Лекція 31.

Моделювання виробничих процесів і систем

Проектування технологічних процесів, перевірка властивостей проекту на моделі, прогнозування ходу виробництва, настроювання моделі для виробничих процесів, керування виробництвом мають ряд особливостей. Ціль даної лекції - обговорити виникаючі при моделюванні виробничих процесів особливості.

Допустимо, що деякий виробничий процес нелінійний. Це значить, що між входом X (те, чим управляємо) і виходом Y (те, що спостерігаємо) є нелінійна залежність. Як змоделювати таку нелінійність?

Приклад 1. Логічні функції. Для початку розглянемо просту нелінійність релейного типу й допустимо, що функція $Y(X)$ оборотна (див. мал. 31.1). Такі залежності називають *статичними* (значення на вході X однозначно, не залежно від передісторії процесу визначає значення на виході Y). Багато виробничих процесів можуть бути описані таким способом.

Наприклад, нехай залежність складається із трьох ділянок (див. мал. 31.1). Часто характерною рисою таких залежностей є наявність у них ділянки, що блокує вихід, лінійної ділянки й ділянки насичення. Перша ділянка вказує на те, що, не подаючи сировини ($X = 0$) на вхід виробничого процесу, не можна сподіватися на який-небудь результат ($Y = 0$). Друга ділянка вказує на той факт, що, збільшуючи кількість сировини на вході, ми забезпечуємо збільшення готової продукції на виході. І, нарешті, зрозуміло, що якщо сировини буде дуже багато ($Y >> 0$), те які-небудь обмеження виробничого процесу (наприклад, продуктивність устаткування, кваліфікація персоналу, фінансові, енергетичні ресурси) однаково не дозволять переробити все це сировина й випустити відповідну кількість продукції. У складних системах така ділянка насичення присутнє обов'язково.

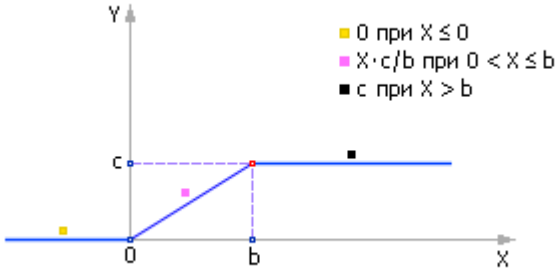


Рис. 31.1. Залежність релейного типу

Зверніть увагу на те, що кожен з ділянок можна легко описати окремо. Представлена на мал. 31.1 залежність математично описується трьома функціями. Три рядки в записі означають, що в кожен окремий момент (тобто, при певнім значенні *X*) з опису використається один з рядків (перша, друга або третя). Тобто мається на увазі (але явно не записується), що рядки з'єднані знайомий **АБО**.

Коли мають справа з обчислювальною технікою й мовами програмування, де умовчувати — небезпечно, цю конструкцію записують у рядок:

$$Y := (0 \text{ при } X \le 0) \text{ АБО } (X \cdot c/b \text{ при } 0 < X \le b) \text{ АБО } (c \text{ при } X > b).$$

Якщо процеси аналогові, тобто якщо ми маємо справу з будь-якими (раціональними) значеннями X, роль операції **АБО** грає знак «+»:

$$Y := (0 \text{ при } X \le 0) + (X \cdot c/b \text{ при } 0 < X \le b) + (c \text{ при } X > b).$$

Справді: якщо умови виключають один одного, то в будь-який момент часу може реалізуватися тільки одне із трьох умов. Воно й дасть необхідну відповідь у суму. Від інших доданків у суму прийде нуль. Додавання шуканого значення з нулями сформує правильну відповідь у цілому.

Розглянемо тепер окремо одну з конструкцій, наприклад, *c* при *X* > *b*. Її використання означає наступну фразу (яку не пишуть і не вимовляють повністю винятково з метою економії місця й часу): «якщо *X* більше *b*, те *Y* привласнити *c*». Скористаємося одиничною функцією Хевісайда (див. мал. 31.2) для перекладу даної фрази на формальну мову. Нагадаємо, що значення одиничної функції дорівнює 0, якщо її аргумент менше або дорівнює 0, і дорівнює 1, якщо її аргумент більше 0.

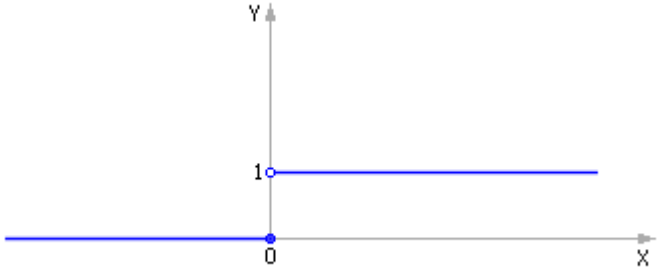


Рис. 31.2. Одинична функція Хевісайда Y = ed(X)

У результаті одержимо: $Y := c \cdot \mathrm{ed}(X - b)$. Перевіримо отриману формулу. Дійсно:

- якщо $X < b$, то аргумент $(X - b)$ одиничної функції негативний, отже, сама функція дорівнює 0, тому $Y = 0$;
- якщо $X = b$, то аргумент $(X - b)$ одиничної функції дорівнює нулю, отже, сама функція дорівнює 0, тому $Y = 0$;
- якщо $X > b$, то аргумент $(X - b)$ одиничної функції позитивний, отже, сама функція дорівнює 1, тому $Y = c$.

Помітимо, що в нашій формулі $Y := c \cdot \mathrm{ed}(X - b)$ знак множення відіграє роль логічного **И**, оскільки для одержання остаточного результату нам, по-перше, потрібне значення «c» **И**, по-друге, щоб істиною (тобто, одиницею) було значення вираження $\mathrm{ed}(X - b)$.

Як бачите, словесна логічна конструкція

якщо **Умова**, *то* **Шукана_величина** := **Факт**

відповідає типовій математичній конструкції

Шукана_величина := **Факт** · ed(**Умова**).

ВАЖЛИВО! Отже, нерівність перетворюється в одиничну функцію (у математику одинична функція еквівалентна поняттю ПРЕДИКАТ), логічне **АБО** заміняється на знак додавання, логічне **И** заміняється на знак множення. Знак «:=» виражає причинно-наслідковий зв'язок, погоджуючи причину й наслідок у явному виді. У випадку неявного завдання функції роль цього знака буде грати знак уравнивания «=».

Тепер переведемо другу частину вираження, $Y := X \cdot c/b$ при $0 < X \leq b$, на формальну мову: $Y := X \cdot c/b \cdot \text{ed}(X) \cdot \text{ed}(b - X)$. Помітимо, що правильний результат вийде, якщо буде дотримано вже три умови одночасно: $(X \cdot c/b)$ И $(X > 0)$ И $(X \leq b)$. Так само надійдемо й із частиною, що залишилася: Y := 0 при 0 ? X. Формальний переклад має вигляд: $Y := 0 \cdot \text{ed}(0 - X)$.

Збираючи всі три частини разом, маємо:

$$Y := 0 \cdot \text{ed}(0 - X) + X \cdot c/b \cdot \text{ed}(X) \cdot \text{ed}(b - X) + c \cdot \text{ed}(X - b).$$

Зверніть увагу, що перший доданок взагалі ж можна не писати, тому що воно завжди дорівнює 0. Залишається:

$$Y := X \cdot c/b \cdot \text{ed}(X) \cdot \text{ed}(b - X) + c \cdot \text{ed}(X - b).$$

Ще одне зауваження. Якщо протестувати функцію в крапці $X = b$, то виявиться, що й перший доданок, і друге одночасно рівні 0, а вся функція буде мати вигляд, показаний на мал. 31.3. Така ситуація називається прощелкой. (А є чи прощелка між $0 \cdot \text{ed}(0 - X)$ і $X \cdot c/b \cdot \text{ed}(X) \cdot \text{ed}(b - X)$? Якщо «так», то при якому X? Перевірте себе.)

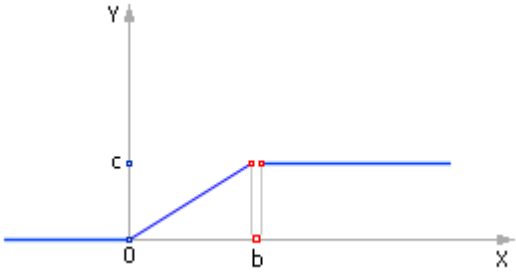


Рис. 31.3. Вид логічної функції з помилкою типу прощелки в описі

Щоб уникнути прощелок, треба писати формули акуратніше:

$$Y := X \cdot c/b \cdot \mathrm{ed}(X) \cdot \mathrm{not}(\mathrm{ed}(X - b)) + c \cdot \mathrm{ed}(X - b).$$

Дійсно, у цьому випадку або $\mathrm{not}(\mathrm{ed}(X - b))$ (з першого доданка), або $\mathrm{ed}(X - b)$ (із другого доданка) виявиться обов'язково рівним 1, і прощелка зникне. Помітимо, що можна не вводити додаткову операцію $\mathrm{not}(q)$, а використати її аналог: $1 - \mathrm{ed}(q)$.

Отже, узагальнимо: якщо в записі залежності є n відрізків і при цьому X на i-ом відрізку задовольняє умові: $a_i < X \le a_{i+1}$, те маємо:

$$Y := \sum_{i=1}^{n} f_i(X) \cdot \mathrm{ed}(X - a_i) \cdot \mathrm{not}(\mathrm{ed}(X - a_{i+1}))$$

Зверніть увагу. Функція вийшла однозначної, тобто тому самому X завжди відповідає те саме Y. Зручність таких описів у тім, що Y обчислюється в будь-який момент для будь-якого X, досить підставити потрібне будь-яке значення X у формулу.

Вид запису залежить від системи, який прийде виконувати такий запис. Якщо система не розуміє запис, то вона не зможе її виконати, і такий запис марний. Мова такий мертва й нічого не означає, тому що не служить засобом комунікації, передачі інформації від одного агента іншому. (Забігаючи вперед, відзначимо, що в мови є й ще інші функції, наприклад, здатність маніпуляції об'єктами, які він описує, функція вирахування.) Приміром, якщо виконавцем є алгоритмічна машина, то конструкція

$$Y := X \cdot c/b \cdot \mathrm{ed}(X) \cdot \mathrm{not}(\mathrm{ed}(X - b)) + c \cdot \mathrm{ed}(X - b).$$

буде мати вигляд, показаний на мал. 31.4.

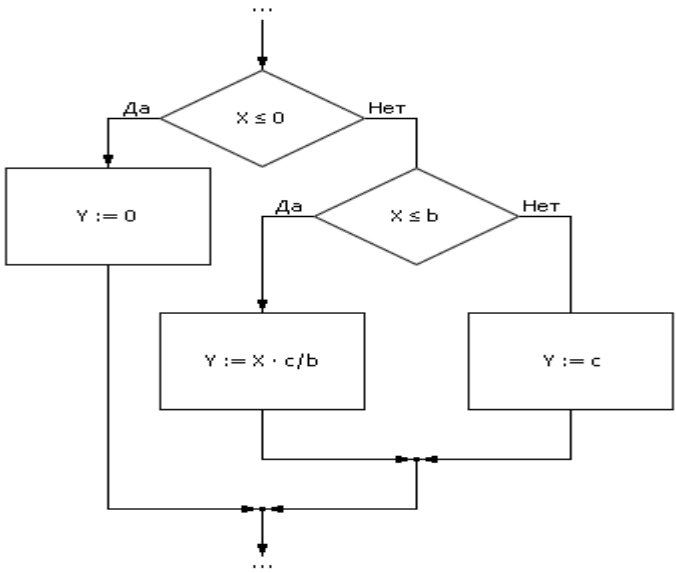


Рис. 31.4. Алгоритмічна реалізація виробничих моделей у вигляді логічних функцій з використанням умовних конструкцій

Як бачите, є відповідність у мовах опису. Умовна конструкція в алгоритмах відповідає «багатоповерховій» формулі в математику або конструкції

$$\textbf{Искомая\_величина} := \sum_{i=1}^{n} \left(\textbf{Факт}_i \cdot \prod_{j=1}^{k} \text{ed}(\textbf{Условие}_j) \right)$$

у формалізованій мові моделювання. Або, з використанням логічних функцій **И**, **АБО**, **НЕ**:

$$\textbf{Искомая\_величина} := \underset{i=1}{\overset{n}{\textbf{ИЛИ}}} \left(\textbf{Факт}_i \cdot \underset{j=1}{\overset{k}{\textbf{И}}} \textit{если}(\textbf{Условие}_j) \right)$$

Приклад 2. Генератор пилкоподібного сигналу (ГПС). Сконструюємо періодичну пилкоподібну функцію (див. мал. 31.5) двома різними способами.

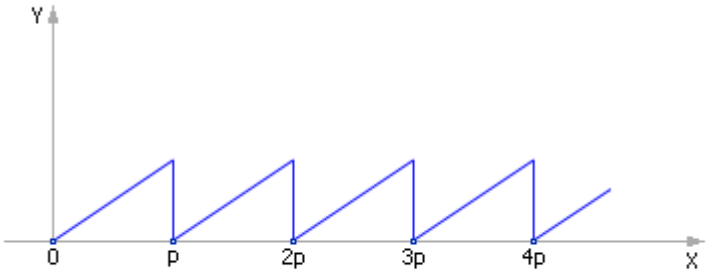


Рис. 31.5. Вид залежності Y(X) для ГПС

Варіант 2.1. ГПС на основі математичної функції mod. Відома математична функція mod(X, a) повертає нам остача від ділення X на a, а остача, як відомо, росте лінійно з ростом X, а потім стає рівним 0 (коли X й a стають кратними), повторюючи цю закономірність періодично. У середовищі Stratum-2000 це запишеться так: $Y := X\%a$. Період й амплітуда «пилки» регулюються значеннями X й a.

Варіант 2.2. ГПС на основі рівняння з пам'яттю. $Y := (Y + d) \cdot \mathrm{ed}(p - Y)$, де d — крок зміни, p — період. Допустимо, що спочатку Y досить мало, тому $p > Y$. Якщо $p - Y > 0$, то для позитивного аргументу одинична функція повертає 1, і вираження має вигляд: $Y := (Y + d) \cdot 1$, тобто Y збільшується на кожному такті на величину d, а значить — лінійно росте.

Рано або пізно Y стає дорівнює або більше p, і умова стає $p - Y \leq 0$. Для негативного й нульового аргументу одинична функція повертає 0, а значить вираження в цей момент приймає вид: $Y := (Y + d) \cdot 0$, тобто Y на даному кроці скидається в нуль, і ситуація повторюється: знову Y стає малим і починає рости, поки не досягне значення p. При подачі на вертикальну координату осцилографа обчисленого значення Y, а на горизонтальну його координату значення X, сформованого як $X := X + 1$, можна побачити «пилку» з періодом p (див. мал. 31.6).

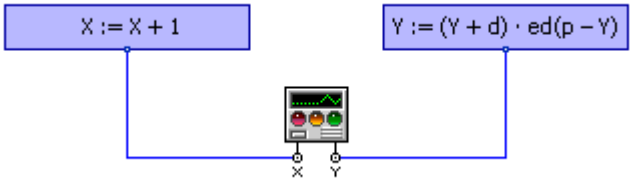


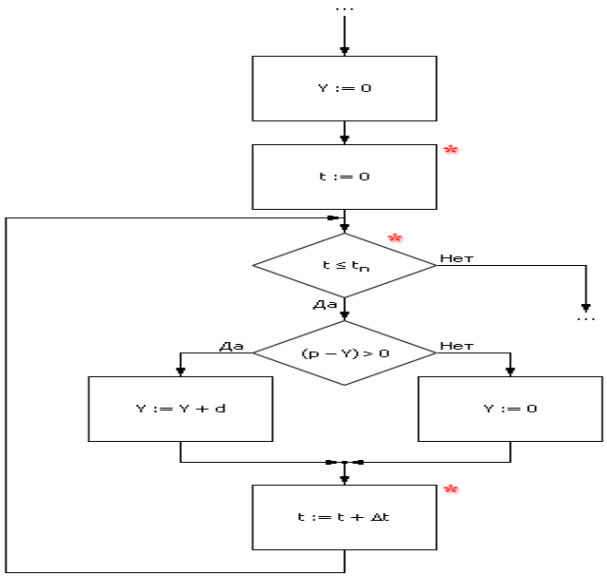
Рис. 31.6. Схема проекту в середовищі Stratum-2000, що
реалізує генератор пилкоподібного сигналу
на основі рівняння з пам'яттю

Зверніть увагу. По-перше, як бачите, у вираженні $Y := (Y + d) \cdot \mathrm{ed}(p - Y)$, щоб обчислити чергове значення Y, треба знати попереднє. Тому говорять, що це **рівняння зі зворотним зв'язком** або **рівняння з пам'яттю**: щоб обчислити N-i значення, потрібно обчислювати всі попередні значення функції на ряді кроків. По-друге, функція вийшла неоднозначна. Тобто при різних початкових даних графік функції буде різний. Тому самому X можуть відповідати різні Y. Це відповідає сімейству графіків, породжуваних диференціальним рівнянням. Даний запис **НЕЯВНА**, тому що вона не містить значення X безпосередньо. Тому не можна підставити довільне значення X і відразу довідатися, чому буде дорівнює Y при цьому значенні. Щоб визначити значення Y, необхідно пройти (перебрати) всі значення Y від 0 до шуканого.

Первый варіант запису, $Y := X\%a$, представляє нам **ЯВНУ** залежність Y від X. У будь-який момент можна підставити будь-яке значення X й одержати відповідне йому значення Y.

Кожен варіант запису має свої переваги й недоліки. Не завжди вдається одержати явний запис, тому що записати закон функціонування системи звичайно набагато простіше, ніж знайти його рішення. Але і явний варіант має той недолік, що це тільки одне з рішень, тоді як сам закон містить потенційно всю множину можливих інших рішень, багатозначний.

Якщо реалізовувати вираження $Y := (Y + d) \cdot \mathrm{ed}(p - Y)$ мовою алгоритмів, то знадобиться циклічна конструкція (див. мал. 31.7).



Примечание. В системах реального времени блоки, помеченные звёздочкой, не ставятся, алгоритм зацикливается.

Рис. 31.7. Алгоритмічна реалізація моделі пилкоподібного сигналу з використанням циклічної конструкції

Між системою з пам'яттю й циклічною конструкцією є глибокий зв'язок, а саме: система з пам'яттю — це *динамічна система*, що відповідає в математику *диференціальному рівнянню*, покроковий розрахунок якого на комп'ютері вимагає *циклічного алгоритму*. Якщо рішення диференціального рівняння можна знайти аналітично, тобто виразити шукану величину залежно від інших явно, те циклічної конструкції не знадобиться, досить алгоритму лінійної структури. Але це буде властиво не рівняння, а присвоєння, тобто рішення рівняння, точніше, одне з рішень рівняння.

Приклад 3. Генератор прямокутних імпульсів (ГПІ). У багатьох додатках потрібно імітувати періодичне відкривання й закривання на якийсь час якихось пристроїв, вентилів, каналів. Для цього зручно використати генератор імпульсів, який би на виході видавав значення «0» й «1», тобто генератор прямокутних імпульсів. За параметри генератора зручно прийняти частоту змін з 0 на 1 й обернено й шпаруватість - співвідношення часу втримання на виході «1» вчасно всього циклу.

Варіант 3.1. ГПІ на основі періодичної функції сінус. Можна використати кілька прийомів моделювання генератора прямокутних імпульсів, що мають свої достоїнства й недоліки. Розглянемо спочатку генератор, що використає властивості періодичної функції, наприклад, математичної функції сінус (див. мал. 31.8).

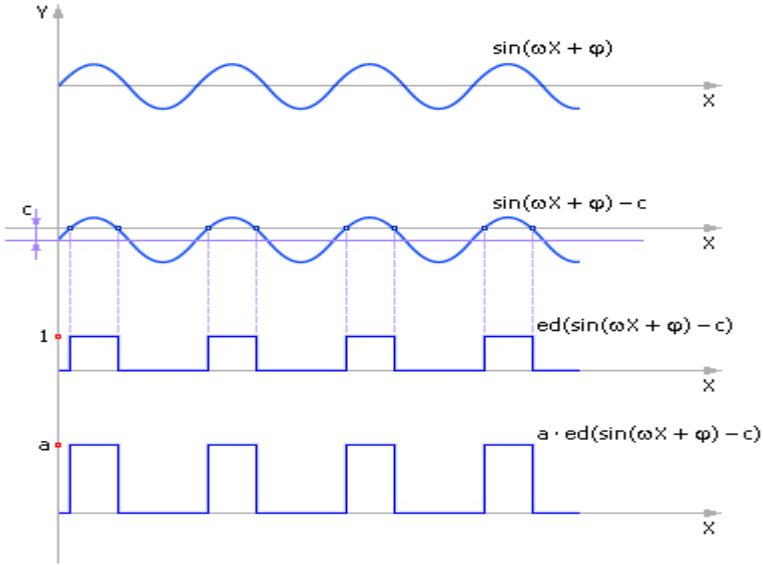


Рис. 31.8. Генерація послідовності прямокутних імпульсів засобами періодичної функції

$Y := a \cdot \mathrm{ed}(\sin(\omega \cdot X + \varphi) - c)$, де ω — частота генератора, що задає його період; φ — фаза генератора; a — амплітуда імпульсів; c — заданий параметр, шпаруватість. Шпаруватість указує на те, протягом якої частини періоду сінуса Y буде дорівнює a, і протягом якої частини періоду Y буде дорівнює 0. Поки $\sin(\omega \cdot X + \varphi) > c$, одинична функція повертає 1, $Y = a$; як тільки $\sin(\omega \cdot X + \varphi) \leq c$, одинична функція буде повертати 0, $Y = 0$.

Опять помітимо, що це ЯВНИЙ запис, що визначає достоїнства й недоліки цього варіанта.

Варіант 3.2. ГПІ на основі рівняння з пам'яттю. Інший варіант — використати для генерації прямокутних імпульсів раніше створений генератор пилкоподібного сигналу, який використав послідовне обчислення значень від крапки до крапки (рівняння з пам'яттю).

$Y := (Y + d) \cdot \mathrm{ed}(p - Y)$, $Z := a \cdot \mathrm{ed}(Y - c)$, де d — крок зміни; p — період; a — амплітуда імпульсів; c — заданий параметр, шпаруватість.

Оскільки Y генерує пилкоподібний сигнал, то досить порівнювати його значення із заданим значенням c. Якщо $Y - c \leq 0$, то $Z = 0$, якщо $Y - c > 0$, то $Z = a \cdot 1$. У результаті виходить періодичний одиничний сигнал. При подачі на осцилограф на вертикальну координату обчисленого значення Z, а на горизонтальну координату — значення X, сформованого як $X := X + 1$, можна побачити «забір» з періодом p (див. мал. 31.9).

Даний запис має НЕЯВНИЙ вигляд, що визначає достоїнства й недоліки цього варіанта.

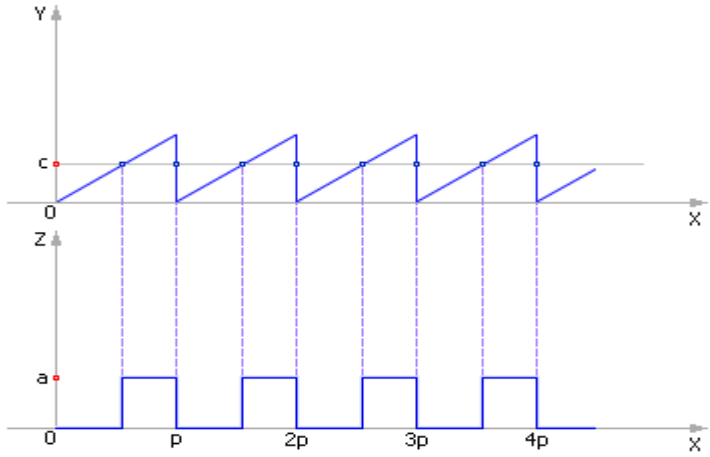


Рис. 31.9. Генерація послідовності прямокутних імпульсів
з використанням вираження з пам'яттю

Приклад 4. Комутатор. Часто використовуваним у виробничих процесах технологічним пристроєм є комутатор, метою якого є розподіл якого-небудь основного потоку (матеріального, енергетичного, інформаційного) на ряд приватних потоків за рахунок перемикання подачі з одного напрямку на інше (див. мал. 31.10). Це дозволяє ділити потоки, міняти їхній напрямок.

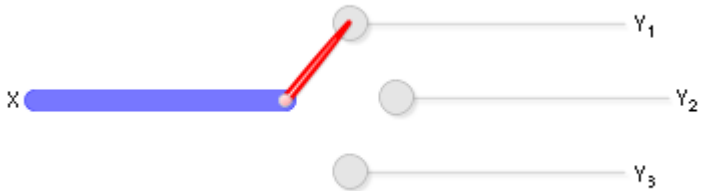


Рис. 31.10. Схема комутатора

Нехай є один вхід, позначений як X, і, наприклад, три виходи: Y_1, Y_2, Y_3. Потрібен сигнал X у перший момент часу транслювати на Y_1, у другий момент часу — передавати на Y_2, а в третій момент — на Y_3, після чого циклічно повторювати цей процес, починаючи знову з Y_1. (Природно, можливі й інші варіанти комутаторів.) Тимчасова діаграма розподілу основного потоку на подпотоки дискретним комутатором показана на мал. 31.11.

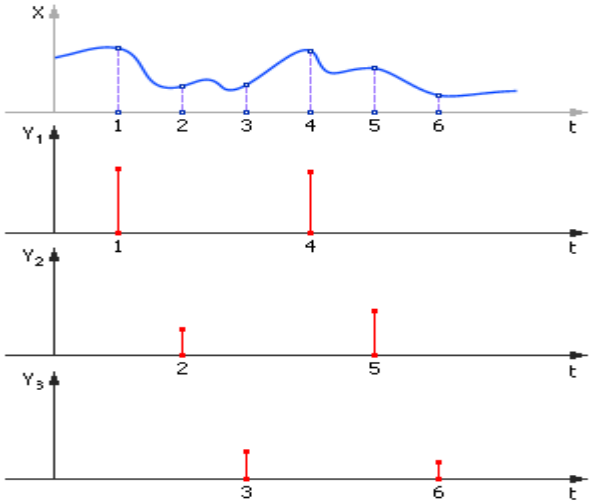


Рис. 31.11. Тимчасова діаграма роботи дискретного комутатора з одним входом і трьома виходами

Для цього нам знадобиться дельта-функція Дірака: $Y = \delta(X)$. Ця функція повертає 1, якщо її аргумент дорівнює 0, в інших випадках вона повертає нуль (див. мал. 31.12).

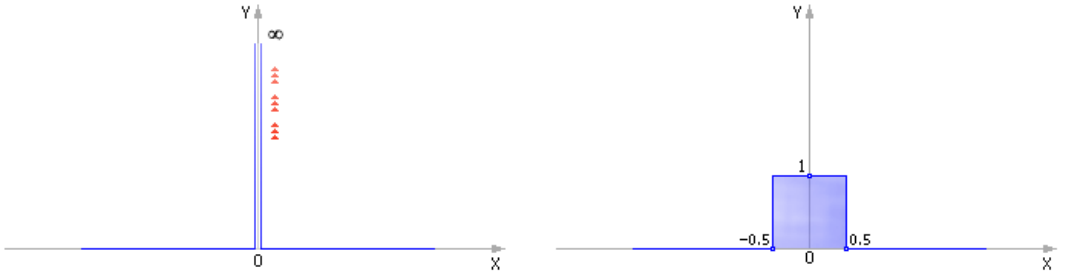


Рис. 31.12. Дельта-функція Дірака Y = δ(X) (ліворуч — теоретичний вид, праворуч — її наближений дискретний аналог)

За допомогою дельта-функції вираження, що імітують комутатор, будуть виглядати так (у наведеній нижче записі змінна i відіграє роль часу t):
$i := (i + 1) \cdot \mathrm{ed}(2 - i)$
$Y_1 := X \cdot \delta(i - 0)$
$Y_2 := X \cdot \delta(i - 1)$
$Y_3 := X \cdot \delta(i - 2)$.

Таким чином, i, завдяки першому вираженню, пробігає циклічно значення 0, 1, 2, 0, 1, 2, 0, 1, 2, … Якщо i виявилося дорівнює 0, то $\delta(i - 0) = 1$ й $Y_1 := X \cdot 1$; при цьому $\delta(i - 1) = 0$ й $Y_2 := X \cdot 0$ й $\delta(i - 2) = 0$ й $Y_3 := X \cdot 0$.

Якщо i виявилося дорівнює 1, то $\delta(i - 1) = 1$ й $Y_2 := X \cdot 1$, а $\delta(i - 0) = 0$ й $Y_1 := X \cdot 0$ й $\delta(i - 2) = 0$ й $Y_3 := X \cdot 0$.

Якщо i виявилося дорівнює 2, то $\delta(i - 2) = 1$ й $Y_3 := X \cdot 1$, а $\delta(i - 0) = 0$ й $Y_1 := X \cdot 0$ й $\delta(i - 1) = 0$ й $Y_2 := X \cdot 0$.

Для організації циклічного комутатора нам знадобилася «пилка», що ми побудували раніше.

Помітимо, що ще краще було б зробити так: окремо записати вираження для системи керування, де описуються інформаційні сигнали i, f_1, f_2, f_3:

$i := (i + 1) \cdot \mathrm{ed}(2 - i)$;

$f_1 := \delta(i - 0)$;

$f_2 := \delta(i - 1)$;

$f_3 := \delta(i - 2)$.

Прапори f_i приймають значення 0 й 1, як це й покладено логічним змінним. Модель інформаційної частини проекту показана на мал. 31.13.

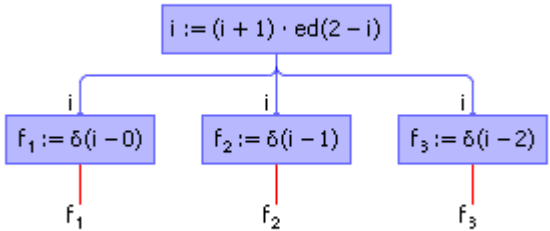


Рис. 31.13. Схема інформаційно-керуючих сигналів для комутатора

Далі окремо записати вираження для матеріальних потоків Y й X, які перекриваються сигналами керування f_i (див. мал. 31.14):

$Y_1 := X \cdot f_1$;

$Y_2 := X \cdot f_2$;

$Y_3 := X \cdot f_3$.

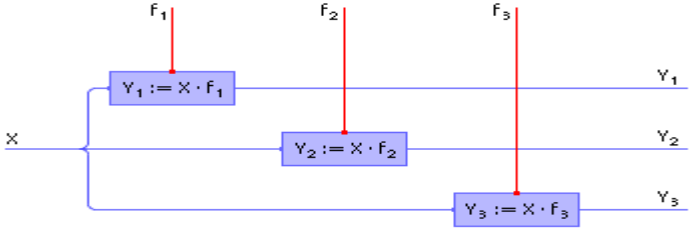


Рис. 31.14. Схема проекту, виконаного в середовищі Stratum-2000, що реалізує комутатор. Відбито частину проекту без формувача прапорів

Змінні Y, X приймають дійсні значення, як це й покладено змінним, які імітують матеріальні потоки. Можна уявити собі аналогію: потік X переходить у потік Y, якщо кран f відкритий ($f = 1$), або, навпаки, значення потоку X не переходять на змінну Y, тому що кран закритий ($f = 0$). Така схема надалі більше краща, тому що відокремлює матеріальну систему від інформаційної, роблячи проект більше зрозумілий і прозорим (див. також приклад 5 нижче).

Об'єднаємо інформаційну схему зі схемою, що імітує матеріальні потоки (див. мал. 31.15).

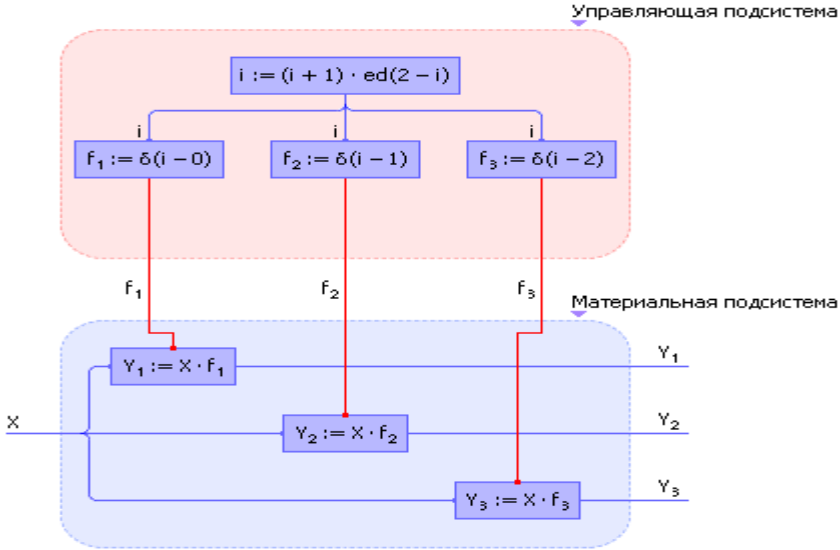


Рис. 31.15. Схема проекту «Модель комутатора», представленого композицією інформаційної й матеріальної підсистем. (Проект виконаний у середовищі Stratum-2000)

На мал. 31.16 показана взаємодія двох підсистем (інформаційної й матеріальної) між собою в загальному виді, якого рекомендується досягати в ідеалі для прозорості процесу проектування. Всі залежності й закони, що стосуються описи властивостей матерії й енергії, рекомендується включати до складу моделі матеріальних потоків (іноді в цій підсистемі додатково розділяють енергетичні й матеріальні потоки), а закони виміру, керування, регулювання, формування цілей - включати до складу моделі інформаційних сигналів.

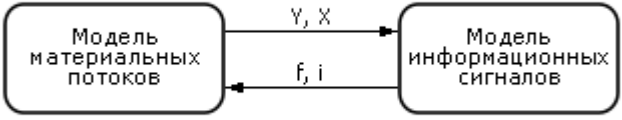


Рис. 31.16. Поділ матеріальної й керуючої моделей на окремі підсистеми

Приклад 5. Транспортування виробів з одного вузла на інший. Частий випадок у виробництві - організація транспортування виробів з одного виробничого вузла на інший. Під вузлом тут будемо розуміти сукупність складу й органа, що здійснює транспортування (наприклад, транспортний засіб, канал). Склад зберігає виробу, кількість яких може зменшуватися або збільшуватися, якщо транспортний орган бере на себе їхнє переміщення з вузла на вузол.

Допустимо, що на першому вузлі є X_1 виробів, а на другому — X_2 виробів. Позначимо як U кількість виробів, що транспортують, за такт. Тоді:
$X_1 := X_1 - U \cdot f$;
$X_2 := X_2 + U \cdot f$.
На мал. 31.17 показаний зв'язок двох вузлів транспортним каналом, що здійснює перекачування виробів зі складу одного вузла на склад іншого вузла із продуктивністю U штук в одиницю часу.

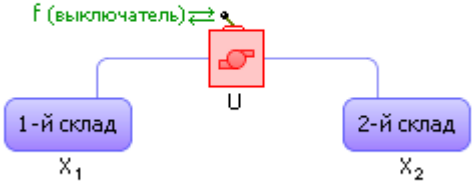


Рис. 31.17. Умовне зображення модуля
транспортування з одного складу на інший

Якщо прапор f дорівнює 1, то U виробів іде з першого вузла й стільки ж *у цей момент* приходить на другий вузол. Якщо прапор f дорівнює 0, то перекидання виробів не відбувається, тобто вони не віднімаються в першому вираженні й не складаються в другому, а X_1 й X_2 залишаються незмінними величинами.

Якщо виріб необхідно транспортувати через ланцюг вузлів (див. мал. 31.18), те, позначаючи об'єми перевезень між вузлами U_1, U_2, …, U_{n-1}, маємо:

$X_1 := X_1 - U_1 \cdot f$;

$X_2 := X_2 + U_1 \cdot f - U_2 \cdot f$;

$X_3 := X_3 + U_2 \cdot f - U_3 \cdot f$;

…;

$X_n := X_n + U_{n-1} \cdot f$...

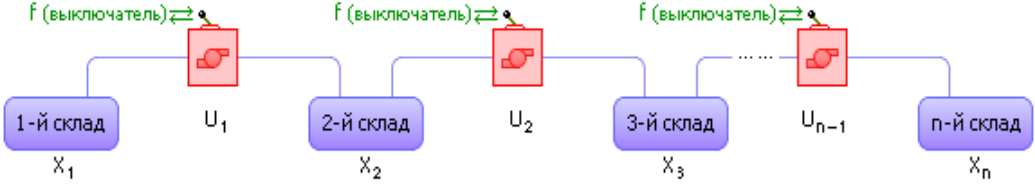


Рис. 31.18. Схема транспортування виробів через ланцюг складів

Якщо засувками f необхідно управляти окремо, то логічно ввести змінні f_i, управляючи засувкою в кожному вузлі окремо.

Варто помітити, що дану модель для поліпшення її адекватності можна модернізувати:

$X_1 := X_1 - U_1 \cdot f \cdot \text{ed}(X_1)$;

$X_2 := X_2 + U_1 \cdot f \cdot \text{ed}(X_1) - U_2 \cdot f \cdot \text{ed}(X_2)$;

$X_3 := X_3 + U_2 \cdot f \cdot \text{ed}(X_2) - U_3 \cdot f \cdot \text{ed}(X_3)$;

…;

$X_n := X_n + U_{n-1} \cdot f \cdot \text{ed}(X_{n-1})$...

Вираження $\text{ed}(X_i)$ відіграє роль своєрідного прапора дозволу транспортування, забороняючи її у випадку, якщо у вузлі немає виробів. Якщо такого прапора ні, то можливі дивні варіанти перекидання фіктивних виробів з вузла на вузол, наприклад, поява негативних виробів.

Ще одне зауваження. На цифровій техніці рівняння вирішуються з деяким тактом. Якщо протягом такту виявиться, що U_i більше X_i, то за такт із вузла зникне більше виробів, чим у ньому було, і кількість виробів стане менше нуля.

По-перших, на це можна не обертати уваги, оскільки це погрішність чисельного розрахунку. Другий варіант — запобігти все-таки таким колізіям.

Для цього є два способи. Перший полягає в тому, щоб зменшити час dt такту розрахунку (до цього ми приймали його рівним 1): $X_1 := X_1 - U_1 \cdot f \cdot \mathrm{ed}(X_1) \cdot dt$. Другий спосіб: віднімати й додавати значення U_i, якщо $U_i < X_i$, або віднімати й додавати X_i, якщо $U_i \geq X_i$, тобто віднімати й додавати мінімальне із двох чисел U_i й X_i: $X_1 := X_1 - U_1 \cdot f \cdot \mathrm{ed}(X_1) \cdot \mathrm{ed}(X_1 - U_1) - X_1 \cdot f \cdot \mathrm{ed}(X_1) \cdot \mathrm{not}(\mathrm{ed}(X_1 - U_1))$. У даному записі вираження $\mathrm{ed}(X_i)$, що грає роль своєрідного прапора дозволу транспортування, стає зайвим, тому його можна опустити: $X_1 := X_1 - U_1 \cdot f \cdot \mathrm{ed}(X_1 - U_1) - X_1 \cdot f \cdot \mathrm{not}(\mathrm{ed}(X_1 - U_1))$. З обліком усього вищесказаного, напишемо вираження для X_2: $X_2 := X_2 + U_1 \cdot f \cdot \mathrm{ed}(X_1 - U_1) + X_1 \cdot f \cdot \mathrm{not}(\mathrm{ed}(X_1 - U_1)) - U_2 \cdot f \cdot \mathrm{ed}(X_2 - U_2) - X_2 \cdot f \cdot \mathrm{not}(\mathrm{ed}(X_2 - U_2))$.

Якщо уявити собі, що ми маємо справу з виробничою технологічною лінією, то як склади в цьому ланцюжку виступають запаси незавершеної продукції (проміжні склади у виробництві), а роль транспортних артерій - верстати, які обробляють виробу, тим самим як би «проштовхуючи» їх з однієї ділянки на іншій.

Тоді U грає в цих вираженнях роль продуктивності верстатів: чим більше U, тим більше виробів за одну одиницю часу переходить із одного складу на інший, а X відіграє роль кількості виробів на проміжних складах у виробництві. Таким чином, U характеризує динаміку виробничого процесу, а множина X — його статику. Тобто X характеризує, скільки виробів перебуває у виробництві в різній стадії готовності, а U — скільки виробів і з якою швидкістю обробляється в кожен момент часу, як швидко перетікають виробу з вузла на вузол. Задаючи різні значення U, можна імітувати різні виробничі ситуації.

Іногда при описі виробництва задають закон руху U. Наприклад, $U = k \cdot X$. Логіка дії такого закону полягає в тому, що чим більше виробів X накраплюється на складі, тим більше повинна бути інтенсивність їхньої обробки U, щоб уникнути надмірного нагромадження виробів на проміжних складах. Як раніше було описано, закон руху, на відміну від рівнянь матеріального балансу, являє собою інформаційний потік, це добре видно на мал. 31.19.

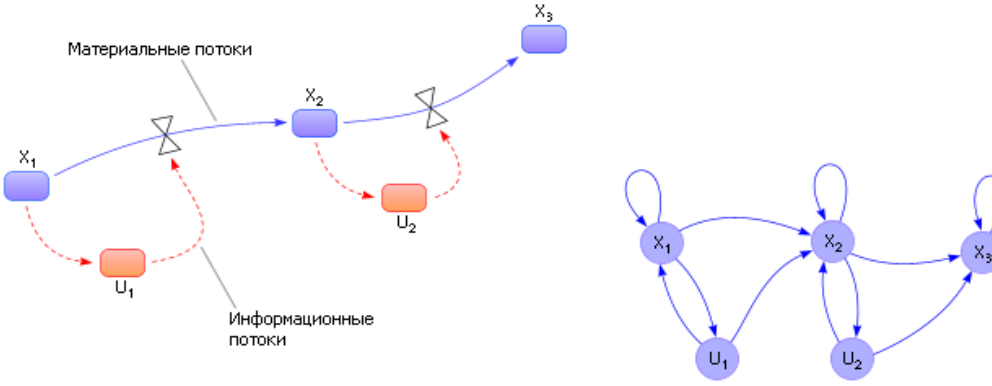


Рис. 31.19. Два варіанти зображення схеми моделі технологічної лінії з інформаційними й матеріальними зв'язками з різним рівнем деталізації. Ліворуч — позначення по Форрестеру з поділом матеріальних й інформаційних потоків, праворуч — граф залежностей змінних

Можна задати й інші закони, наприклад, $U_i := U_i + k \cdot (X_i - X_{i+1})$. Логіка цього закону полягає в тому, що якщо на $i + 1$-ом складі затоварення ($X_{i+1} > X_i$), те треба пригальмовувати U_i, якщо недолік, тобто $X_{i+1} < X_i$, те розганяти продуктивність даного вузла.

Вимірювальна частина системи керування може фіксувати загальна кількість виробів P_1 на всіх складах, їхню сумарну вартість P_2, загальну продуктивність верстатів P_3, нерівномірність розподілу виробів у виробництві P_4, виконання плану X_z — P_5 і так далі:

$P_1 = X_1 + X_2 + X_3 + \ldots + X_n$;

$P_2 = X_1 \cdot c_1 + X_2 \cdot c_2 + X_3 \cdot c_3 + \ldots + X_n \cdot c_n$;

$P_3 = U_1 + U_2 + U_3 + \ldots + U_n$;

$P_4 = \text{abs}(X_1 - X_2) + \text{abs}(X_2 - X_3) + \ldots + \text{abs}(X_{n-1} - X_n)$;

$P_5 = X_n - X_z$...

За даними показниками можна набудовувати регулятори у виробництві, домагаючись прийнятних значень у цих показників.

За допомогою такого підходу, використовуючи логічні комбінації (**И**, **АБО**, **НЕ**) і умови, відбивані одиничними функціями, можна описати поділу, злиття потоків, перехід партій, зборку, шлюб, перешкоди, затримки, перехід виробів з лінії на лінію, і взагалі будь-яку виробничу топологію.

Задачі, які можна вирішити на такій моделі:

- які регулятори підібрати для того, щоб забезпечити значення, що цікавить, *показників* $_{\text{p}}1$ - $_{\text{p}}5$;
- за який час можна виконати заданий план, які керування U для цього варто задати;
- як перейти від однієї картини заповнення складів (X_1, X_2, X_3, …, X_n) у виробництві до іншої (X_1^*, X_2^*, X_3^*, …, X_n^*), змінюючи множину U;
- чи компенсується і як регуляторами випадкове збурювання типу «шлюб»;
- який рівень запасів виробів варто мати у виробництві, щоб виключити дефіцит;
- наскільки можна зупинити якийсь із верстатів (поломка), щоб не зірвати план;
- які керуючі величини U варто застосувати при заданому заповненні складів (X_1, X_2, X_3, …, X_n), щоб забезпечити рівномірність руху й розподілу виробів по технологічній лінії.

Приклад 6. Обробка партії виробів у термічній печі. Розглянемо обробку партії виробів у термічній печі. Якщо температура в печі до 50 градусів, то партія знову надходить в обробку. Якщо температура в печі між 50 й 100 градусами, то партія йде на склад готових виробів. Якщо температура в печі більше 100 градусів, то партія надходить у брак.

Блок інформаційних сигналів має вигляд:
$f_1 := \text{ed}(50 - T)$;
$f_2 := \text{not}(\text{ed}(50 - T)) \cdot \text{ed}(100 - T)$;
$f_3 := \text{not}(\text{ed}(100 - T))$.

Блок імітації матеріальних процесів:
Обр := Партія $\cdot f_1$;
Готові := Готові + Партія $\cdot f_2$;
Брак := Брак + Партія $\cdot f_3$.

Зверніть увагу: на схемі моделі (мал. 31.20) окремо сформовані всі можливого й істотні для процесу умови «ed(Умова)», окремо сформовані інформаційні сигнали (f_1, f_2, f_3), окремо описані матеріальні потоки.

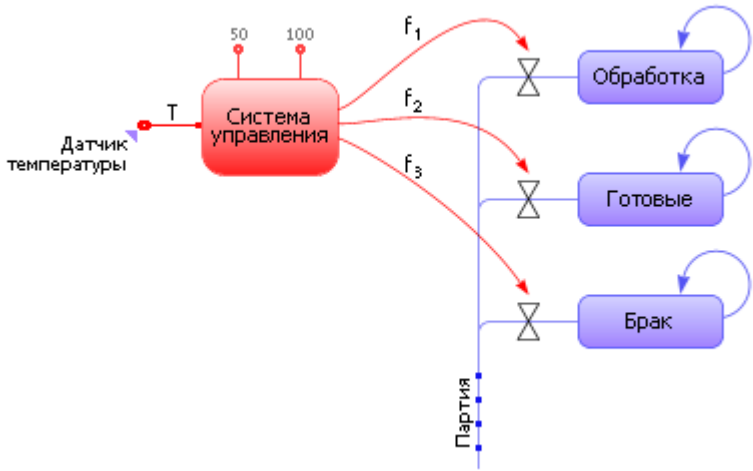


Рис. 31.20. Схема моделі «Обробка виробів у термічній печі»

Приклад 7. Навантаження поїзда. Розглянемо процес навантаження поїзда, що складає з електровоза й причеплених до нього трьох вагонів (див. мал. 31.21). Над першим вагоном перебуває бункер, з якого насипається вантаж. Вагон коштує на вагах, що вимірюють масу насипаного вантажу. Вантажопідйомність вагонів задана й відома системі керування з довідника в базі даних АСУ. Система керування (СУ) повинна визначити момент (і видати керуючий сигнал електровозу), коли варто поставити під завантаження наступний вагон. Після завантаження останнього вагона варто закрити заслінку бункера.

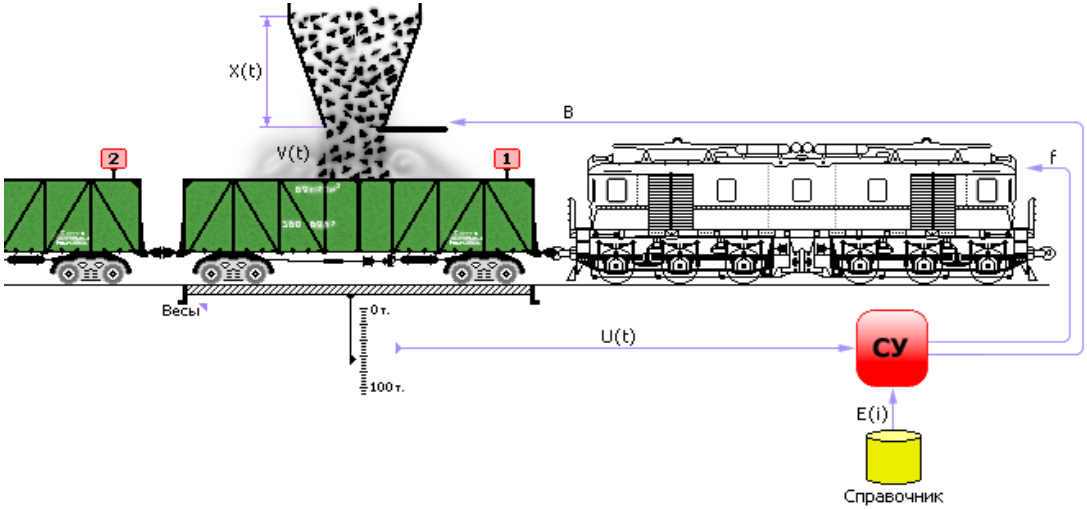


Рис. 31.21. Навантаження поїзда й система керування нею

Нехай $U(t)$ — поточне завантаження вагона, значення на вагах; $V(t)$ — інтенсивність надходження вантажу у вагон з бункера, кількість за такт; i — номер вагона (1, 2, 3); $E(i)$ — вантажопідйомність *i-го* вагона (заданий); $X(t)$ — рівень сировини в бункері; f — сигнал машиністові електровоза: «продьорнуть потяг на один вагон уперед».

Вираження, що моделюють навантаження й реакції системи керування, повинні імітувати процес навантаження (закони зміни $U(t)$, $X(t)$ і $V(t)$), сигнали АСУ (f й i).

$U(t) := (U(t) + V(t)) \cdot \text{ed}(E(i) - U(t))$ — завантаження вагона збільшується, якщо вагон не повний, і зупиняється й скидається в нуль, якщо вагон завантажений до норми $E(i)$;
$f := \text{not}(\text{ed}(E(i) - U(t)))$ — сигнал машиністові електровоза: «продерни склад»;
$i := i + f$ — лічильник вагонів збільшується в момент просмикування складу;
$X(t) := X(t) - V(t)$ — рівень сировини в бункері знижується;
$a := k \cdot X(t)$ — чим менше рівень X, тим повільніше висипає сировина, тому що верхні шари менше давлять на нижні;
$V(t) := a \cdot B$ — інтенсивність надходження вантажу у вагон з бункера;
$B := \text{ed}(3 - i + 1) \cdot \text{ed}(X(t))$ — сировина надходить у вагон, якщо його номер не більше третього **И** сировина в бункері є.

На основі побудованої моделі можна вирішити ряд задач, наприклад, визначити час, за яке буде завантажений весь склад, або з'ясувати інші якісь інші питання. У загальному випадку, те, яка задача буде зважуватися на моделі, багато в чому залежить від цілей людини, що використає цю модель.

Приклад 8. Тригер. Часто для керування інформаційними сигналами потрібен тригер. Таблиця істинності тригера представлена в табл. 31.1.

Таблиця 31.1.

Таблиця істинності тригера

Входи			**Новий стан тригера, вихід**
Скидання тригера	**Старий стан**	**Інформаційний вхід**	
R	**Q**	**S**	**Q**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Це пристрій, як видно з табл. 31.1, легко описується логічними функціями, але при цьому одночасно є системою з пам'яттю (усередині тригера перебуває ланцюг зворотного зв'язку, математично: $Q := F(R, S, Q)$, див. мал. 31.22).

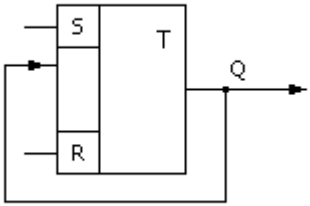


Рис. 31.22. Умовна позначка тригера на схемах

Представимо модель тригера, використовуючи логічні функції його стану. У табл. 31.1 виділимо всі рядки з $Q = 1$ (стовпець «Новий стан тригера, вихід»). Їх три. Запишемо три ситуації через операцію **АБО**: $Q := c_1$ АБО c_2 АБО c_3. Кожна ситуація, щоб рядок видав 1 на виході, повинна бути ОДНОЧАСНОЮ комбінацією сигналів R, Q, S, тобто R И Q И S. Якщо вхідний сигнал у рядку дорівнює 1, то змінну треба взяти «як є», якщо дорівнює 0, то інвертувати функцією НЕ. Наприклад, рядок

R	**Q**	**S**	**Q**
0	0	1	1

дає: НЕ(R) І НЕ(Q) И S И Q, або, використовуючи функції в записі, як їх звичайно реалізують у формальних мовах, маємо: not(R) · not(Q) · S · Q, або ще варіант: and(and(not(R), not(Q)), and(S, Q)). Вся таблиця у вигляді логічної функції опишеться так:

$$Q := (\text{not}(R) \cdot \text{not}(Q) \cdot S) \text{ АБО } (\text{not}(R) \cdot Q \cdot \text{not}(S)) \text{ АБО } (\text{not}(R) \cdot Q \cdot S).$$

У табл. 31.2 для довідки наведені значення загальновикористовуваних логічних функцій **И**, **АБО**, **НЕ**. Остаточно, приводячи подібні, виносячи за дужки й використовуючи формальний загальноприйнятий язиковий запис, одержимо:

$$Q := \text{and}(\text{not}(R), \text{or}(S, \text{and}(Q, \text{not}(S)))).$$

Таблиця 31.2.
Значення логічних функцій И, АБО, НЕ

A	**B**	**A and B**	**A or B**	**not A**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Приклад 9. СТАРТІВ-СТІП або Пастка. Тепер допустимо, що треба змоделювати такий процес, що починається при виконанні якогось сигналу Y, потім іде якийсь час T_p і зупиняється сам по собі по проходженні такого часу. Якщо під час виконання процесу надходить знову сигнал Y, то процес ігнорує такий сигнал Y. Якщо сигнал Y приходить знову, коли процес зупинений, то процес готовий сприйняти сигнал і відреагувати на нього знову.

Пасткою така конструкція називається тому, що модель на час захоплює вхідний сигнал і не реагує протягом цього часу на інші вхідні сигнали. За допомогою таких конструкцій часто імітують процес обробки виробів в окремих вузлах технологічної лінії. Виріб попадає у вузол, як у пастку, і виходить із нього тільки через певний час. Протягом усього цього часу вузол зайнятий і не готовий прийняти інші вироби (вхідні сигнали). Виходить, що й виріб зайнятий вузлом, і вузол зайнятий виробом.

Назвемо для наочності прапори:
«Можна_почати» — прапор установлюється в 1, якщо можна почати процес, а процес можна почати, якщо прийшов сигнал Y **И** процес зараз не йде;
«Процес_іде» — прапор установлюється в 1, якщо процес відбувається, тобто вже почався **И** поки не закінчився. Процес іде, якщо можна почати процес **АБО** якщо процес уже йде **И** поки не зупинений;
«Стіп» — прапор установлюється в 1, якщо треба зупинити процес, а це відбувається, якщо процес уже або поки не йде **АБО** час протікання процесу закінчилося.

Теперь запишемо рівняння відповідно до логіки (у записі використані знаки & (И) і | (АБО)):

Можна_почати := ed(Y) & not(Процес_іде);
Процес_іде := Можна_почати | (Процес_іде & not(Стіп));
$t := (t + h) \cdot \text{ed}(T_p - t) \cdot$ Процес_іде;
Стіп := not(Процес_іде) | not(ed(t)).