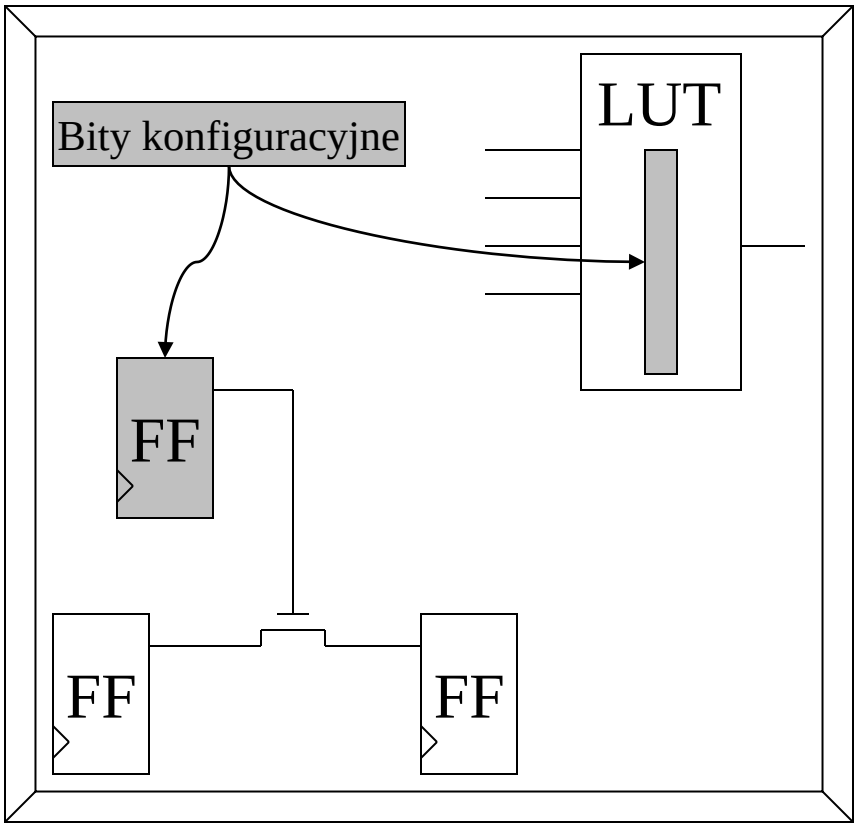
REμP

Pierwsze pomysły

Cel zasadniczy:

Dynamicznie rekonfigurowalny układ FPGA
jako
procesor ogólnego przeznaczenia

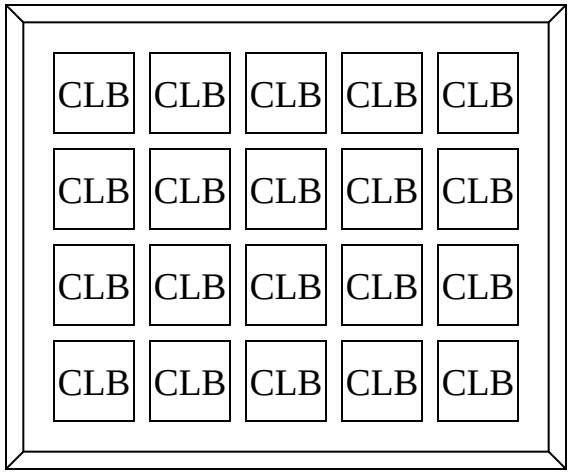
Co to jest FPGA?



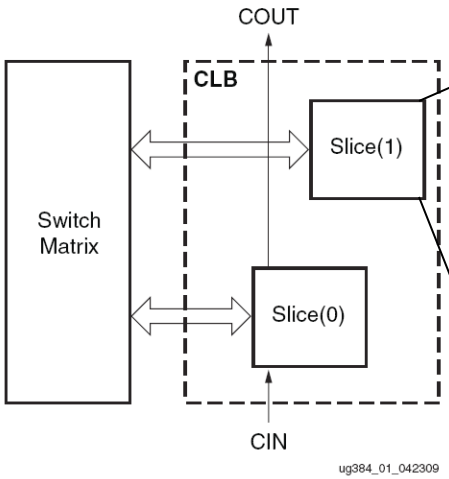
FPGA – Field Programmable Gate Array

Układ cyfrowy, którego wewnętrzna architektura definiowana jest zestawem bitów konfiguracyjnych. Określają one strukturę połączeń pomiędzy wbudowanymi elementami układu, takimi jak przerzutinki, multiplexery, bramki logiczne, itp.. Funkcje kombimacyjne realizowane są za pomocą małych bloków również konfigurowalnej pamięci.

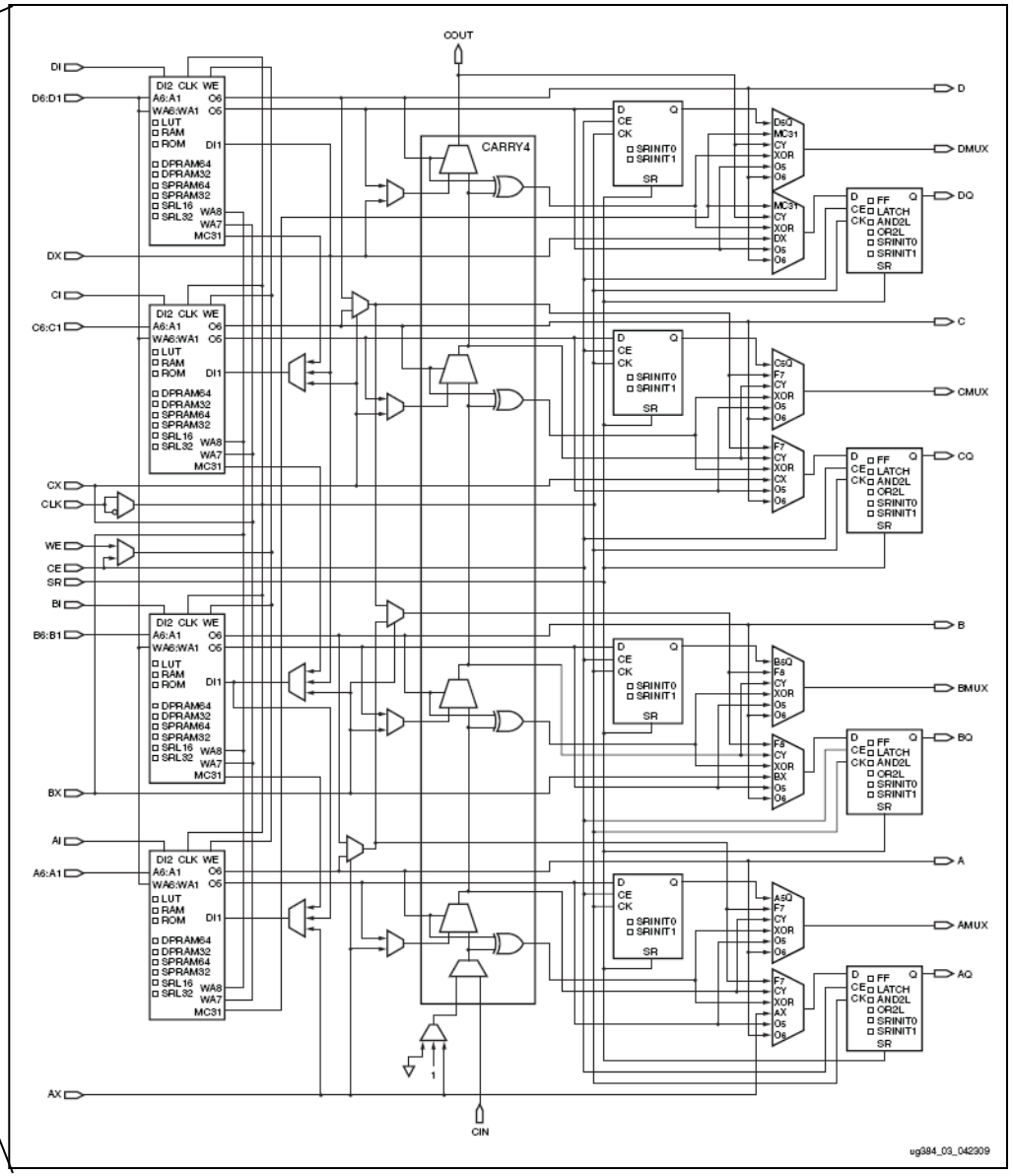
Zasoby układu pogrupowane są w macierz bloków CLB (Configurable Logic Blok).



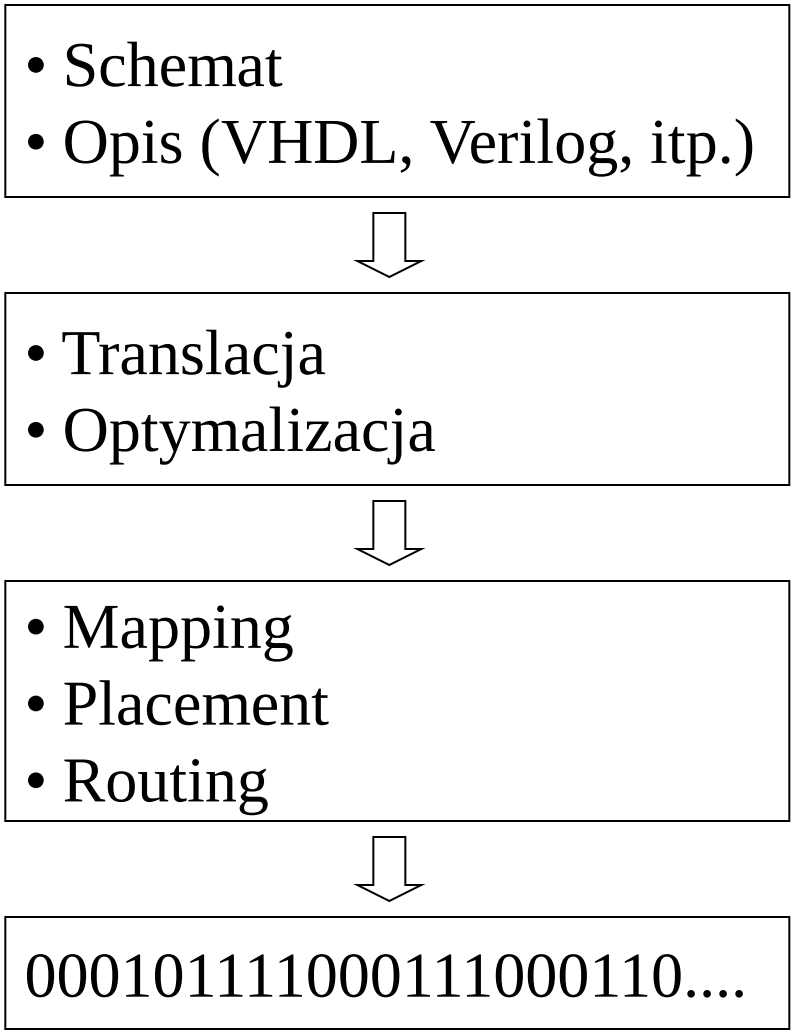
FPGA – trochę szczegółów



Pomiędzy blokami CLB, wzdłuż kolumn i wierszy, rozmieszczone są kanały routing'u, z również konfigurowalnymi macierzami przełączników (Switch Matrices)



FPGA – jak tego używać?



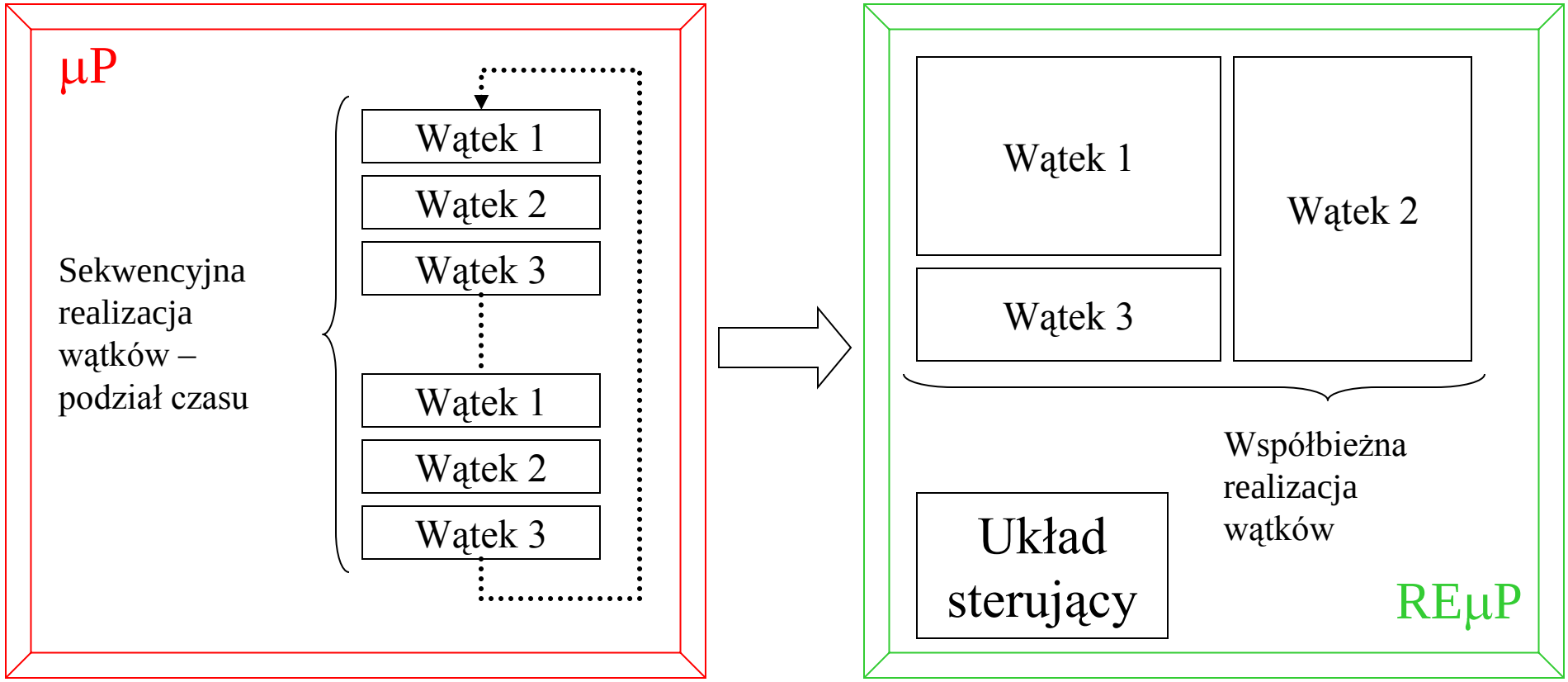
Specyfikacja projektu

Synteza logiczna

Synteza fizyczna

Zestaw bitów konfiguracyjnych (bitstream)

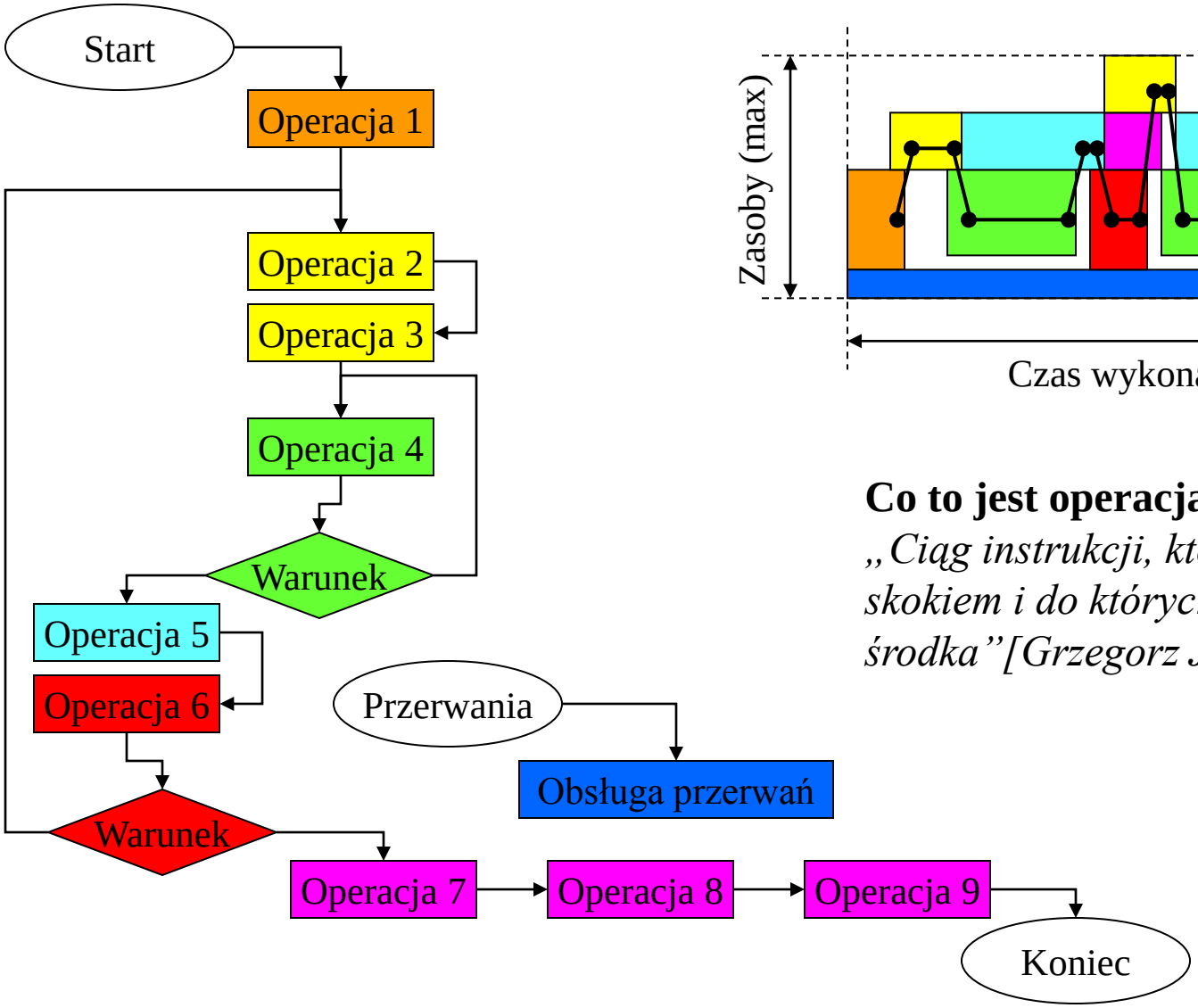
FPGA jako procesor



Zrównoleglenie jest **pierwszym** etapem przyspieszenia obliczeń (odpowiednik **łatwo skalowalnego** procesora wielordzeniowego)

Uwaga:
Wątek jest zbyt duży by zostać w całości zaimplementowanym w sprzęcie!!!

Partycjonowanie wątków

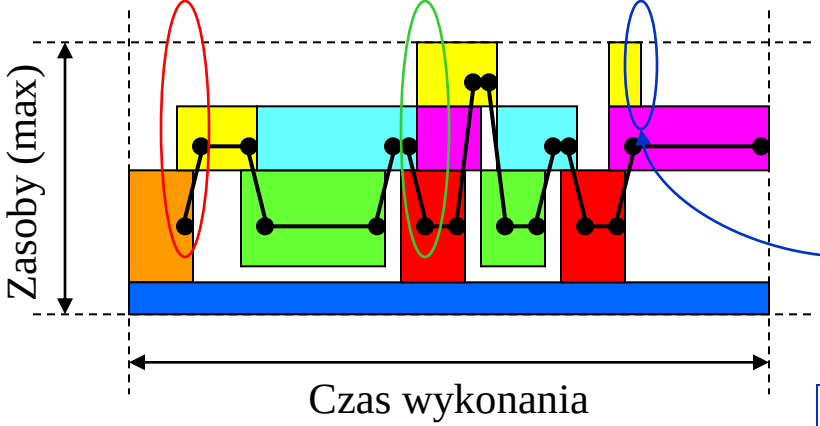


Co to jest operacja? Jedna z propozycji:
„Ciąg instrukcji, które nie kończą się skokiem i do których nie wskakuje się do środka”[Grzegorz Jabłoński]

Implementacja partycji

Przed „zakończeniem” danej partycji „pojawia” się partycja kolejna (ALAP)

W przypadku rozgałęzień, „pojawiają” się obie następne partycje (ALAP)



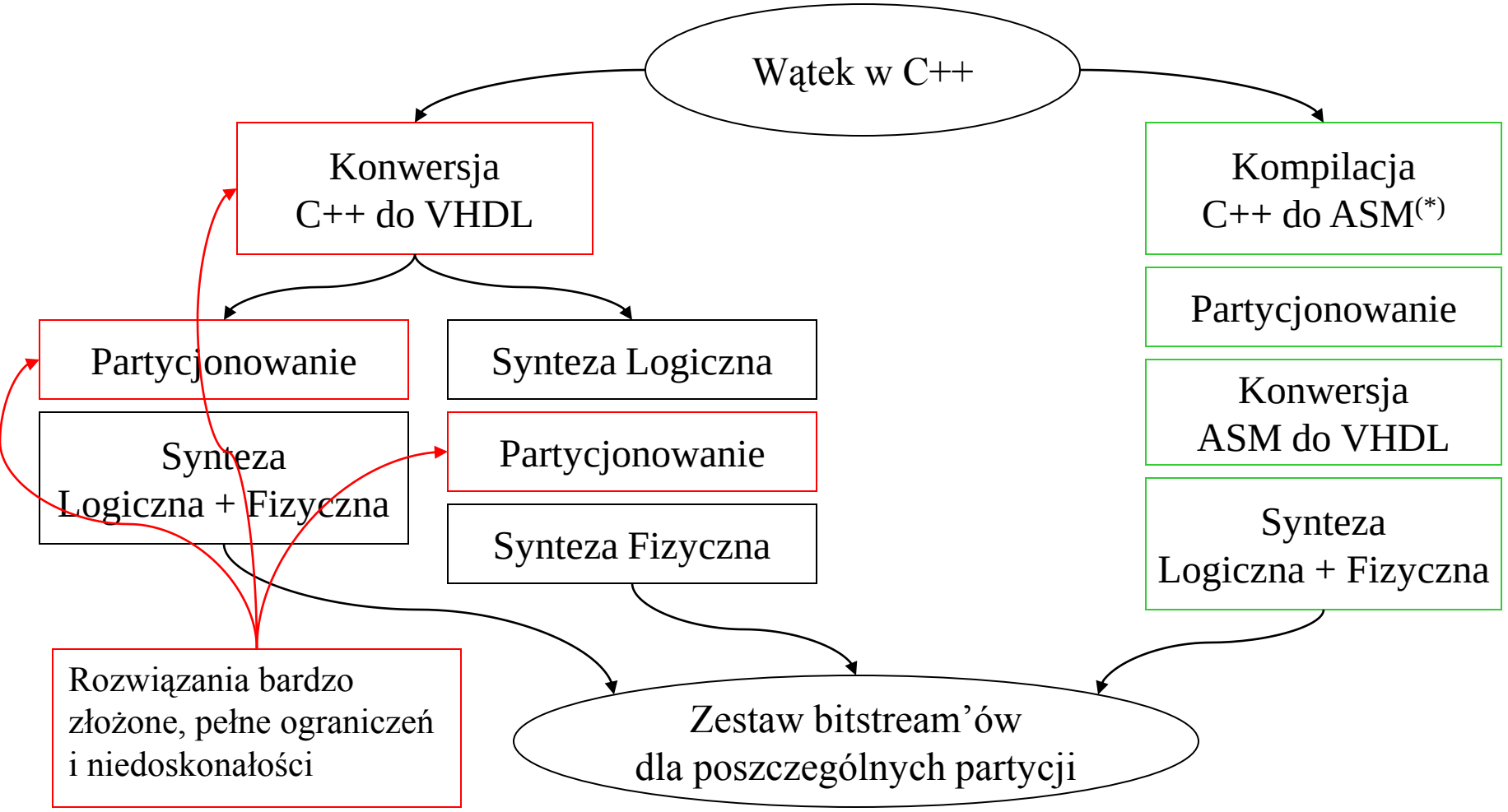
Nieużywane partycje „znikają” (ASAP)

Uwagi:

• Konstrukcja partycji musi umożliwiać umieszczenie ich w różnych miejscach układu

• „Rozgałęziające” się partycje muszą informować układ sterujący o „wybranym” rozgałęzieniu!!!

Konwersja wątków „soft” do wątków „hard”

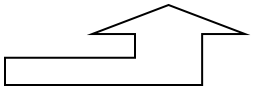


(*) ASM użyty dla uproszczenia rozważań. Docelowo lepszym kandydatem wydaję się być GIMPLE

Konwersja ASM do VHDL

```
char fun(char a, char b)
{
        char c, d, r;
        c = a * 2;
        d = b * 4;
        r = c - d;
        return r;
}

int main(void)
{
        ...
        c = fun(a,b);
        ...
}
```

```
LDS     R25,0x0186      Load direct from data space
LDS     R24,0x0182      Load direct from data space

LSL     R25             Logical Shift Left
LSL     R24             Logical Shift Left
LSL     R24             Logical Shift Left
SUB     R25,R24         Subtract without carry

STS     0x0183,R25      Store direct to data space
```



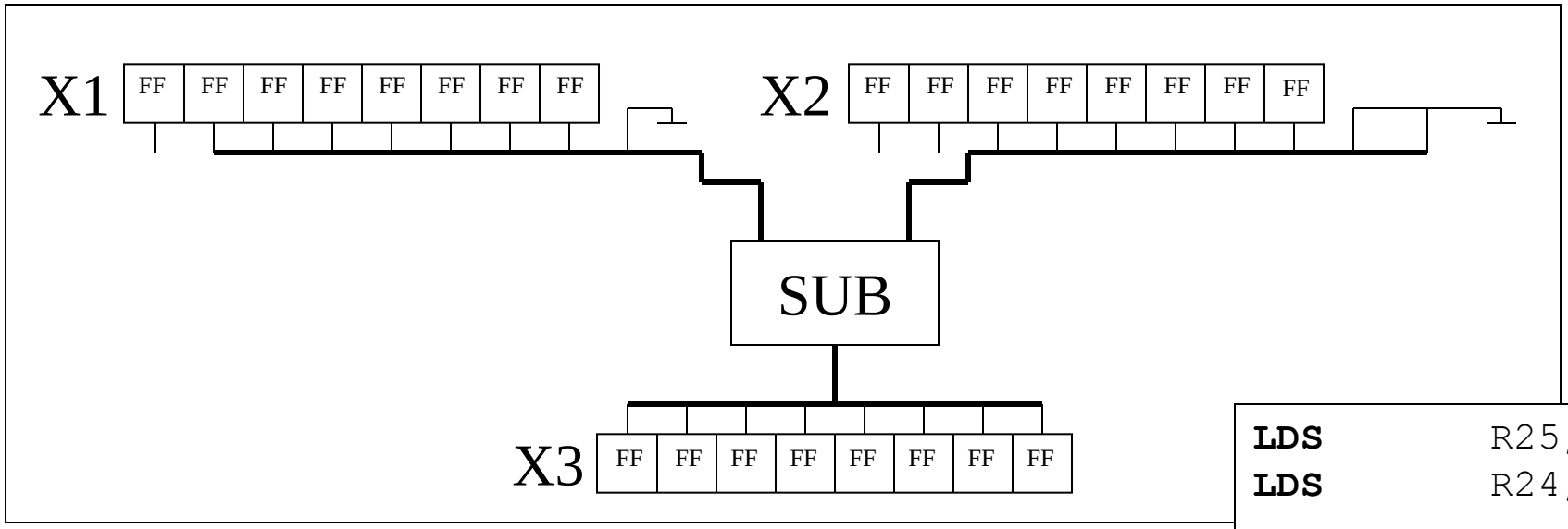
```
LDS   R,X
                    X : in std_logic_vector(N-1 downto 0);
                    variable R : std_logic_vector(X'high downto X'low);
                    R := X;
LSL   R
                    R := R(R'high-1 downto R'low) & '0';
SUB   Rx, Ry
                    Rx := Rx - Ry;
STS   X,R
                    X : out std_logic_vector(R'high downto R'low);
                    X <= R;
```

Konwersja ASM do VHDL - przykład

```
LDS     R25,X1
LDS     R24,X2

LSL     R25
LSL     R24
LSL     R24
SUB     R25,R24

STS     X3,R25
```

```
entity TEST is
        port(X1, X2 in: ...;
             X3 out: ...);

architecture TEST_behav of entity TEST is
begin

        process(clk)
                variable R25 : std_logic_vector(X1'high downto X1'low);
                variable R24 : std_logic_vector(x2'high downto X2'low);
        begin
                if rising_edge(clk) then
                        R25 := X1;
                        R24 := X2;
                        R25 := R25(R25'high-1 downto R25'low) & '0';
                        R24 := R24(R24'high-1 downto R24'low) & '0';
                        R24 := R24(R24'high-1 downto R24'low) & '0';
                        R25 := R25 - R24;
                        X3 <= R24;
                end if;
        end process;
end architecture;
```

Dopasowanie sprzętu do potrzeb wątku




```
LDS     R25,X1
LDS     R24,X2

LSL     R25
LSL     R24
LSL     R24
SUB     R25,R24

STS     X3,R25
```

Generacja sprzętu dedykowanego do wykonywanych operacji jest **drugim (zasadniczym)** etapem przyspieszenia obliczeń.

Opóźnienia i zasoby

```
LDS         R25,X1
LDS         R24,X2

LSL         R25
LSL         R24
LSL         R24
...
...
...
...
...
Rxx,Ryy
...
...
...
...
...
SUB
R25,R24

STS         X3,R25
```

W przypadku konwersji długiego ciągu rozkazów można:

• rozbić ten ciąg na mniejsze fragmenty, wprowadzając **rejestry przechowujące wynik pośredni**, tak aby opóźnienie na najdłuższej ścieżce kombinacyjnej nie przekraczało założonego okresu zegara

• narzucić na ścieżki między rejestrami wejściowymi a wyjściowymi **ograniczenie typu „multicycle”**

Zarówno w pierwszym jak i drugim rozwiązaniu, w wyniku procesu syntezy określona zostanie **liczba taktów** niezbędnych do realizacji wchodzących w skład partycji operacji (**czas „życia” partycji**).

Czas ten powinien być tak zbilansowany, aby **w trakcie pracy aktywnych partycji możliwe było skonfigurowanie kolejnych**. Jednocześnie partycje powinny być możliwie małe, by pozwolić na **współbieżną realizację jak największej liczby wątków**.

RISC czy CISC?

```
LDS         R25,X1
LDS         R24,X2

LSL         R25
LSL         R24
LSL         R24
???
???
???
???
???
???
???
???
???
???
???
SUB
R25,R24

STS         X3,R25
```

Optymalizacja wygenerowanego opisu w języku VHDL bierze pod uwagę nie pojedyncze rozkazy ale ich sekwencję. Pojęcia RISC i CISC tracą w tej sytuacji swoje standardowe znaczenie (nie ma czegoś takiego jak **opcode rozkazu**).

Jednak w reprezentacji pośredniej (ASM) pojawiają się **mnemoniki** odpowiadające operacjom, jakie należy wykonać na rejestrach. Operacje te mogą być znacznie bardziej złożone od tych, które można znaleźć w zbiorze rozkazów procesorów CISC. Ich implementacja polegać będzie jedynie na określeniu odpowiadającego im opisu w języku VHDL.

W tym znaczeniu możemy nazwać nasz procesor EISC (Extended Instruction Set Computer) albo VCISC (Very Complex Instruction Set Computer).

Co rekonfigurować?

Należy wybrać jedno z dwóch, niestety wykluczających się podejść:

- **Standardowe (stosowane w układach komercyjnych).**

\+ W tym podejściu będzie można dokonać **wstępnej weryfikacji** zaproponowanych rozwiązań (translacji, konwersji, zarządzania partycjami) za pomocą **ogólnodostępnych narzędzi i układów** (np. z rodziny Virtex).

– Pociągnie to jednak za sobą konieczność uwzględniania **wielu ograniczeń** (stosowania modułów Bus Macro do łączenia modułów dynamicznych ze statycznymi, używania zunifikowanych interfejsów bloków dynamicznych, itp.), co może znacząco zniwelować atrakcyjność zaproponowanych rozwiązań.

– Poza tym, nawet komercyjne narzędzia wspierające dynamiczną rekonfigurację są ciągle niedoskonałe (Early Access Partial Reconfiguration Lounge).

– **Przeniesienie** wypracowanych w tym podejściu technik rekonfiguracji na (ewentualny) własny, dedykowany układ ASIC będzie wymagało **dodatkowego, dużego nakładu pracy**.

- **Niestandardowe**

– Realizacja projektu w oparciu o własny, dedykowany układ rekonfigurowalny (w pierwszym etapie można by się oprzeć o jego model, nawet behawioralny) uniemożliwi wstępną, **fizyczną weryfikację** zaproponowanych rozwiązań.

– W celu otrzymania jakichkolwiek wyników będzie **wymagane opracowanie własnych narzędzi syntezy fizycznej.**

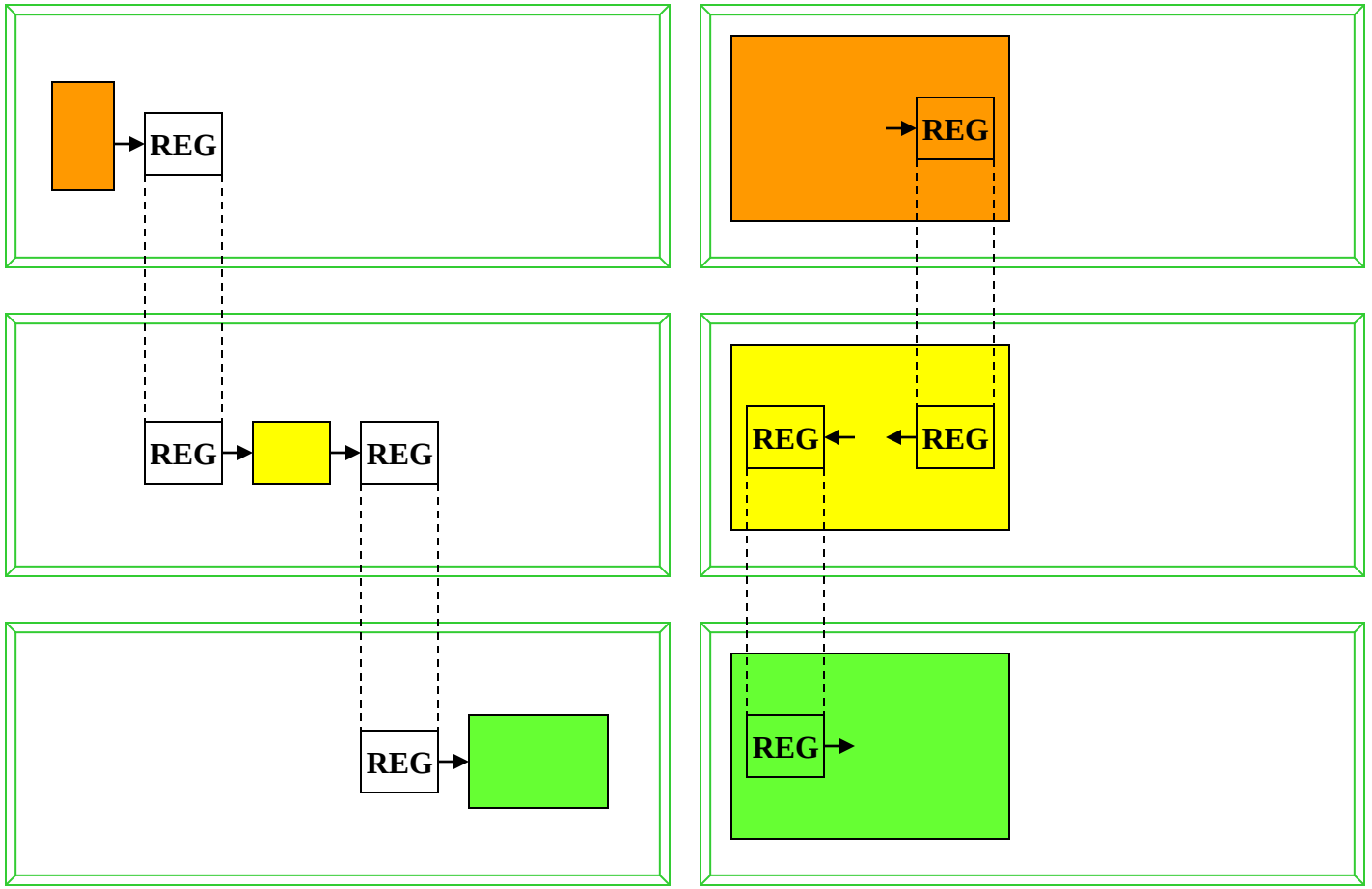
– Prawdopodobnie nie uda się też wykonać fizycznego układu mogącego **konkurować z komercyjnymi**.

\+ Podejście to jednak pozwoli **uniknąć wielu istotnych ograniczeń**, jakie narzucają rozwiązania komercyjne.

\+ Będzie się też lepiej wpisywało w **naukowy charakter projektu (nowa architektura).**

Jaka rekonfiguracja?

„Pełzająca” czy „wielokontekstowa” ?



Odpowiedź na to pytanie wymaga głębszej analizy i w dużej mierze zależy od odpowiedzi na pytanie „co rekonfigurować” – układy komercyjne wspierają jedynie rekonfigurację „pełzającą”.

Krótkie pytania, krótkie odpowiedzi

- **Architektura Harwardzka czy von Neumana?**
  Skoro procesor ogólnego przeznaczenia to może von Neumana? Należy jednak pamiętać, że w proponowanym rozwiązaniu nie ma typowego programu, jest zbiór bitstream'ów.
- **SIMD czy MIMD?**
  Jeśli to ma być uniwersalny procesor, to oczywiście MIMD.
- **Jaki model zarządzania pamięcią?**
  Nie wiem.
- **Jaki model ochrony?**
  Nie wiem.
- **Jaki model zarządzania segmentacji-stronicowani-haszowania?**
  Nie wiem.
- **Czy można skorzystać z opensoftprocesor'ów/opensoftcor'ów?**
  Nie. Nasz pomysł ma pójść znacznie dalej niż zarządzanie mniejszymi rdzeniami.
- **Jaki system operacyjny?**
  Pierwsze testy bez systemu operacyjnego. Ale należy o tym myśleć już teraz!
- **Co z operacjami zmiennoprzecinkowymi?**
  Do wykazania atrakcyjności naszego rozwiązania wystarczą operacje stałoprzecinkowe. W finalnej wersji projektu operacje zmiennoprzecinkowe mogą być implementowane dynamicznie tak jak wszystkie inne. I tu mogą się przydać gotowe IP-core'y.

Krótkie pytania, krótkie odpowiedzi

- **Czy transfer danych konfiguracyjnych nie będzie zbyt duży?**

  W dedykowanym układzie rekonfigurowalnym (nasz własny ASIC) można zoptymalizować proces rekonfiguracji (szeroka szyna danych konfiguracyjnych, możliwość rekonfiguracji niedużych fragmentów, itp.). Dodatkowo, np. konfiguracja wielokontekstowa pozwoli zminimalizować liczbę niezbędnych „przeładowań" bitstream'u – nieużywana w danej chwili konfiguracja może czekać w zapasowych przerzutnikach konfiguracyjnych.

- **Czy „zmniejszymy Grzesia do rozmiarów mikro, sklonujemy i będziemy dodawali jako kompilator i zarządcę zasobów do każdego naszego procesora"?**

  Zdecydowanie NIE!!! ;-)

Podsumowanie

- **Interdyscyplinarny projekt, znakomicie wpisujący się w zakres zainteresowań naszej Katedry, wymagający synergicznej współpracy specjalistów z wielu dziedzin:**
  - układy dynamicznie rekonfigurowalne
  - języki opisu sprzętu
  - architektura komputerów
  - procesory
  - techniki kompilacji
  - systemy operacyjne
  - układy ASIC
- **Niezbędne badania literaturowe**
  - potwierdzające unikalność naszego pomysłu
  - pozwalające zdecydować o jego wykonalności
- **Sposób realizacji projektu jest ciągle zagadnieniem otwartym, zatem wszystkie pomysły są mile widziane**
- **Projekt o dużym znaczeniu komercyjnym. Ma on mieć jednak charakter głównie naukowy, można więc pozwolić sobie na odważne propozycje i jednocześnie na niezbędne uproszczenia, ułatwiające wyciągnięcie ciekawych wniosków.**

Głosy w dyskusji

• **W procesie translacji ciągu rozkazów do sprzętu istotnym problemem mogą być wskaźniki**

• **Duży wpływ na efektywność zrównoleglenia wątków będzie miał czas dostępu do pamięci.**

• **Oprócz zrównoleglenia wątków i dopasowania sprzętu do zestawu instrukcji, duży wpływ na szybkość obliczeń może mieć zrównoleglenie przetwarzania w ramach pojedynczej operacji.**
Np.: jeśli w tłumaczonym na sprzęt ciągu instrukcji znajdą się:
SUB R1,R2
SUB R3,R4
to w wyniku translacji powinny powstać dwa niezależne układy odejmujące, realizujące operacje odejmowania współbieżnie (na marginesie: przy zaproponowanym sposobie translacji ASM do VHDL taka funkcjonalność jest niejako wbudowana).

• **Rekonfiguracja wielokontekstowa wydaję się być pomocną przy ukrywaniu latencji dostępu do pamięci (konfiguracja, która czeka na dane z pamięci jest zastępowana inną, która może pracować). Ten sposób rekonfiguracji może jednak stwarzać problemy przy realizacji rozgałęzień.**

• **Należy rozważyć możliwość zrównoleglania pętli**

• **W trakcie przeglądu literatury warto zwrócić uwagę (kolekcjonować i rozpowszechniać) na słowa kluczowe**