Spis stosowanych skrótów

ASIC –

FDD –

FPGA – Field Programmable Gate Array

# Słowa kluczowe

* ASIC,
* winda,

# Keywords

* ASIC,
* elevator,

# Bibliografia

1. Antoszkiewicz K., *Generacja i synteza częstotliwości*, Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2015, str. 260-271.
2. Bhasker J., *A VHDL Primer,* Upper Saddle River: Prentice Hall PTR, 1999.
3. Naylor D., Jones S., *VHDL: a logic synthesis approach,* London: Chapman & Hall, 1997.
4. Pellegrino R., *The Electronic Arts of Sound and Light*, New York: Van Nostrand Reinhold Company, 1983.
5. Strakosch The verical transportation nhandbook
6. Elevator control strategies j axelsson,s bernlind
7. Algorytm Scan http://aragorn.pb.bialystok.pl/~wkwedlo/OS1-9.pdf
8. Elevators https://www.slideshare.net/saramesallam/shanghai-tower-elevator-and-escalators
9. Shanghai towr <https://en.wikipedia.org/wiki/Shanghai_Tower>
10. Elevator gruop <https://link.springer.com/content/pdf/10.1023%2FA%3A1007518724497.pdf>
11. Designing Elevator Installations Using Modern Estimates of Passenger Demand <https://www.peters-research.com/index.php?option=com_content&view=article&id=57%3Alift-passenger-traffic-patterns-applications-current-knowledge-and-measurement&catid=3%3Apapers&Itemid=1>
12. Elevators UE <https://ec.europa.eu/growth/sectors/mechanical-engineering/lifts_en>
13. Directive 95/16/EC - Lifts - European standards for existing lifts – references
14. <http://www.elca-eu.org/>
15. [*https://www.slideshare.net/yogeshwarank1/asic-design-flow-71285160*](https://www.slideshare.net/yogeshwarank1/asic-design-flow-71285160)
16. [*http://www.abpenterprises.com/gen2-premier.html*](http://www.abpenterprises.com/gen2-premier.html)
17. *Paul morgan* Energy Analysis and Optimisation Techniques for

Automatically Synthesised Coprocessors

http://www.worldofarchi.com/2012/08/shanghai-tower-elevator-system-drawings.html

https://vignette.wikia.nocookie.net/elevation/images/a/aa/BigTransit.jpg/revision/latest?cb=20130115104717&format=original

http://www.neii.org/destdispatch.cfm

<https://www.tripadvisor.com.au/LocationPhotoDirectLink-g53449-d294369-i310683750-Cathedral_of_Learning-Pittsburgh_Pennsylvania.html>

<https://elevation.fandom.com/wiki/Destination_dispatch>

<https://www.retrofitmagazine.com/destination-dispatch-system-improves-buildings-elevator-performance-and-efficiency/>

<http://global.ctbuh.org/resources/papers/download/399-elevator-traffic-simulation-procedure.pdf>

https://link.springer.com/content/pdf/10.1023%2FA%3A1007518724497.pdf

# Dodatek A.

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \* Script Name : Genus Legacy syntesis script

date

**set** LOCAL\_DIR **"[exec pwd]/"**

**set** LIB\_PATH **"/tools/DesignKits/pssw/TIMING13 /tools/DesignKits/pssw/LEF"**

**set** RTL\_PATH **"$LOCAL\_DIR/RTL"**

**set** myFiles **[list** $LOCAL\_DIR**/**RTL**/**elevator.v $LOCAL\_DIR**/**RTL**/**buttons\_res.v**]** ;#

**set** basename elevator ;# name of top level module

**set** myClk clock ;# clock name

**set** myPeriod\_ps 1000000000 ;# Clock period in ps 1kHz

**set** myInDelay\_ps 2000000 ;# delay from clock to inputs valid 2us

**set** myOutDelay\_ps 2000000 ;# delay from clock to output valid 2us

**set** runname RTL ;# name appended to output files

# Baseline Libraries

**set** LIB\_LIST **{** \

tt\_g\_1v20\_25c.lib \

tt\_hvt\_1v20\_25c.lib \

**}**

**set** LEF\_LIST **{** \

tsmc13fsg\_8lm\_tech.lef \

tsmc13g\_m\_macros.lef \

tsmc13hvt\_m\_macros.lef \

**}**

**set** CAP\_TABLE\_FILE **/**tools**/**DesignKits**/**pssw**/**tsmc13fsg.capTbl

set\_attribute hdl\_track\_filename\_row\_col true **/**

set\_attribute lp\_power\_unit mW **/**

set\_attribute init\_lib\_search\_path $LIB\_PATH **/**

set\_attribute init\_hdl\_search\_path $RTL\_PATH **/**

set\_attribute library $LIB\_LIST

set\_attribute lef\_library $LEF\_LIST /

set\_attribute cap\_table\_file $CAP\_TABLE\_FILE /

# Analyze and Elaborate the HDL files

read\_hdl elevator.v

elaborate elevator

# Apply Constraints and generate clocks

set clock [define\_clock -period ${myPeriod\_ps} -name ${myClk} [clock\_ports]]

external\_delay -input $myInDelay\_ps -clock ${myClk} [find / -port ports\_in/\*]

external\_delay -output $myOutDelay\_ps -clock ${myClk} [find / -portports\_out/\*]

# check that the design is OK so far

check\_design -unresolved

report timing -lint

# Synthesize the design to the target library

synthesize -to\_mapped

# Write out the reports

report timing > ${basename}\_${runname}\_timing.rep

report gates > ${basename}\_${runname}\_cell.rep

report power > ${basename}\_${runname}\_power.rep

# Write out the structural Verilog and sdc files

write\_hdl -mapped > ${basename}\_${runname}.v

write\_sdc > ${basename}\_${runname}.sdc

write\_design ${basename} -gzip\_files -innovus -tcf

# Dodatek B. Spis zawartości dołączonej płyty CD

– strona internetowa zawarta w bibliografii na pozycji,

– strona internetowa zawarta w bibliografii na pozycji,

Mateusz\_Dyrdół\_276528.doc – Tekst pracy zapisany w formacie MS Word,

Mateusz\_Dyrdół\_276528.pdf – Tekst pracy zapisany w formacie rozszerzonym,

# Spis ilustracji

[*Rysunek 1‑1 Ruch głowicy dysku twardego zgodny z algorytmem SCAN* 9](#_Toc12485187)

[*Rysunek 1‑2 System strefowy w budynku Shanghai Tower* 11](#_Toc12485188)

[*Rysunek 1‑3 Przykład połączenia dwóch stref* 12](#_Toc12485189)

[*Rysunek 1‑4 Model windy dwuprzystankowej* 15](#_Toc12485190)

[*Rysunek 1‑5 Wybór piętra w systemie z wysyłką docelową* 21](#_Toc12485191)

[*Rysunek 1‑6 Przyciski żądań w windzie z wysyłką docelową i klasycznej windzie* 22](#_Toc12485192)

[*Rysunek 1‑7 Identyfikacja wind w systemie wysyłkowym* 23](#_Toc12485193)

[*Rysunek 1‑8 Nowa generacja mechanizmów wind GEN2 LIFT firmy OTIS* 24](#_Toc12485194)

[*Rysunek 2‑1 Schemat projektowania układu ASIC* 26](#_Toc12485195)

[*Rysunek 2‑2 Ruch wertykalny na końcowych platformach* 28](#_Toc12485196)

[*Rysunek 2‑3 Algorytm ruchu wertykalnego dla ośmiopiętrowej windy* 29](#_Toc12485197)

[*Rysunek 2‑4 Algorytm na pełnym piętrze* 30](#_Toc12485198)

[*Rysunek 2‑5 Algorytm przejazdu między piętrami* 31](#_Toc12485199)

[*Rysunek 2‑6 Układ cyfrowy do rejestru przycisku zewnętrznego* 33](#_Toc12485200)

[*Rysunek 2‑7 Kod dla resetowalnych przycisków* 34](#_Toc12485201)

[*Rysunek 2‑8 Maszyna stanów obsługi drzwi* 35](#_Toc12485202)

[*Rysunek 2‑9 Przykładowy testbench* 37](#_Toc12485203)

[*Rysunek 3‑1 Struktura układu po syntezie* 41](#_Toc12485204)

[*Rysunek 3‑2* 42](#_Toc12485205)

[*Rysunek 3‑3 Komórki standardowe po imporcie do programu Innovus* 44](#_Toc12485206)

[*Rysunek 3‑4 Układ po planowaniu podłoża* 45](#_Toc12485207)

[*Rysunek 3‑5* 46](#_Toc12485208)

[*Rysunek 3‑6 Komórka wypełniająca przerwę między komórkami standardowymi* 47](#_Toc12485209)

# Spis tabel

**Nie można odnaleźć pozycji dla spisu ilustracji.**

1. Norma energtyczna – (należy dopisać) [↑](#footnote-ref-1)
2. HDL – Hardware Description Language – opis jezyku sprzetu [↑](#footnote-ref-2)
3. RTL – Register Transfer Level oznacza metodę proejkowania [↑](#footnote-ref-3)
4. IO – ang. in-out, porty wejścia-wyjścia [↑](#footnote-ref-4)