

1. Поняття про мережі Петрі
2. Прості мережі Петрі
3. Розширені мережі Петрі
4. Приклади реалізації мереж Петрі

1. Поняття про мережі Петрі

Мережі Петрі призначені для представлення координації асинхронних подій і застосовуються для описування взаємовідносин між паралельними процесами та їх синхронізацією.

Мережа Петрі - це орієнтований, дводольний граф з мітками (марками). Це визначення треба розуміти так (рис.3.1):



Вузли:	а) 	Пункт (може бути вільним, або поміченим), “Стан”
	б) 	Перехід, “Дія”(ACTION)
Ребра		Орієнтовані зв'язки між процесорами

Рис.3.1. Елементи мережі Петрі

Кожна мережа Петрі є графом з двома різними групами вершин: вузли та переходи. Між вузлами та переходами можуть міститися орієнтовані ребра (дуги), але два вузли або два переходи не можуть з'єднуватися ребрами. Між кожною парою вузол/перехід може існувати максимально одне ребро від вузла до переходу (ребро входу) і максимально одне ребро від переходу до вузла (ребро виходу). Вузли можуть бути вільними або зайнятими міткою (маркованими); переходи не можуть бути маркованими. Вузли, що є стартовими пунктами одного ребра до одного переходу t , називаються далі вхідними вузлами переходу t . Вузли, що є кінцевими пунктами ребра від переходу t називаються вихідними вузлами переходу t .

На рис.3.2 показано просту мережу Петрі, що має один перехід, три ребра і три вузли, два з яких марковані і один не маркований. Кожен вузол зв'язаний з переходом за допомогою одного ребра.

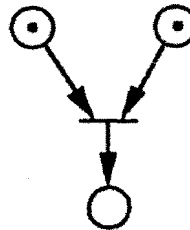


Рис.3.2.

2. Прості мережі Петрі

Визначення:

Активізація (Стан): перехід t активізований, якщо всі вхідні вузли p_i цього переходу марковані.

Тобто, активізація - це залежна від часу властивість переходу і описує деякий стан. Перехід мережі Петрі на рис.3.2 активізований, бо обидва вузли ребер, що входять у перехід, марковані.

Ввімкнення (Подія) : активізований перехід t може вмикатися. Тоді зникають марки з всіх вхідних вузлів p_i переходу t і маркуються всі вихідні вузли p_j цього переходу.

Процес увімкнення переходу має передумовою його активізацію.

Як видно з рис.3.3, в процесі ввімкнення відбувається зміна маркування вузлів мережі Петрі: перехід активізований, бо обидва вузли вхідних ребер марковані. Після ввімкнення переходу маркування обох верхніх (вхідних) вузлів зникає, в той час як на нижньому (вихідному) вузлі з'являється нова марка. Загальна кількість маркувань у будь-якій мережі Петрі не залишається постійною. Якби на вихідному вузлі вже була марка, то вона б переписалася, тобто вузол як і раніше був би зайнятий маркою.



Рис.3.3. Перемикання переходу

Невизначеність: якщо одночасно активізовані декілька переходів, то не зрозуміло, який з них перемкнеться першим.

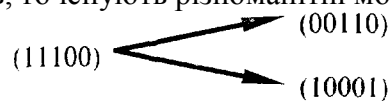
Зроблені вище визначення не дають уявлення про порядок перемикання декількох активізованих переходів. Одночасне перемикання їх неможливе! Як показано на рис.3.4, у цьому випадку можуть відбуватися два процеси і який з них фактично має місце, в мережі Петрі визначити не можна. У зв'язку з тим, що ввімкнення одного або іншого переходу не може бути здійснене за допомогою зовнішніх параметрів, мережа Петрі має в цьому випадку невизначеність.

Стан: стан маркування (або, коротко, стан) мережі Петрі до деякого моменту часу T визначено як сукупність маркувань кожного окремого вузла мережі.



Рис.3.4. Недермінованість перемикань мережі Петрі

У простих мережах Петрі стан маркування можна відобразити за допомогою деякої послідовності бінарних цифр (двійковий рядок). Можливе перемикання переходу задається за допомогою переходу в послідовність стану (якщо за аналогією з показаним на рис.3.4 можуть перемикатися декілька переходів, то існують різноманітні можливі послідовності станів):



У тому випадку, коли перемикання кожного переходу визначається цим правилом, всі послідовності станів одного заданого початкового стану можуть бути "обчислені" комп'ютером і можуть бути розпізнані вірогідні блокування.

Генерування марок: перехід, що не має жодного вхідного ребра, завжди активізований і може постійно видавати нові марки на вихідні вузли, що з'єднані з ним.

Знищення марок: перехід, що не має жодного вихідного ребра і має тільки одне вхідне ребро, завжди активізований тоді, коли це місце марковано і він може постійно знищувати марку.

"Мертвий" стан: Мережа Петрі перебуває в "мертвому" стані (блокована), якщо жоден з переходів не активізований.

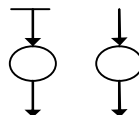


Рис.3.5. Генерування та гасіння марок

Блокована мережа Петрі є статичною, тобто в ній немає нових послідовностей станів. Наприклад, обидві мережі Петрі, що показані на рис.3.4, блоковані.

“Живий” стан: мережа Петрі перебуває в “живому” стані (не блокована в жодному моменті часу), якщо хоча б один з її переходів активізований і це справедливо також для кожного наступного стану.

Зауважимо, що “живий” стан не є протилежністю “мертвого” стану. Мережа Петрі в “живому” стані не блокована і не перебуває в жодному із можливих послідовностей станів, в той час як не блокована мережа Петрі не обов'язково має бути в “живому” стані. Наприклад, блокування може відбутися тільки після багатьох послідовних кроків. На рис. 3.6 показано два приклади мереж.

Ліва мережа Петрі (рис.3.6) є “живою” тому, що її маркування змінюється від одного вузла до іншого, але не зникає; перехід завжди тут активізований. В правій мережі Петрі можуть перемикатись або перехід u (і тоді мережа негайно переходить у “мертвий” стан), або перехід s . Нарешті можуть перемикатись один раз переходи t і U , що веде також до “мертвого” стану мережі Петрі. За допомогою мереж Петрі можна уникнути взаємоблокування процесів або виникнення суперечних даних у тих випадках, коли два процеси мають звернутися до однієї і тієї області даних (рис.3.7).

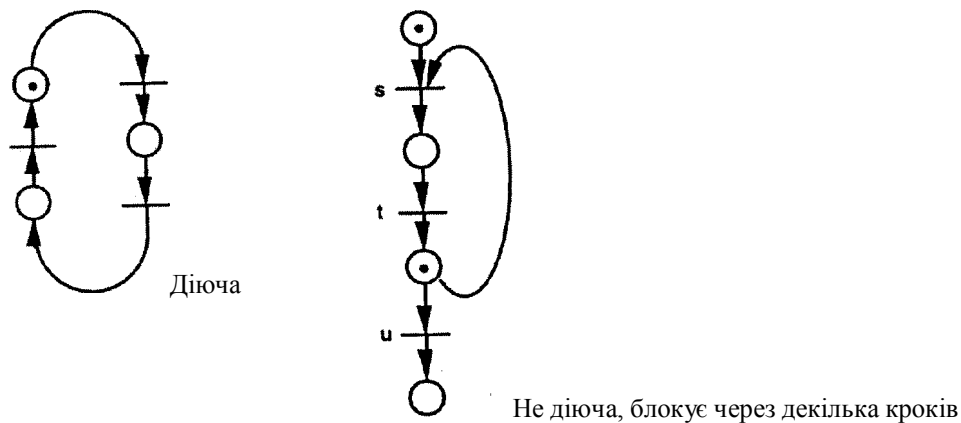


Рис. 3.6. Діюча та блокуюча мережі Петрі



Рис.3.7. Доступ двох процесів P_1 і P_2 до загальних даних

Процес P_1 (виробник) генерує тут незалежно від процесу P_2 дані, записує їх у буферну область пам'яті і хоче без затримки працювати далі. Процес P_2 (споживач) читає дані із цього буфера і використовує їх паралельно з процесом P_1 . Щоб уникнути суперечливих даних треба виконати таку умову:

одночасний доступ процесів в одну і ту саму область пам'яті має бути виключений.

Це викликає потребу в синхронізації процесів, тобто один з них має деякий час почекати.

На рис.3.8 показано простий приклад для випадку, коли процеси P_1 та P_2 мають бути синхронізовані, тому, що вони, наприклад, хочуть користуватися одними і тими самими даними.

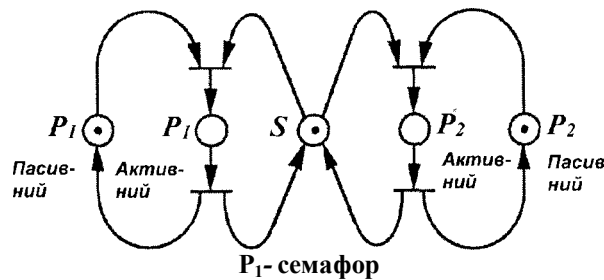


Рис. 3.8. Мережа Петрі для синхронізації процесів

Кожний процес має тут два стани - активний і пасивний, які символізуються двома вузлами. Процес перебуває в тому стані, який позначається відповідною маркою (на рис.3.8 обидва процеси пасивні). Кожен з двох процесів може змінювати свій стан з пасивного на активний і навпаки, перемикаючи відповідний перехід. Обидва цикли стану для P_1 і P_2 аналогічні простому контуру, що зображений зліва на рис.3.6, але ці цикли пов'язані між собою за допомогою семафора S . Процес, що хотів би змінити свій стан з пасивного на активний (щоб звернутися до загальних даних), потребує для цього переходу маркування в S . Це означає, що він тільки тоді може змінити свій стан на активний, коли марка семафора не використовується іншим процесом (який в цей час перебуває в активному стані). При переході в активний стан семафор S звільняється від маркування; у тому випадку, коли інший процес запросить перехід з пасивного стану в активний (доступ до загальних даних), він має чекати, поки перший процес не перейде знову зі стану активного в пасивний і знову з'явиться семафор-марка S .

Синхронізація за допомогою цієї мережі Петрі є високонадійною, оскільки, в кожний момент часу тільки один процес перебуває в активному стані. Таким чином, в разі одночасного запиту процеси з паралельних перетворюються в послідовні і їх блокування виникнути не зможе.

3. Розширені мережі Петрі

За допомогою трьох простих розширень прості мережі Петрі стають суттєво продуктивнішими. Як показано *Хопкрофтом і Ульманом*, "розширені мережі Петрі" (навіть без додатково введених нижче ваг дуг) за своєю ефективністю рівні машині Тьюрінга, тобто вони можуть застосовуватись як загальна модель обчислюваності.

Розширення:

Багаторазове маркування.



відповідає



Кожен вузол може мати довільну кількість маркувань (при зображенні мережі Петрі допускається маркування коротко позначати числом). Правила активізації та перемикань змінюються відповідно:

- перехід активізований тільки тоді, коли число, що відповідає кількості маркувань кожного його вхідного вузла, є більшим чи дорівнює одиниці;
- якщо активізований перехід перемикається, то числа-маркування усіх вхідних вузлів цього переходу зменшуються, а усіх вихідних вузлів - збільшуються на одиницю.

Тобто кількість маркувань одного вузла може бути як завгодно великою, але відповідне число не може бути меншим від нуля.

Дуги-заперечення

Дуги-заперечення в мережах Петрі зображаються кружечком на кінці замість стрілки (рис.3.9) і не мають дугової ваги; дуги, що були визначені раніше, розглядаються як позитивні.

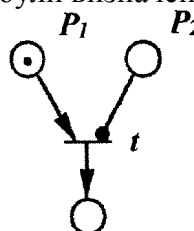


Рис. 3.9. Дуга-заперечення

Вони завжди направлені тільки від деякого вузла до деякого переходу, зворотного напрямку не дозволяється. Введення дуг-заперечень зумовлює оновлені пристосування правил активізації та перемикачів:

- перехід активізований тільки тоді, коли кількість маркувань кожного вхідного вузла з позитивною дугою більша або дорівнює одиниці і коли кількість маркувань кожного вхідного вузла з дугою-запереченням дорівнює нулю (на рис.3.9 перехід t активізований, якщо P_1 маркований і P_2 не маркований);

- якщо активізований перехід перемикається, то кількість маркувань усіх вхідних вузлів з позитивними дугами цього переходу зменшується на одиницю, в той час як кількість маркувань вхідних вузлів з дугами-запереченнями залишається незмінною. Числа, що відповідають кількості маркувань усіх вихідних вузлів, як і раніше, збільшуються на одиницю.

Вага дуг

Кожна дуга, що не є дугою-запереченням, може мати постійну цілочислову вагу, що більша або дорівнює одиниці (число, що використовується за замовчанням, рис. 3.10). Для активізації і перемикачів переходу діє правило:

- перехід активізований тільки тоді, коли кількість маркувань кожного його вхідного вузла більша або дорівнює відповідній вазі дуги, а для дуги-заперечення дорівнюють нулю;

- при перемикачів переходу кількість маркувань кожного вхідного вузла зменшується на вагу відповідної вхідної дуги (вона залишається незмінною для дуг-заперечень); кількість маркувань кожного вихідного вузла збільшується на вагу відповідної вихідної дуги.



Рис.3.10. Вага дуг

4. Приклади реалізації мереж Петрі

В принципі кожен програму, написану на будь-якій мові програмування, можна представити у формі розширеної мережі Петрі. При цьому треба мати на увазі, що "пам'ять маркувань" деякого вузла може містити тільки додатне число або нуль. Від'ємні числа можуть, наприклад, представлятися додатковим вузлом, що визначає знаки чисел. Кожна наведена тут мережа має спеціальний "стартовий" вузол, з якого починається обчислення, і вузол "готовності", який маркується тоді, коли обчислення повністю завершено і з'явився результат. Спеціальні старт - і готовність - вузли особливо важливі тоді, коли треба скласти розширену мережу Петрі із наявних елементів.

Суматор

Суматор (рис. 3.11) повинен порахувати маркування від пункту Y до маркувань в Z , тобто створити суму $Z+Y$. Стартовий вузол на початку операції зайнятий маркуванням "1". Після перемикачів переходу s ця марка зникне з місця старту, а марка у вузлі готовності з'явиться тільки тоді, коли закінчиться процес додавання і сума буде у вузлі Z .

Фініш

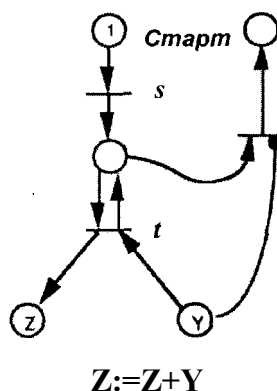


Рис. 3.11. Мережа Петрі як суматор

Після перемикання стартового переходу s маркується середній вузол мережі Петрі. Крок за кроком, перемикаючи перехід t , одне маркування береться з вузла Y (вхід) і одночасно вводиться у вузол Z (вихід). Середній вузол залишається при цьому маркованим, бо він містить у собі як вихідну, так і вхідну дуги ($1-1+1=1$). Однак цей вузол не є зайвим, бо він дбає про те, щоб операція додавання відбувалась як результуюча після розблокування стартового вузла. В кінці операції, коли $Y=0$, сума з'являється у вузлі Z і перехід t більше не активізується. Для завершення операції призначений перехід u , що тепер активізований; якщо він перемикається, то марка із середнього вузла виводиться і генерується марка у вузлі готовності, що сигналізує про закінчення обчислення.

Необхідно враховувати, що початкове числове значення Y знищується (у нашому випадку стає нулем) і не може використовуватися для можливих подальших операцій. Якщо змінна Y потрібна для інших операцій, що відбуваються після отримання суми, то мережу Петрі треба будувати так, щоб в ній відновлювалося значення величини Y .

Пристрій для віднімання

Простий пристрій для віднімання (рис.3.12 зліва) можна побудувати із суматора (рис.3.11), в якому змінено напрямки дуги від Z (штрихова стрілка). На кожному кроці тепер маркування не йде на місце Z , а навпаки, виводиться з Z . Ця методика функціонує тільки за тих умов, що $Z \geq Y$; в інших випадках віднімання зупиняється без маркування кінця операції у вузлі готовності. Це, звичайно, не має сенсу, тому на рис.3.12 (праворуч) показано розширений пристрій, який обчислює симетричну різницю.

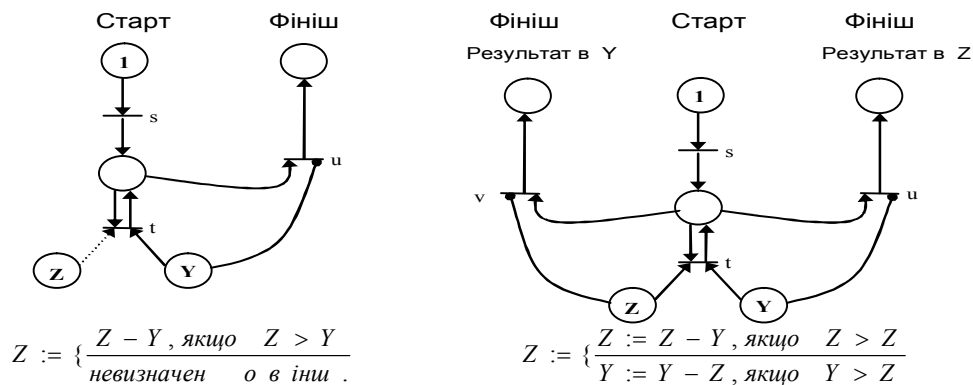


Рис.3.12. Мережа Петрі для операції віднімання

При симетричному відніманні різниця $Z-Y$ заноситься в Z , якщо $Z > Y$, а різниця $Y-Z$ заноситься в Y , якщо $Y > Z$. У зв'язку з цим мережа Петрі доповнюється симетрично переходом v .

Помножувач

Помножувач (рис. 3.13) має складнішу будову, але в переходах s , t і v можна впізнати основні елементи суматора. Це змінює одночасно принцип роботи помножувача: під час кожного проходу X зменшується на одиницю і Y покроково додається з Z . У вузлі "пам'ять" копіюється початкова величина Y , і в кінці операції через перехід u на Y робиться нова копія, щоб Y можна було використати в наступних розрахунках.

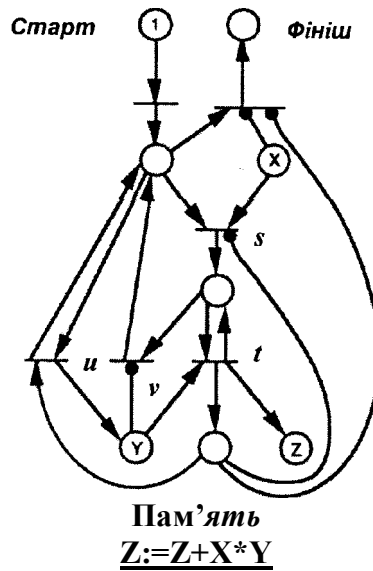


Рис. 3.13. Мережа Петрі для операції множення

Новий обчислювальний цикл починається, якщо "пам'ять" дорівнює нулю; потім перехід знову активізується через дугу заперечення. Цикли, в яких величина Y додається з Z , виконуються доти, поки X стане дорівнювати нулю, тобто добуток $X * Y$ буде додано з Z .

На рис.3.14 показано можливість розмноження значень змінної величини. На рис.3.15, 3.16 наведено приклади складних мереж Петрі, як скомпоновані з простіших модулів за методом чорного ящика.

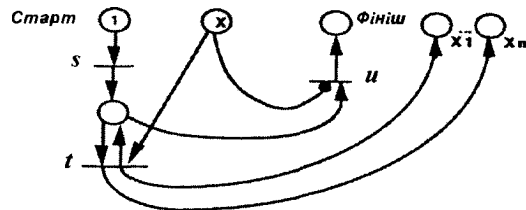


Рис. 3.14. Розмноження значень змінної X

Послідовні обчислення

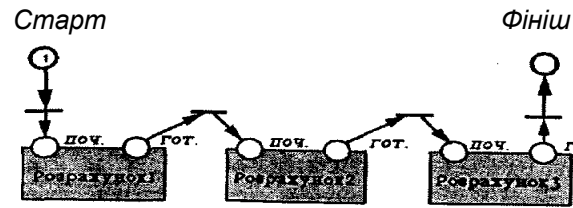


Рис. 3.15. Послідовне використання мережі Петрі

Паралельні обчислення

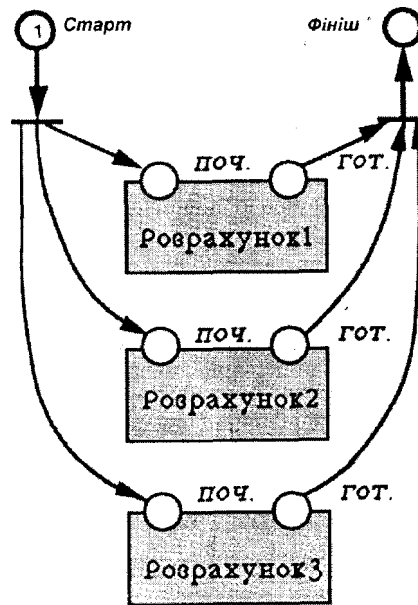


Рис. 3.16. Паралельне використання мережі Петрі

Розширена мережа Петрі, оптимально, тобто з мінімальною кількістю вузлів та переходів реалізовує наступні обчислення.

Задано цілі додатні числа $u_1, u_2, v_1, v_2, w_1, w_2$. Знайти :

$$\left\lfloor \frac{u + v + w}{u^2} \right\rfloor + v^2 u - \left\lfloor \frac{5w^2 v}{13u} \right\rfloor,$$

якщо $u = u_1 + iu_2$; $v = v_1 + iv_2$, $w = w_1 + iw_2$ – комплексні числа.

[a/b]– ціла частина від ділення.

Нижче наведено малюнок мережі Петрі в нерозгорнутому варіанті.

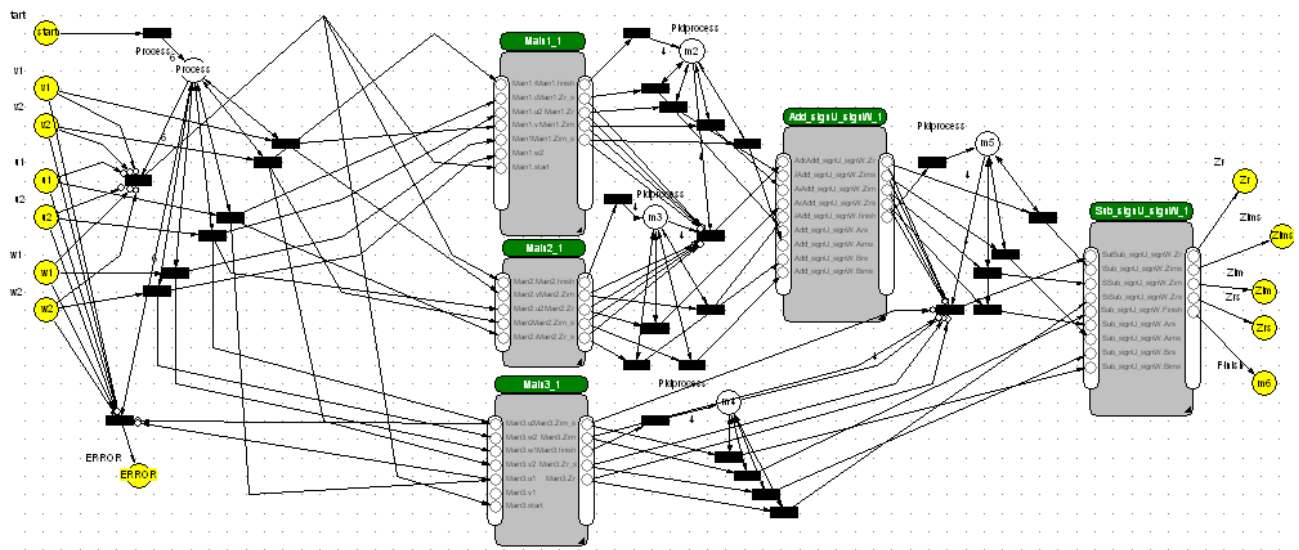


Рис.1 Мережа Петрі яка виконує поставлене завдання

В даній мережі є:

Всього 390 елементів враховуючи внутрішню структуру блоків.

-154 зв'язки

-31 перехід

-5 допоміжних вершин

-13 глобальних вершин

-5 функціональних блоків.

Перелік блоків що входять в склад схеми:

Main1, Main2, Main3, Add_signU_signW, Sub_signU_signW.

Розглянемо детально кожен з них з розкриттям їхньої функціональної бази:

Блок Main1:

Main1=(u+w+v)/u*u

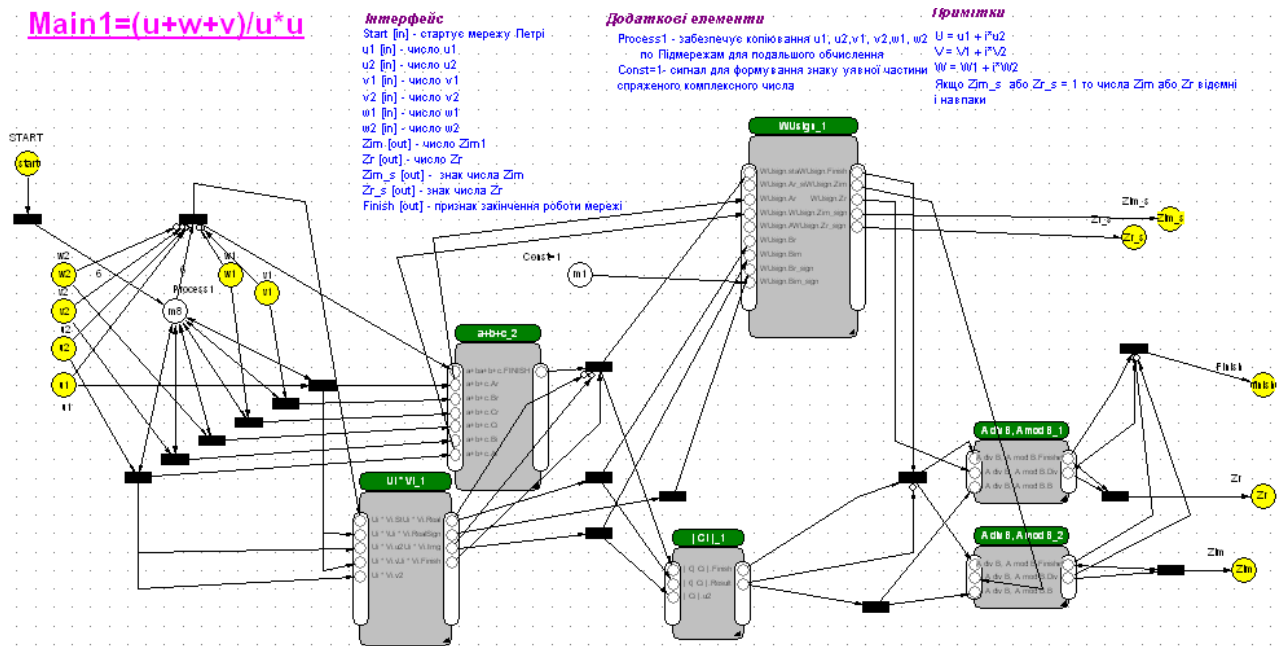


Рис.2 Блок виконання Main1=(u+w+v)/u*u

Перелік блоків: Abs(U), WUsign, a+b+c, Ui*Vi, A div B, A mod B.
 Блок Main2.

Main2=V*V*U

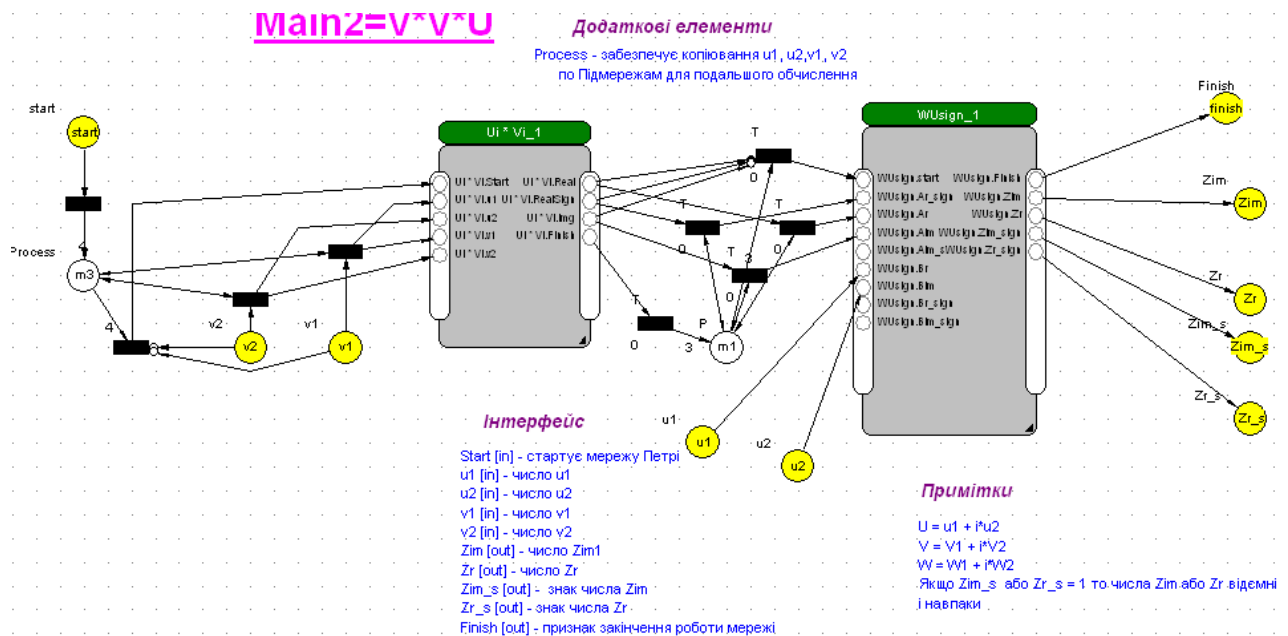


Рис.3 Блок виконання main2=v*v*u

З блоків Ui*Vi та WUsign.
 Блок Main3.

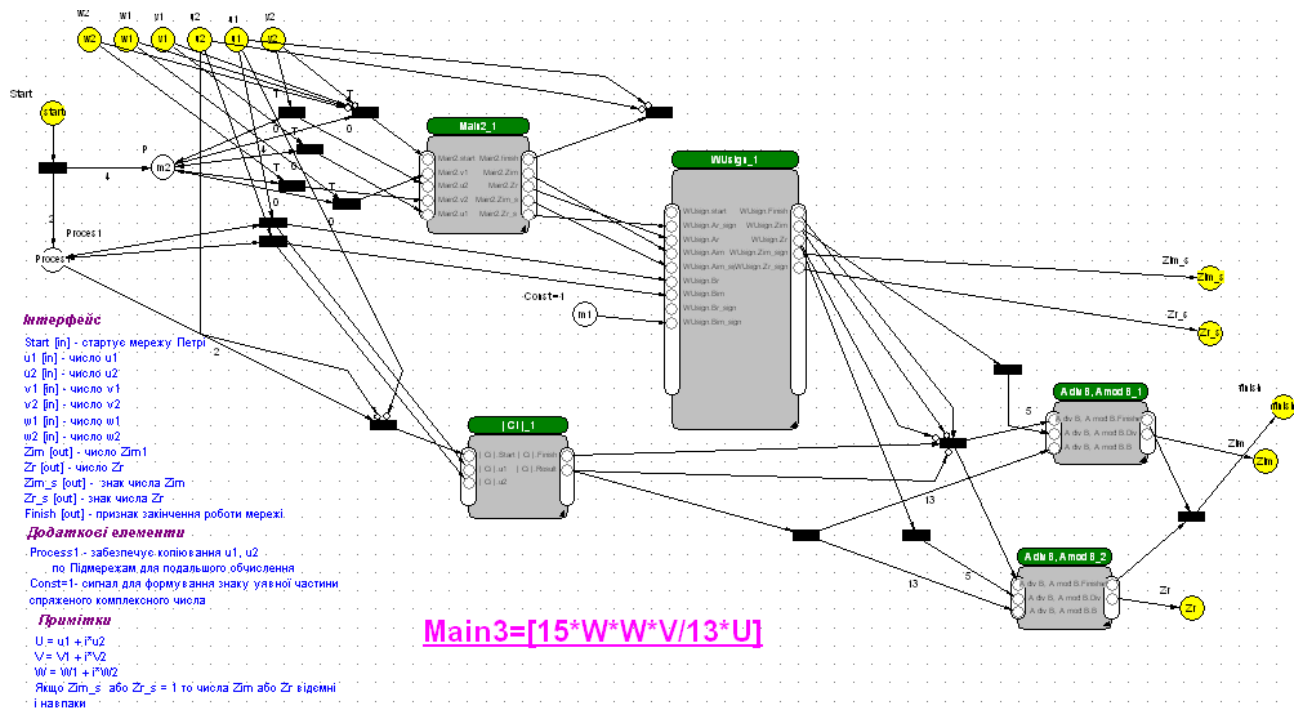


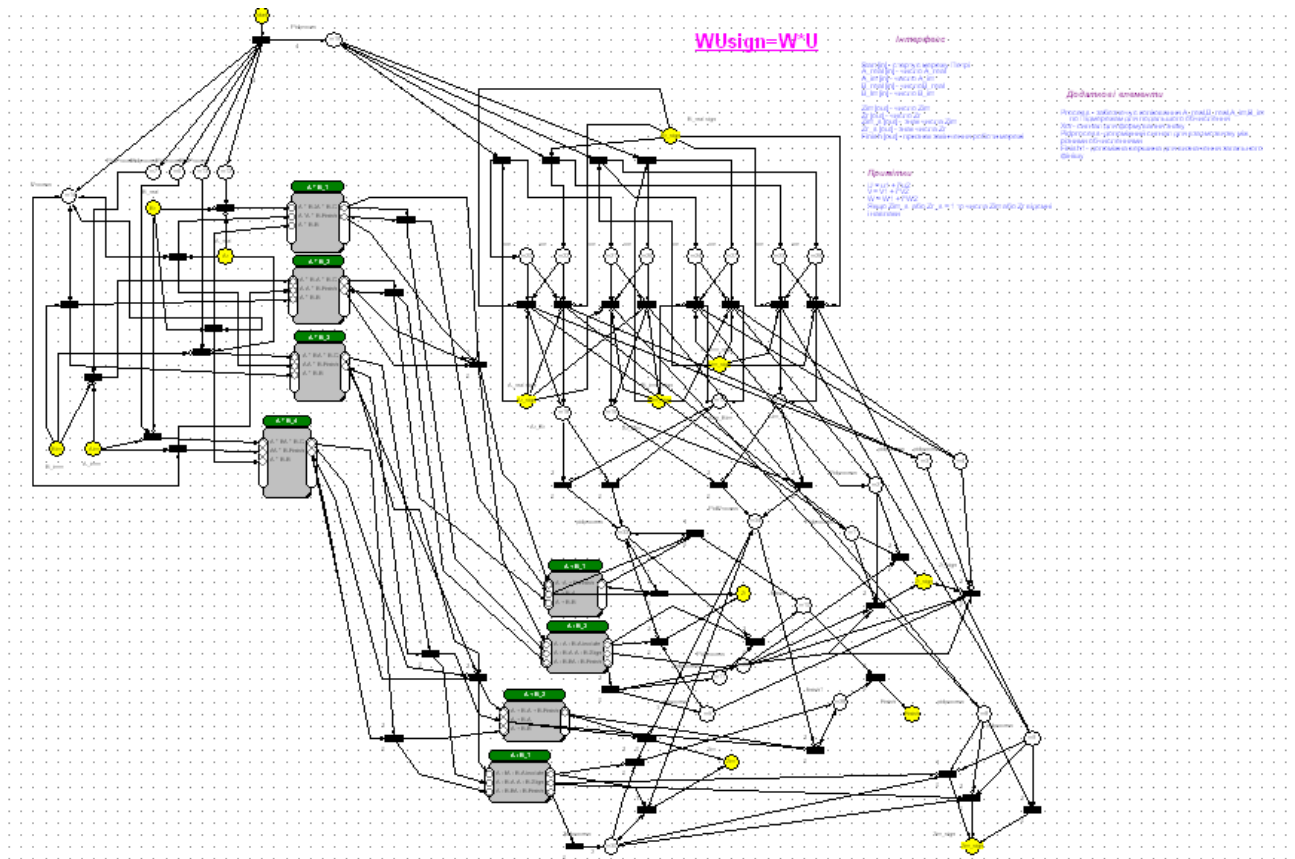
Рис.4 Блок виконання $main3=[15*w*w*v/13*u]$

Складається з блоків Main2, WUsign, ABS(V), $A \div B_A \bmod B$.

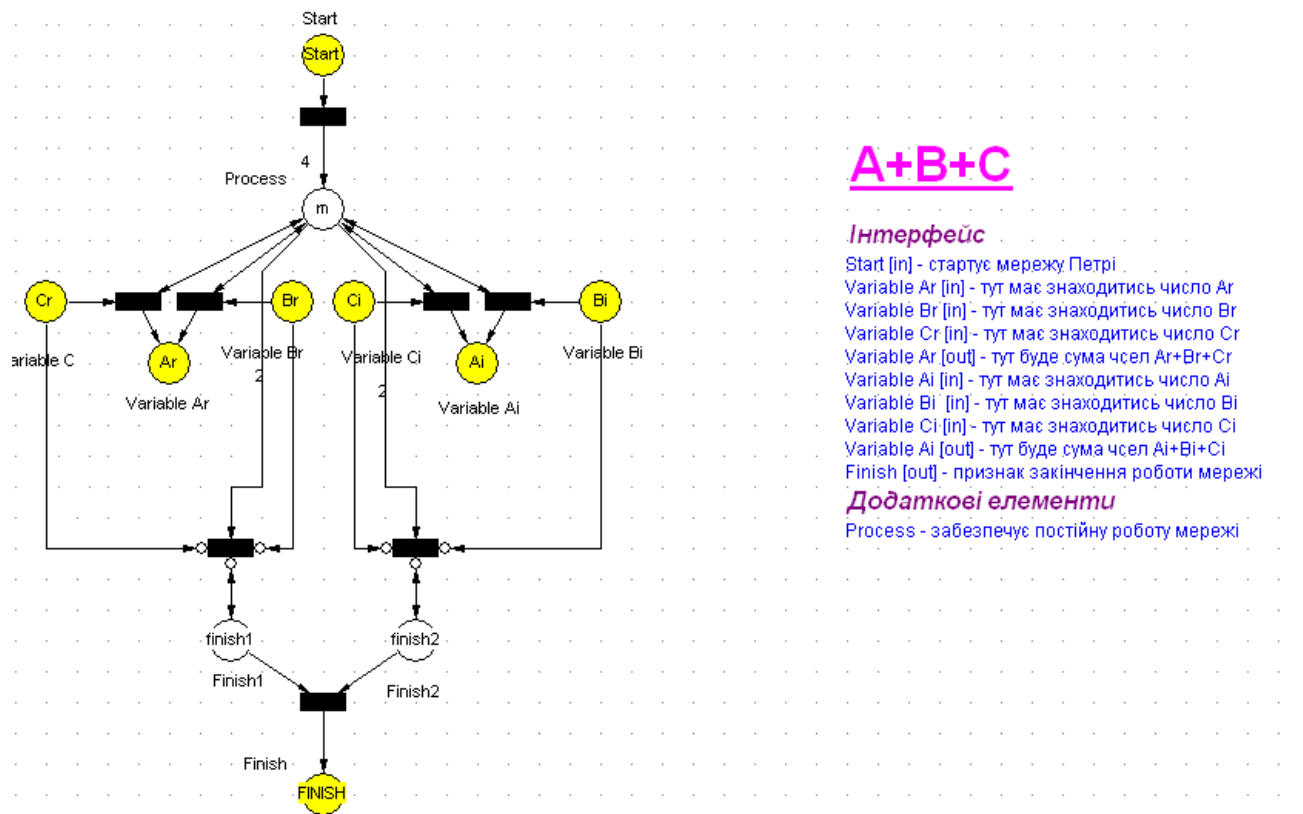
Розглянемо будову кожного з складових блоків.

Блок WUsign- Цей блок виконує знакове множення двох комплексних чисел: спершу обчислює відповідні добутки а потім на основі знаку виконує додавання чи віднімання.

Цей блок містить у складі блоки звичайного множення двох чисел , додавання і віднімання – наводити їхню структуру немає потреби, оскільки це є елементарні блоки обчислень що входять до мереж Петрі.

Рис.5 Блок виконання $WU_{sign}=w*u$

Наступний блок $a+b+c$ – додає 3 комплексні числа.

Рис.6 Блок виконання $a+b+c$

Блок $U_i \cdot V_i$ - виконує звичайне множення комплексних чисел при цьому формується тільки знак дійсної частини результату. В складі блоку є елементарні додавачі, віднімачі, помножувачі.

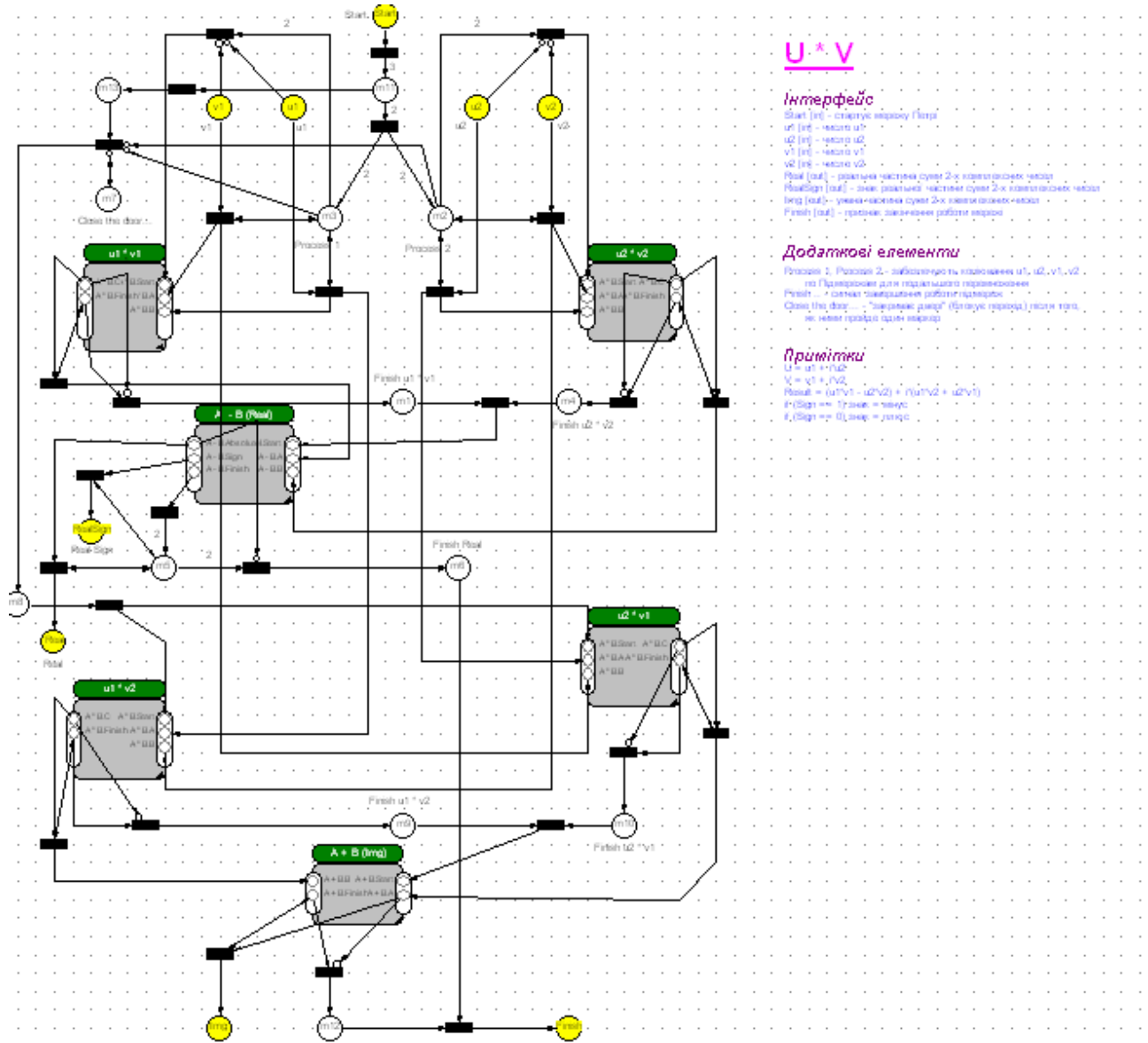
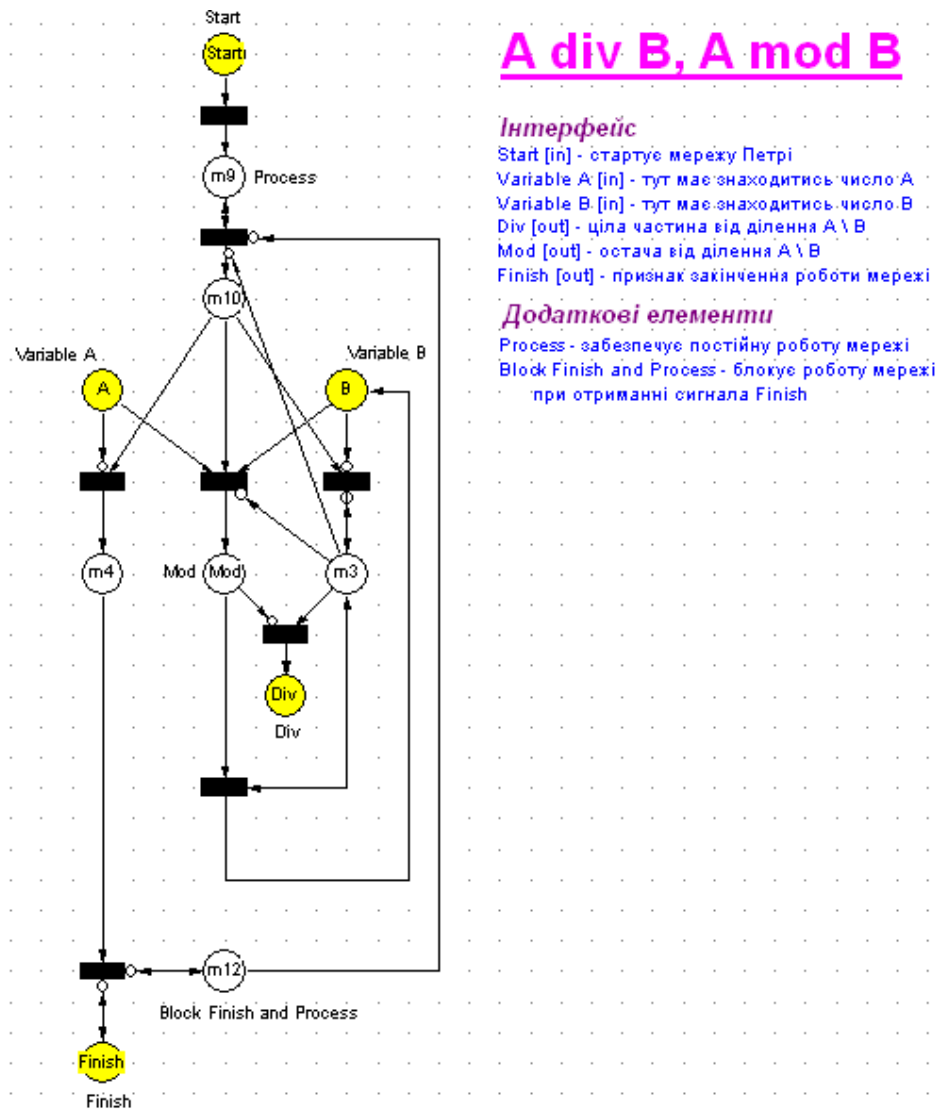


Рис.7 Блок виконання $u \cdot v$

Блок ABS(U)- шукає модуль комплексного числа. Також має суматор і простий помножувач.

Блок $A \div B, A \bmod B$ - шукає цілу частину від ділення A на B .

Рис.9 Блок виконання $a \div b, a \bmod b$.

Блок Add_signU_signW- цей блок виконує знакове додавання двох комплексних чисел: відповідні числа семуються а потів аналізуються їхні знаки і на основі аналізу відбувається вибірка правильного результату та встановлення його знаку. Тут використано такі вершини process- для завантаження даних до підмереж. Help1 – для запобігання передчасного хибного обрахунку знаку результату. Сор- для копіювання значень знаку оскільки на потрідно перевіряти знак але з одною вершиною ми це не зробим. Xor- умова пересилання результату. Fin- результат завершення роботи помереж.

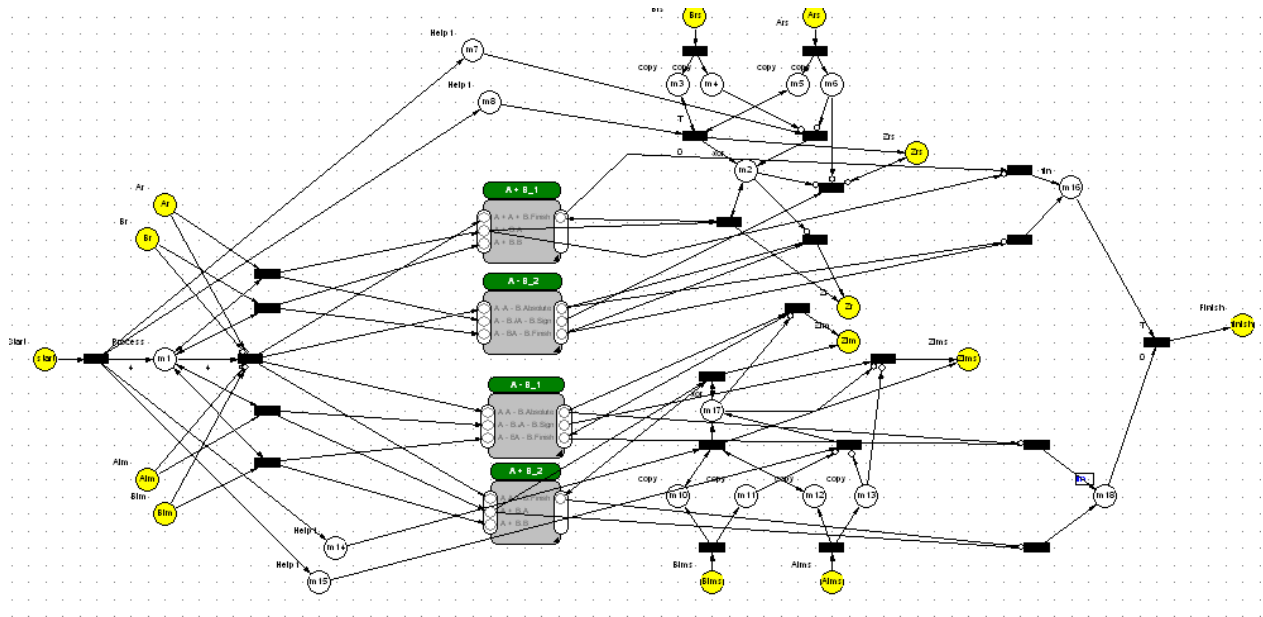


Рис.10 Блок виконання U+W

Блок Sub_signU_signW-знакове віднімання комплексних чисел. Назви проміжних вершин виконують те саме призначення що і у попередньому блоці. Складається тож з простих суматорів та помножувачів.

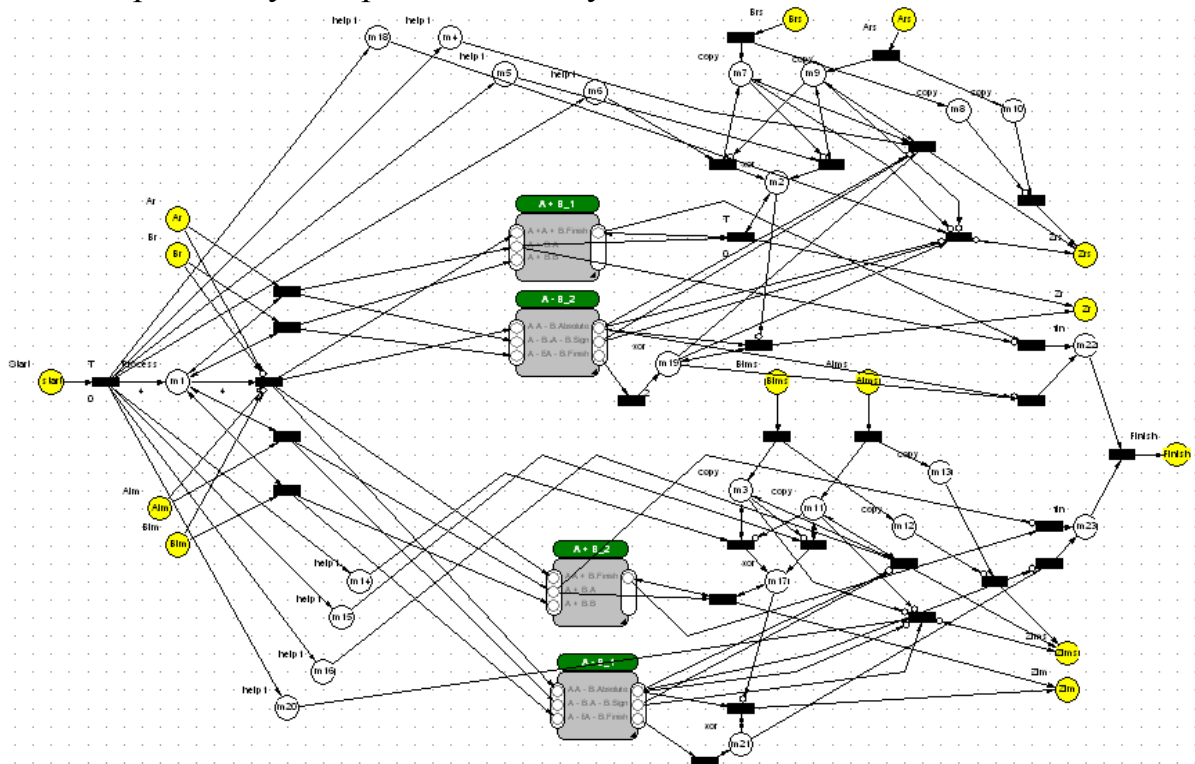


Рис.11 Блок виконання U-W

Зауваження

Вершини вхідних даних містять цілі невід'ємні числа і можуть з'явитися лише один раз. У мережі мають бути помічені вершини "старт", "фініш" та "помилка" (при потребі). Знак результату має встановлюватися за допомогою додаткової вершини. При необхідності, результат може знаходитися у декількох вершинах (наприклад ціла і дробові частини).