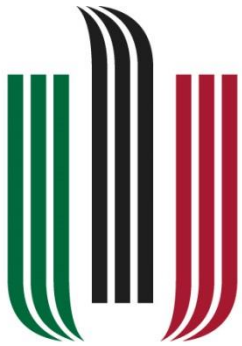


AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

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(-a) 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 2018 r. poz. 1191 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(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

……

(czytelny podpis studenta)

Spis treści

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 których trzeba dopasować odpowiednie podejście transportu ludzi na piętra. Zwiększenie poziomu niezawodności, komfortu, bezpieczeństwa i optymalizacji zużycia energii windy to priorytetowe zadania. Wybór odpowiedniej strategii przez projektanta świadczy o wysokiej jakości usługi i jest opłacalną inwestycją. 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 (ang. *top-down)*. Opis behawioralny układu cyfrowego został napisany 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 łatwy w obsłudze dla użytkownika, dlatego sterowanie zgodnie ze strategią zbiorowej kontroli odbywa się za pomocą przycisków wewnątrz i na zewnątrz windy. Algorytm opisany w tej pracy został stworzony dla ośmiopiętrowego budynku. Analogiczna funkcjonalność pięter została wykorzystana 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 celu podróży. 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, zgodnie z normą PN-EN ISO 25745-1. 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.

Rozdział 1
Technologia dźwigów osobowych

1.1 Pierwsze systemy dźwigowe

Prymitywne windy były używane już w III wieku p.n.e. i były obsługiwane przez ludzi, zwierzęta lub koła wodne. W 1743 r. zbudowano dla króla Ludwika XV przeciwwagę napędzaną przez człowieka jako 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 zaprojektowano nie więcej niż 150 lat temu. Windy parowe i hydrauliczne zostały już wprowadzone w 1852, kiedy Elisha Otis stworzył jeden z najważniejszych wynalazków związanych z windą: sprzęgło, które uniemożliwia 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 ewoluowały od napędzanych parą do mocy hydraulicznej. Pierwsze hydrauliczne dźwigi zostały zaprojektowane z wykorzystaniem ciśnienia wody jako źródła energii. Technologia silnika i metody sterowania zmieniały się szybko, a elektryczność stała się akceptowanym źródłem energii. Pierwsza elektryczna winda została zbudowana przez niemieckiego wynalazcę Wernera 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 urządzenia, których zadaniem jest zarówno przewóz osób, jak i towaru,
- dźwigi szpitalne, zwykle przeznaczone 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 budowane głównie na podstawie logiki przekaźnikowej, układów PLC lub mikrokontrolerów. Windę można uznać za złożony system reaktywny, który wymaga równoległego przetwarzania zdarzeń z wieloma wejściami i wyjściami.

1.2 Strategie sterowania windą

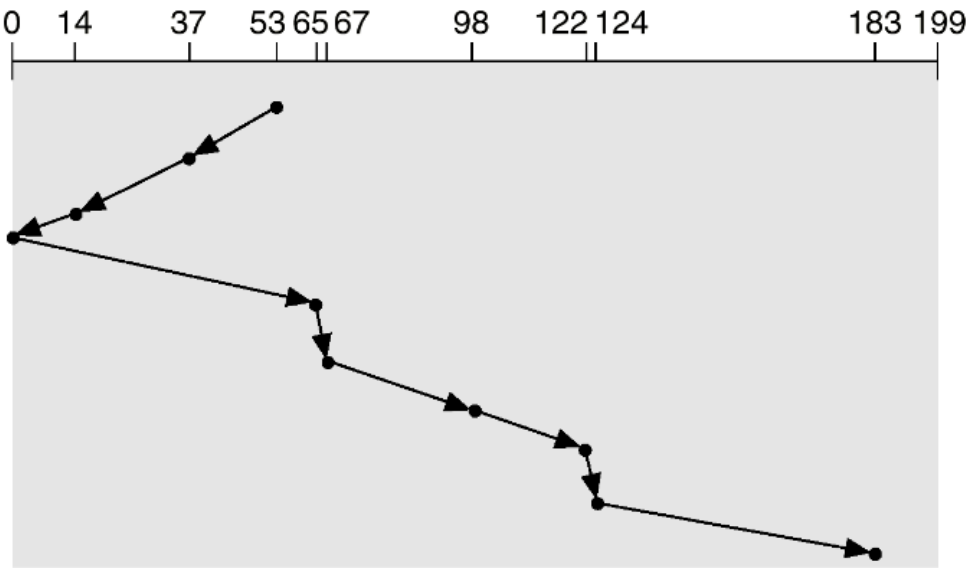
Przez lata opracowano wiele strategii sterowania windą. Pierwsze z nich były to bardzo proste strategie a dziś powszechne są algorytmy wykorzystujące sztuczną inteligencję i uczenie maszynowe w celu planowania ruchów wind. W wysokich budynkach z dużym przepływem pasażerów, zwłaszcza w biurach, strategia dla systemów 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 we wszelakich 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.

1.2.1 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 opuści ją ostatni pasażer. Jedyną wadą tej strategii jest zjawisko zwane *grupowaniem*, gdzie kilka systemów 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. [1]

Popularność tej strategii spowodowała użycie jej w innej technologii. Podobnie jest w sterowaniu ramieniem dysku twardego. Dostęp do danych na dysku 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 wewnątrz, 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.*



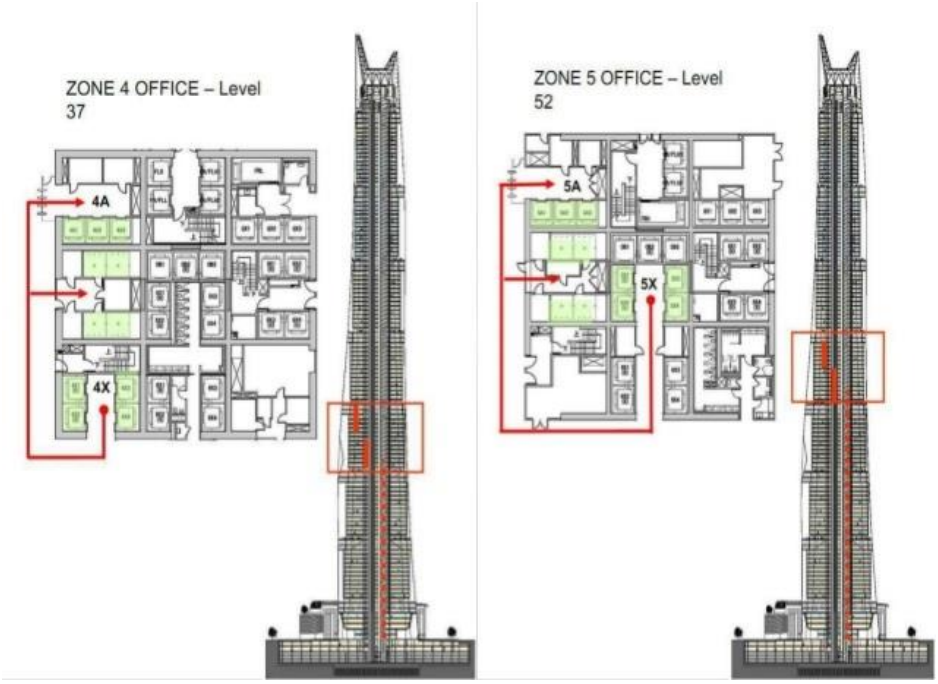
Rysunek 1-1: Ruch głowicy dysku twardego zgodny z algorytmem SCAN [8]

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

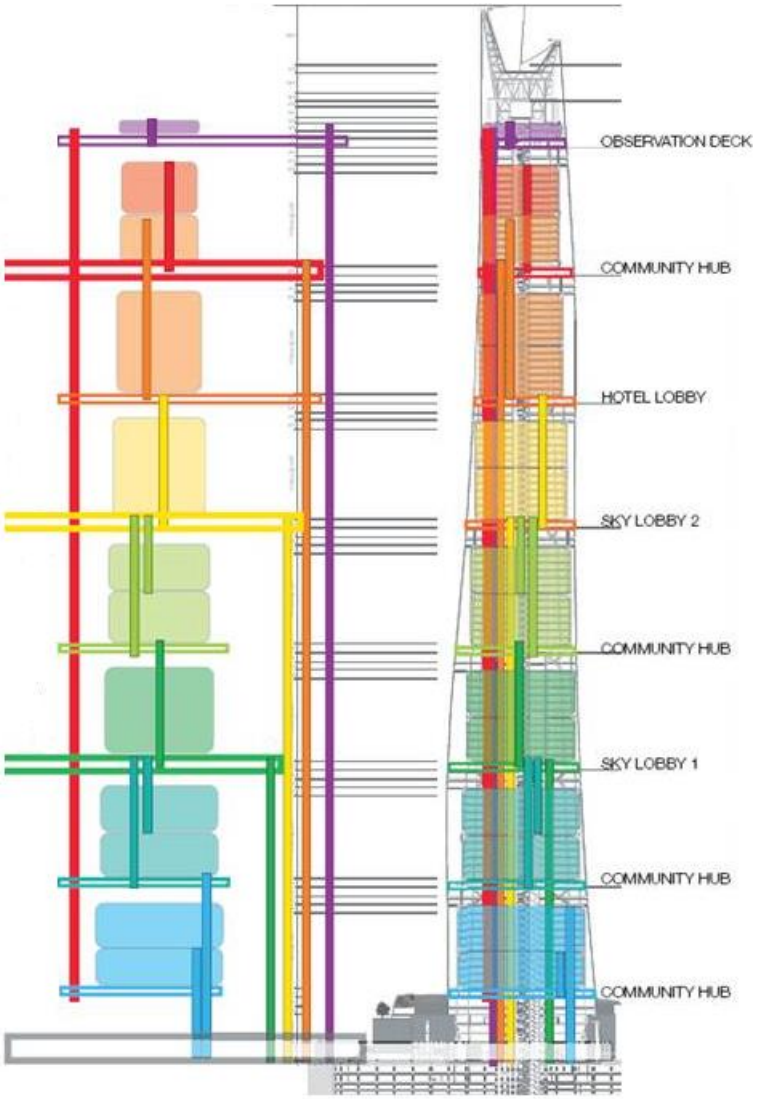
1.2.2 Strategia strefowa

Pierwszy opis strategii strefowej został stworzony 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 wózków wind w oddzieleniu i uniknięciu zjawiska grupowania. System ten jest odpowiedni dużego natężenia ruchu, gdy połączenia hali są rozłożone na wszystkie strony budynku, ale jednocześnie traci dużą elastyczność [1] kiedy windy 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. Strefy można podzielić w zależności od ilości populacji pięter lub z uwagą na to, czy jest ważne piętro, takie jak piętro kierownicze. Ogólnym pojęciem jest posiadanie tylu stref, ile jest dostępnych szybów windowych. [3]



Rysunek 1-2: Przykład połączenia dwóch stref [17]



Rysunek 1-3: System strefowy w budynku Shanghai Tower [17]

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 1-3. Połączenia między strefami znajdują się w różnych miejscach piętra. Ogromna ilość 128 pięter jest skomunikowana przez 108 wind. Dodatkowo windy poruszające się w tym budynku osiągają największe prędkości sięgające do 20,5 m/s.

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 szybkie windy wahadłowe, które poruszają się z prędkością 18 metrów na sekundę. Te trzy windy wahadłowe są uzupełnione trzema dodatkowymi windami, 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 specjalnych pięter schronienia, rozmieszczonych w regularnych odstępach na całej wysokości wieży. [18]

1.2.3 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 niezachłannym. 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. Niezachłanne algorytmy odkładają swoje zadania lub rozważają je w świetle zaktualizowanych informacji, które mogą otrzymać w postaci 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. [3]

Niezależnie od podejścia, system wybiera kabinę, która minimalizuje czas oczekiwania, czas podróży i liczbę pasażerów. System również wybiera współczynniki i szacowanie funkcji. Symulacje przed uruchomieniem 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. [1]

1.3 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 komercyjnym i około 60 sekund w budynku mieszkalnym. [16]

Dobrą analogią jest porównanie transportu wertykalnego do przepływu ciągłego wody. System ciągłego przepływu transportuje wodę ze zbiornika strumieniem lub wężem, która jest przenoszona do miejsca przeznaczenia. System wsadowy przenosi zmierzone ilości do zbiornika, gdzie się gromadzą, dopóki kolejna partia nie zostanie przeniesiona, zwykle w wiadrze. Podobnie można przedstawić działanie windy 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 wymaganie dla dobrej obsługi windy: system musi zapewniać wystarczającą ilość usług windy dla maksymalnej stawki pasażerów oczekiwanej w szczytowym okresie ruchu. To jest możliwe do osiągnięcia przez platformę o wystarczającej powierzchni, aby pomieścić wszystkie osoby czekające 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.

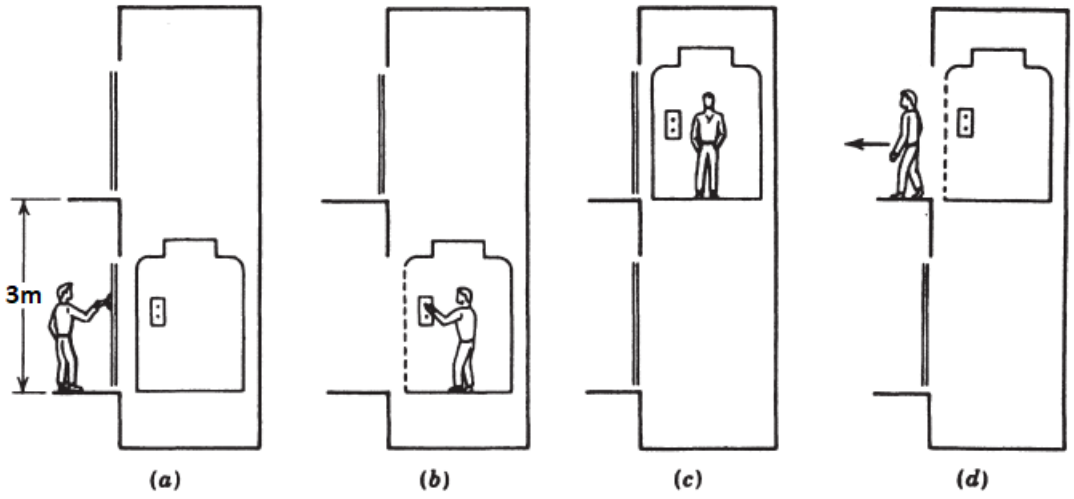
Kolejnym podobnym przykładem systemu zbliżonego do windy są schody ruchome. Platformy są dostępne przy minimalnym czasie oczekiwania, aby osoba miała natychmiastowy dostęp do schodów. Każda z platform jest wystarczająco duża, aby pomieścić tylko jedną lub dwie osoby. W przypadku, gdy więcej osób czeka na platformę w tej samej chwili, ktoś musi czekać. Potencjalni pasażerowie nie niecierpliwią 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 schody, 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. 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 różnić się w zależności od celu użycia windy. [7]

1.3.1 Zależności czasowe

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



Rysunek 1-4: Model windy dwuprzystankowej [7]

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 dwóch sekund, w zależności od szerokości i rodzaju drzwi. Kolejne dwie sekundy trwa wejście pasażera do środka i naciśnięcie przycisku (b). Drzwi zamykają się (około 3 sekundy), a wózek 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 muszą 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. Ten czas się nazywa „interwałem" między obsługą windy na parterze. Jeśli uznamy to za 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 windy. 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 większą ilością przystanków. [7]

1.3.2 Interfejs użytkownika w standardowej windzie

W nowoczesnych budynkach ludzie są przyzwyczajeni do przycisku przywołania windy, a 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 podróż od nowa. Zaletą systemu jest to, że ma tendencję do zmniejszania liczby zatrzymań windy i poprawia obsługę podróżnych. Ta różnica w obsłudze pasażerów odniosła wielki sukces w budynkach biurowych, w których ludzie zapoznają się z podejściem, ale ograniczono użycie tego w hotelach, gdzie nieznane systemu powodują trudności.

1.4 Układy elektroniczne sterujące windą

Główna część systemu windy nazywana jest kontrolerem. Funkcjonalnie jest to urządzenie logiczne, które monitoruje różne funkcje bezpieczeństwa 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, głównie dlatego, że przekaźniki są droższe niż elementy elektroniczne. Ponadto układy PLC ułatwiają proste połączenia elektryczne i zmniejszają ryzyko awarii. W starszych wariantach wind kierunek wózka 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 jest główną część systemu lub jest urządzeniem, które wysyła sygnały do niezależnego systemu sterowania ruchem. Architektura sterownika jest skończoną maszyną stanu lub kilkoma maszynami stanów, które odczytują dane wejściowe systemu i na podstawie aktualnego stanu 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 równoległe. 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 układy FPGA zastępuje się układami o specjalizowanej funkcjonalności (tzw. ASIC). Te układy są znacznie bardziej wydajne niż FPGA ze względu na 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 powierzchnię niż odpowiadające ich układy PLC. Celem sterowania windą za pomocą układów ASIC jest

zwiększenie niezawodności, redukcja wielkości jednostek sterujących windą oraz zmniejszenie poboru energii.

1.5 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 w 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 dostosowana do nowej polityki ram prawnych i ma zastosowanie od 20 kwietnia 2016 r., zastępując poprzednią dyrektywę 95/16/WE. [13]

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. [12]

W odniesieniu dostępu dla osób niepełnosprawnych do kabin wind, kraje należące do 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.). [14]

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 powinien 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. [13]

1.6 System przyszłości

1.6.1 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. To 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 ludzi, takie jak wdrożenie strefowego systemu windy, a nawet zastosowanie dwupoziomowych wind. Wraz z rozwojem technologii doprowadziło to do stworzenia optymalnego systemu dystrybucji wind, znanego jako *wysyłka docelowa*, który w ostatnich latach stał się bardziej atrakcyjną technologią dla wind, przerywając stosowanie tradycyjnych systemów 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 piętra przed szybem windy, na urządzeniu takim jak klawiatura lub ekran dotykowy.

Rysunek 1-5: Wybór piętra w systemie z wysyłką docelową [19]

System wysyłania do 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. Kontroler kieruje poszczególnymi wagonami windy w grupie, aby obsługiwały tylko określone piętra. 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 w mniejszym skupisku ludzi i większym komfortem.

Rysunek 1-6: Przyciski w windzie z wysyłką docelową i klasycznej windzie [11]

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 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.

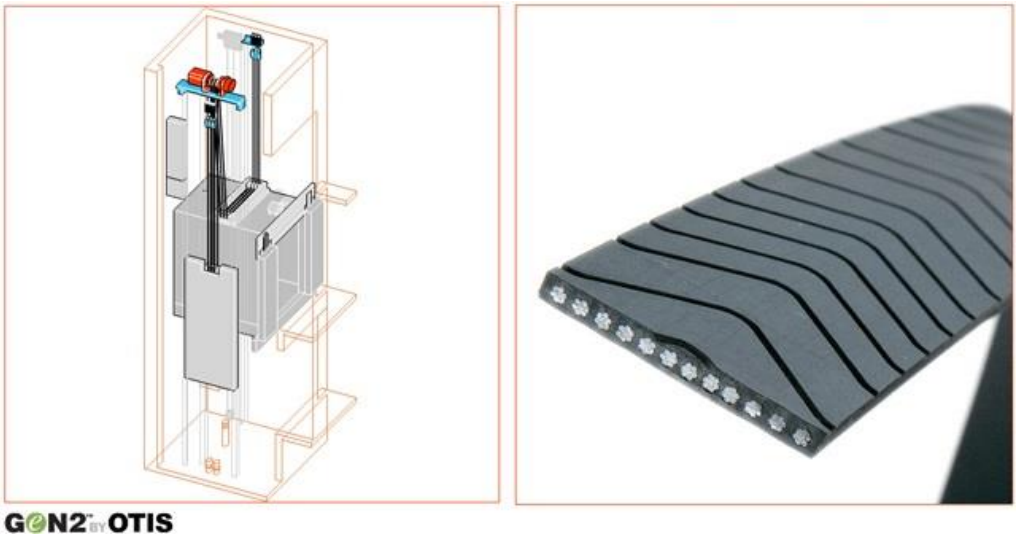
Windy wyposażone w docelowy system wysyłkowy mają co najmniej dwie klawiatury numeryczne lub ekrany dotykowe w każdym holu, które zastępują tradycyjne przyciski żądań. Jeśli windy mają ekrany dotykowe, zazwyczaj zawierają listę dostępnych pięter obsługiwanych przez windy. Istnieją również tabliczki identyfikacyjne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

Rysunek 1-7: Identyfikacja wind w systemie wysyłkowym [10]

Pasażerowie są przypisywani do danych szybów i nie mają wymogu wybierania pięter w środku kabiny. Wewnątrz pozostały 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.

1.6.2 Kierunki rozwoju technologii

Pomimo tak wyrafinowanych systemów operacyjnych wind, wciąż istnieją ogromne możliwości do 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 [15]

Brak maszynowni pozwala architektom na większą elastyczność projektowania. Konstruktorzy korzystają z kontrolowanego, usprawnionego procesu instalacji i minimalnej ingerencji w inne elementy budowli. Dla właścicieli budynków systemu Gen2 przekłada się na niższe koszty budowy i znaczny wzrost powierzchni do wynajęcia. Modułowy kontroler, zaprojektowany dla grup składających się z maksymalnie trzech wózków, 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 windy. Rezultatem jest system wyjątkowej niezawodności. [15]

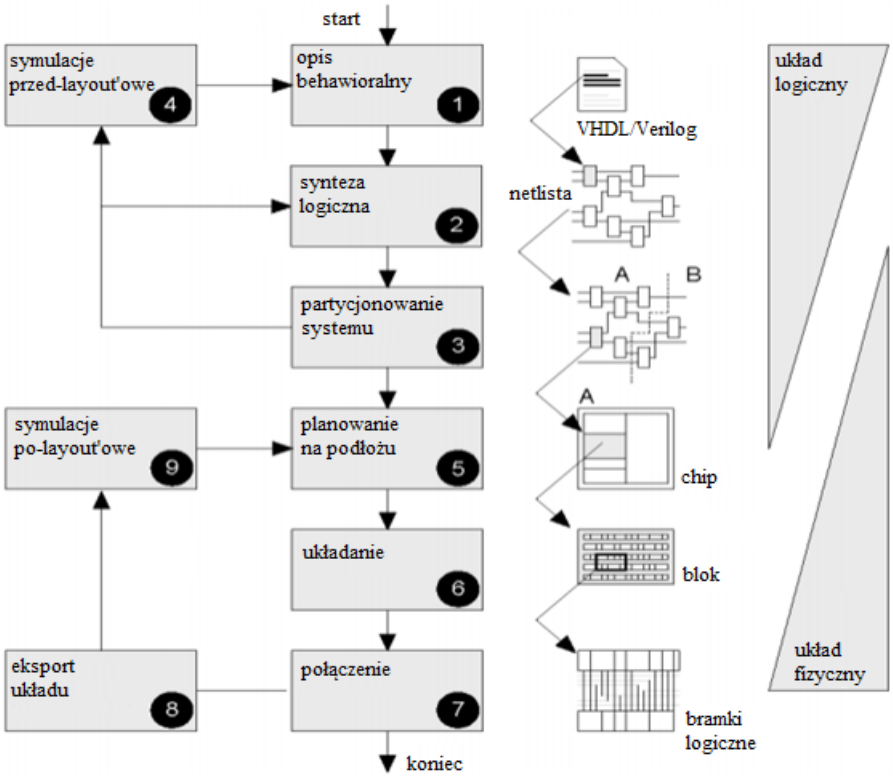
Trend będzie kontynuowany przy użyciu lżejszych materiałów i bardziej wydajnych napędów regeneracyjnych. Ulepszenia w zdalnym monitorowaniu, niezawodności systemu i konserwacji opartej na użytkowaniu będą kontynuowane przez bardziej zaawansowaną diagnostykę kontrolerów oraz wyższy poziom integracji z 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. Wraz z rozwojem technologii następuje ciągłe ulepszanie interfejsów użytkownika i urządzeń bezpieczeństwa dla wind.

Rozdział 2
Opis algorytmu w języku Verilog

2.1 Tworzenie modułów z wykorzystaniem języka Verilog

Język Verilog HDL (ang. *Hardware Description Language*) zawiera funkcje opisujące zachowanie modułu, przepływ danych, skład strukturalny, opóźnienia i mechanizm generowania przebiegów, a także aspekty monitorowania odpowiedzi i weryfikacji, przy użyciu jednego języka. Ponadto Verilog 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. 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 jest 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 łatwiejszy w obsłudze dla użytkownika. Bezpośredni dostęp do wewnętrznych sygnałów pozwala na szybką implementację oraz weryfikację 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 [6]

2.2 Rozwój oprogramowania w oparciu o cechy użytkowe

Wybraną metodologią 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:

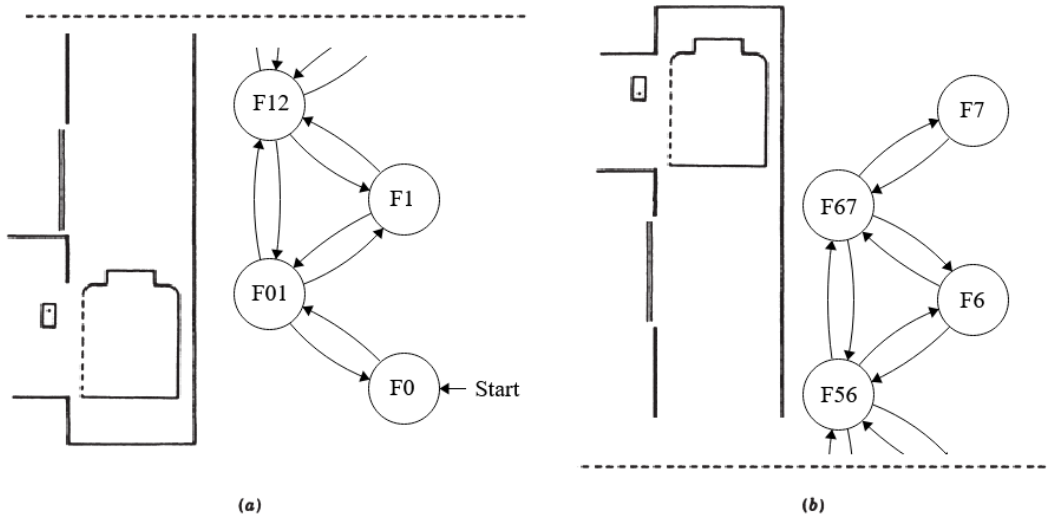
1. Opracowanie ogólnej architektury produktu.
2. Budowanie listy cech produktu.
3. Zaplanowanie w czasie implementacji cech.
4. Projektowanie każdej z planowych cech.
5. Budowanie oprogramowania w nastawieniu na cechy.

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 z tych powodów 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 funkcjonalność nie jest całkowicie zdefiniowana od początku, a struktura projektu jest głównie oparta 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 cech 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 implementację algorytmu sterowania windą. Przy zwinnym, iteracyjnym podejściu do rozwoju oprogramowania należy pracować wraz z systemem do kontroli wersji, aby mieć możliwość powrotu do poprzedniego stanu kodu.

2.3 Architektura systemu

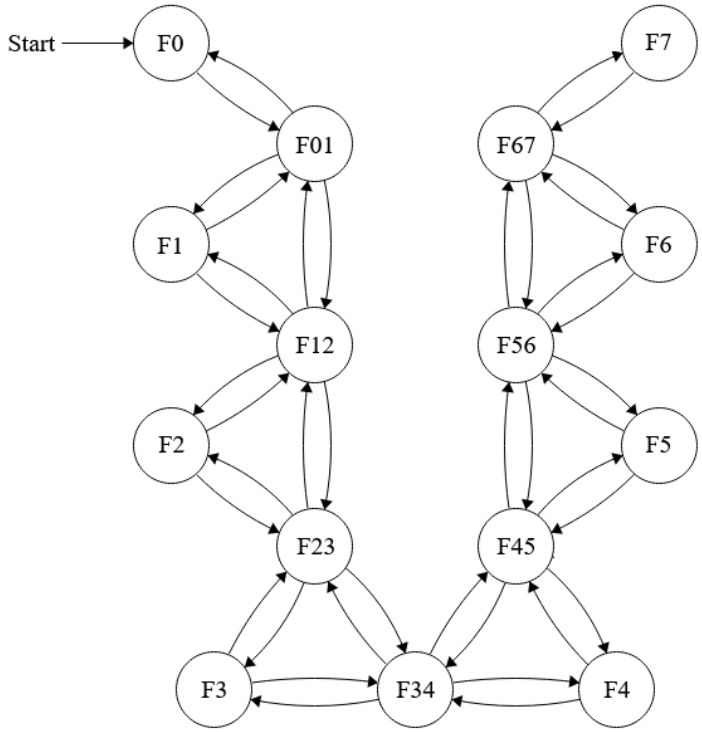
Bazując na cechach modelu strategii zbiorowej kontroli, został stworzony algorytm kontrolera windy. Główną częścią w systemie jest maszyna stanów obsługująca poruszanie się po piętrach oraz sterowanie drzwiami windy. Podstawowym założeniem dla projektu jest zaproponowanie systemu dla ośmiu pięter. Wertykalny 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 [opr. 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 część algorytmu jest zbudowana z podobnych modułów, różniących się numerem piętra oraz innymi rejestrami przycisków, które są sprawdzane w czasie działania systemu. Więcej na temat poszczególnych stanów algorytmu zostanie omówione w paragrafach 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.

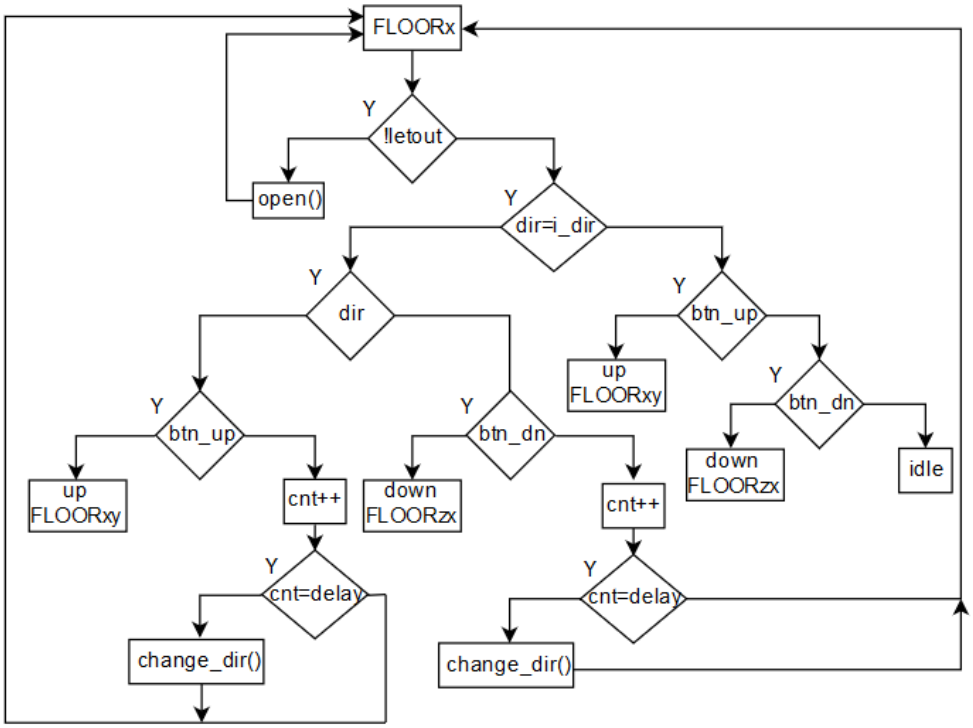


Rysunek 2-3: Algorytm ruchu wertykalnego dla ośmiopiętrowej windy [opr. własne]

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.

2.3.1 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 i 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 ruchu w tym stanie. System zakłada 5 sekund na wejście pasażera i naciśniecie przycisku. Dodatkowo przed zmianą kierunku odliczane jest 500 milisekund.

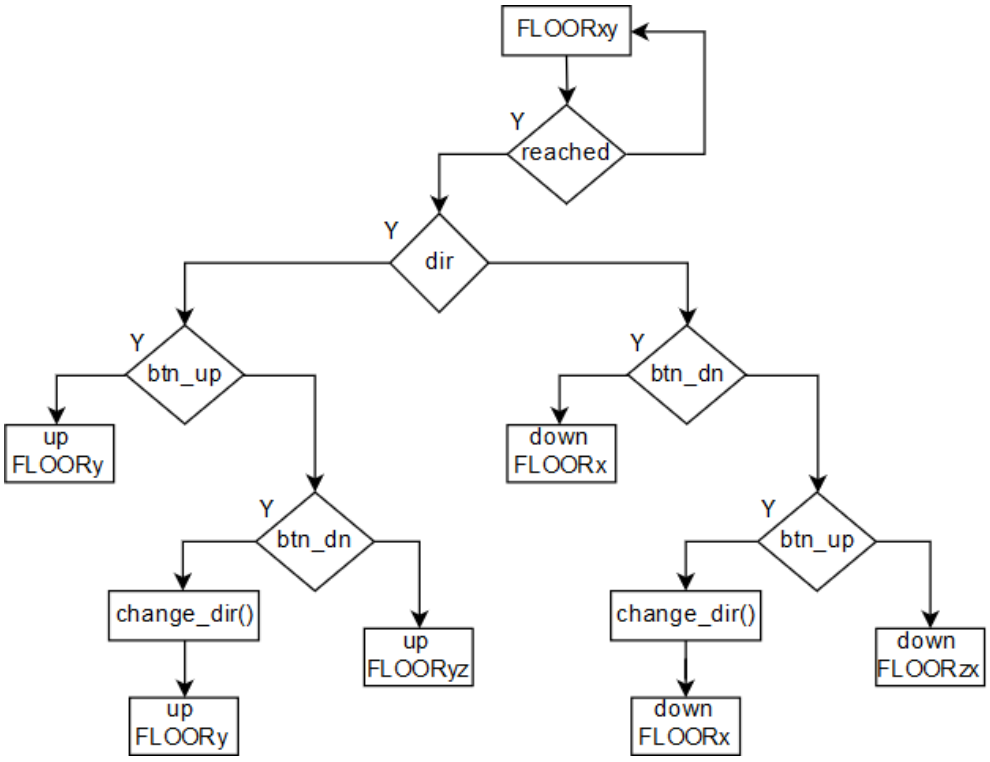


Rysunek 2-4: Algorytm na pełnym piętrze [opr. własne]

Po zmianie kierunku wewnątrz stanu algorytm kieruje się na prawą stronę ścieżki algorytmu. Pierwsza sprawdzana jest kombinacja przycisków w górę, choć w tej sytuacji kolejność może być dowolna, gdyż zakładamy, że kontynuowanie kierunku już zostało zaniechane. Ostatecznie przy braku jakichkolwiek poleceń system przechodzi w stan oszczędzania energii. W przypadku piętra 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 naciśnięty żaden z przycisków.

2.3.2 Moduł piętra przejściowego

Stan przejścia między piętrami zależy głównie od pozycji kabiny względem platformy na piętrze. Zgodnie z rysunkiem 2-5 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 windy równą 1m/s oznacza, że winda pokona tylko jeden milimetr i system podejmie decyzję, czy zatrzymać się, czy kontynuować ruch.

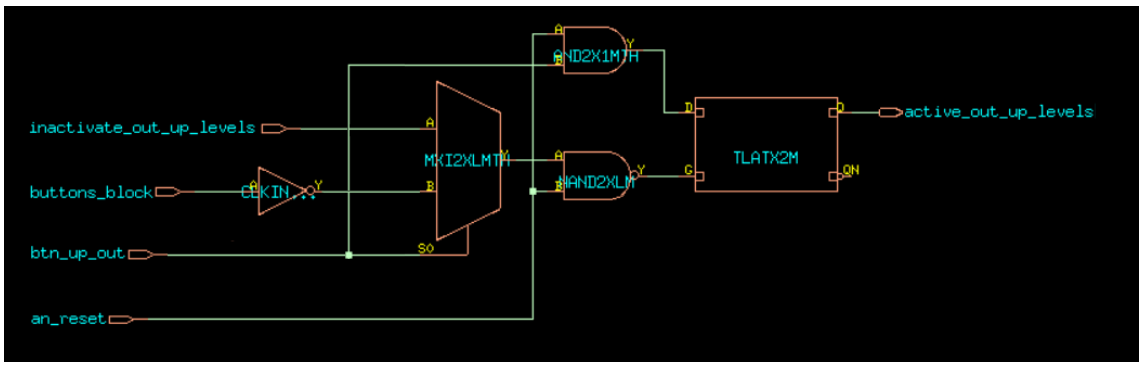


Rysunek 2-5: Algorytm przejazdu między piętrami [opr. własne]

Na rysunku 2-5 nazwą FLOORxy zostały oznaczone 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 piętra 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 celu podróży przez pasażera i usunięcie niepotrzebnych przystanków na piętrach. Warto również wspomnieć, że przy odpytywaniu o żądania, brany jest pod uwagę czujnik tensometryczny. Zgodnie z wymaganiami 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.

2.3.3 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 [opr. 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 do 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 naciśnięty omyłkowo. Algorytm tych przycisków jest bardziej skomplikowany i wymagana wyjaśnienia. Poniższy kod opisuje zachowanie resetowalnych przycisków.

```
for(index=0; index<8; index+=1){
    if (inactivate_in_levels[index] == 1){
        if (l_inactivate_in_levels[index] == 0){
            if (active_in_levels[index] == 1){
                active_in_levels[index] = 0;
                buttons_state[index]=!buttons_state[index];
            }
        }
    }else{
        if(!buttons_block){
            if(btn_in[index] == 1){
                if(l_btn_in[index] == 0){
                    if(buttons_state[index])
                        active_in_levels[index] = 1;
                    else
                        active_in_levels[index] = 0;
                    buttons_state[index]=!buttons_state[index];
                }
            }
        }
    }
    l_btn_in[index] = btn_in[index];
    l_inactivate_in_levels[index] = inactivate_in_levels[index];
}
```

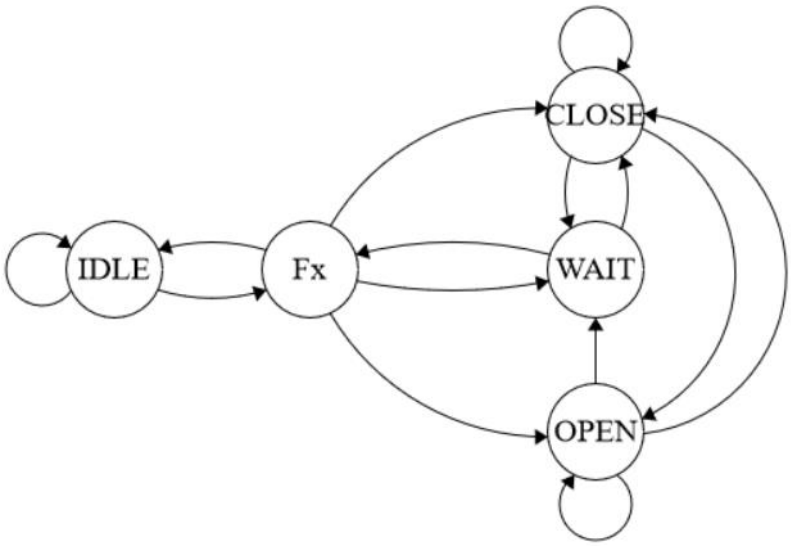
Rysunek 2-7: Kod dla resetowalnych przycisków [opr. własne]

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 stan przycisku,
- 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 przyciśnięcie, w tym przypadku nazwany buttons_state. W zależności od tej wartości przyciśnięcie aktywuje lub dezaktywuje polecenie. Przytrzymanie przycisku, nie zmienia wartości, gdyż 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 [opr. 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.

2.4 Weryfikacja kodu

Zgodnie z metodologią tworzenia oprogramowania na podstawie cech, 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 wymagane do zakończenia prac nad implementacją kodu. Najlepszą metodą przy testowaniu algorytmu, którego kod jest znany, są testy strukturalne (ang. *white box*). Trzy wykorzystane techniki w testowaniu:

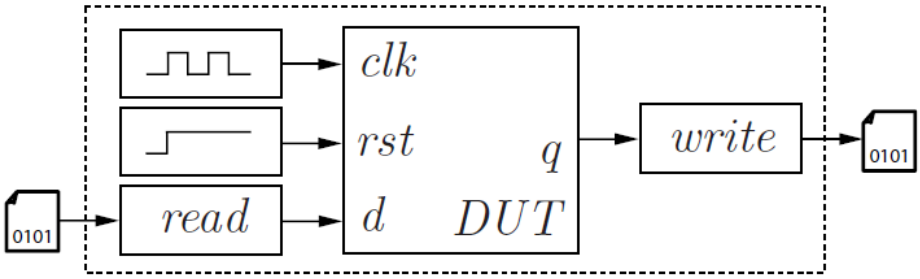
- Pokrycie linii kodu *(ang. statement coverage)* – technika testowania sprawdzająca wszystkie linijki implementowanego kodu, znane również jako pokrycie segmentu. Określa, które fragmenty programu pokrywają się ze sobą, czyli zostały wywołane przez zestaw testowy oraz wskazuje na te, których nie udało 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. Z reguły 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 polega na testowaniu ocenę programu w czasie jego wykonywania. Analizy statyczne i dynamiczne, rozpatrywane razem, są czasami nazywane testami „na szkle". [20]

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

2.4.1 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 [opr. własne]

Do przetestowania przy użyciu wcześniej wspomnianej metody utworzono dwadzieścia 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 typów przycisków. Sterowanie drzwiami przy pomocy przycisków zostało sprawdzone przez dwa testy, rozdzielone na otwieranie i zamykanie drzwi. 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 pozostałych 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.

Rozdział 3
Projektowanie układu scalonego

3.1 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 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, synteza RTL, tworząca poziom bramki netlist z zachowaniem RTL i synteza behawioralna (wysokiego poziomu), która tworzy opis RTL z reprezentacji algorytmicznej. Wszystkie z nich używają plików typu *Liberty* do opisania zachowania bramek. Pliki te zawierają opisy dla różnych stylów projektowania, a także służą do optymalizacji prędkości, powierzchni i mocy 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:

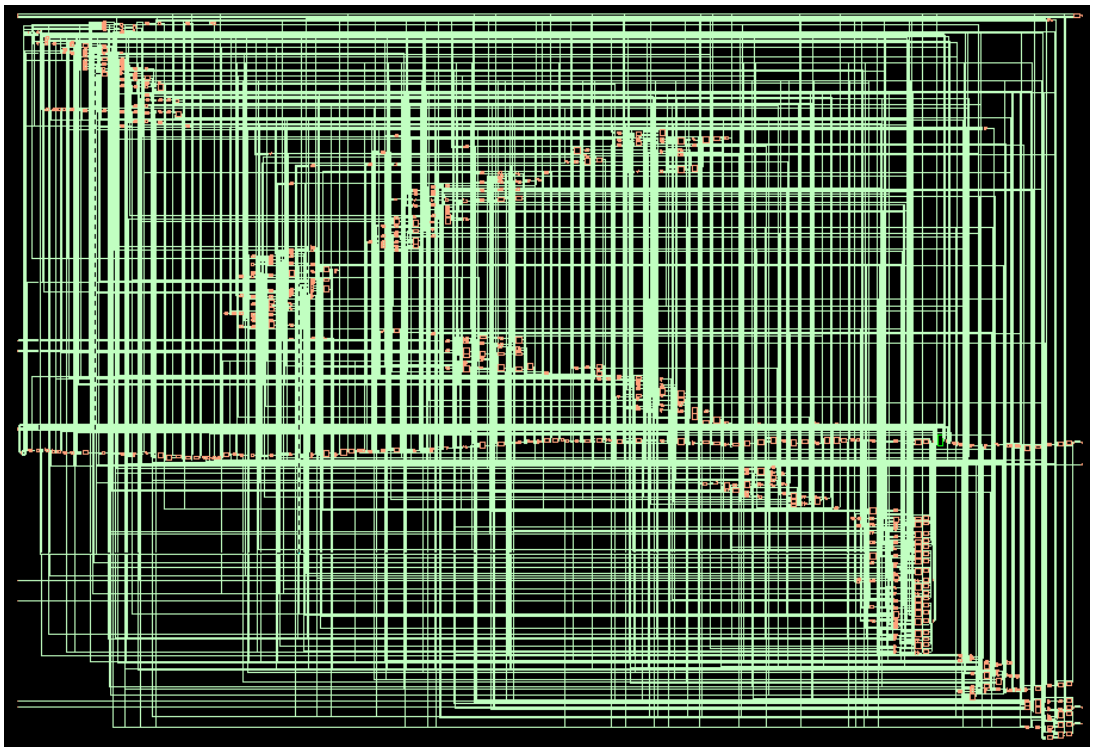
1. Ustawienie zmiennych i parametrów, które będą używane do syntezowania. Najczęściej są to ścieżki do plików z modułami do syntezy oraz biblioteki, które zawierają opisy strukturalne.
2. Odczyt plików, analiza pod kątem składni językowej oraz translacja.
3. Ustawienie ograniczeń układu w związku z taktowaniem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

Skrypt syntezy 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 zsyntezuje układ, konieczne jest podanie, które biblioteki powinny być używane do tworzenia modułu. 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 wysokich 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 w wyższej częstotliwości.

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 program 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 strukturalnego są sprawdzane przez komendę *check_design -unresolved.* Ostatecznie komenda *synthesize -to_mapped* wykonuje właściwą syntezę. Dodatkowy argument *-to_mapped* wskazuje syntezerowi, by używał bibliotek dodanych przez użytkowania, niż 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.



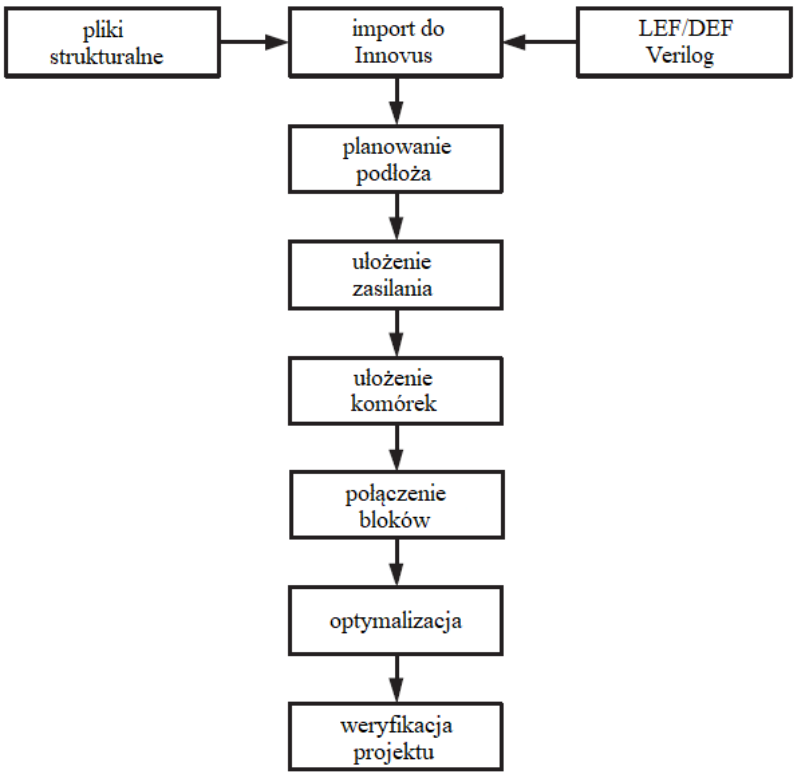
Rysunek 3-1: Struktura układu po syntezie [opr. własne]

3.1.1 Symulacja po syntezie

Po zakończeniu syntezy z behawioralnego opisu do strukturalnego należy sprawdzić ponownie działanie modułu. Z powodu dużej ilości połączeń mogą występować opóźnienia, co skutkuje przekłamaniami i błędami w działaniu układu. Do sprawdzenia funkcjonalności należy użyć tego samego testu lub testów co przed syntezą. Jeżeli testy zakończą się wynikiem pozytywnym, można przechodzić do kolejnego punktu. W przypadku ograniczeń czasowych dla opisywanego kontrolera windy taktowanie sygnału zegarowego 1kHz pozostawia duży margines na opóźnienia sygnałowe. Generacja układu po syntezie została wykonana w programie Cadence Virtuoso. Ostateczne wyniki są zgodne z oczekiwanymi, struktura po syntezie ma tę samą funkcjonalność.

3.2 Projektowanie topografii

Proces polegający na pobraniu plików z opisem strukturalnym i stworzenie z niego fizycznego układu jest nazywane układaniem i połączeniem (ang. *place&route*). Jak sugeruje nazwa, ten etap składa się z dwóch kroków. Pierwszy to umieszczenie, które 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: Schemat projektowania topografii [opr. własne]

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

3.2.1 Generacja układu

Do stworzenia topografii, pliki LEF muszą zostać zaimportowane w celu utworzenia biblioteki fizycznej i logicznej. W nich zawierają się informacje opisujące technologię jako komórki i bloki. Jeśli używane są komórki pamięci, odpowiedni plik LEF też musi być dodany. Import można wykonać za pomocą GUI lub za pomocą skryptu. Dołączone pliki muszą być zgodne z użytymi do syntezy, aby zachować jednolitość opisu 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).

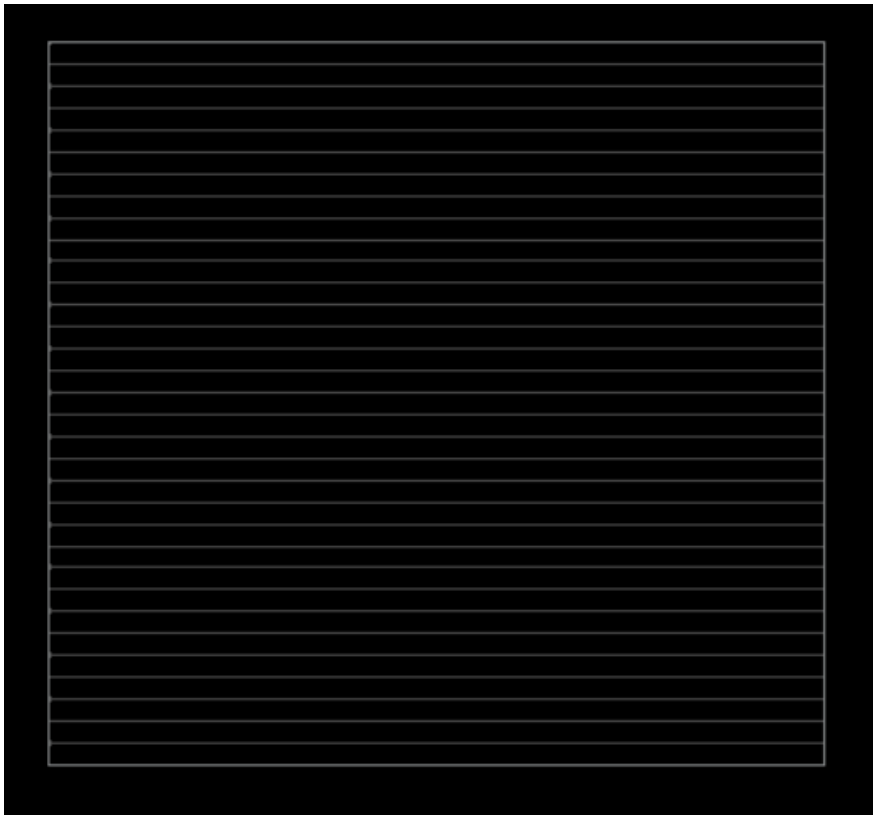
Posługując się skryptem z poprzedniego rozdziału o syntezie w programie Genus, wszystkie wymienione wyżej pliki są generowane automatycznie. Dwa ostanie pliki nie mają swojego miejsca do zadeklarowania podczas importowania. Dołączenie źródeł odbywa się plik MMMC w formacie mmode.tcl. Plik ten wywołuje 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 informacja o trzy warstwowym procesie m AMI C5N 50nm jest podana w pliku UofU_Digital_v1_2.lef. Dodatkowo wszelkie twarde makra obecne w projekcie muszą mają również odpowiedni plik LEF, ponieważ nie można ich zbudować z listy komórek obecnych w standardowej bibliotece makra LEF.

Netlista projektu jest zwykle niezmodyfikowanym plikiem wynikowym z takiego narzędzia syntezy jako kompilator projektu. Jedną z ważnych kwestii jest to, że wszystkie typy komórek instancji na liście muszą 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. Ostatecznie wyniki mogą zostać poprawione przez plik SDC, choć nie jest to wymagane do stworzenia układu scalonego. Analogicznie jak netlistę, można ten plik utworzyć z poziomu kompilatora projektu za pomocą komendy write_sdc po syntezie.

Informacje o ograniczeniach czasowych 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.

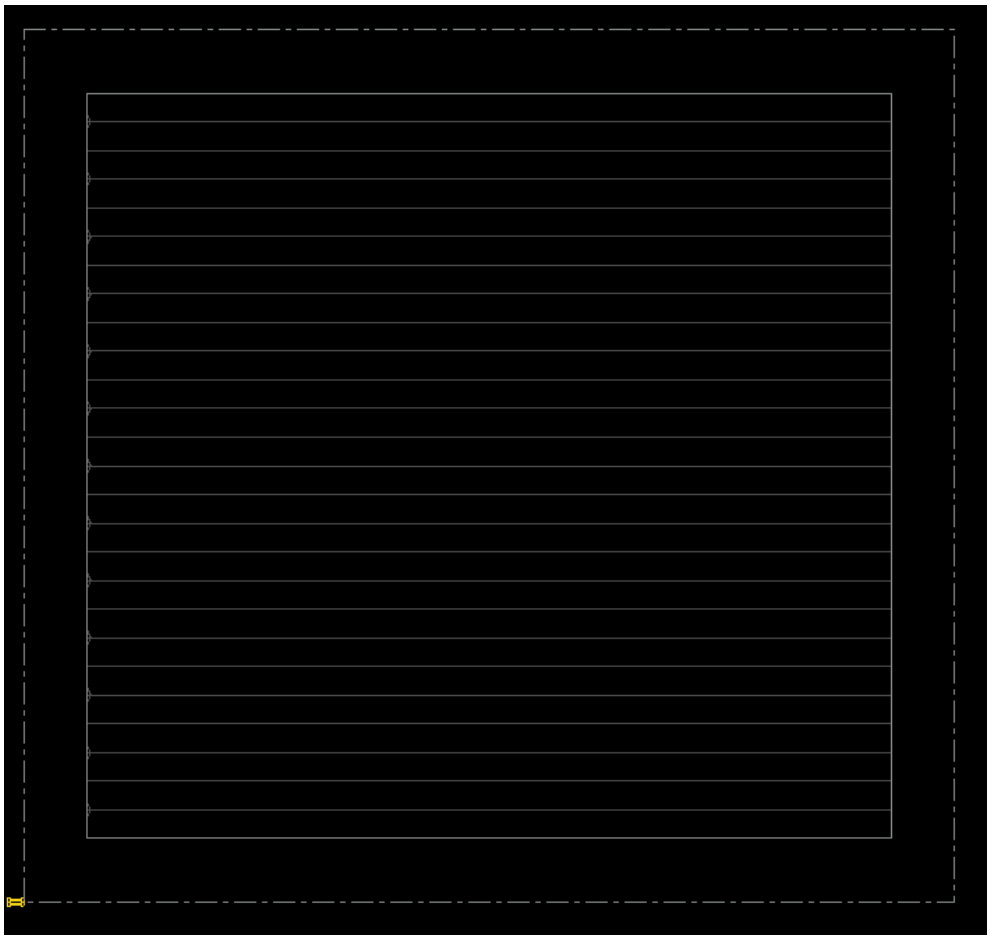


Rysunek 3-3: Komórki standardowe po imporcie do programu Innovus [opr. własne]

Przy importowaniu struktury wymagane jest podanie nazw sygnału zasilania i masy. Dobrą 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.

3.2.2 Planowanie podłoża

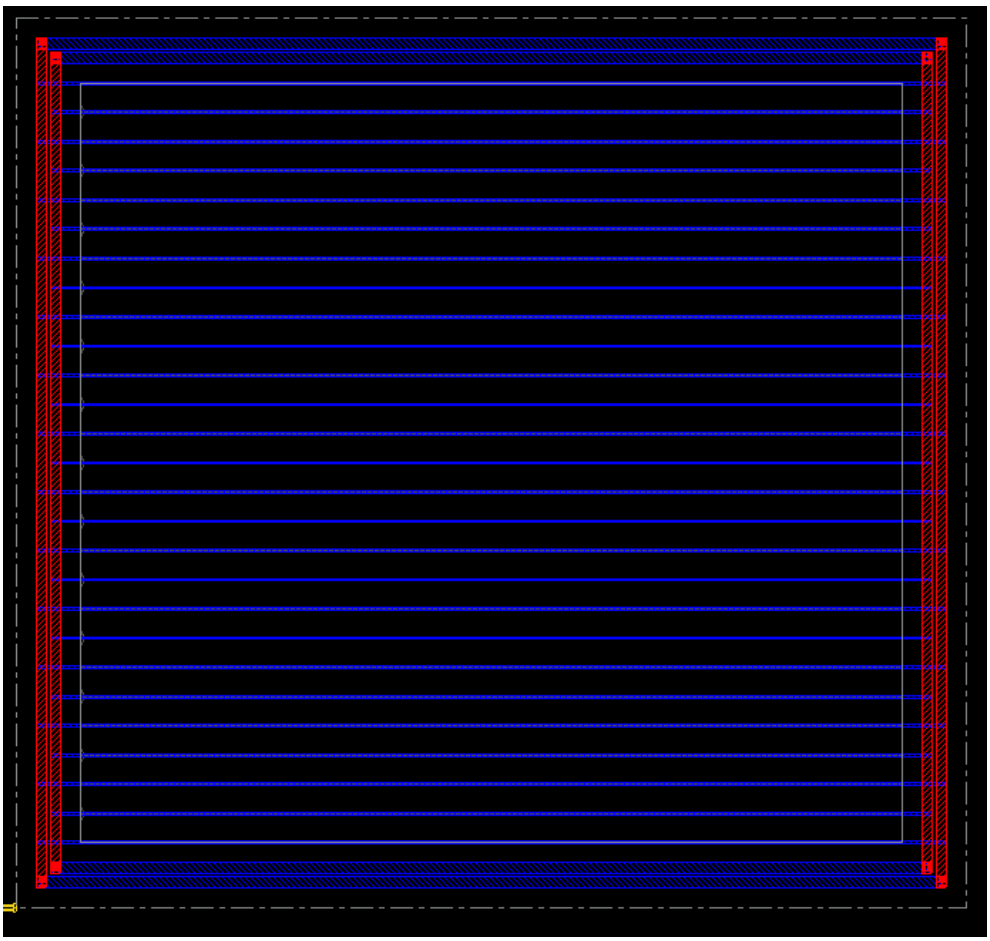
Planowanie podłoża projektu jest pierwszym krokiem po zaimportowaniu plików technologicznych na poziomie netlisty bramek. W tym kroku plan układu musi być przygotowany na zasilanie, rozmieszczenie klocków wejścia/wyjścia (ang. *input/output*, I/O), twarde makra (np. pamięć RAM) i standardowe komórki. Projektant jest zobowiązany do określenia rozmiaru matrycy przez rozmieszczenie IO na ramie podkładki i ustawienie ramy podkładki względem odległości od rdzenia. Co więcej, wiersze rdzenia muszą zostać przerwane, gdy zostają umieszczone twarde makra. [5] W przypadku tego projektu, istnieje tylko jedna instancja, co bardzo ułatwia planowanie podłoża. Jedyną rzeczą jest zostawienie miejsca na warstwy zasilania. Zaplanowany obszar został oznaczony przerywaną linią na poniższym rysunku.



Rysunek 3-4: Układ po planowaniu podłoża [opr. własne]

3.2.3 Układanie warstw zasilania

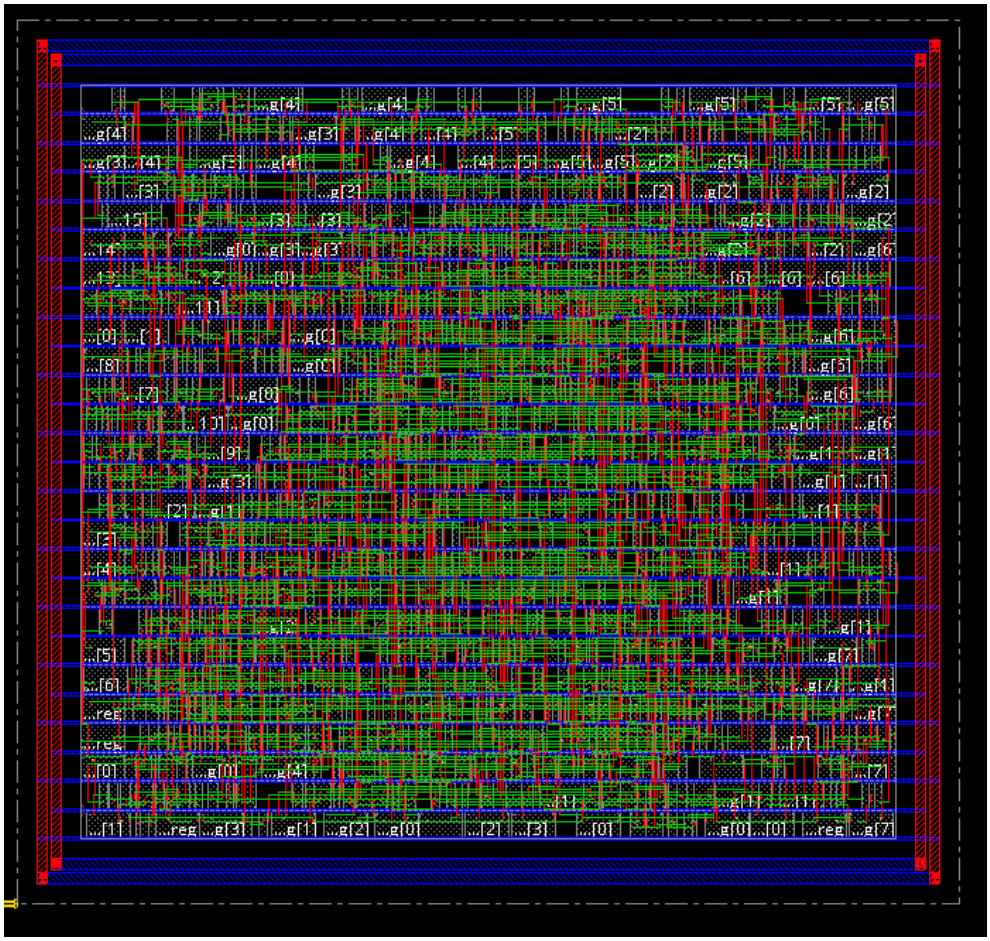
Planowanie i układanie warstw zasilania jest niezbędne do dostarczenia napięcia do standardowych komórek i makr. W pierwszym kroku należy dodać pierścienie, które umieszcza się na obwodzie rdzenia i zasilania do pasków, które przenoszą moc przez chip. Oddzielny pierścień służy do zasilania (vdd!) i masy (gnd!). 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, wystarczające jest dołączenie pierścieni zasilania wokół pierścienia bez potrzeby dodawania pasków mocy. Paski dodajemy przy użyciu instrukcji *special route.*



Rysunek 3-5: Zasilanie dołączone do układu scalonego [opr. własne]

3.2.4 Planowanie komórek standardowych

Po zaplanowaniu podłoża układu oraz dołączeniu warstw zasilających rdzeń jest gotowy na dołączenie komórek standardowych i makr. Po zakończeniu tego etapu standardowe komórki i bloki makr, które implementują funkcjonalność netlist, powinny być umieszczone w rzędach topografii. W wielu przypadkach ułożenie komórek standardowych odbywa się wraz z próbnym połączeniem tak jak w przypadku poniższego układu:



Rysunek 3-6: Widok fizyczny po ułożeniu komórek standardowych [opr. własne]

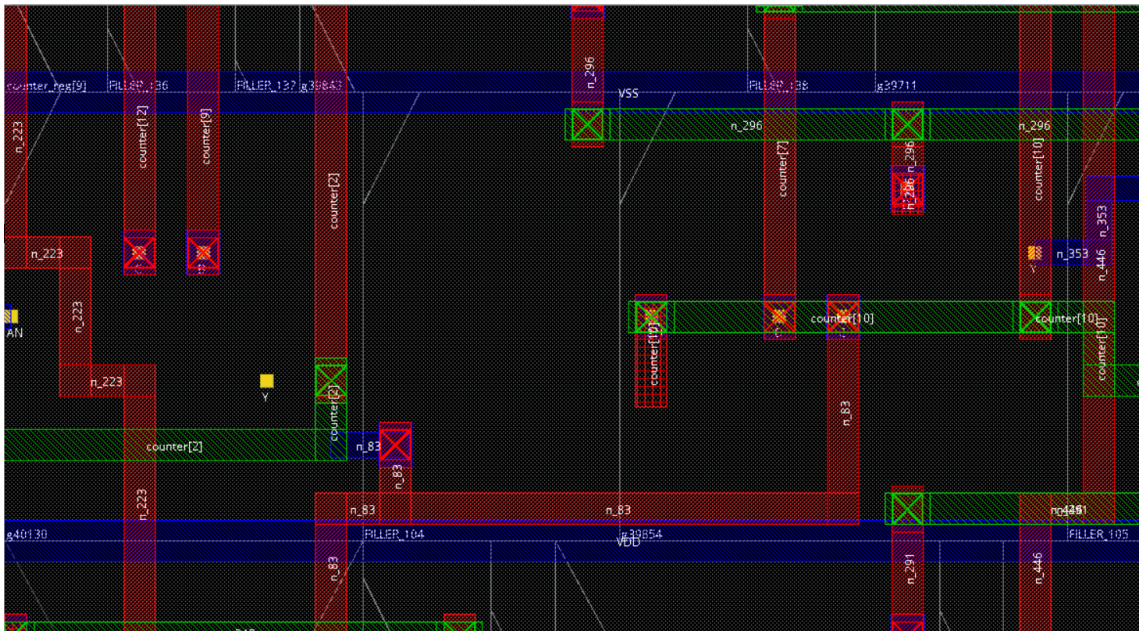
Pierwsze ułożenie struktury i połączenie warstwami metali jest nieoptymalne i wymaga dalszego procesowania. Im większy projekt, tym większa szansa, że duża ilość połączeń może zostać usunięta. Podczas projektowania ważne jest doprowadzanie sygnału zegarowego, który po optymalizacji zmniejsza swoje opóźnienie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

3.2.5 Optymalizacja

Na etapie łączenia występują trzy fazy optymalizacji, gdzie każda następuje po zmianie topografii pod pewnym kryterium. Pierwsza optymalizacja jest wykonywana bezpośrednio po ułożeniu komórek standardowych. Automatyczne narzędzie wbudowane w oprogramowanie Innovus próbnie łączy bloki sprawdzając parametry przebiegów czasowych. Na każdym z etapów można zdecydować jakie czynniki są pożądane przez projektowany układ.

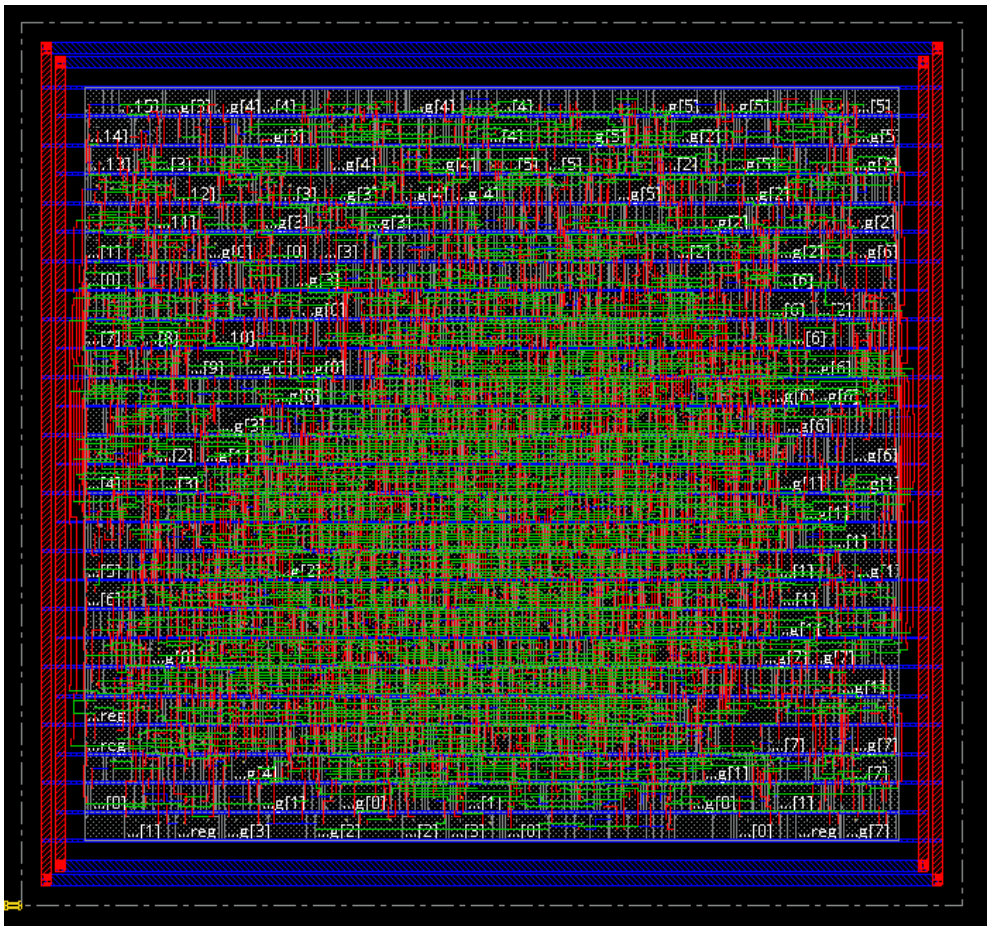
Kolejnym krokiem przed drugą fazą optymalizacji jest synteza drzewa zegarowego. Cel tego zabiegu jest prosty: ma pomóc w spełnieniu założonych ograniczeń przebiegu czasowego. Do poprawy działania sygnału zegarowego wykorzystywane są komórki inwerterów lub zbudowane wcześniej gotowe bufory zegarowe. Ten proces może dodawać nowe komórki i zmieniać dotychczasowe. Po zakończeniu prac nad drzewem zegarowym niezbędna jest kolejna optymalizacja. Ponownie układ jest analizowany do połączenia przy najlepszych ustawieniach dla spełnienia założeń. Kolejną fazą jest ostateczne połączenie komórek (*ang. routing*). [2]

Przed routingiem dodaje się komórki wypełniające, aby uzupełnić luki w układzie fizycznym. Ten krok zapewnia pojemność sprzęgającą i uzupełnia połączenia zasilania i masy standardowych komórek. Następnie wykonuje się specjalny routing na sieciach zasilania i masy; vdd! i gnd!. Po dodaniu dodatkowych komórek można wykonać połączenie wszystkich elementów za pomocą narzędzia NanoRoute.



Rysunek 3-7: Wypełnienie przerw między komórkami standardowymi [opr. własne]

Zaawansowane narzędzie do łączenia NanoRoute obsługuje wszystkie wyzwania routingu zarówno na poziomie bloków, jak i pełnych układów. Łączy ono w sobie charakterystykę routera opartego na sieci z elastycznością poza siecią, a jednocześnie ocenia i optymalizuje topologię połączeń na podstawie efektów 3D dotyczących czasu, obszaru, mocy, zdolności produkcyjnej i wydajności. Przez użycie wielowątkowego przetwarzania danych na płycie układu, NanoRoute wykonuje miliony połączeń sieci na godzinę. To narzędzie 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. [9] 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. [2]



Rysunek 3-8: Finalny projekt topografii [opr. własne]

3.2.6 Weryfikacja

Pierwszym krokiem w weryfikacji jest sprawdzenie poprawności połączeń układu. Wymagane jest pokrycie połączeń zawartych w netliście. Wszelkie problemy są zgłaszane przez kompilator i muszą zostać poprawione przed kolejnym działaniem. Jeżeli nie ma błędów, zwrotna wiadomość powinna wyglądać tak:

```
******** End: VERIFY CONNECTIVITY ********
Verification Complete : 0 Viols. 0 Wrngs.
```

Drugim etapem jest weryfikacja geometrii układu. W tym kroku sprawdzane jest ułożenie komórek standardowych w granicach rzędów, nachodzenie się obszarów i kontaktów. Warto zauważyć, że ta weryfikacja nie zastępuje ostatecznej weryfikacji zgodnej z regułami projektowymi. Sprawdzany jest głównie widok abstrakcyjny, który nie zawiera warstw metalicznych. Przed kolejnym krokiem należy wykonać tą weryfikacje i poprawić ewentualne błędy. Poprawne działanie układu jest sygnalizowane informacją:

```
**********End: VERIFY GEOMETRY**********
Verification Complete : 0 Viols. 0 Wrngs.
```

Ostatecznym etapem jest weryfikacja reguł projektowych (ang. *Design Rule Check*). W inżynierii elektronicznej regułą projektową jest ograniczenie geometryczne nałożone na płytkę drukowaną, urządzenie półprzewodnikowe i projektory układów scalonych, aby zapewnić, że ich konstrukcje działają prawidłowo, niezawodnie i mogą być wytwarzane z akceptowalną wydajnością. Zasady projektowania produkcji są opracowywane przez inżynierów procesowych na podstawie zdolności ich procesów do realizacji zamierzeń projektowych. Automatyzacja projektowania elektronicznego jest szeroko stosowana w celu zapewnienia, że projektanci nie naruszają zasad projektowania. DRC jest ważnym krokiem podczas podpisywania fizycznej weryfikacji projektu, co obejmuje również kontrole LVS (układ w porównaniu ze schematem), kontrole XOR, ERC (kontrola reguł elektrycznych) i kontrole anten. Znaczenie zasad projektowania i DRC jest największe w przypadku układów scalonych, które mają geometrie w skali mikro lub nano; w przypadku zaawansowanych procesów niektóre zakłady również nalegają na stosowanie

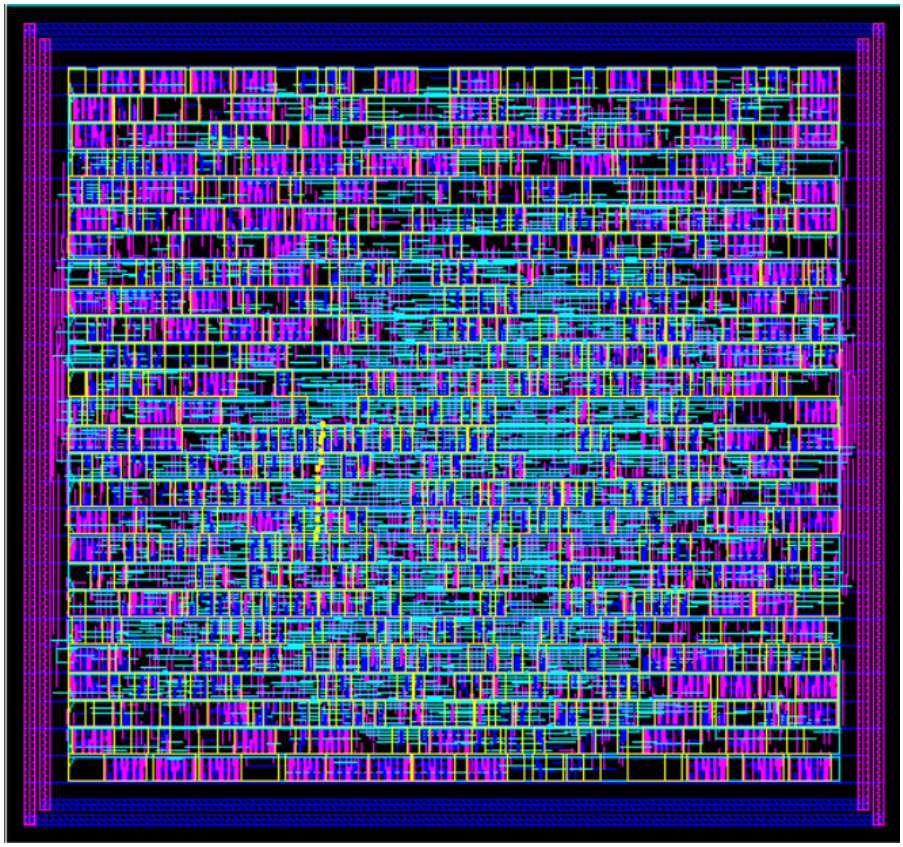
bardziej ograniczonych zasad w celu poprawy wydajności. Prawidłowo zakończona weryfikacja DRC zwraca informację:

```
***End Verify DRC(CPU:0:00:00.1 ELAPSED TIME:0.00 MEM:0.0M)***
                Verification Complete : 0 Viols.
```

Po zakończeniu całkowitej weryfikacji układ może zostać wyeksportowany do włożenia w ramę. Raport z całej weryfikacji projektu został dołączony jako dodatek B.

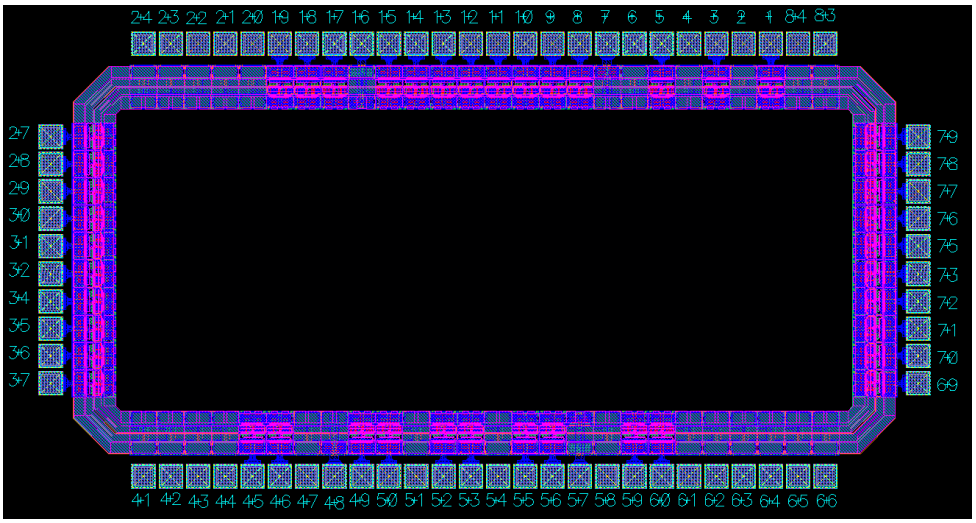
3.3 Przygotowanie do fabrykacji

Przed zakończeniem projektu pozostaje jeszcze ostatni element do dodania. Każdy układ scalony ma specjalną ramę z wyprowadzeniami do połączenia z innymi elementami i zasilaniem. Pierwszym krokiem do fabrykacji jest import topografii do programu Virtuoso:



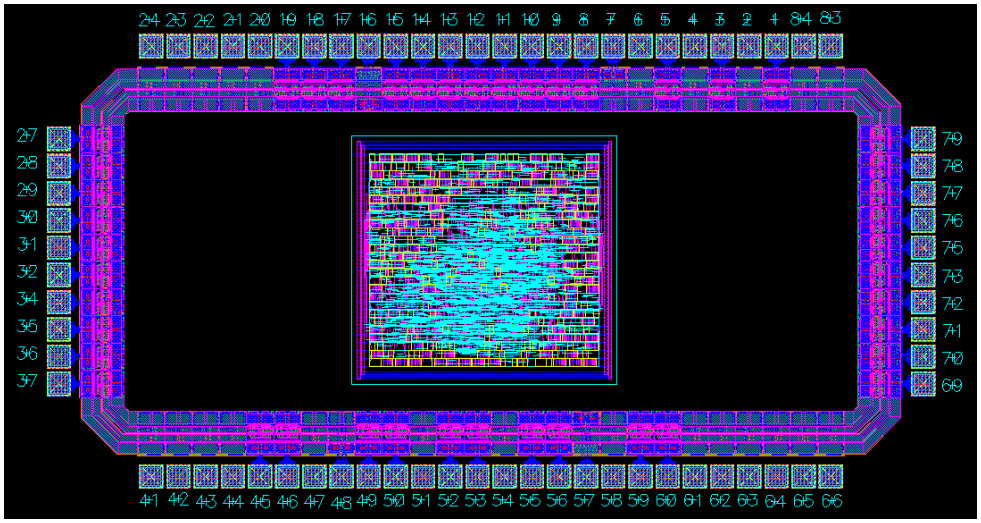
Rysunek 3-9: Moduł w programie Virtuoso [opr. własne]

Po dodaniu układu do Virtuoso widoczny będzie widok tzw. abstrakcyjny, czyli bez detali komórek standardowych. Kolory warstw po imporcie projektu mogą się różnić, gdyż Virtuoso używa ustawień o gamie kolorów zawartych w pliku *display.drf,* których nie ma w programie Innovus. Układ trzeba umieścić w ramie, która zabezpiecza przed wyładowaniami ESD i zapewnia połączenie układu z innymi elementami elektronicznymi. Zaproponowana rama znajduje się na poniższym rysunku:



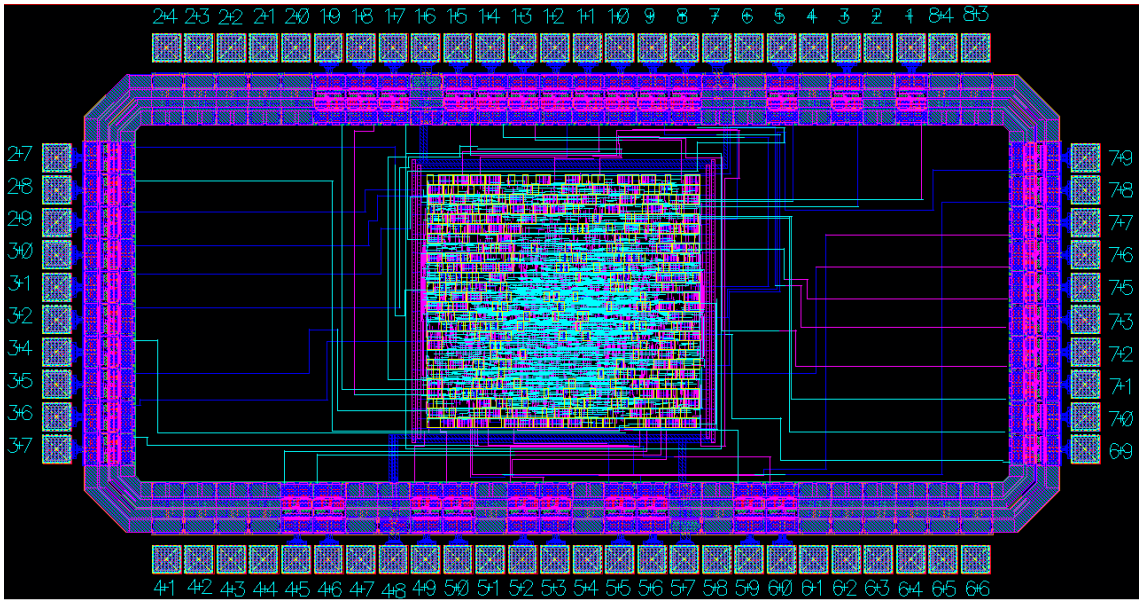
Rysunek 3-10: Rama układu scalonego [opr. własne]

Rama składa się z 68 pinów. Pady są komórkami do połączenia układu z innymi urządzeniami elektronicznymi. Ochrona przed wyładowaniami elektrostatycznymi jest zapewniona przez komórki tranzystorów zawarte w padzie. Zbudowana rama w formie pierścienia nie jest przypadkowa, gdyż przy dużej produkcji jest to wymagane. Niekiedy rama może być zredukowana do minimum przy tworzeniu nietypowej obudowy lub minimalizacji kosztów. Kolejnym krokiem jest umieszczenie modułu lub modułów w środku ramy.



Rysunek 3-11: Umiejscowienie modułu do połączenia z ramą [opr. własne]

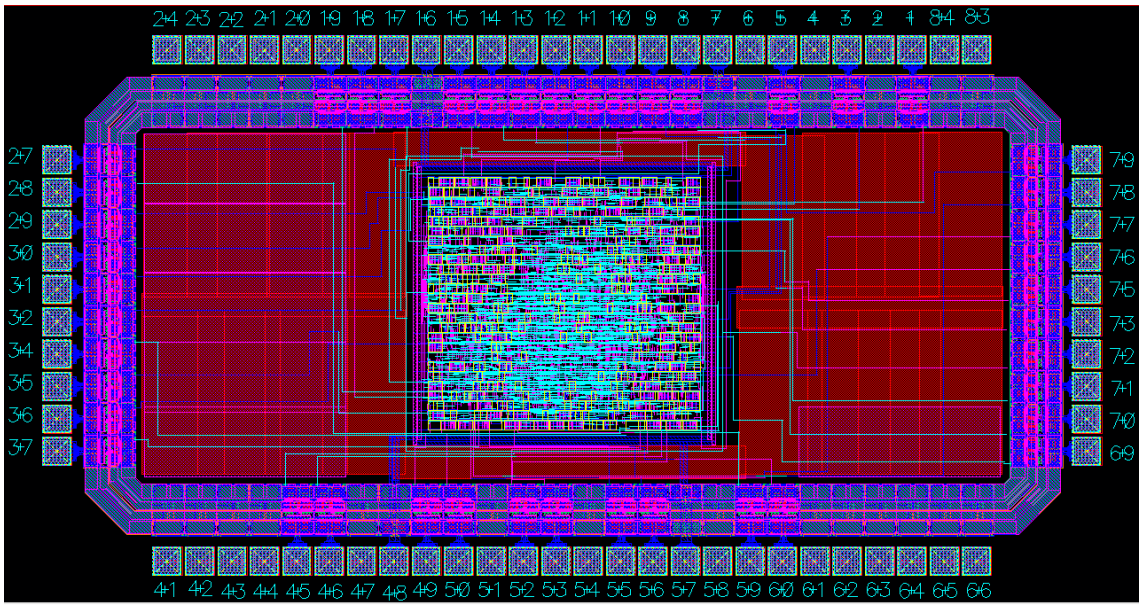
Ostatecznym etapem jest połączenie pinów wejścia/wyjścia z padami ramy. Przy skomplikowanych projektach zalecane jest, by połączenia wykonał program automatycznie, a pozostałe poprawki nałożył projektant. W małych projektach wybór zależy od projektanta. Ręczne połączenie z ramą pozwala na dopasowanie ścieżek do własnych potrzeb, ale wymaga dużej ilości czasu. Po zakończeniu operacji połączenia tzw. *wire bonding* pozostaje sprawdzić, czy wszystkie reguły zostały zachowane.



Rysunek 3-12: Moduł kontrolera połączony z ramą [opr. własne]

Połączenia zostały wykonane automatycznie, sprawdzone i poprawione. Po wykonanych analizach połączeń i sprawdzeniu reguł projektowych układ scalony uznajemy za wykonany. Na tym etapie można zakończyć pracę nad projektem.

W zależności od procesu fabrykacji należy pamiętać o jednym dodatkowym kroku do dodania. Niektóre procesy tworzenia układów scalony używają metody chemiczno-mechanicznego polerowania, które sprawia, że układy są bardziej spłaszczone. Jeżeli proces używa tej metody, niezbędne jest zapewnienie minimalnej gęstości dla warstwy poly oraz niektórych warstw metalicznych. Dla techniki, w której ten układ został zbudowany, czyli AMI C5N, wymagania wypełnienia warstw wynoszą: poly: 15%, metal1: 30%, metal2: 30%. [2] Wolne miejsca zostały zapełnione tymi warstwami. Końcowy wynik prac nad układem scalonym został przedstawiony na rysunku 3-13.



Rysunek 3-13: Finalna wersja układu scalonego [opr. własne]

Przygotowany w ten sposób projekt jest w całości gotowy do fabrykacji. Standardem w branży producentów układów scalonych jest GDSII. Po wyeksportowaniu tego pliku projektant może zamówić realizację swojego układu i czekać na gotowy produkt.

Podsumowanie i wnioski

Projektowanie układów w metodologii od ogółu do szczegółu ASIC jest bardzo skomplikowanym procesem. Każdy osobny projekt wymaga odpowiedniego podejścia i metod, które wynikają z różnorodniej funkcjonalności układów elektroniki. Osoba projektująca takie układy musi mieć niezbędną wiedzę na temat tworzenia modułów w języku opisu sprzętu i znać podstawy syntezowalnej składni. Implementacja kodu dla układów FPGA jest czasochłonna, dlatego doświadczenie projektanta jest sprawą kluczową w tej kwestii. Doświadczona osoba po stworzeniu kilku projektów unika podstawowych błędów i trzyma się konwencji przyjętej w poprzednich projektach. Dodatkowo implementowany kod powinien być testowany wraz z jego rozwojem. Po etapie weryfikacji kodu projektowana jest topografia. Ponadto cały proces się komplikuje, gdyż gama metodologii prowadzenia procesu tworzenia oraz testowania oprogramowania jest szeroka i zależna od wielkości i cech dla danego projektu. Ludzka natura przy tak dużym nakładzie pracy jest skłonna do błędów, dlatego każdy z etapów jest prowadzony przez wykwalifikowaną osobę lub zespół. Profesjonalne firmy zajmujące się produkcją układów scalonych przez lata budują swoją bazę pracowników i narzędzi, które skutkują sukcesami w tworzeniu układów scalonych.

Tworzenie układów scalonych w dużej mierze oprócz czynników ludzkich zależy od narzędzi i bibliotek przyjętych do syntezy. Im więcej komórek standardowych jest dostępnych dla narzędzia syntezującego, tym większa szansa, że wygenerowany układ będzie mniej skomplikowany. Mniejsza ilości bramek i komórek oznacza mniejszą ilość połączeń na topografii i ostatecznie mniejszą szansę na błąd. Skrypt używany do syntezy w tym projekcie posiada podstawowe komendy z domyślnymi ustawieniami, które bardzo dobrze pokazują schemat generacji netlisty. Dla zaawansowanych projektantów interfejs narzędzia syntezującego ma wiele opcji, którymi można manipulować powstający opis strukturalny.

Kolejnym dużym czynnikiem przy projektowaniu topografii jest ilość połączeń. Przy założeniu idealnej, pełnej biblioteki z wszystkimi typami komórek, wciąż pozostaje kwestia zaawansowanego łączenia elementów. Jeżeli w bramkach użyjemy, na przykład warstwy metal1 to zredukuje to możliwości używania tej warstwy w późniejszych połączeniach. W projektach z małymi komórkami ten problem można rozwiązać przez zwieszenie rozmiarów rzędów, które pozwalają na więcej połączeń i przelotek, ale zwiększa to powierzchnię układu scalonego. Większa gęstość warstw metalicznych i ich

rezystancja może powodować problemy z zapewnieniem ograniczeń czasowych przyjętych na etapie syntezy projektu. Nie ma uniwersalnej metody dla tworzenia topografii, ale dzięki zaawansowanym narzędziom do automatycznego łączenia i optymalizacji gęstości (np. NanoRoute) można uzyskać dokładną analizę. Dodatkowym problemem, który może wystąpić w końcowym etapie tworzenia układu, jest sprawdzenie reguł projektowych. Zdarza się, że różnice między plikiem technologicznym .lef a zasadami projektowymi dołączonymi do narzędzia sprawdzającego są różne, co skutkuje fałszywymi błędami. Problem można poprawić w pliku DFII, znając poprawne reguły projektowania, zmieniając warstwy, które powodują błędy.

Proces tworzenia układu scalonego dla kontrolera windy składał się w tylko z jednego elementu dołączonego do ramy. Ze względu na złożoność funkcjonalną duże projekty dzieli się na mniejsze układu, które są osobno implementowane, testowane, syntezowane i ostatecznie razem łączone z ramą. Przy korzystaniu z gotowych makr jak pamięci, lepiej jest zainwestować czas i połączyć układy ręcznie niż korzystać z automatycznych narzędzi. Proces nakładania warstw łączących jest powtarzany kilka razy. Te kroki są niezbędne do optymalizacji topografii, aby zapewnić wszystkie cechy dla układu scalonego. Należy pamiętać, że wszystkie elementy i rama powinny być stworzone w tej samej technologii przy użyciu tym samym bibliotek, aby uniknąć problemów z różnymi warstwami. Po zakończeniu prac należy zapisać projekt do pliku GDSII, który w tej postaci jest gotowy do fabrykacji.

Streszczenie

Tematem pracy jest system sterowania windą, który został zaprojektowany do fabrykacji jako układ ASIC. Oprogramowanie zostało napisane w metodologii nastawionej na rozwój cech. Architektura systemu została zaimplementowana w języku Verilog, wykorzystując narzędzia Aldec Active-HDL oraz Iverilog. Jednym z głównych założeń projektu było stworzenie systemu bazującego na dotychczasowych rozwiązaniach. Dogłębna analiza tematu współczesnych strategii kierowania windą w pierwszym rozdziale pracy wskazuje na wybranie strategii strefowej. Użytkownik wysyła żądania przy użyciu przycisków znajdujących się na zewnątrz szybu oraz wewnątrz kabiny jak w standardowej windzie. Dodatkowo zawarte zostały przypuszczenia odnośnie do przyszłych usprawnień w systemach wind, bazując na zaawansowanych algorytmach wysyłki docelowej stosowanych w najwyższych wieżowcach. W kolejnym rozdziale autor skupia się na architekturze projektowanego systemu oraz o wykorzystanych metodach rozwoju i testowania oprogramowania. Proces w oparciu o cechy został wybrany ze względu na początkowe założenia, jakim było zachowanie windy. Podstawą działania windy jest maszyna stanu obsługująca poruszanie się po piętrach oraz kontrolę nad drzwiami windy. Modułowość kodu udało się osiągnąć dzięki analogii sterowania systemem na każdym z pięter. Zaprojektowane resetowalne przyciski wewnątrz kabiny umożliwiają pasażerowi zmienić wybór celu podróży w sytuacji pomyłki. Ostatnim krokiem w budowaniu opisu behawioralnego jest sprawdzenie wszystkich dodanych funkcji. Przetestowanie jest kluczową sprawą przy projektowaniu układu ASIC, gdyż te układy nie mają możliwości zmiany struktury po fabrykacji. Po otrzymaniu satysfakcjonujących wyników układ został zsyntezowany do opisu strukturalnego układów cyfrowych w technologii CMOS 50nm. Projekt topografii jest wykonany w programie Cadence Innovus w kolejności, która została opisana szczegółowo w pracy. Ostatecznym punktem w tworzeniu układu scalonego jest dodanie ramy z wyprowadzeniami. Zakończony układ jest gotowy do fabrykacji.

Summary

The subject of the thesis is the elevator control system, which was designed for fabrication as an ASIC system. The software was written in a methodology focused on the development of features. The system architecture has been implemented in the Verilog language using the Aldec Active-HDL and Iverilog tools. One of the main assumptions of the project was to create a system based on existing solutions. An in-depth analysis of the topic of modern elevator management strategies in the first chapter of the work indicates the selection of a zone strategy. The user sends requests using buttons located outside the shaft and inside the cabin as in a standard elevator. In addition, there were assumptions about future improvements in elevator systems based on advanced shipping algorithms used in the top high-rise buildings. In the second chapter, the author focuses on the architecture of the designed system and about the methods used to develop and test the software. The process based on features was chosen because of the initial assumption of the elevator. The base of the elevator is a state machine that supports moving over floors and control over the elevator door. The modularity of the code was achieved thanks to the analogy of system control on each of the floors. Designed resettable buttons inside the cab allow the passenger to change the choice of destination in the event of a mistake. The last step in building a behavioral description is to check all the added features. Testing is a key issue when designing the ASIC system, as these systems do not have the ability to change the structure after fabrication. After receiving satisfactory results, the system was synthesized for a structural description of digital circuits in 50nm CMOS technology. The topography design is made in the Cadence Innovus program in the order described in detail in the paper. The final point in creating an integrated circuit is adding a frame with leads. The completed circuit is ready for fabrication.

Spis stosowanych skrótów

ASIC – Application Specific Integrated Circuit

DRC – Design Rule Check

ERC – Electrical Rules Checking

FDD – Feature Driven Development

FPGA – Field Programmable Gate Array

GUI – Graphical User Interface

HDL – Hardware Description Language

I/O – Input/Output

LEF – Library Exchange Format

LVS – Layout Versus Schematic

MMMC – Multi-Mode Multi-Corner

SDC – Standard Design Constraint

RAM – Random Access Memory

RTL – Register Transfer Level

Słowa kluczowe

- ASIC,
- winda,
- Verilog,
- układ scalony,
- Cadence,
- algorytm windy,

Keywords

- ASIC,
- elevator,
- Verilog,
- integrated circuit,
- Cadence,
- elevator algorithm

Bibliografia

[1] Axelsson J., Bernelind S., *Elevator Control Stategies*, praca inżynierska, Sztokholm: Royal Institute of Technology, 2013

[2] Brunvand E., *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*, Addison Wesley Publishing Company, 2010.

[3] Crites R. H., Barto A. G., *Elevator Group Control Using Multiple Reinforcement Learning Agents*, Boston: Kluwer Academic Publishers, 1998

[4] Khomytskyi S., *Simulation of passenger lift in Witness software*, praca magisterska, Praga: Czech Technical University, 2016

[5] Morgan P., *Energy Analysis and Optimisation Techniques for Automatically Synthesised Coprocessors*, rozprawa doktorska, Edinburgh: Uniwersytet Heriot-Watt, Glasgow: Uniwersytet Strathclyde, 2008

[6] Smith M. J. S., *Application-Specific Integrated Circuit*, Addison Wesley Publishing Company, 1997

[7] Strakosch G. R., Caporale R. S., *The Vertical Transportation Handbook, 4th edition*, New Jersey: John Wiley & Sons, 2010

[8] Algorytm Scan, http://aragorn.pb.bialystok.pl/~wkwedlo/OS1-9.pdf, stan na dzień: 01.07.2019.

[9] Cadence NanoRoute Advanced Digital Router, https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/nano-route-advanced-digital-router-ds.pdf, stan na dzień: 01.07.2019.

[10] Destination Dispatch technology overview, https://elevation.fandom.com/wiki/Destination_dispatch, stan na dzień: 01.07.2019.

[11] Destination Dispatch, http://www.neii.org/destdispatch.cfm, stan na dzień: 01.07.2019.

[12] Dyrektywa 95/16/EC - Lifts - European standards for existing lifts, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:01995L0016-20130101, stan na dzień: 01.07.2019.

[13] Elevators UE https://ec.europa.eu/growth/sectors/mechanical-engineering/lifts_en, stan na dzień: 01.07.2019.

[14] European Lift & Lift Component Association, http://www.elca-eu.org/, stan na dzień: 01.07.2019.

[15] GEN2 PREMIER Otis, http://www.abpenterprises.com/gen2-premier.html, stan na dzień: 01.07.2019.

[16] Peters R., Smith R., *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, 01.07.2019.

[17] Shanghai Tower, Elevators and escalators, https://www.slideshare.net/saramesallam/shanghai-tower-elevator-and-escalators, stan na dzień: 01.07.2019.

[18] Shanghai Tower, https://en.wikipedia.org/wiki/Shanghai_Tower, stan na dzień: 01.07.2019.

[19] Otis Compass Destination Dispatch floor selection number pad, https://www.schuminweb.com/life-and-times/life-2016/pittsburgh-2016-part-2/, stan na dzień: 01.07.2019.

[20] White Box Testing: A Complete Guide with Techniques, Examples, & Tools, https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/, stan na dzień: 01.07.2019.

Dodatek A.

Skrypt do syntezy w programie Cadence Genus

```
# ******************************************************
# * Script Name  : Genus Legacy syntesis script

date
set LOCAL_DIR "[exec pwd]/"
set LIB_PATH    "/home/student/DYPLOM/dyrdol/UofU/UofU_Digital_v1_2
/home/student/DYPLOM/dyrdol/UofU/UofU_Digital_v1_2"
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 { \
UofU_Digital_v1_2.lib \
}
set LEF_LIST { \
UofU_Digital_v1_2.lef \
}

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 /

# 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.

Raport weryfikacji topografii

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Jun 27 21:26:57 2019
Design Name: elevator
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (72.7700, 72.2400)
Error Limit = 1000; Warning Limit = 50
Check all nets
Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Jun 27 21:26:57 2019
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)

*** Starting Verify Geometry (MEM: 1149.6) ***
**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc
command. It still works in this release but will be removed in future release.
Please update your script to use the new command.
  VERIFY GEOMETRY ...... Starting Verification
  VERIFY GEOMETRY ...... Initializing
  VERIFY GEOMETRY ...... Deleting Existing Violations
  VERIFY GEOMETRY ...... Creating Sub-Areas
                  ...... bin size: 2160
  VERIFY GEOMETRY ...... SubArea : 1 of 1
  VERIFY GEOMETRY ...... Cells          :  0 Viols.
  VERIFY GEOMETRY ...... SameNet        :  0 Viols.
  VERIFY GEOMETRY ...... Wiring         :  0 Viols.
  VERIFY GEOMETRY ...... Antenna        :  0 Viols.
  VERIFY GEOMETRY ...... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.2  MEM: 114.7M)

innovus 3> #-report elevator.drc.rpt               # string, default="", user
setting
 *** Starting Verify DRC (MEM: 1289.1) ***
  VERIFY DRC ...... Starting Verification
  VERIFY DRC ...... Initializing
  VERIFY DRC ...... Deleting Existing Violations
  VERIFY DRC ...... Creating Sub-Areas
  VERIFY DRC ...... Using new threading
  VERIFY DRC ...... Sub-Area: {0.000 0.000 72.770 72.240} 1 of 1
  VERIFY DRC ...... Sub-Area : 1 complete 0 Viols.
  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:00.1  ELAPSED TIME: 0.00  MEM: 0.0M) ***
```

Dodatek C.

Spis zawartości dołączonej płyty CD

Scalony_system_sterowania_windą_Mateusz_Dyrdół_276528.doc – Tekst pracy zapisany w formacie MS Word,

Scalony_system_sterowania_windą_Mateusz_Dyrdół_276528.pdf – Tekst pracy zapisany w formacie rozszerzonym,

Elevator.zip – Plik skompresowany ze źródłami w języku Verilog,

Genus.zip – Plik skompresowany z projektem w programie Cadence Genus,

Innovus.zip – Plik skompresowany z projektem w programie Cadence Innovus,

Virtuoso.zip – Plik skompresowany z projektem w programie Cadence Virtuoso,

UofU.zip – Plik skompresowany zawierający biblioteki w technologii AMI C5N

Spis ilustracji