Основи рачунарске технике 2

Садржај

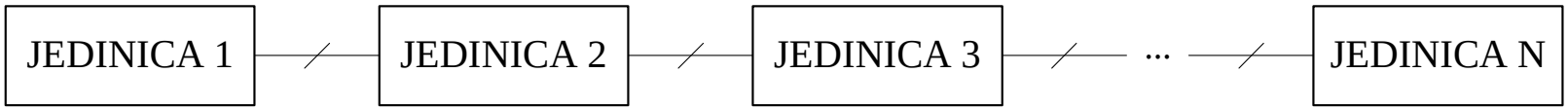
- Подсетник
  - Основни комбинациони модули
  - Основни секвенцијални модули
- Основни појмови
  - Структра дигитаног уређаја
  - Повезивање јединица
- Операциона јединица
  - Пример: Операција множења – MULU
- Управљачка јединица
  - Шетајућа јединица
  - Стандардна секвенцијална мрежа
  - Бројач корака
  - Микропрограмска

Структура дигиталног уређаја

- Дигитални уређај су прекидачке мреже које реализују једну или више операција над подацима који су представљени у бинарном облику.
- Дигитални уређаји су најчешће толико сложене прекидачке мреже да се пројектовање дигиталних уређаја не може реализовати формалним поступцима синтезе прекидачких мрежа.

Структура дигиталног уређаја

- Један од поступака је да се извршавање операције или операција подели на логичке целине које се називају фазе и да за извршавање сваке фазе у дигиталном уређају постоји посебна јединица.



Операциона јединица

- Операциона јединица се састоји од комбинационих и секвенцијалних прекидачких мрежа које се према својој функцији могу поделити у три групе.

1. Секвенцијалне прекидачке мреже које служе за памћење бинарних речи којима се дефинишу операције и подаци (регистри и меморије).
2. Комбинационе и секвенцијалне прекидачке мреже које реализују изабрани скуп микрооперација (мултиплексер, сабирач, АЛУ, декодер, ...).
3. Комбинационе и секвенцијалне прекидачке мреже које формирају сигнале логичких услова. (компаратори и бројачи, али и наменске мреже).

Операциона јединица

- Операције се разлажу на одређен број корака и то тако да у сваки корак уђе једна или више микрооперација.
- Сваки од корака на које је разложена операција се извршава за време трајања једне периоде сигнала такта, па реализација операција захтева онолико тактова на колико корака је разложена операција.
- Под микрооперацијама се подразумевају операције које се реализују за време трајања једне периоде сигнала такта у комбинационим или секвенцијалним мрежама.

Операциона јединица

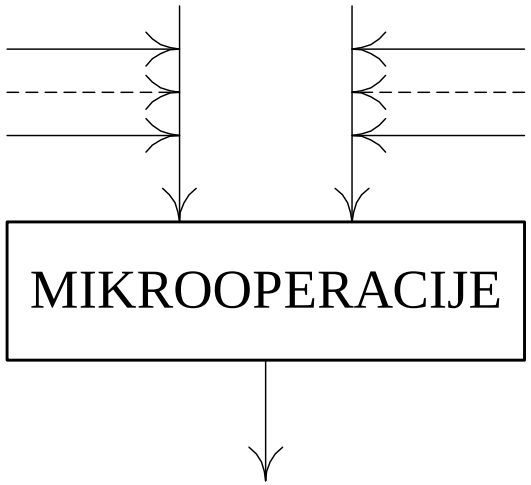
- Време извршавања операције се добија као производ броја корака на које је операција разложена и вредности периоде сигнала такта.
- Ово време се може смањити
  - смањивањем броја корака неопходних за извршавање микрооперација на које је једна операција разложена и
  - смањивањем периоде сигнала такта.

Операциона јединица

- Уређени низ корака на које је разложена нека операција и логички услови на основу којих се одређује редослед корака дефинишу алгоритам операције.
- За представљање алгоритама операција користе се дијаграми тока који се цртају помоћу операционих и управљачких блокова.
- Операциони и условни блокови се повезују линијама тако да излаз неког блока води на улаз једног и само једног блока.

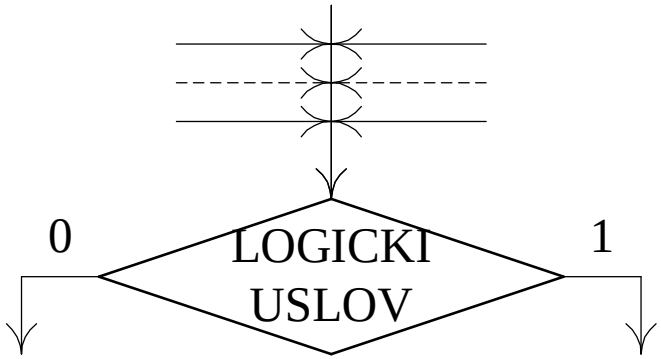
Операциона јединица

- У операциони блок се уписују микрооперације које се реализују у истом кораку алгоритма. Операциони блок има један или више улаза и један излаз.



Операциона јединица

- У условни блок се уписује логички услов које дефинише гранање алгоритма.
- Управљачки блок има један или више улаза и два излаза код обичног гранања и више излаза код вишеструког условног скока.



Управљачка јединица

- Управљачка јединица се састоји из секвенцијалних и комбинационих прекидачких мрежа које сагласно алгоритму операције и вредностима сигнала логичких услова који се формирају у операционој јединици генеришу управљачке сигнале за операциону јединицу.
- Ови сигнали одређују које микрооперације и по ком редоследу треба извршавати у операционој јединици да би се као резултат после одређеног броја сигнала такта реализовала операција.

Управљачка јединица

- Управљачки сигнали су сигнали којима се у операционој јединици
  - одређују садржаји који у неком кораку треба да се пропусте кроз мултиплексере и појаве не излазима мултиплексера,
  - одређује која аритметичка или логичка операција треба да се реализује у ALU,
  - врши упис у регистре,
  - инкрементира или декрементира садржај бројача итд.

Управљачка јединица

- Управљачка јединица се реализује као секвенцијална мрежа која има онолико стања колико има корака на које је разложена операција.
- Посебно стање секвенцијалне мреже се додељује сваком кораку.
- У зависности од тога како се стање секвенцијалне мреже користи за генерисање управљачких сигнала операционе јединице разликују се две групе реализација управљачких јединица и то реализације са ожиченим и микропрограмским генерисањем управљачких сигнала.

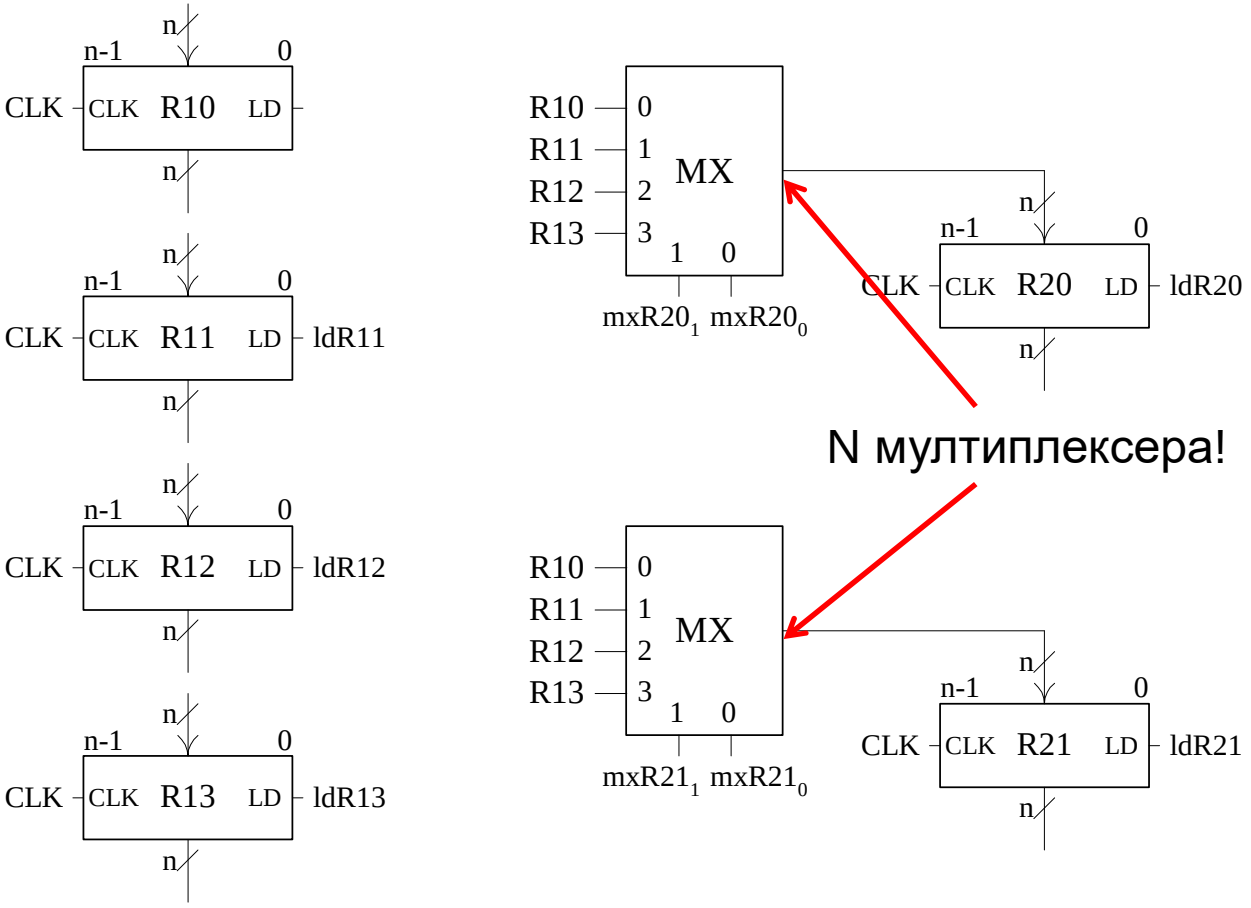
Управљачка јединица

- У случају ожичених реализација сигнал декодованог стања секвенцијалне мреже придружен неком кораку се користи за генерисање оних управљачких сигнала операционе јединице који у датом кораку треба да имају вредност 1.
- У случају микропрограмских реализација стање секвенцијалне мреже се користи као адреса микропрограмске меморије из које се чита контролна реч формирана за дати корак и на основу њеног садржаја генеришу одговарајуће вредности управљачких сигнала операционе јединице.

Повезивање јединица

- Повезивања регистара се може реализовати на више начина, при чему су они варијанте два основна начина повезивања:
  - повезивање директним везама и
  - повезивање магистралом.

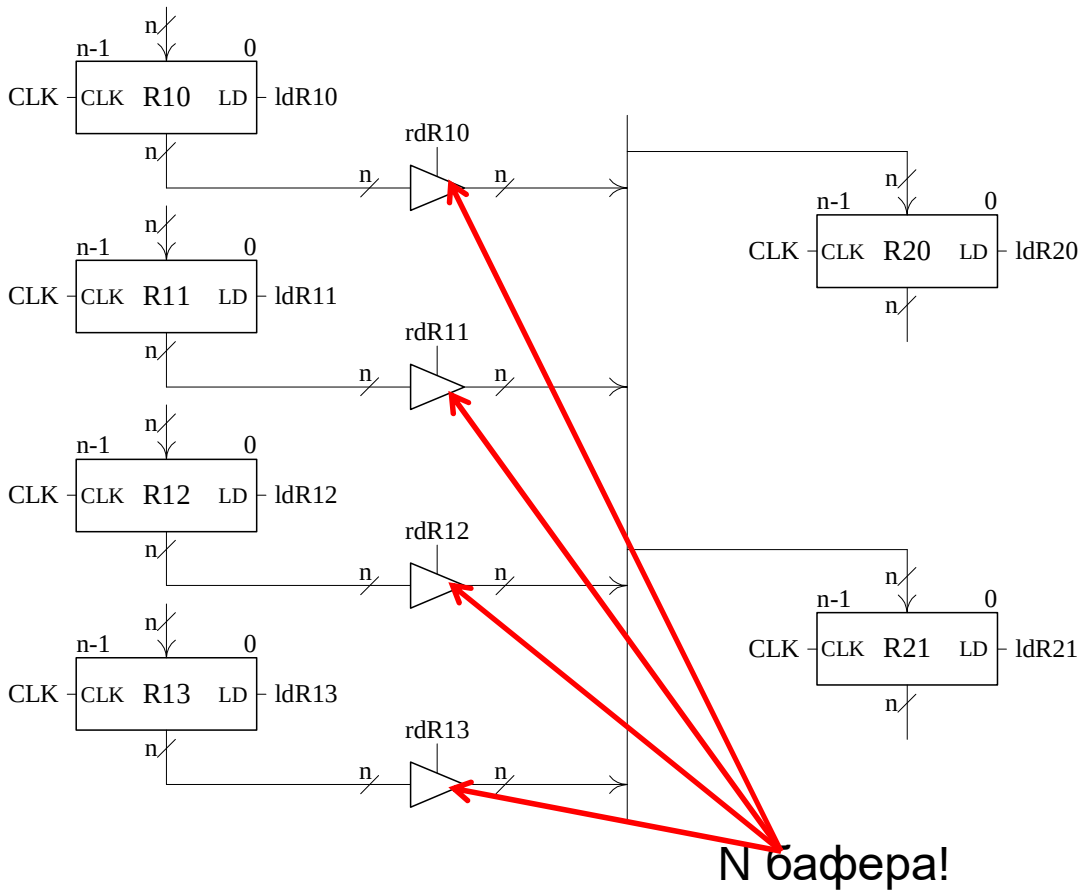
Повезивање директним везама



Повезивање директним везама

- Добра страна повезивања директним везама је да се у истом кораку могу реализовати микрооперације преноса садржаја од **више** изворишних регистара до **више** одредишних регистара
- Лоша страна повезивања директним везама је да је потребан велики број мултиплексера.

Повезивање магистралом



Повезивање магистралом

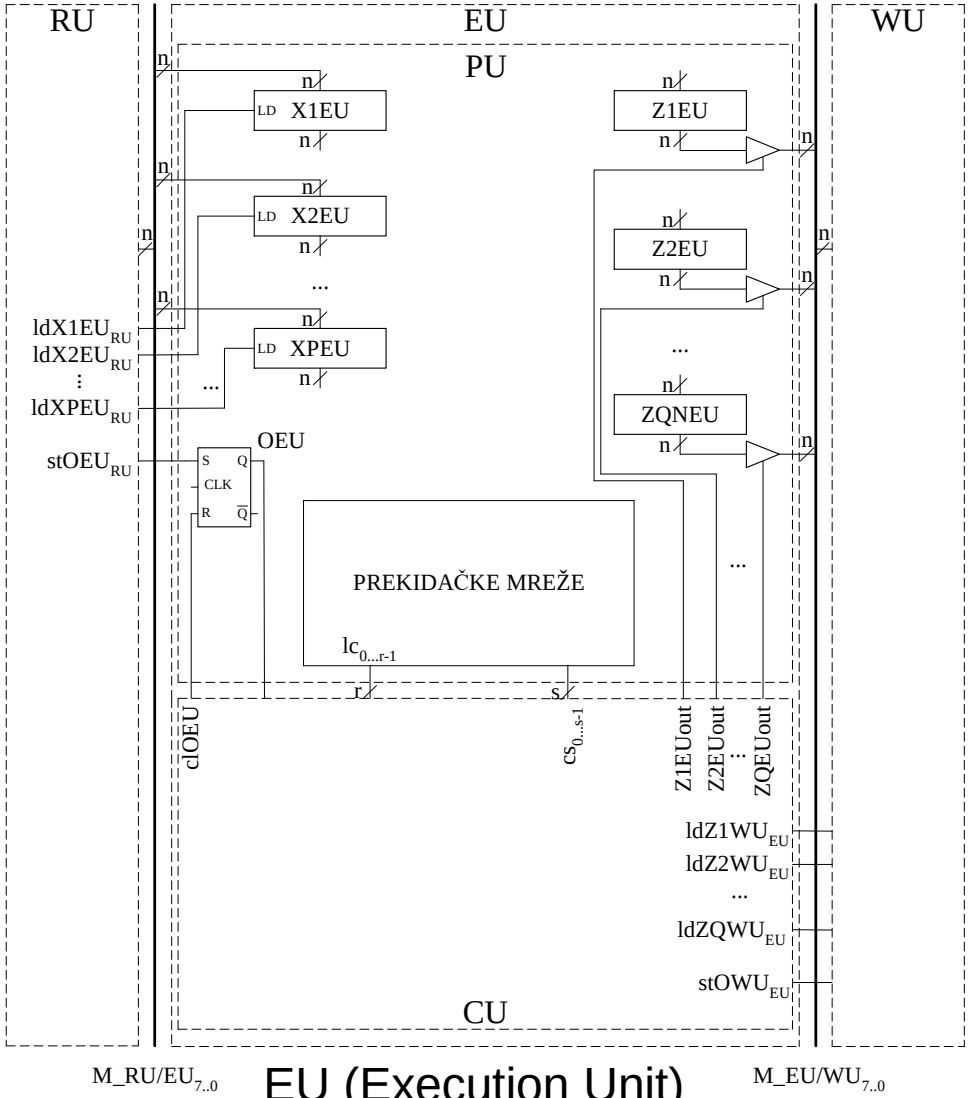
- Добра страна повезивања магистралом је да не постоји потреба за мултиплексерима, већ да се уместо њих користе бафери са три стања.
- Лоша страна је да се истовремено не може преносити садржај више од једног изворишног регистара у одредишне регистре.

Реализација јединица са једном и више операција

RU (Read Unit)

PU (Processing Unit)

CU (Control Unit)



WU (Write Unit)

EU (Execution Unit)

Операција множења – MULU

$$P = X \cdot Y = X \cdot \left(\sum_{i=0}^{n-1} Y_i \cdot 2^i \right) = \sum_{i=0}^{n-1} X \cdot Y_i \cdot 2^i$$

$P(0)=0+X\cdot Y_0\cdot 2^0$

$P(1)=P(0)+X\cdot Y_1\cdot 2^1$

...

$P(i)=P(i-1)+X\cdot Y_i\cdot 2^i$

...

$P(n-1)=P(i-2)+X\cdot Y_{n-1}\cdot 2^{n-1}$

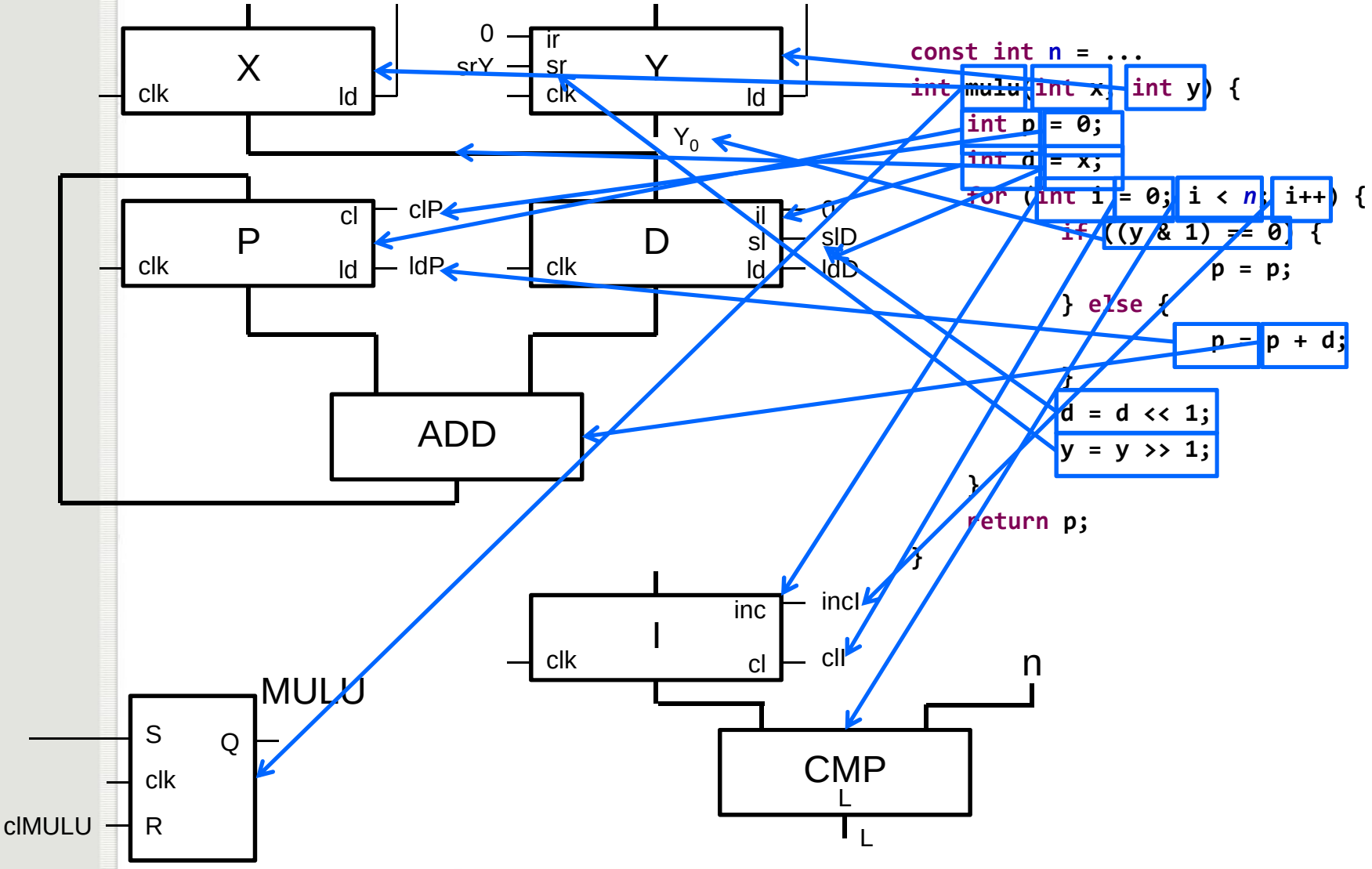
Операција множења – MULU

- Приликом рачунања парцијалног производа чланови се могу груписати и на следећи начин:
- $P(i)=P(i-1)+(X\cdot 2^i)\cdot Y_i$, где је $P(0)=0$.
- Израз $X\cdot 2^i$ се може исказати рекурзивно $D(i)=D(i-1)\cdot 2$, где је $D(0)=X$.
- Овим се добије да је $P(i)=P(i-1)+D(i)\cdot Y_i$,
  - у случају када је Y_i једнако нула даје $P(i)=P(i-1)$,
  - у случају када је Y_i једнако један даје $P(i)=P(i-1)+D(i)$.

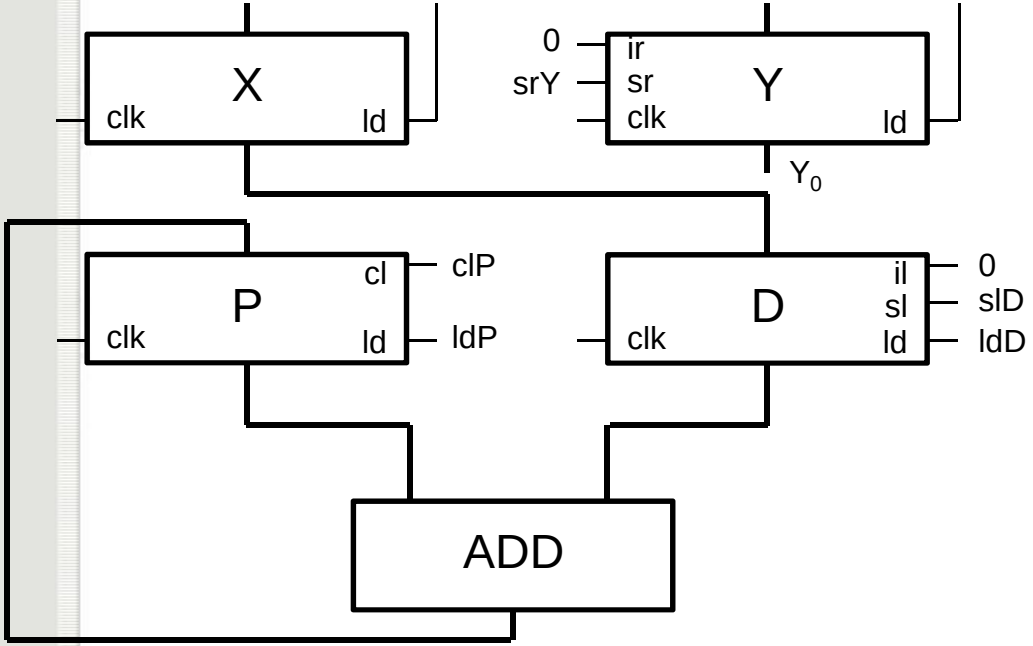
Операција множења – MULU

```
int mulu(int x, int y) {
   int p = 0;
   int d = x;
   for (int i = 0; i < n; i++) {
       if ((y & 1) == 0) {
              p = p;
       } else {
              p = p + d;
       }
       d = d << 1;
       y = y >> 1;
   }
   return p;
}
```

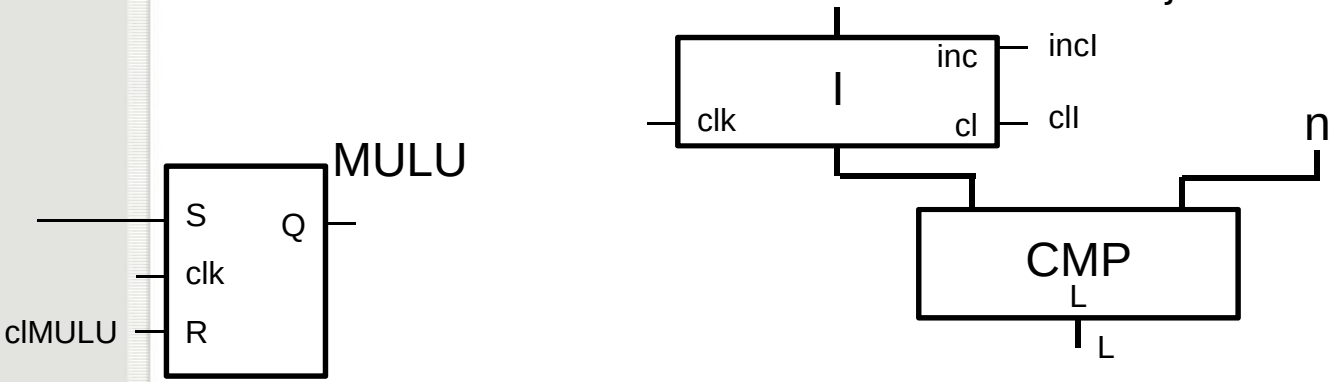
Операција множења – MULU



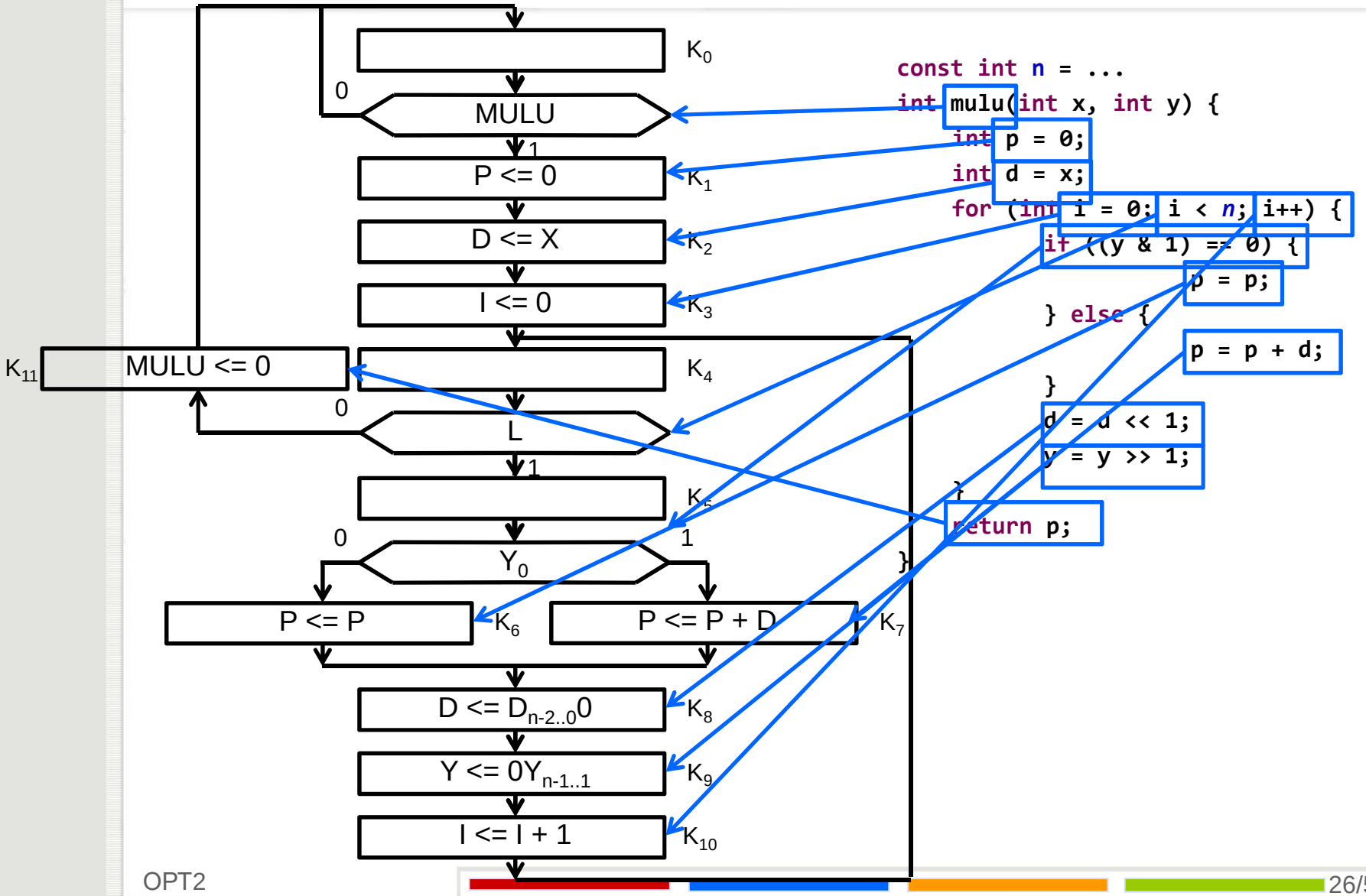
Операција множења – MULU



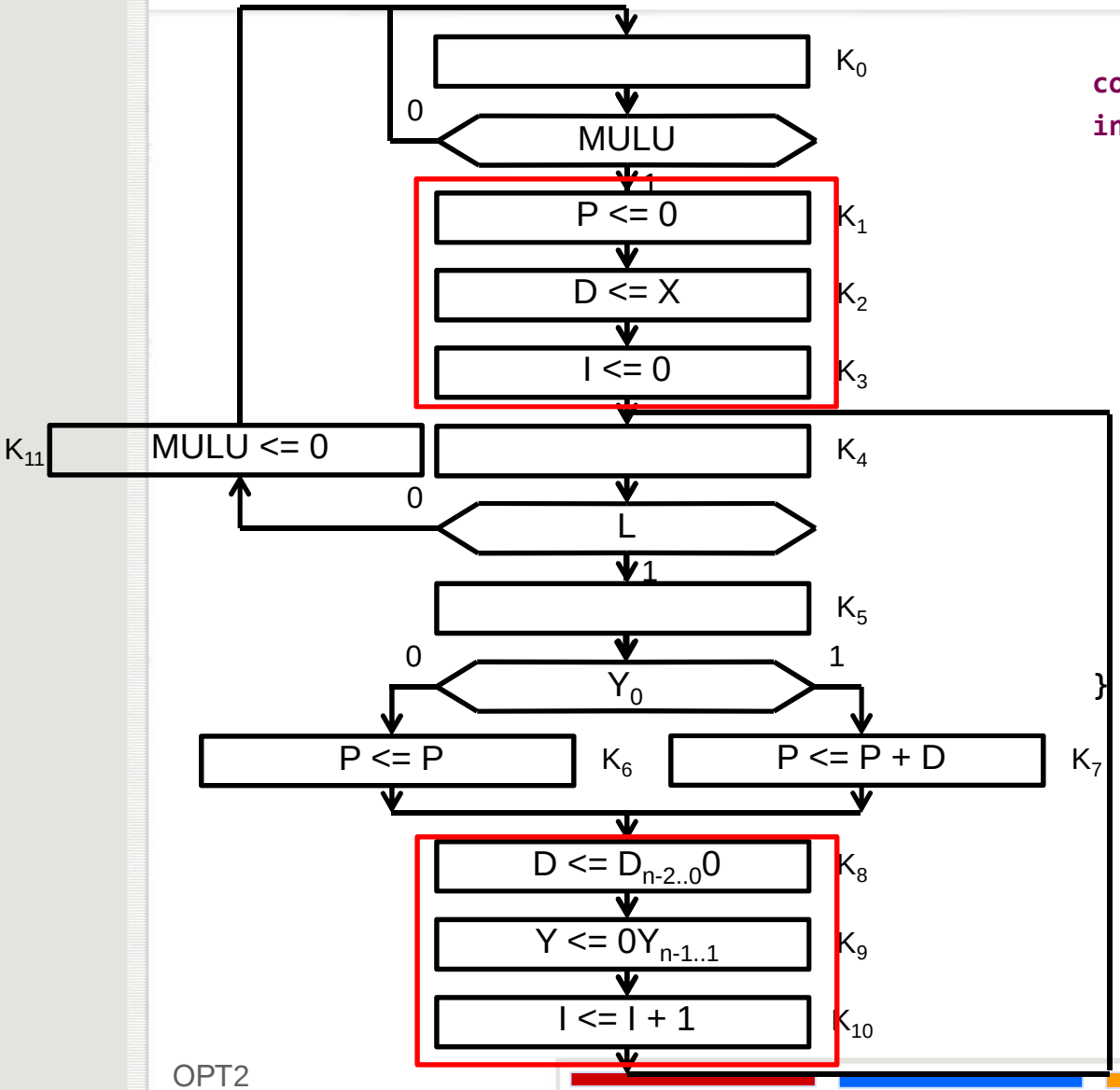

```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```



Операција множења – MULU

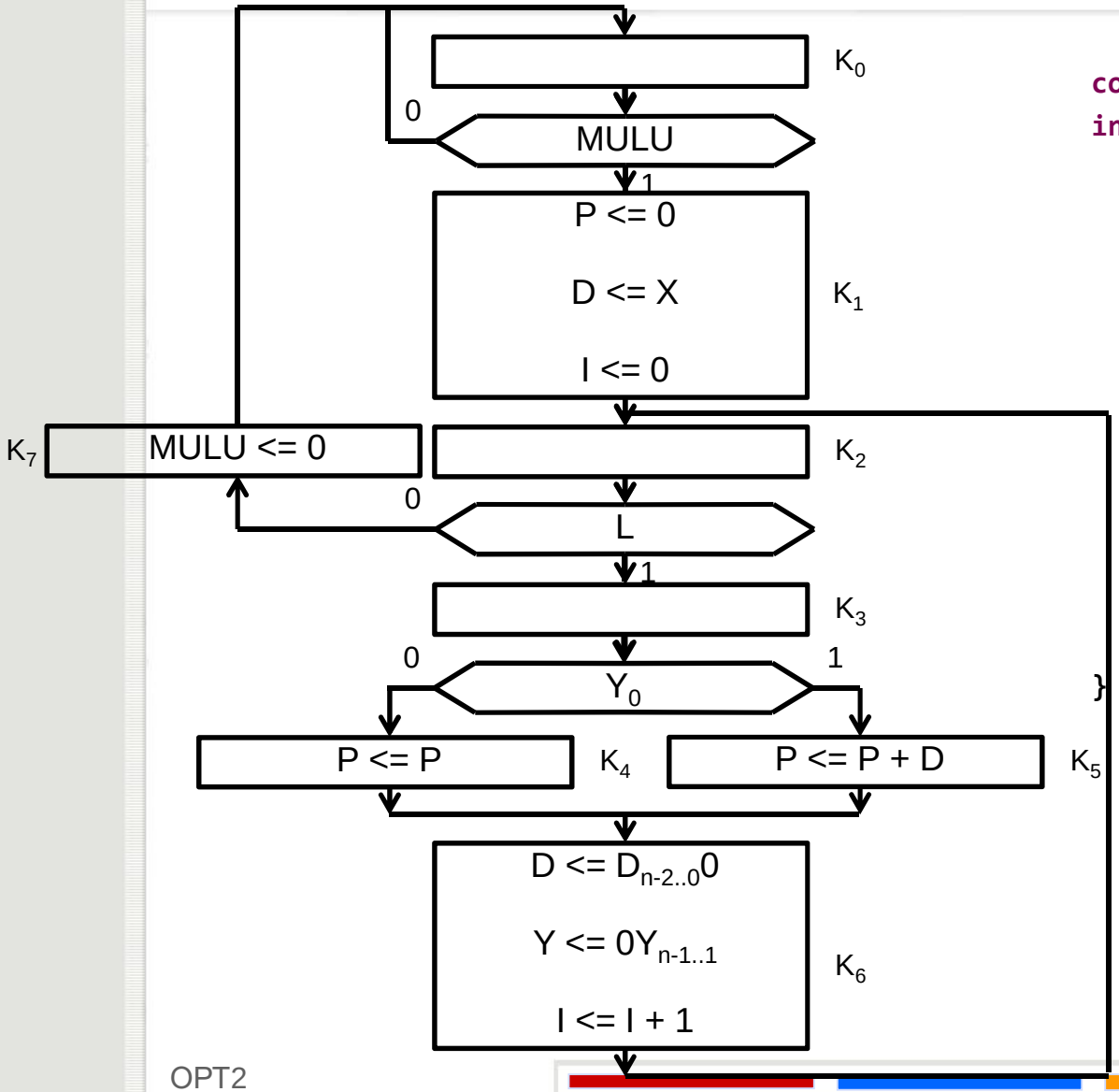


Операција множења – MULU



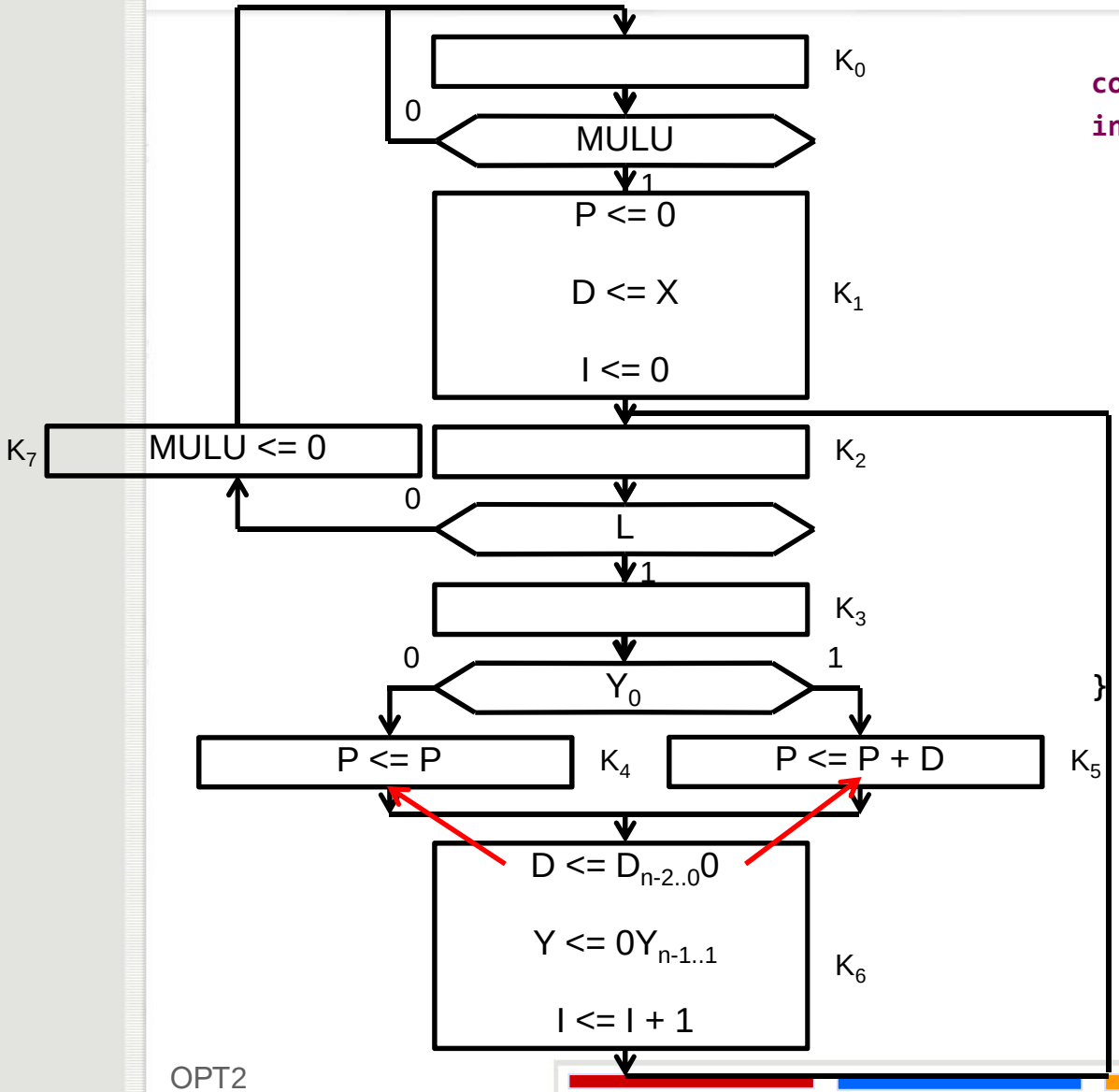

```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```

Операција множења – MULU



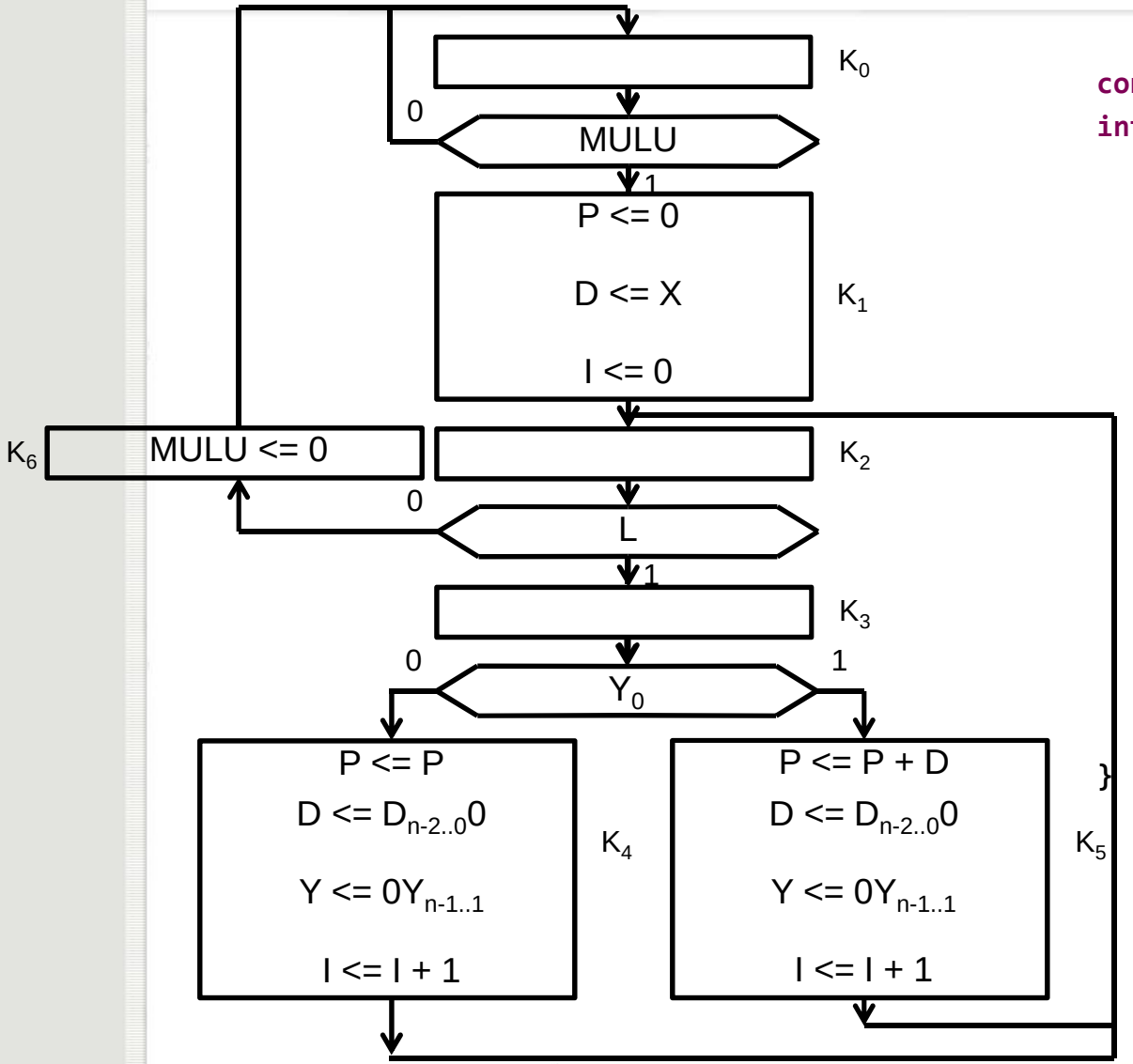

```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```

Операција множења – MULU



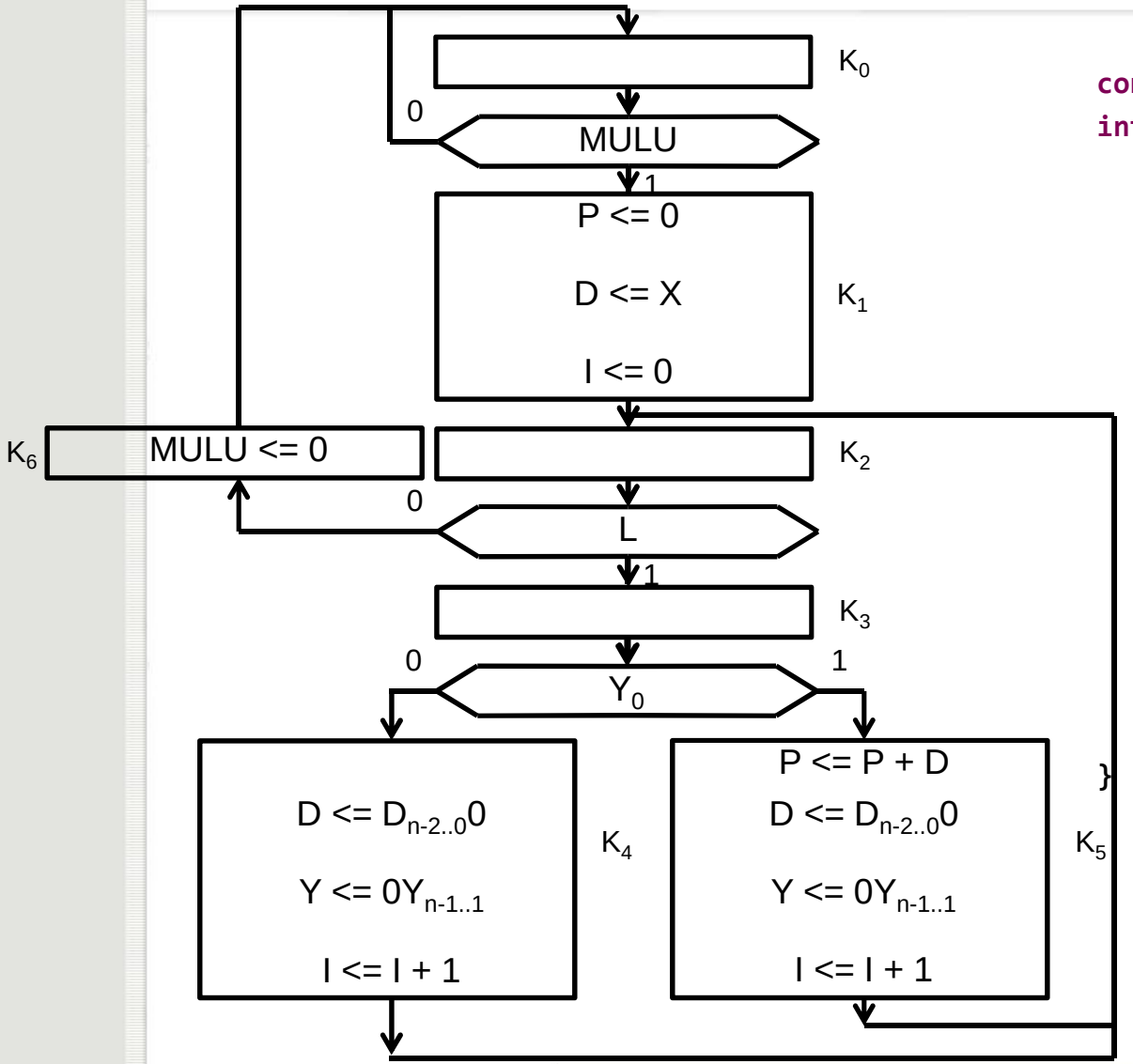

```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```

Операција множења – MULU



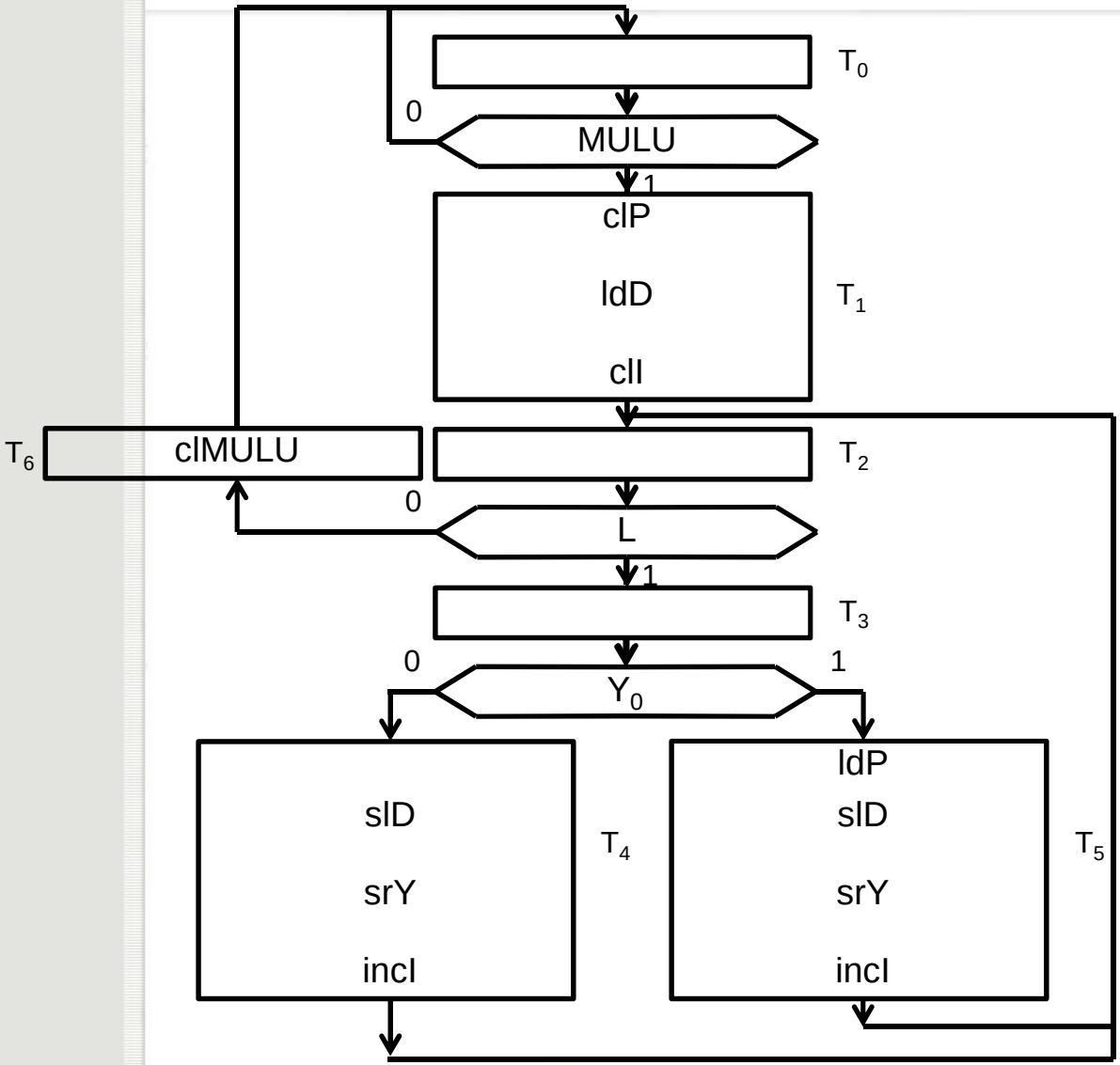

```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            p = p;
            d = d << 1;
            y = y >> 1;
        } else {
            p = p + d;
            d = d << 1;
            y = y >> 1;
        }
    }
    return p;
}
```

Операција множења – MULU

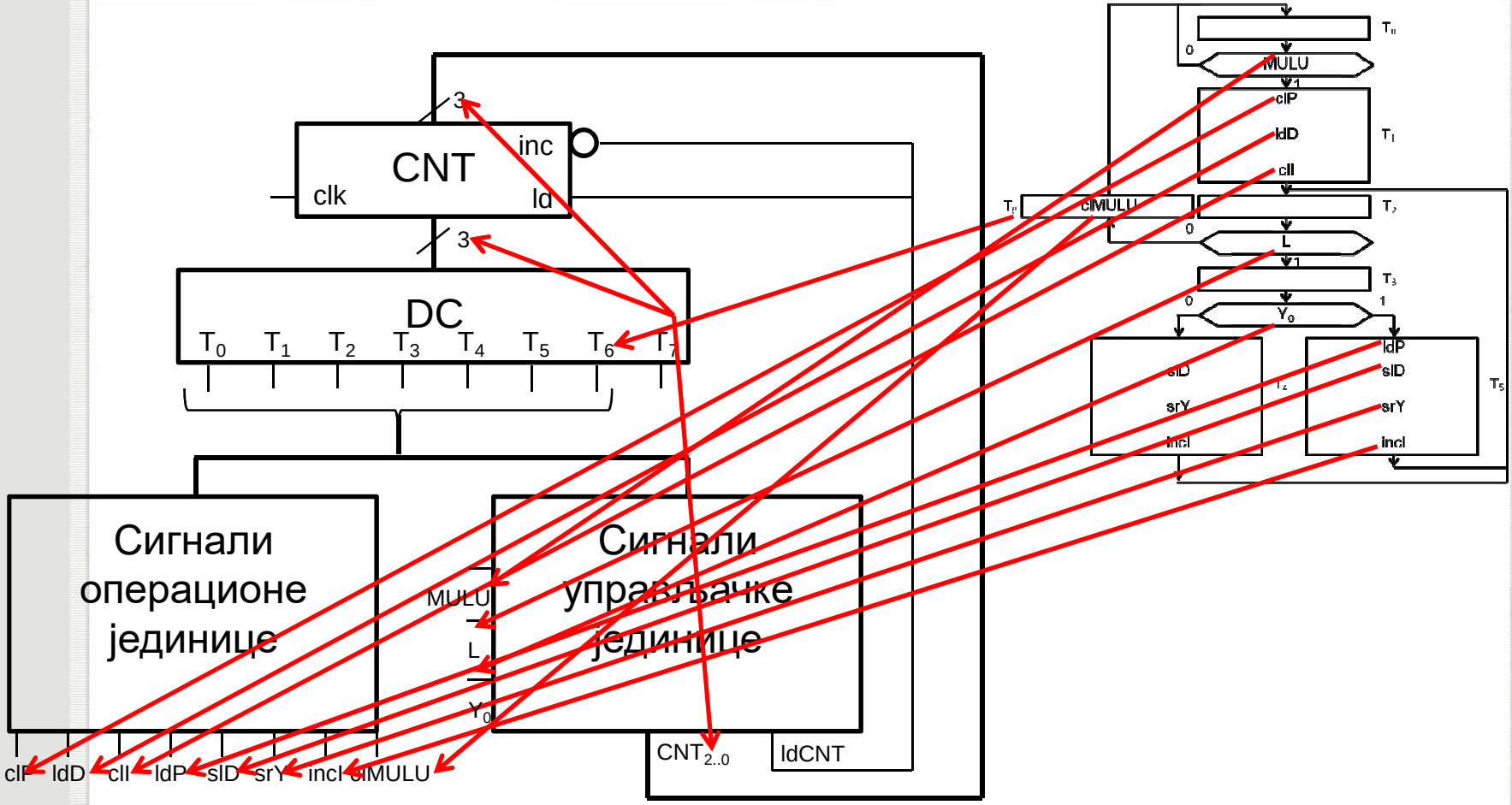



```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = 0; i < n; i++) {
        if ((y & 1) == 0) {
            d = d << 1;
            y = y >> 1;
        } else {
            p = p + d;
            d = d << 1;
            y = y >> 1;
        }
    }
    return p;
}
```

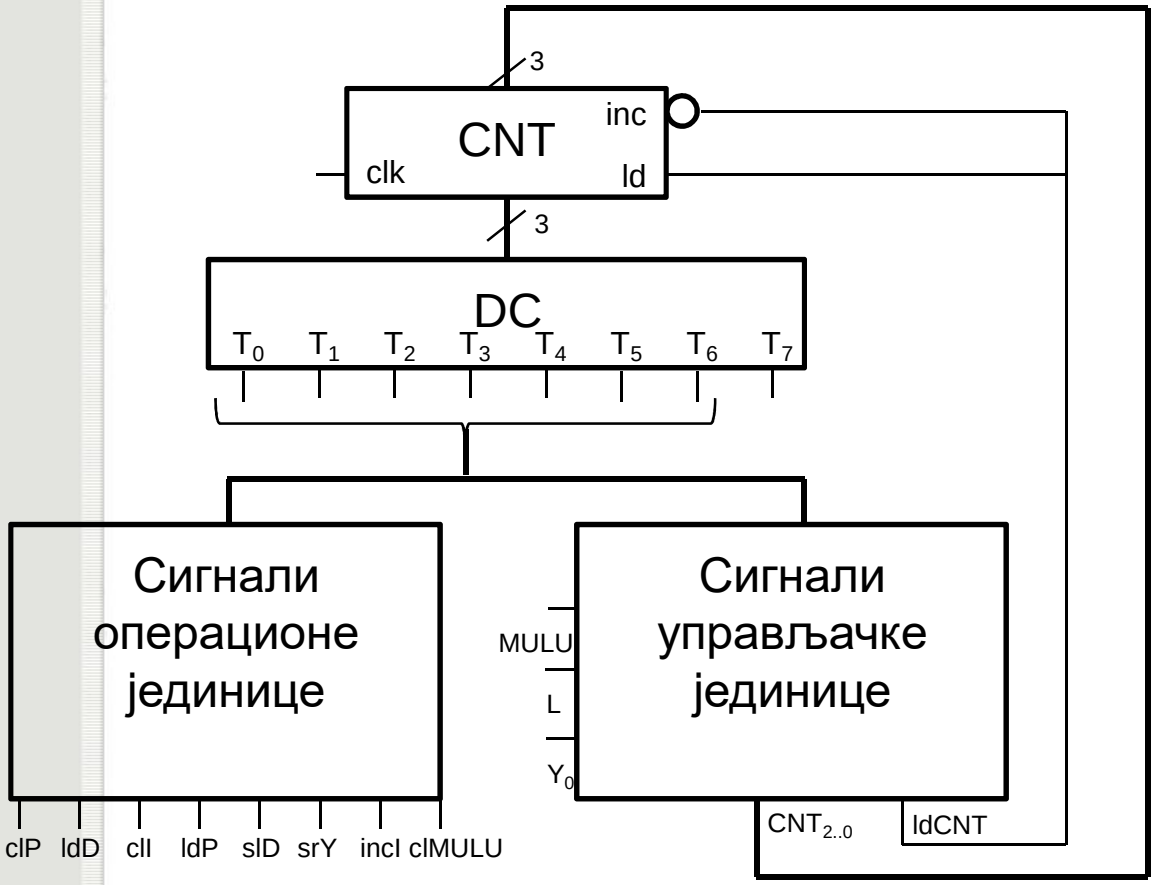
Операција множења – MULU

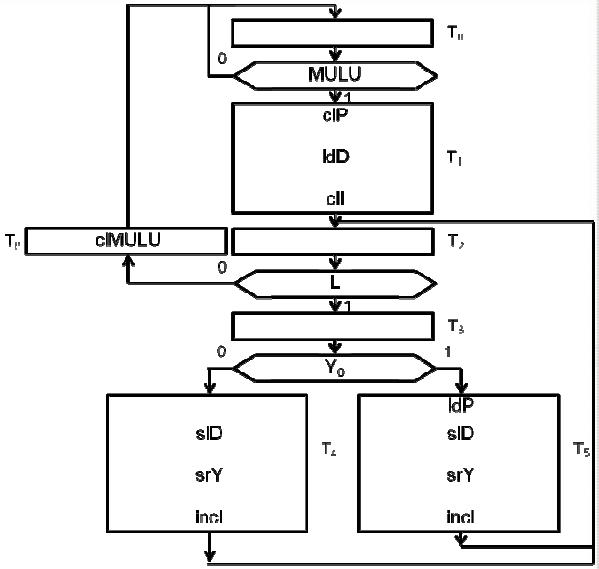


Операција множења – MULU



Операција множења – MULU





Операција множења – MULU

Сигнали операционе јединице

ldP = T_5

clP = T_1

ldD = T_1

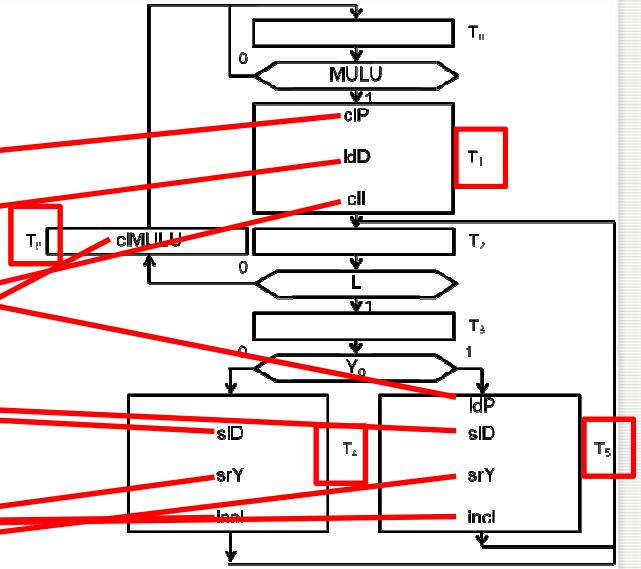
slD = $T_4 + T_5$

clI = T_1

incI = $T_4 + T_5$

srY = $T_4 + T_5$

clMULU = T_6



Операција множења – MULU

Сигнали операционе јединице

ldP = T_5

clP = T_1

ldD = T_1

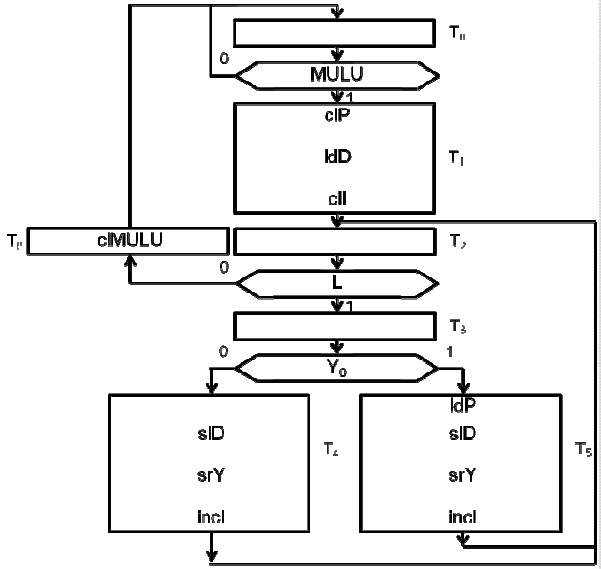
slD = $T_4 + T_5$

clI = T_1

incI = $T_4 + T_5$

srY = $T_4 + T_5$

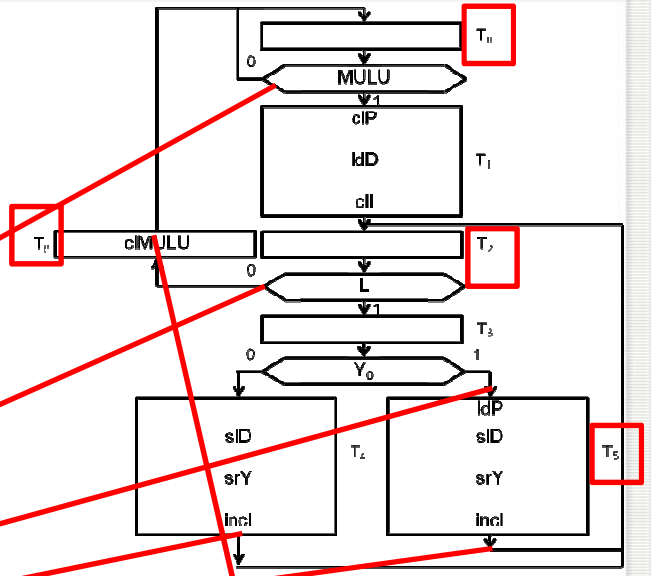
clMULU = T_6



Мрежа Мурогов типа

Операција множења – MULU

Сигнали управљачке јединице



$$ldCNT = T_0*\overline{MULU} + T_2*\overline{L} + T_3*Y_0 + T_4 + T_5 + T_6$$

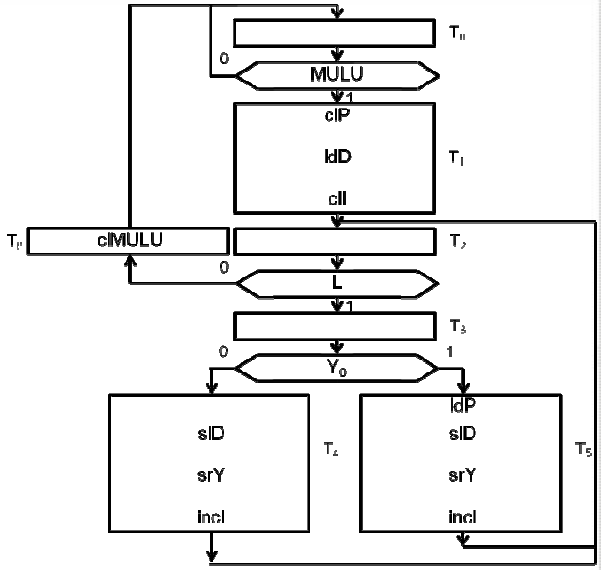
$$CNT_2 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_3*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

$$CNT_1 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_3*Y_0*0 + T_4*1 + T_5*1 + T_6*0$$

$$CNT_0 = T_0*\overline{MULU}*0 + T_2*\overline{L}*0 + T_3*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

Операција множења – MULU

Сигнали управљачке јединице



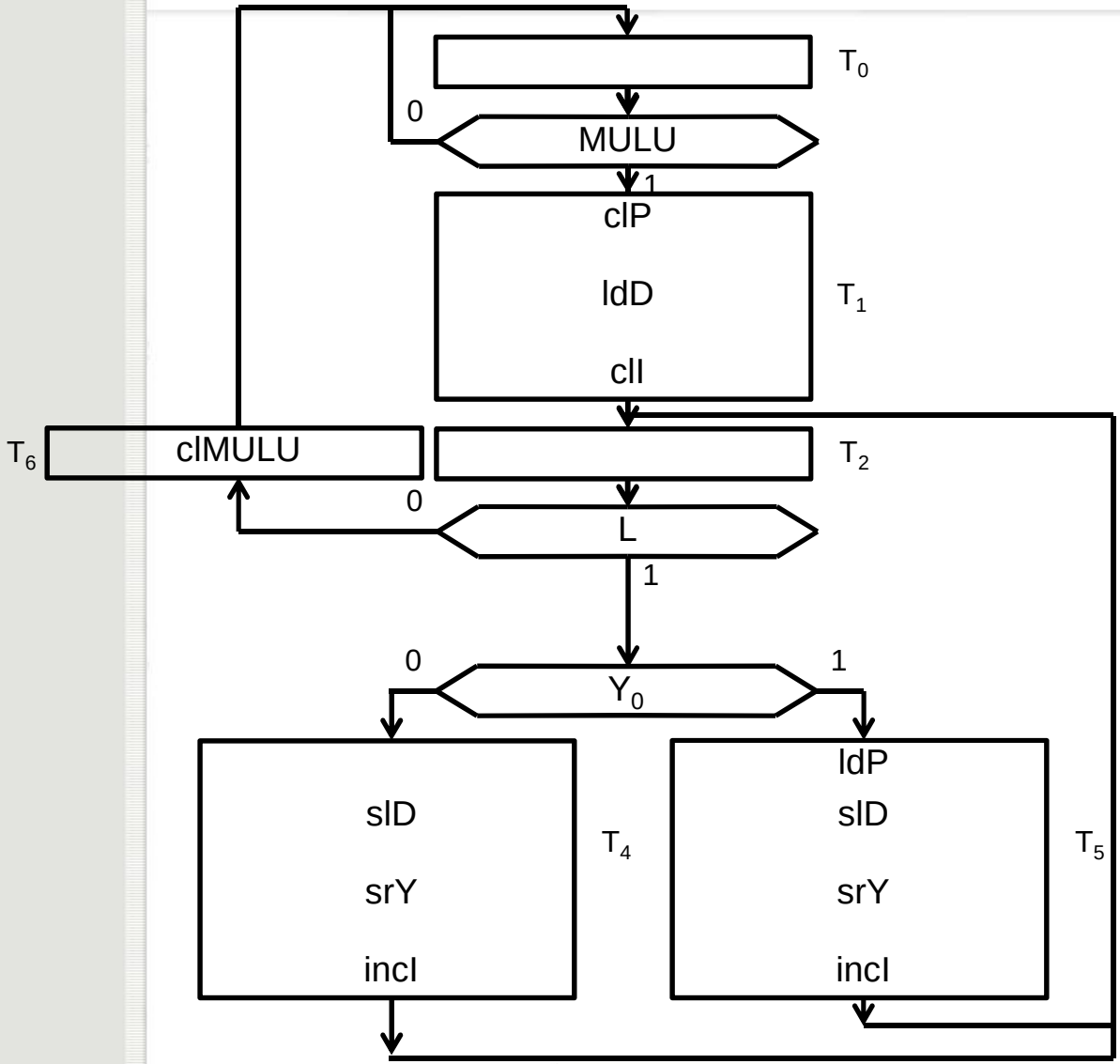
$ldCNT = T_0 * \overline{MULU} + T_2 * \overline{L} + T_3 * Y_0 + T_4 + T_5 + T_6$

$CNT_2 = T_2 * \overline{L} + T_3 * Y_0$

$CNT_1 = T_2 * \overline{L} + T_4 + T_5$

$CNT_0 = T_3 * Y_0$

Операција множења – MULU v2



Операција множења – MULU v2

Сигнали операционе јединице

ldP = T_5

clP = T_1

ldD = T_1

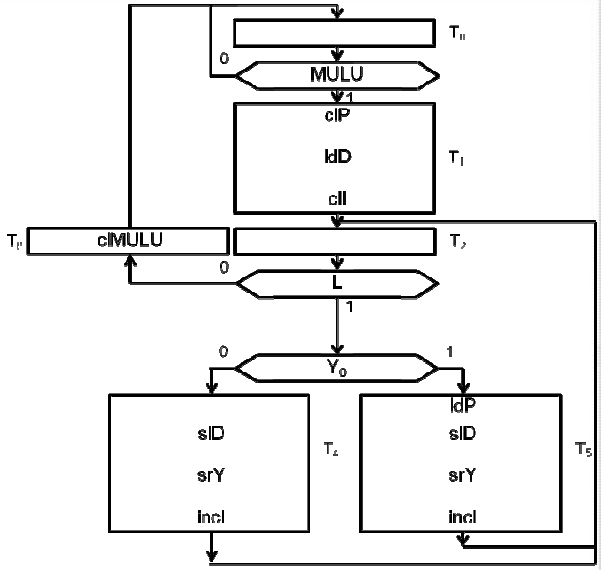
slD = $T_4 + T_5$

clI = T_1

incI = $T_4 + T_5$

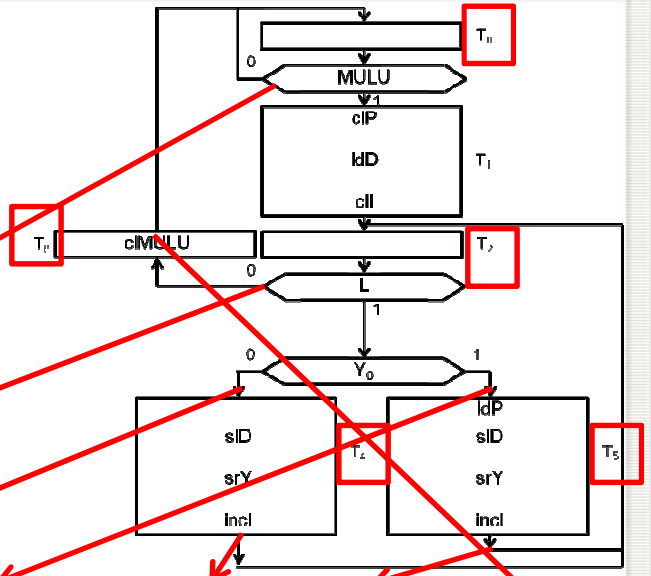
srY = $T_4 + T_5$

clMULU = T_6



Операција множења – MULU v2

Сигнали управљачке јединице



$$ldCNT = T_0*\overline{MULU} + T_2*\overline{L} + T_2*L*\overline{Y_0} + T_2*L*Y_0 + T_4 + T_5 + T_6$$

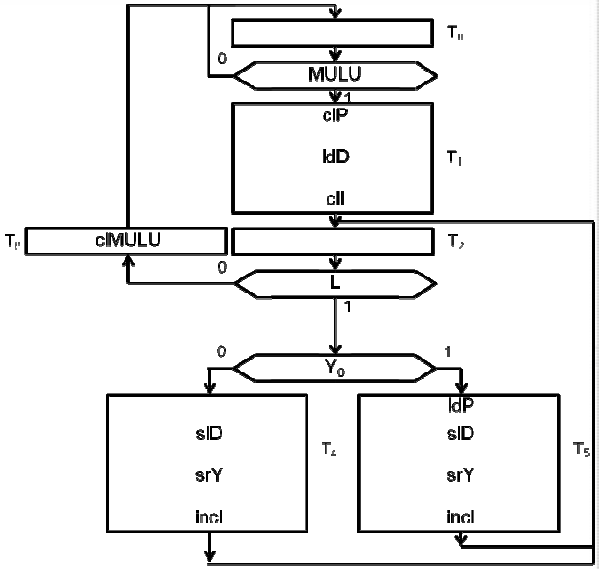
$$CNT_2 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_2*L*\overline{Y_0}*1 + T_2*L*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

$$CNT_1 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_2*L*\overline{Y_0}*0 + T_2*L*Y_0*0 + T_4*1 + T_5*1 + T_6*0$$

$$CNT_0 = T_0*\overline{MULU}*0 + T_2*\overline{L}*0 + T_2*L*\overline{Y_0}*0 + T_2*L*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

Операција множења – MULU v2

Сигнали управљачке јединице



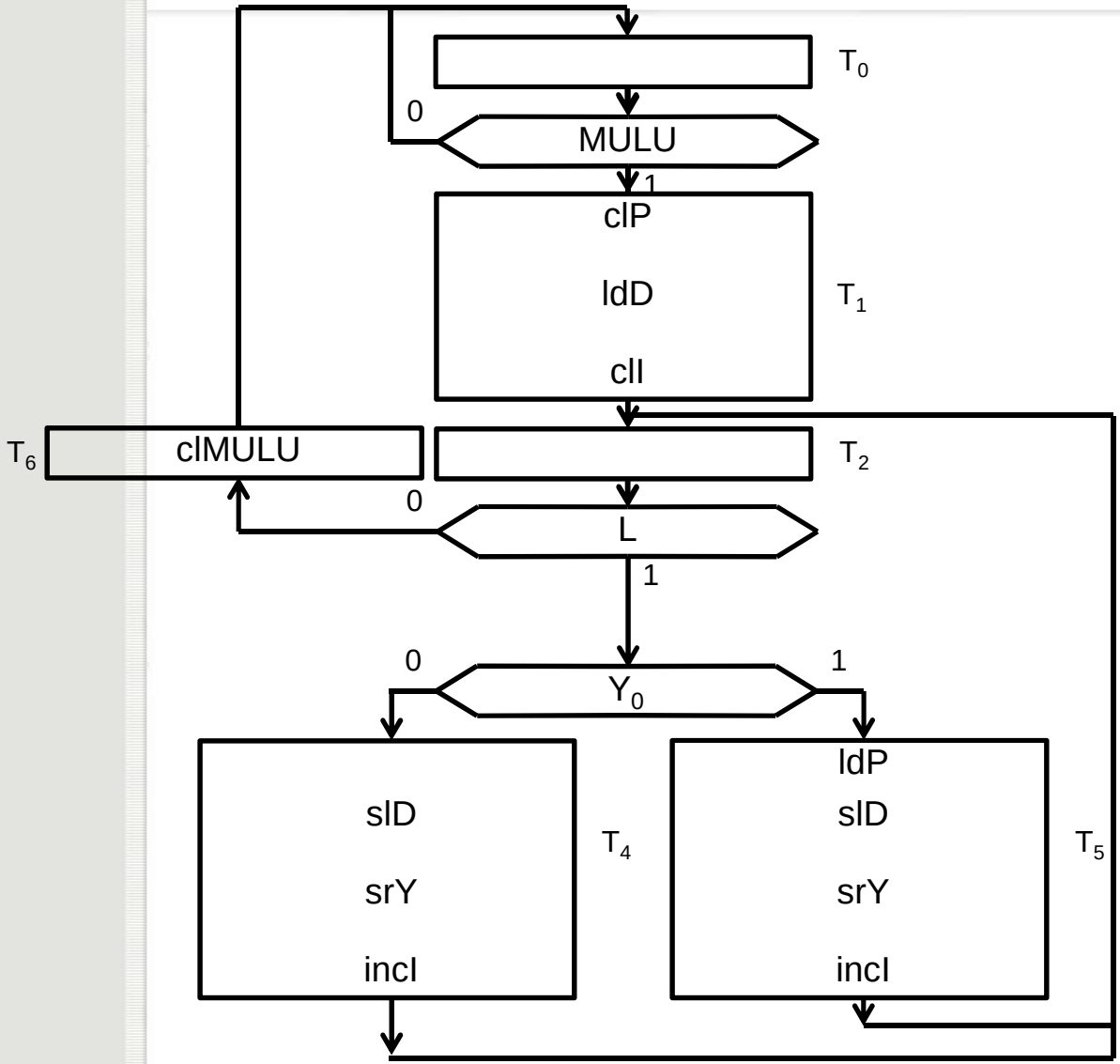
$$ldCNT = T_0 * \overline{MULU} + T_2 + T_4 + T_5 + T_6$$

$$CNT_2 = T_2$$

$$CNT_1 = T_2 * \overline{L} + T_4 + T_5$$

$$CNT_0 = T_2 * L * Y_0$$

Операција множења – MULU v2’



Операција множења – MULU v2’

Сигнали операционе јединице

ldP = T_4

clP = T_1

ldD = T_1

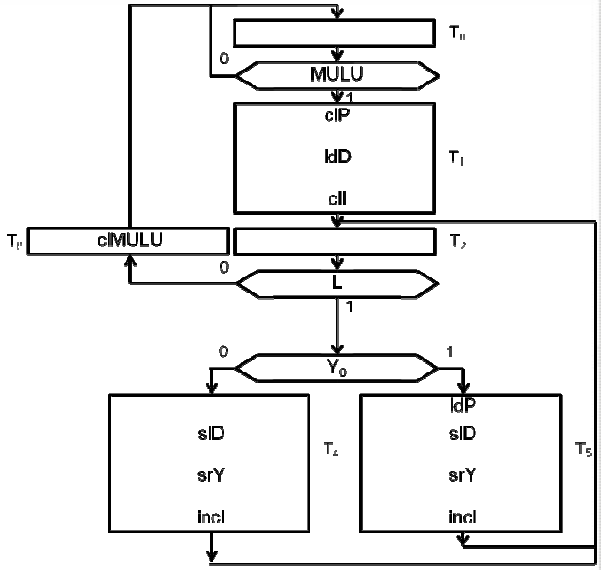
slD = $T_3 + T_4$

clI = T_1

incI = $T_3 + T_4$

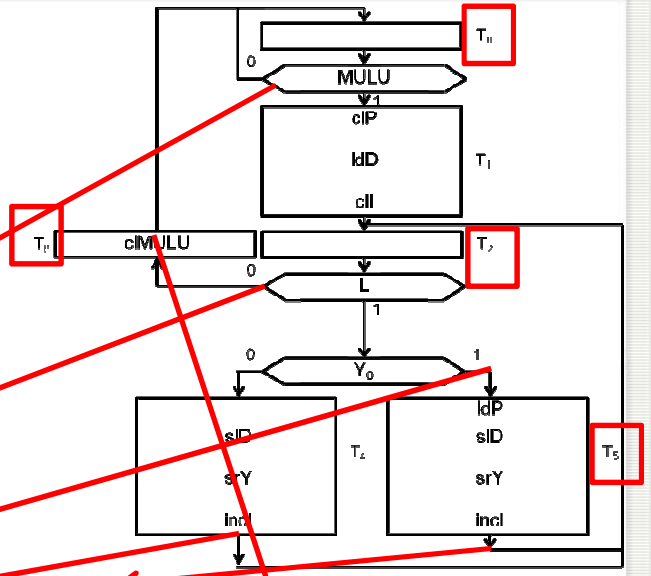
srY = $T_3 + T_4$

clMULU = T_5



Операција множења – MULU v2’

Сигнали управљачке јединице



$$ldCNT = T_0*\overline{MULU} + T_2*\overline{L} + T_2*L*Y_0 + T_4 + T_5 + T_6$$

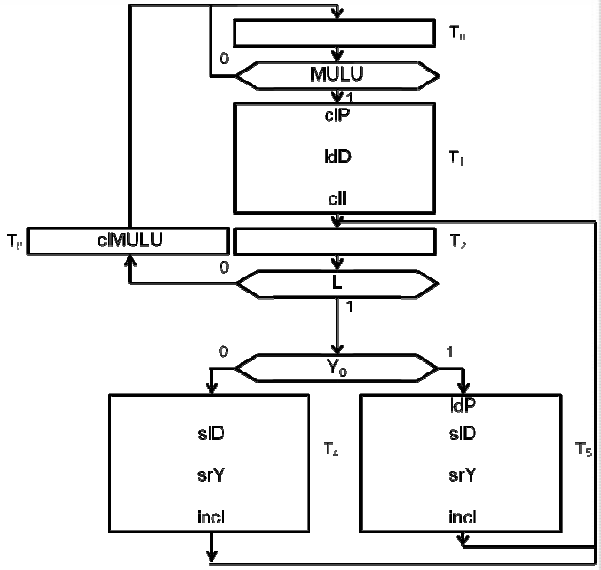
$$CNT_2 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_2*L*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

$$CNT_1 = T_0*\overline{MULU}*0 + T_2*\overline{L}*1 + T_2*L*Y_0*0 + T_4*1 + T_5*1 + T_6*0$$

$$CNT_0 = T_0*\overline{MULU}*0 + T_2*\overline{L}*0 + T_2*L*Y_0*1 + T_4*0 + T_5*0 + T_6*0$$

Операција множења – MULU v2’

Сигнали управљачке јединице



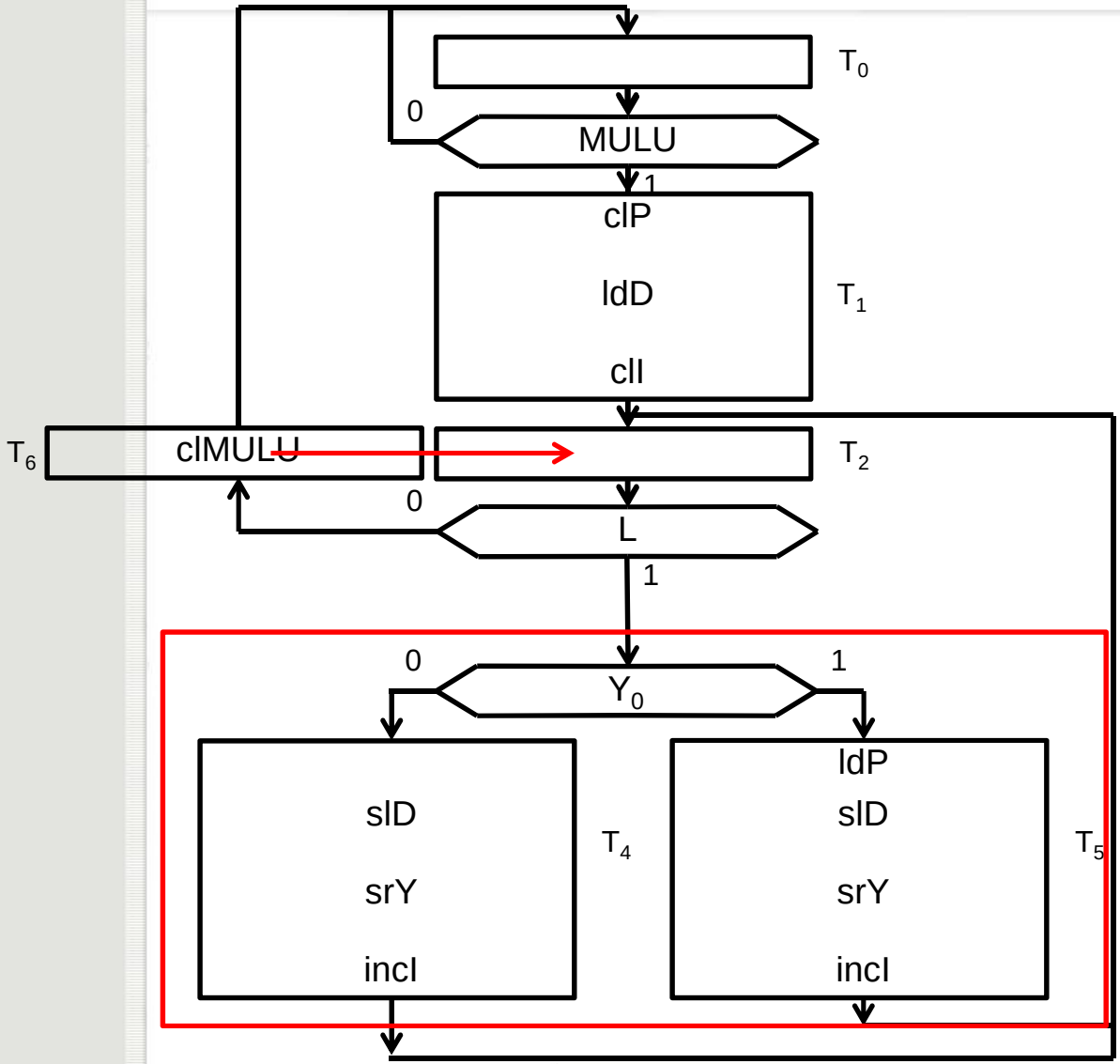
$ldCNT = T_0 * \overline{MULU} + T_2 * \overline{L} + T_2 * L * Y_0 + T_4 + T_5 + T_6$

$CNT_2 = T_2 * \overline{L} + T_2 * L * Y_0$

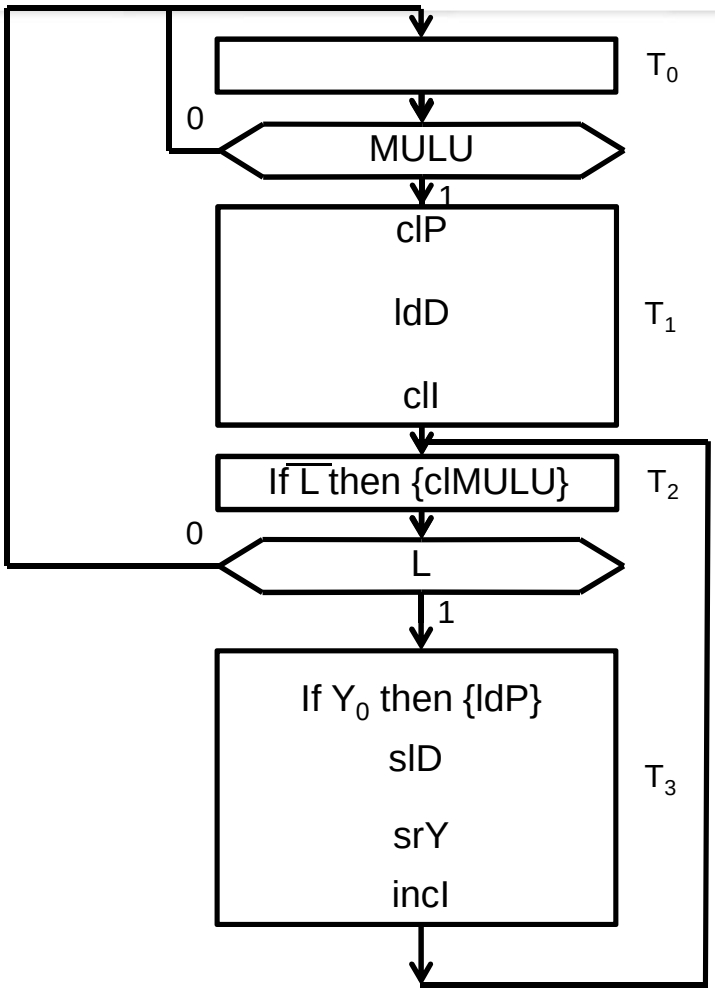
$CNT_1 = T_2 * \overline{L} + T_4 + T_5$

$CNT_0 = T_2 * L * Y_0$

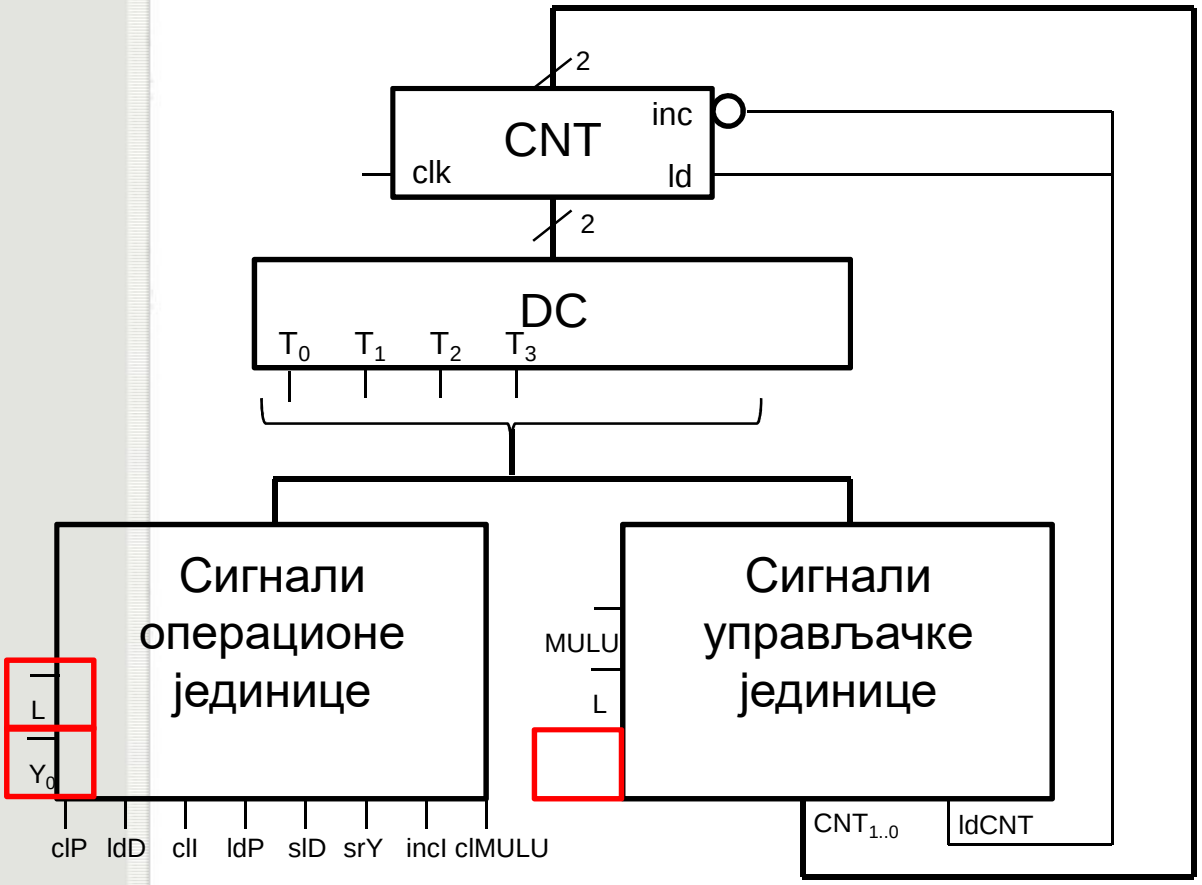
Операција множења – MULU v3

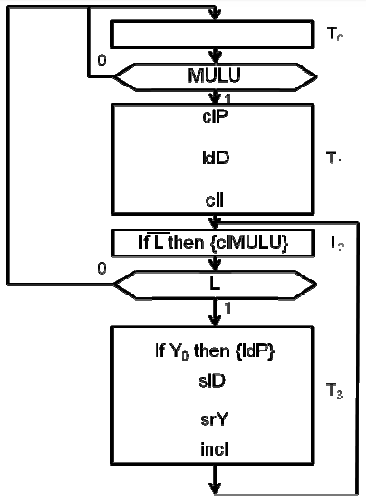


Операција множења – MULU v3



Операција множења – MULU





Операција множења – MULU v3

Сигнали операционе јединице

ldP = $T_3 * Y_0$

clP = T_1

ldD = T_1

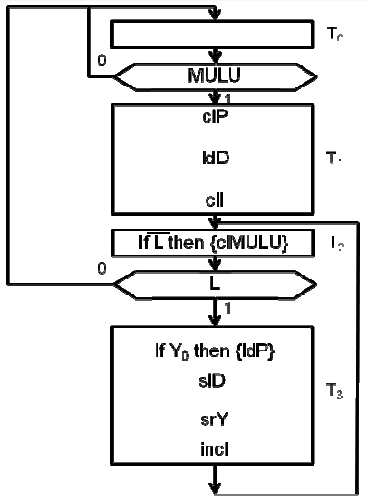
slD = T_3

clI = T_1

incI = T_3

srY = T_3

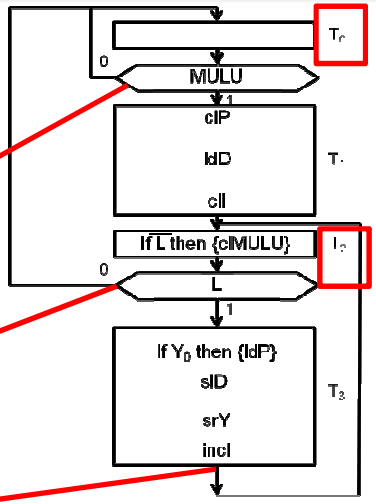
clMULU = $T_2 * \overline{L}$



Мрежа Милијевог типа

Операција множења – MULU v3

Сигнали управљачке јединице



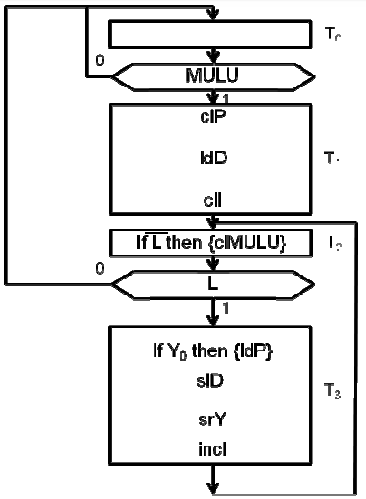
$$ldCNT = T_0 * \overline{MULU} + T_2 * \overline{L} + T_3$$

$$CNT_1 = T_0 * \overline{MULU} * 0 + T_2 * \overline{L} * 0 + T_3 * 1$$

$$CNT_0 = T_0 * \overline{MULU} * 0 + T_2 * \overline{L} * 0 + T_3 * 0$$

Операција множења – MULU v3

Сигнали управљачке јединице



$ldCNT = T_0 * \overline{MULU} + T_2 * \overline{L} + T_3$

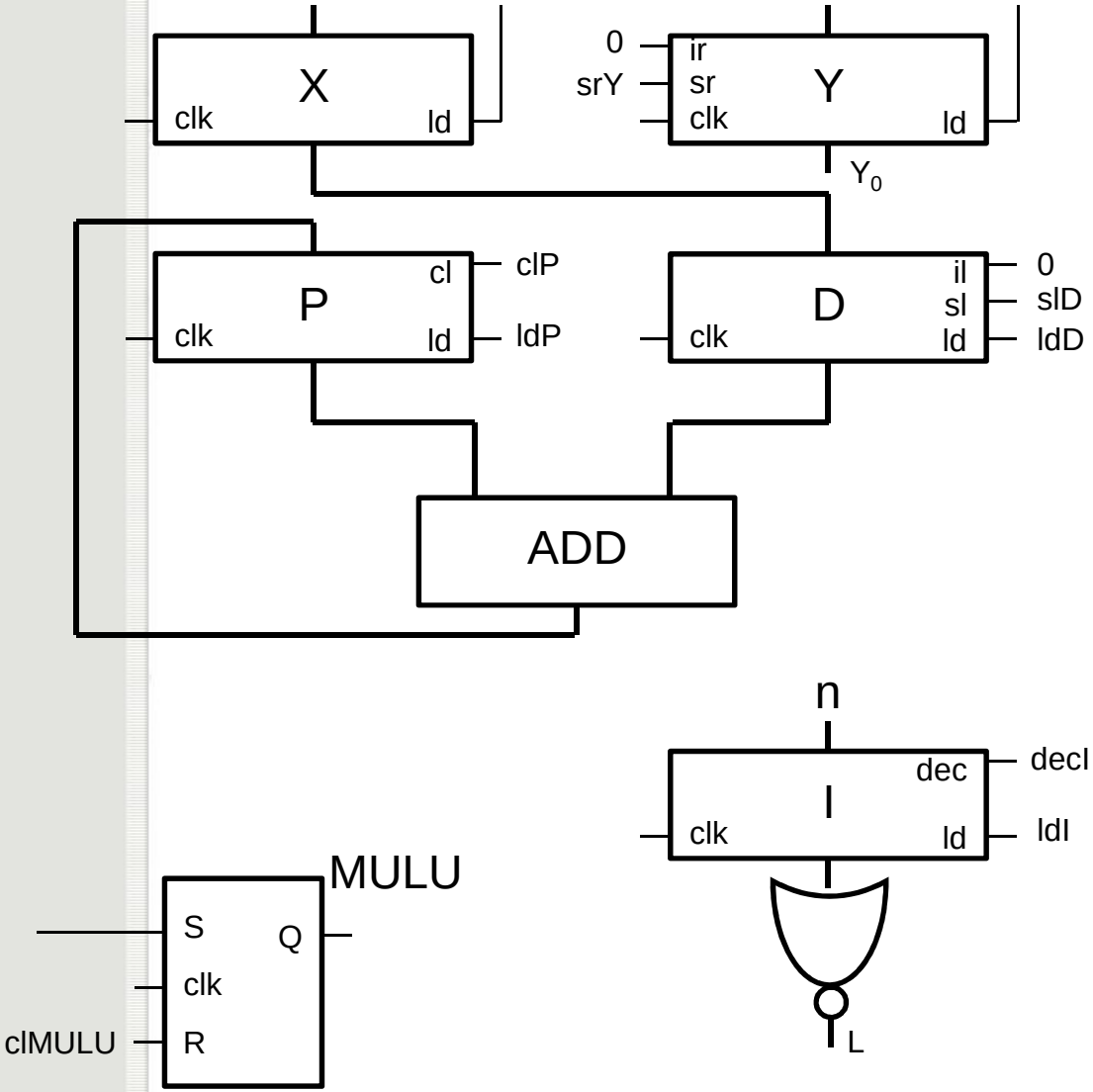
$CNT_1 = T_3$

$CNT_0 = 0$

Операција множења – MULU

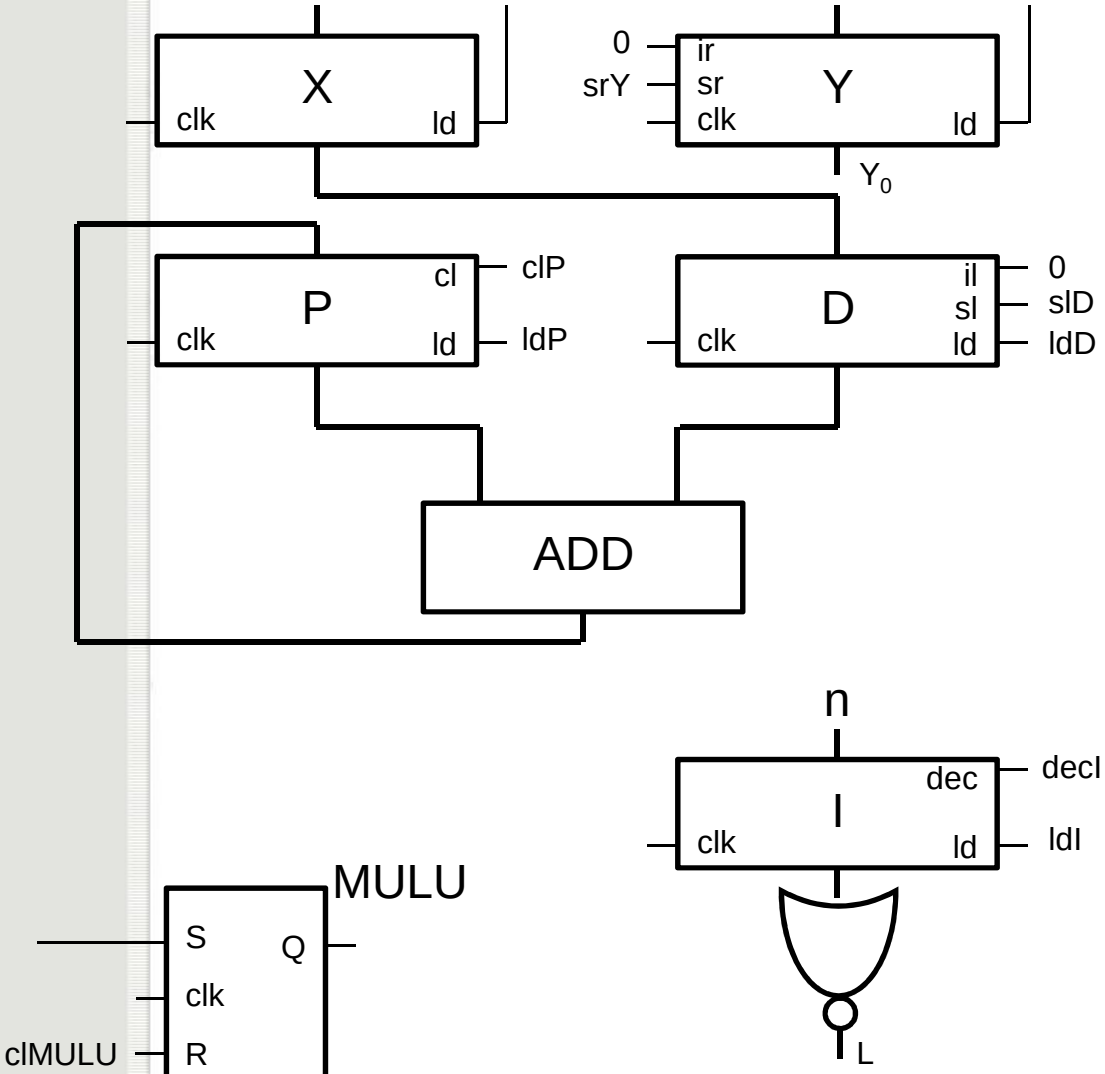
```
int mulu(int x, int y) {
   int p = 0;
   int d = x;
   for (int i = n; i != 0; i--) {
       if ((y & 1) == 0) {
              p = p;
       } else {
              p = p + d;
       }
       d = d << 1;
       y = y >> 1;
   }
   return p;
}
```

Операција множења – MULU




```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = n; i != 0; i--) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```

Операција множења – MULU




```
const int n = ...
int mulu(int x, int y) {
    int p = 0;
    int d = x;
    for (int i = n; i != 0; i--) {
        if ((y & 1) == 0) {
            p = p;
        } else {
            p = p + d;
        }
        d = d << 1;
        y = y >> 1;
    }
    return p;
}
```

Која је величина регистара и сабирача?

Питања?

Електротехнички Факултет
Универзитет у Београду
http://rti.etf.bg.ac.rs/rti/ir2ort2/index.html