EE 044252: Digital Systems and Computer Structure

Spring 2018

Lecture 7: Equivalence, Faults, Pipeline

EE 044252: Digital Systems and Computer Structure

Topic	wk	Lectures	Tutorials	Workshop	Simulation
Arch	1	Intro. RISC-V architecture	Numbers. Codes		
Comb	2	Switching algebra & functions	Assembly programming		
	3	Combinational logic	Logic minimization	Combinational	
	4	Arithmetic. Memory	Gates		Combinational
Seq	5	Finite state machines	Logic		
	6	Sync FSM	Flip flops, FSM timing	Sequential	Sequential
	7	FSM equiv, scan, pipeline	FSM synthesis		
	8	Serial comm, memory instructions	Serial comm, pipeline		
µArch	9	Function call, single cycle RISC-V	Function call		
	10	Multi-cycle RISC-V	Single cycle RISC-V		Multi-cycle
	11	Interrupts, pipeline RISC-V	Multi-cycle RISC-V		
	12	Dependencies in pipeline RISC-V	Microcode, interrupts		
	13		Depend. in pipeline RISC-V		

Agenda

- Equivalence of States and Machines
- Fault Detection in Sequential Systems
- Pipeline

שקילות מצבים וצמצום מכונות

- לעיתים קרובות, תכנון המכונה מתוך סיפור המעשה מביא להגדרת מצבים יתירים (redundant states) : הפונקציה שהם ממלאים ניתנת להשגה באמצעות מצבים אחרים

- כיוון שמספר רכיבי הזיכרון הדרוש למימוש המכונה גדל עם מספר מצביו (אם n הוא מספר המצבים, נדרשים $\lceil \log n \rceil$ רכיבי זיכרון), צמצום מספר המצבים יביא למימוש זול יותר ויאפשר השוואה בין מכונות

- <u>המטרה:</u> בהנתן מכונה סופית, מצא מכונה המבצעת אותה משימה בדיוק (=עבור כל קלט תפיק אותו פלט) בעלת מינימום מספר מצבים

הגדרות

- הגדרה: מצבים בני-הפרדה
  - שני מצבים S_i ו- S_j של מכונה M נקראים <u>בני-הפרדה</u> (distinguishable), אם קיימת סדרת כניסה אחת לפחות (סדרת-הפרדה) של M, המספקת יציאות שונות עבור המצבים ההתחלתיים S_i ו- S_j
- הגדרה: מצבים k-בני-הפרדה
  - שני מצבים (S_i ו- S_j) ייקראו <u>k –בני-הפרדה</u> (k-distinguishable), אם קיימת עבורם סדרת הפרדה באורך k

דוגמה

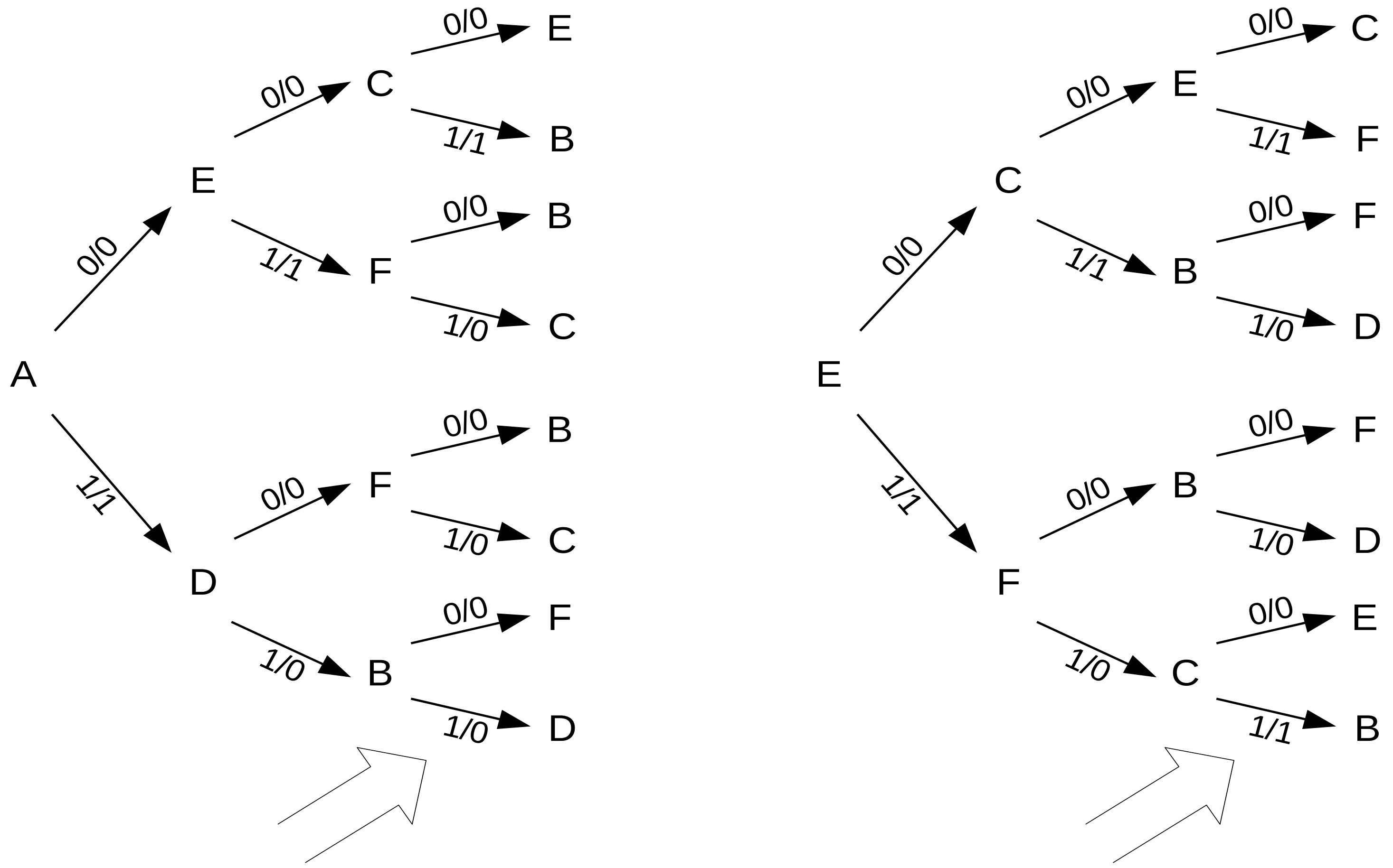
נתונה המכונה $M1$:

PS	$NS,\ z$	
	$x=0$	$x=1$
A	$E,0$	$D,1$
B	$F,0$	$D,0$
C	$E,0$	$B,1$
D	$F,0$	$B,0$
E	$C,0$	$F,1$
F	$B,0$	$C,0$

- (A ,B) הם 1 - בני-הפרדה, שכן תחת הקלט 1 תפיק M1 פלט 1 ממצב A ופלט 0 ממצב B

PS	NS, z	
	x=0	*x=1*
A	*E,0*	*D,1*
B	*F,0*	*D,0*
C	*E,0*	*B,1*
D	*F,0*	*B,0*
E	*C,0*	*F,1*
F	*B,0*	*C,0*

- (A, E) הם 3 – בני הפרדה:



- אין אף סדרה באורך 2 המפרידה בין מצבים אלה, אך הסדרה 111 נותנת יציאה 100 מ- A, ו- 101 מ- E. זו סדרת ההפרדה היחידה באורך 3.

הגדרה : מצבים שקולים

שני מצבים S_i ו-S_j של מכונה M נקראים שקולים (equivalent) אם כל סדרת-כניסה אפשרית של M מפיקה אותה סדרת יציאה, בין אם המצב ההתחלתי הוא S_i או S_j

נסמן $S_i \equiv S_j$

$\equiv$ הוא יחס שקילות. יחס שקילות מקיים את שלוש התכונות הבאות:

רפלקסיביות (1) $S_i \equiv S_i$

סימטריות (2) $S_i \equiv S_j \Rightarrow S_j \equiv S_i$

טרנזיטיביות (3) $S_i \equiv S_j, \quad S_j \equiv S_k \Rightarrow S_i \equiv S_k$

הגדרות

- יחס שקילות מחלק קבוצה (במקרה זה קבוצת המצבים של המכונה) למחלקות שקילות
- כל חברי אותה מחלקה שקולים זה לזה, ואינם שקולים לאף חבר של אף מחלקה אחרת
- איחוד כל המחלקות נותן את כל הקבוצה, וחיתוך כל שתי מחלקות הוא כמובן קבוצה ריקה (כלומר המחלקות זרות הדדית)

S_i ו-S_j שקולים $\Longleftrightarrow$ S_i ו-S_j אינם בני-הפרדה

נגדיר גם:

מצבים S_i ו- S_j הם k-שקולים $\Longleftrightarrow$ S_i ו- S_j אינם – k בני-הפרדה

k -שקילות אף הוא יחס שקילות, ומתקיים:

S_i ו- S_j שקולים $\Leftrightarrow$ S_i ו- S_j k – שקולים לכל k

וכן מתקיים:

S_i ו- S_j k-שקולים $\Leftarrow$ S_i ו-S_j r – שקולים לכל $r < k$

- את הגדרת השקילות ניתן להרחיב לשני מצבים S_i ו-S_j משתי **מכונות שונות** M1 ו-M2 עם אותו אלפאבית כניסה

האלגוריתם של Moore לצמצום מכונה

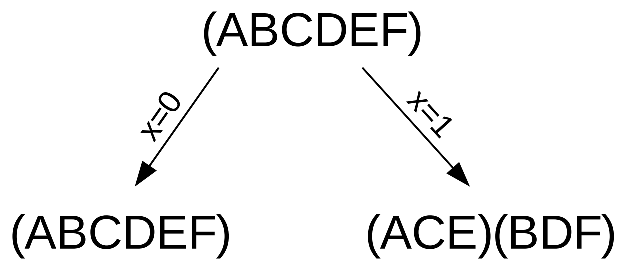
	NS, z	
PS	x=0	x=1
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

- נפתח בקבוצת כל המצבים, שהם 0 – שקולים

P – חלוקת השקילות

$$P_0 = (ABCDEF)$$

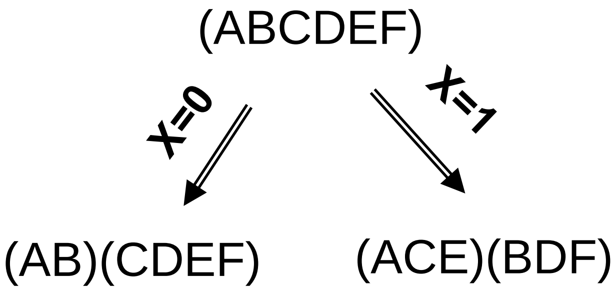
- נבצע חלוקה על P_0 למצבים 1 – שקולים



- המצבים ב- (ACE) נותנים אותה יציאה (0) עבור כניסה 0, ואותה יציאה (1) עבור כניסה 1
- כמו כן, המצבים ב- (BDF) נותנים אותה יציאה (0) עבור כניסה 0 או 1
- לכן כל המצבים ב- (ACE) הם 1 – בני-הפרדה מהמצבים ב- (BDF)
- סדרת ההפרדה בין (ACE) לבין (BDF) היא X=1. בדוגמא אין סדרת הפרדה אחרת. החלוקה החדשה היא

$$P_1 = \overset{\text{מחלקה}}{(ACE)}\,\overset{\text{מחלקה}}{(BDF)}$$

הגדרות: מצב עוקב



דוגמא אחרת: מהי החלוקה P_1 המתקבלת מ-P_0
כתוצאה מחלוקה כזו למצבים 1-שקולים?

הגדרות:

מצב 0-עוקב (0-successor) של S_i – המצב שעוברים אליו מ- S_i בגין כניסה *0*
מצב 1-עוקב (1-successor) של S_i – המצב שעוברים אליו מ- S_i בגין כניסה *1*

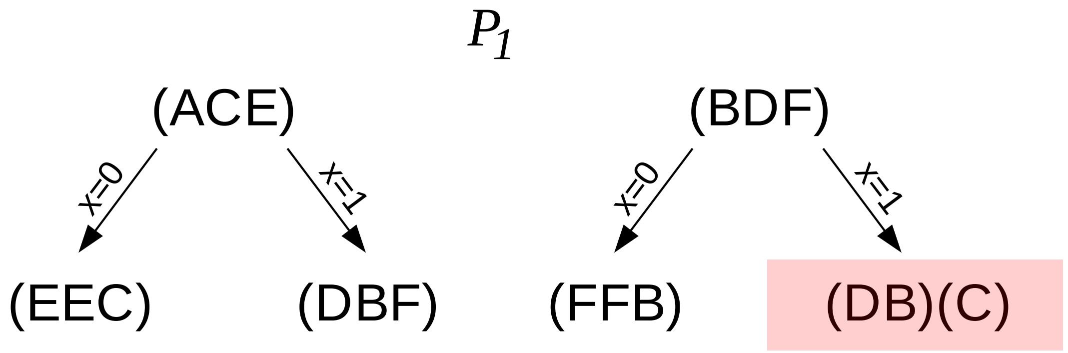
מצב X-עוקב (X-successor) של מצב S_i – מצב שניתן להגיע אליו מ- S_i תחת סדרת הכניסה X

המשך האלגוריתם (לכל שלב החל מהשני והלאה)

נמצא את מחלקות המצבים ה-k-שקולים, שהם:

- (k-1) – שקולים
- גם המצבים העוקבים שלהם (בהתאמה) הינם (k-1) – שקולים
- נקבל חלוקה שהיא עידון של (כלומר לא ניתן לאחד מחלקות
- קודמות או לערבב ביניהם; מותר רק לפצל מחלקות):

PS	x=0	x=1
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0



הערה: כעת החיצים מסמנים את המעבר ממצב למצב העוקב שלו

$$
\begin{tikzcd}
 & (ACE) \arrow[dl, "x=0"'] \arrow[dr, "x=1"] & & & (BDF) \arrow[dl, "x=0"'] \arrow[dr, "x=1"] & \\
(EEC) & & (DBF) & (FFB) & & (DB)(C)
\end{tikzcd}
$$

	$NS,\ z$	
PS	**x=0**	**x=1**
A	**E,0**	**D,1**
B	**F,0**	**D,0**
C	**E,0**	**B,1**
D	**F,0**	**B,0**
E	**C,0**	**F,1**
F	**B,0**	**C,0**

ולכן:

$$P_2=(ACE)(BD)(F)$$

סדרת ההפרדה בין (BD) לבין (F) היא X=11

כך נמשיך הלאה:

$$P_3=(AC)(E)(BD)(F)$$

$$P_4=(AC)(E)(BD)(F)$$

עד אשר **חלוקת שקילות** $P_{k+1}=P_k$ ("תנאי עצירה")

משפט: חלוקת שקילות

משפט:

חלוקת השקילות P_k היא יחידה

הוכחה:

- בשלילה, נניח כי קיימות שתי חלוקות שקילות P_a ו-P_b שונות זו מזו
- אזי, קיימים שני מצבים S_i ו-S_j הנמצאים באותה מחלקה בחלוקה אחת (נניח P_a) ובמחלקות שונות בחלוקה האחרת (נניח P_b)
- מהחלוקה P_b נובע כי קיימת סדרת כניסה המפרידה בין S_i ו-S_j
- מכאן ש- S_i ו-S_j אינם יכולים להיות באותה מחלקה ב-P_a

משפט (תכונת העצירה של אלגוריתם Moore):

אם S_i ו- S_j שני מצבים בני-הפרדה במכונה M בעלת n מצבים, כי אז קיימת סדרת-הפרדה באורך של $n-1$ לכל היותר

הוכחה:

באלגוריתם של Moore, אם $i<j$, P_i מכילה (לפחות) מחלקה אחת יותר מ-P_j(פרט לצעד האחרון).

המשפט נובע מכך שמספר המחלקות הוא n לכל היותר.

שקילות בין מכונות

- הגדרה:
  - שתי מכונות M ו-'M תיקראנה שקולות אם לכל מצב ב-M קיים מצב שקול מתאים ב- 'M, ולהיפך
- המושג שקילות בין מצבים הוא כמו במכונה בודדת:
  - משני מצבים שקולים תתקבל אותה סדרת פלט תחת אותה סדרת קלט
- בהינתן מכונה M, נמצא מכונה M^* השקולה ל- M ובעלת מספר מצבים מינימלי
  - M^* תיקרא הצורה <u>המינימלית</u>, או <u>מצומצמת</u> של M
- כל מצב במכונה M^* יתאים <u>למחלקת שקילות</u> של מצבים בחלוקת השקילות של M, כיוון שחלוקת השקילות יחידה, ולכן לא ייתכן כי מצב ב- M^* יהיה שקול לשני מצבים לא-שקולים ב- M!

$$P_3 = (AC)(E)(BD)(F)$$

נחזור למכונה M_1 :

PS	NS, z	
	$x=0$	$x=1$
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

נחליף כל מצב בטבלת המצבים במחלקת השקילות שלו:

PS	NS, z: x=0	NS, z: x=1
(AC)	(E),0	(BD),1
(BD)	(F),0	(BD),0
(AC)	(E),0	(BD),1
(BD)	(F),0	(BD),0
(E)	(AC),0	(F),1
(F)	(BD),0	(AC),0

רואים שיש שתי שורות מיותרות.
נכנה את מחלקות השקילות כדלקמן:

$$P_3 = (\overset{\alpha}{AC})(\overset{\beta}{E})(\overset{\gamma}{BD})(\overset{\delta}{F})$$

PS	NS, z: x=0	NS, z: x=1
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

ונקבל את טבלת המצבים הבאה עבור M_1^* :

	PS	NS, z x=0	NS, z x=1
(AC) $\longrightarrow$	α	$\beta,0$	$\gamma,1$
(E) $\longrightarrow$	β	$\alpha,0$	$\delta,1$
(BD) $\longrightarrow$	γ	$\delta,0$	$\gamma,0$
(F) $\longrightarrow$	δ	$\gamma,0$	$\alpha,0$

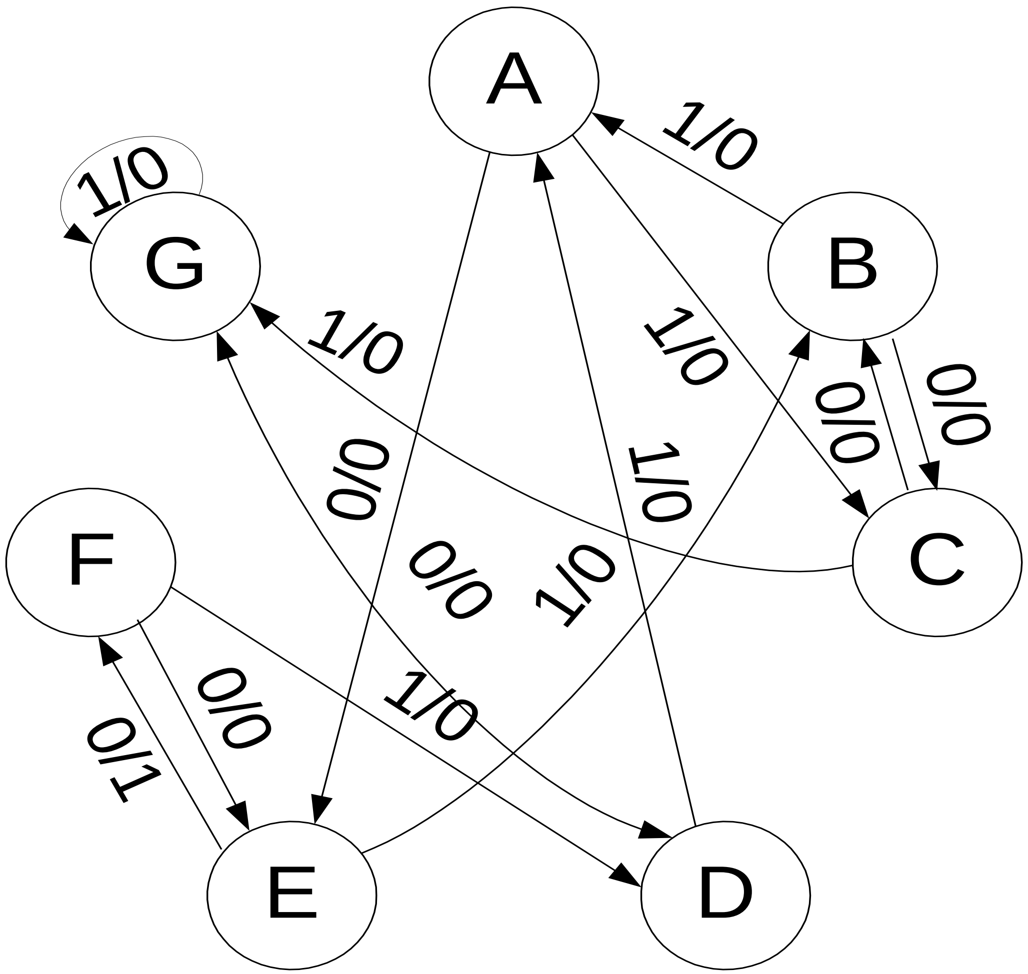
$$P_3 = (\overset{\alpha}{AC})(\overset{\beta}{E})(\overset{\gamma}{BD})(\overset{\delta}{F})$$

המכונה M_1^* שקולה למכונה M_1 .

PS	NS, z x=0	NS, z x=1
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

דוגמה נוספת- מכונה $M2$:

PS	NS, z (x=0)	NS, z (x=1)
A	$E,0$	$C,0$
B	$C,0$	$A,0$
C	$B,0$	$G,0$
D	$G,0$	$A,0$
E	$F,1$	$B,0$
F	$E,0$	$D,0$
G	$D,0$	$G,0$



סדרת חלוקות השקילות היא (בכל שלב מצויינת סדרת הפרדה בין שתי מחלקות):

$$P_o = (ABCDEFG)$$

$$P_1 = (ABCDFG)^o (E)$$

$$P_2 = (AF)^o (BCDG)(E)$$

$$P_3 = (AF)(BD)^1 (CG)(E)$$

$$P_4 = (A)^1 (F)(BD)(CG)(E)$$

$$P_5 = (A)(F)(BD)(CG)(E)$$

	$NS, \quad z$	
PS	x=0	x=1
A	E,0	C,0
B	C,0	A,0
C	B,0	G,0
D	G,0	A,0
E	F,1	B,0
F	E,0	D,0
G	D,0	G,0

PS	NS, z (x=0)	NS, z (x=1)
A	E,0	C,0
B	C,0	A,0
C	B,0	G,0
D	G,0	A,0
E	F,1	B,0
F	E,0	D,0
G	D,0	G,0

$$P_4 = (A)(F)(BD)(CG)(E)$$

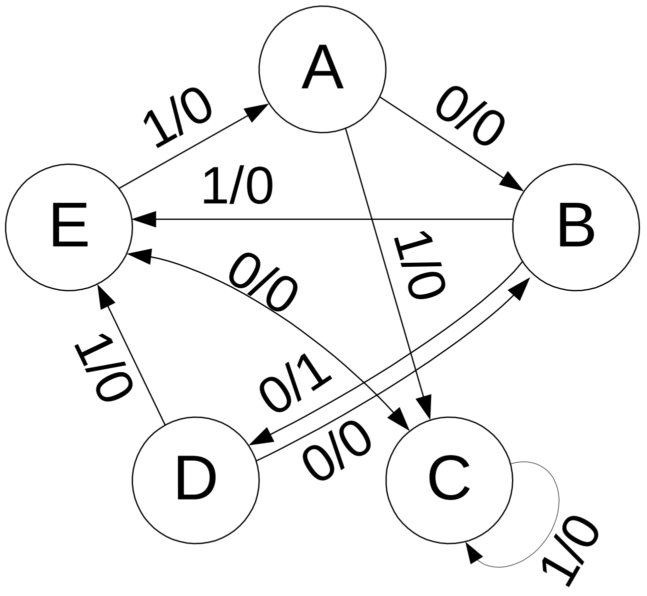
מכונה מצומצמת M_2^* :

	PS	NS, z (x=0)	NS, z (x=1)
(A)	$\longrightarrow \alpha$	$\varepsilon,0$	$\delta,0$
(F)	$\longrightarrow \beta$	$\varepsilon,0$	$\gamma,0$
(BD)	$\longrightarrow \gamma$	$\delta,0$	$\alpha,0$
(CG)	$\longrightarrow \delta$	$\gamma,0$	$\delta,0$
(E)	$\longrightarrow \varepsilon$	$\beta,1$	$\gamma,0$

	PS	NS, z: x=0	x=1
(A)	$\longrightarrow \alpha$	ε,0	δ,0
(F)	$\longrightarrow \beta$	ε,0	γ,0
(BD)	$\longrightarrow \gamma$	δ,0	α,0
(CG)	$\longrightarrow \delta$	γ,0	δ,0
(E)	$\longrightarrow \varepsilon$	β,1	γ,0

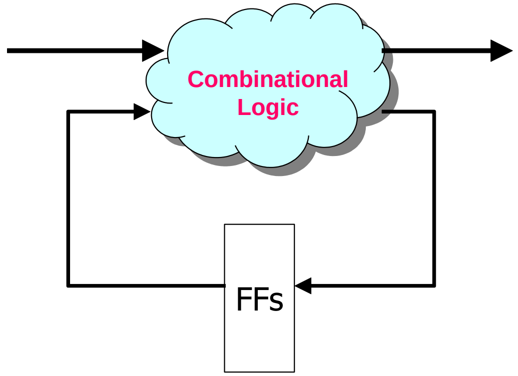
- שתי מכונות זהות הנבדלות רק בשמות המצבים נקראות **איזומורפיות** (שוות צורה)
- כדי לוודא ששתי מכונות הן איזומורפיות זו לזו נגדיר צורה סטנדרטית או קנונית, בה נתחיל ממצב כלשהו ושמות המצבים ייקבעו לפי סדר הופעתם
- אם למשל נבחר את האותיות A,B,C,D,E במקום האותיות α,ε,δ,β,γ בהתאמה, נקבל מכונה איזומורפית לראשונה:

	PS	NS, z: x=0	x=1
$\alpha \longrightarrow$	A	B,0	C,0
$\varepsilon \longrightarrow$	B	D,1	E,0
$\delta \longrightarrow$	C	E,0	C,0
$\beta \longrightarrow$	D	B,0	E,0
$\gamma \longrightarrow$	E	C,0	A,0



Agenda

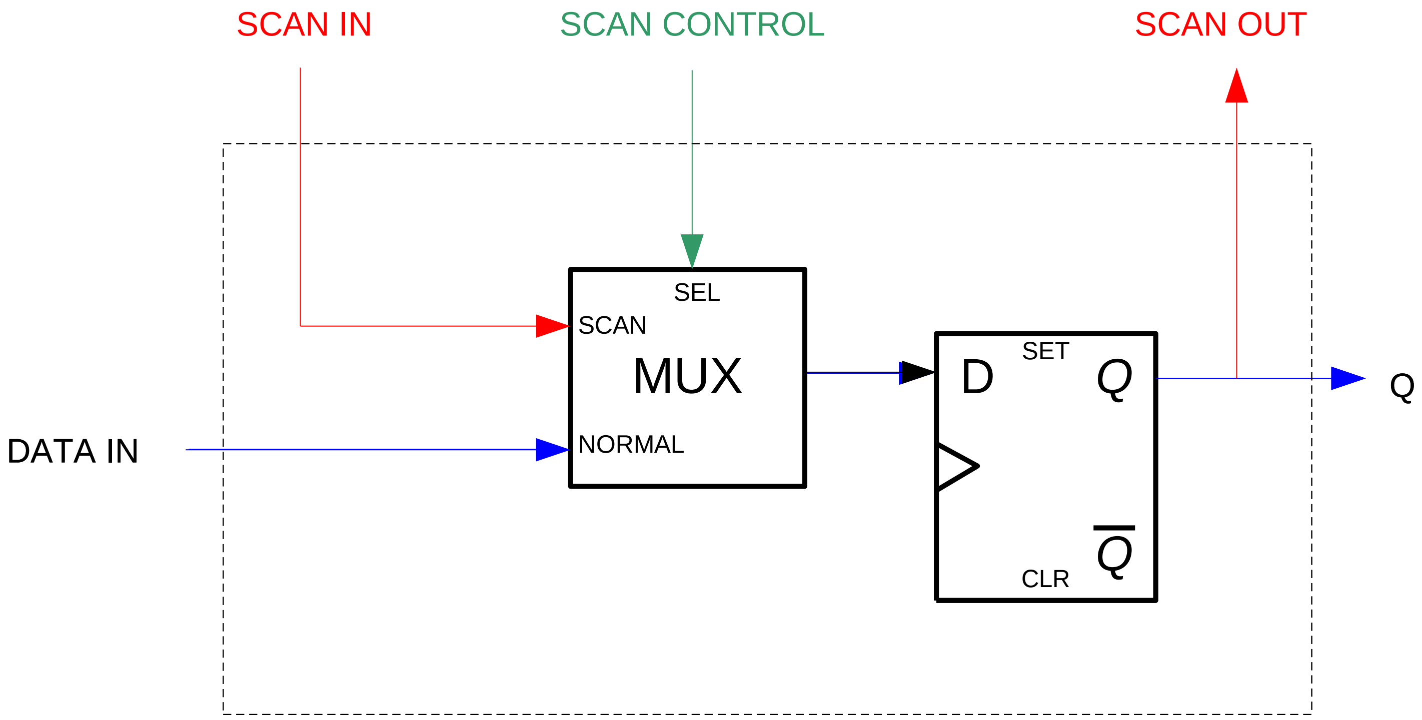
- Equivalence of States and Machines
- **Fault Detection in Sequential Systems**
- Pipeline



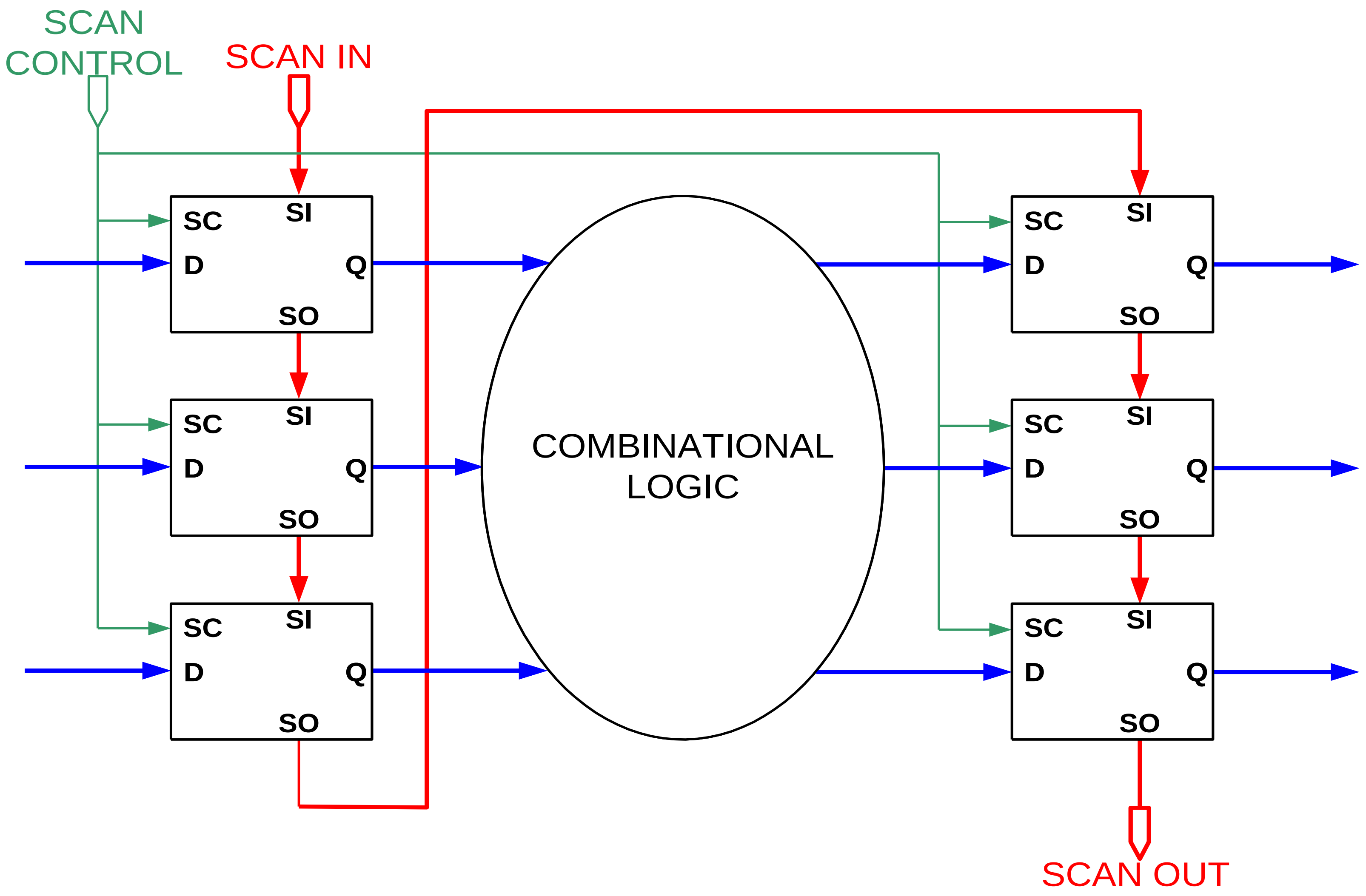
גילוי תקלות במערכות עקיבה

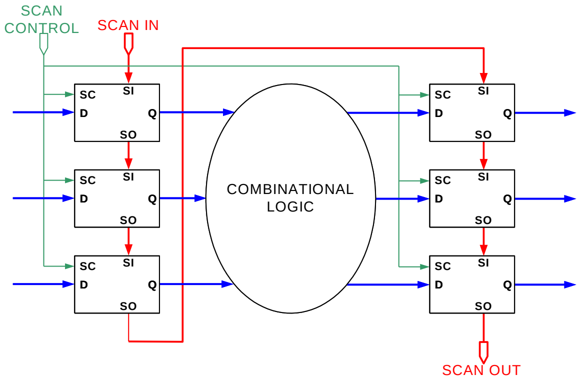
- בדקנו תקלות במערכת צירופית על ידי בדיקות וניסוי
- קשה יותר לבדוק מערכות עקיבה
- כדאי לפרק מערכות עקיבה למעגלים צירופיים ולרכיבי הזיכרון, ולבדוק כל אחד לחוד
- שיטת הסריקה (Scan):
  - החלף כל FF ב- Scanned FF
  - חבר את כל ה- Scanned FFs בשרשרת אחת, ההופכת אותם לרגיסטר הזזה אחד
  - חבר כניסת בקרה SCAN CONTROL לכל ה-FF-ים

Scanned FF


SCAN IN
SCAN CONTROL
SCAN OUT
SEL
SCAN
MUX
NORMAL
DATA IN
D
SET
Q
CLR
$\overline{Q}$
Q

מערכת עקיבה עם אפשרות סריקה

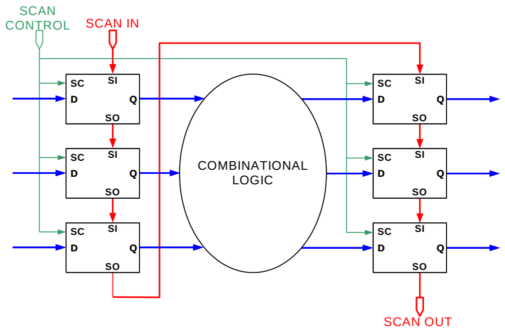




מהלך הסריקה

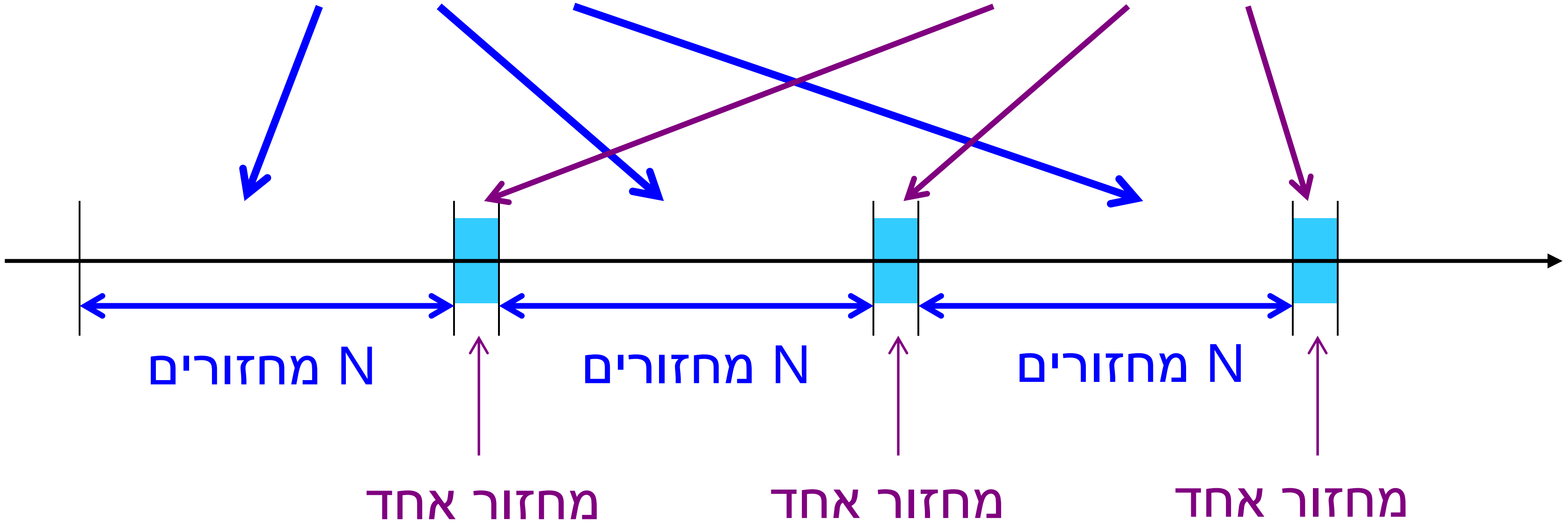
- בפעולה רגילה, SCAN CONTROL = 0 וה-FF מתנהגים כרגיל
- לצורך בדיקה, SCAN CONTROL = 1 ובמשך N מחזורי שעון מכניסים לכל FF את הכניסה הדרושה על מנת לבצע בדיקה אחת
- מעבירים את SCAN CONTROL שוב ל-0 למשך מחזור שעון אחד, מבצעים את הבדיקה, וכל ה-FF טוענים לתוכם את תוצאות הבדיקה
- במשך N מחזורי השעון הבאים SCAN CONTROL=1 ותוצאות הבדיקה נקראות החוצה ובו זמנית נטענות הכניסות הדרושות לבדיקה הבאה, וחוזר חלילה

מהלך הסריקה



קריאת N סיביות מ-N פלופים
והטענת N סיביות חדשות

מחזור חישוב אחד במעגלים הצירופיים,
טעינת כל ה-FF בתוצאות החישוב



גילוי תקלות בזיכרונות

- שיטת הסריקה איננה מתאימה לזיכרונות
- שיטה אפשרית אחרת היא לכתוב לכל תא בזיכרון, לקרוא אותו, ולהשוות את מה שנכתב למה שנקרא
  - יש צורך לחזור על כך עם תוכן שונה בכל פעם, בכדי לבדוק תקלות שונות התלויות **בתוכן הנכתב**. זהו תהליך ארוך ויקר.
- במקום זאת משתמשים במכונות מצבים המייצרות תוכן אקראי לכאורה, כותבות לזיכרון וקוראות בחזרה
  - משתמשים בקודים לגילוי שגיאות, הפוטרים את מכונת המצבים מלזכור מה נרשם לתוך הזיכרון
  - המכונה יכולה לחשב האם הייתה שגיאה (כלומר תקלה) על פי התוכן הנקרא

גילוי תקלות בזיכרונות (המשך)

- מכונות אלו בנויות יחד עם הזיכרון והפעלתן איננה דורשת מכשיר בדיקה חיצוני.
  שיטה זו קרויה
  Built In Self Test (BIST)
- ניתן להוסיף לכל זיכרון כמות קטנה של זיכרון רזרבי:
  - המכונה לגילוי התקלות יכולה לתכנת את הזיכרון כך שאיזור זיכרון שיש בו תקלה יוחלף בזיכרון רזרבי
  - שיטה זו קרויה Built In Self Repair (BISR) והיא נפוצה מאוד במוצרי זיכרון שונים

Agenda

- Equivalence of States and Machines
- Fault Detection in Sequential Systems
- **Pipeline**

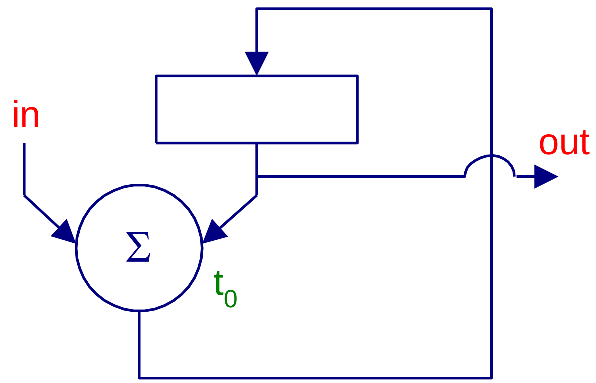
Pipeline

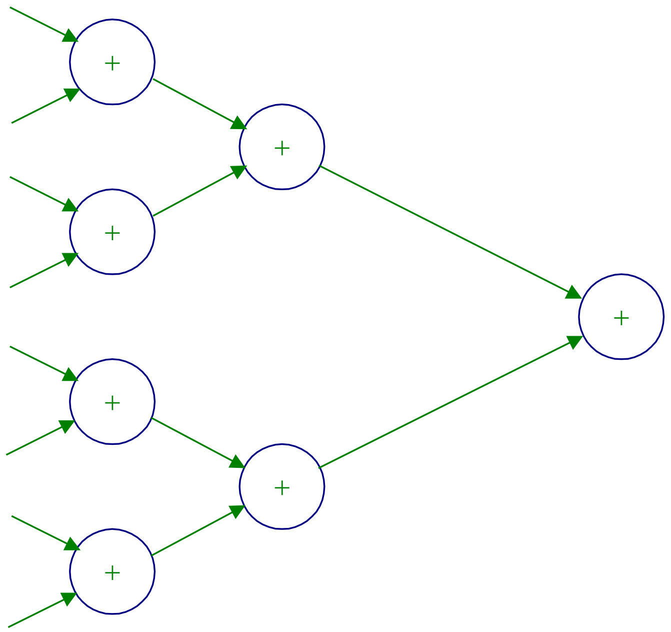
שני מדדים לביצועי מערכת ספרתית

- זמן שעובר מתחילת חישוב ועד סופו
  - **השהיה, Latency**
  - הרחבה של Propagation Delay למערכות כלליות
- קצב החישוב
  - מספר החישובים שניתן לעשות ביחידת זמן
  - **ספיקה, Throughput**

- **מדדים אפשריים נוספים: הספק, שטח, אמינות וכו'**

דוגמה להשהיה במימושים סדרתי וצירופי

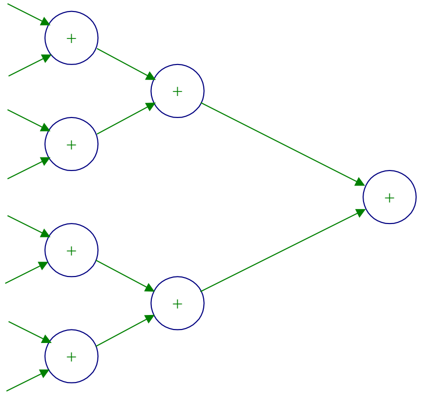




- המשימה: חיבור N מספרים
- מימוש סדרתי
  - על-ידי מסכם יחיד ורגיסטר
  - זמן לחיבור יחיד t_0
  - זמן כולל: $N \times t_0$
- שני ערוצים מקבילים
  - זמן כולל $(N/2 + 1) \times t_0$
- מימוש צירופי
  - עץ מסכמים בינארי
  - "עומק" העץ $\sim \log_2 N$
  - זמן כולל $\sim \log_2(N) \times t_0$

Throughput – ספיקה

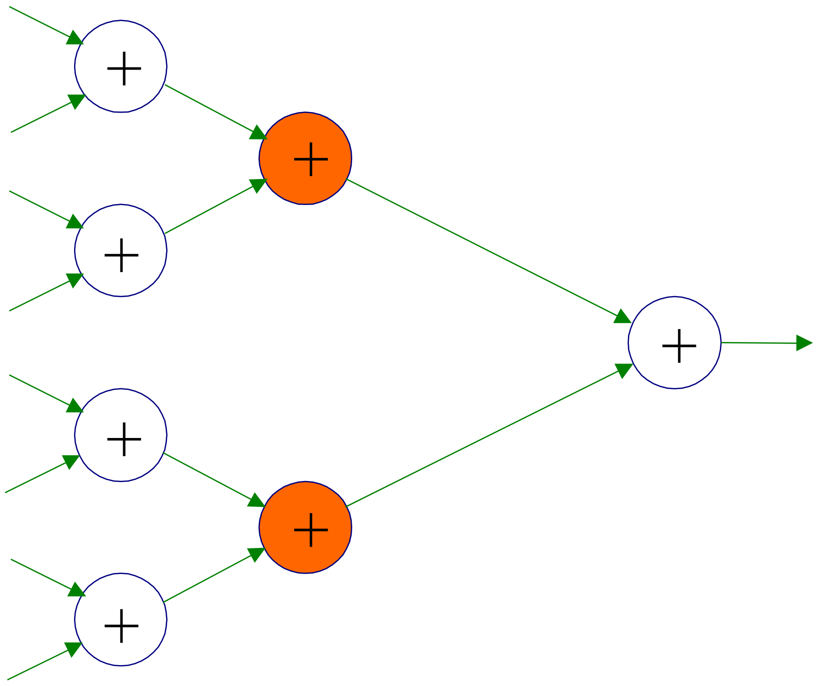
- מהו ה-Throughput של מערכת חישוב צירופית?
- דוגמה: עץ המסכמים
  - ההשהיה ~ $\log_2(N)\times t_0$
  - רק לאחר סיום החישוב ניתן לספק נתונים חדשים
  - לכן מספר קבוצות הנתונים ליחידת זמן:



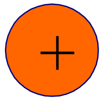
$$Throughput = \frac{1}{Latency} = \frac{1}{\log_2(N)\times t_0}$$

$$\underset{\text{Combinational Example}}{Throughput} = \frac{1}{3\times t_0}$$

אבטלת חמרה



- בעץ המסכמים, לאחר זמן t_0 השלב הראשון בעץ לא עושה דבר.
- האם ניתן לנצל טוב יותר את החמרה ?
- כן, אם :
  - יש לבצע הרבה חישובים זהים על נתונים שונים
  - ניתן לקבוע כרצוננו את קצב הגעת הנתונים
- כיצד? לאחר זמן t_0 נכניס נתונים חדשים
- בעיה : עץ המסכמים צירופי – אסור לשנות את כניסותיו עד לסיום החישוב הכולל


אדום ממאמץ :

שיפור ספיקה בעזרת Pipeline

Real-World Pipelines: Car Washes

Sequential

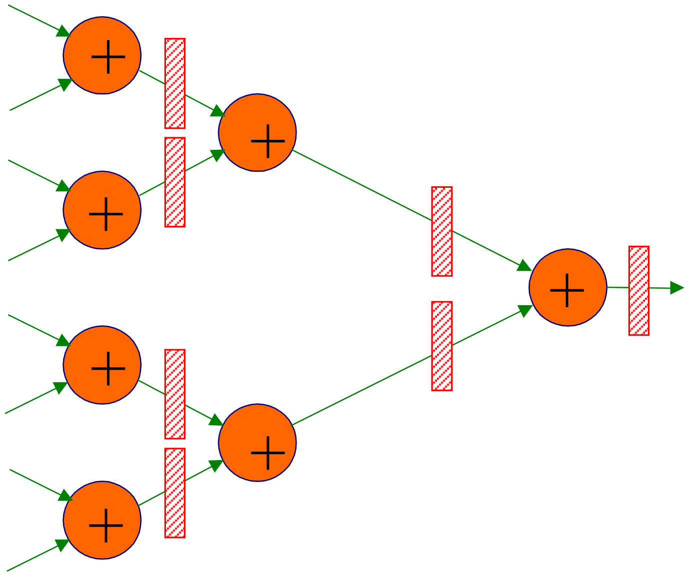
Parallel

Pipelined

- **Idea**
  - Divide process into independent stages
  - Move objects through stages in sequence
  - At any instant, multiple objects being processed

שיפור ספיקה בעזרת Pipeline

- פתרון: "נלכוד" את תוצאות הביניים באוגרים (רגיסטרים), ואז נוכל לשנות את הכניסה לפני שנגמר החישוב כולו

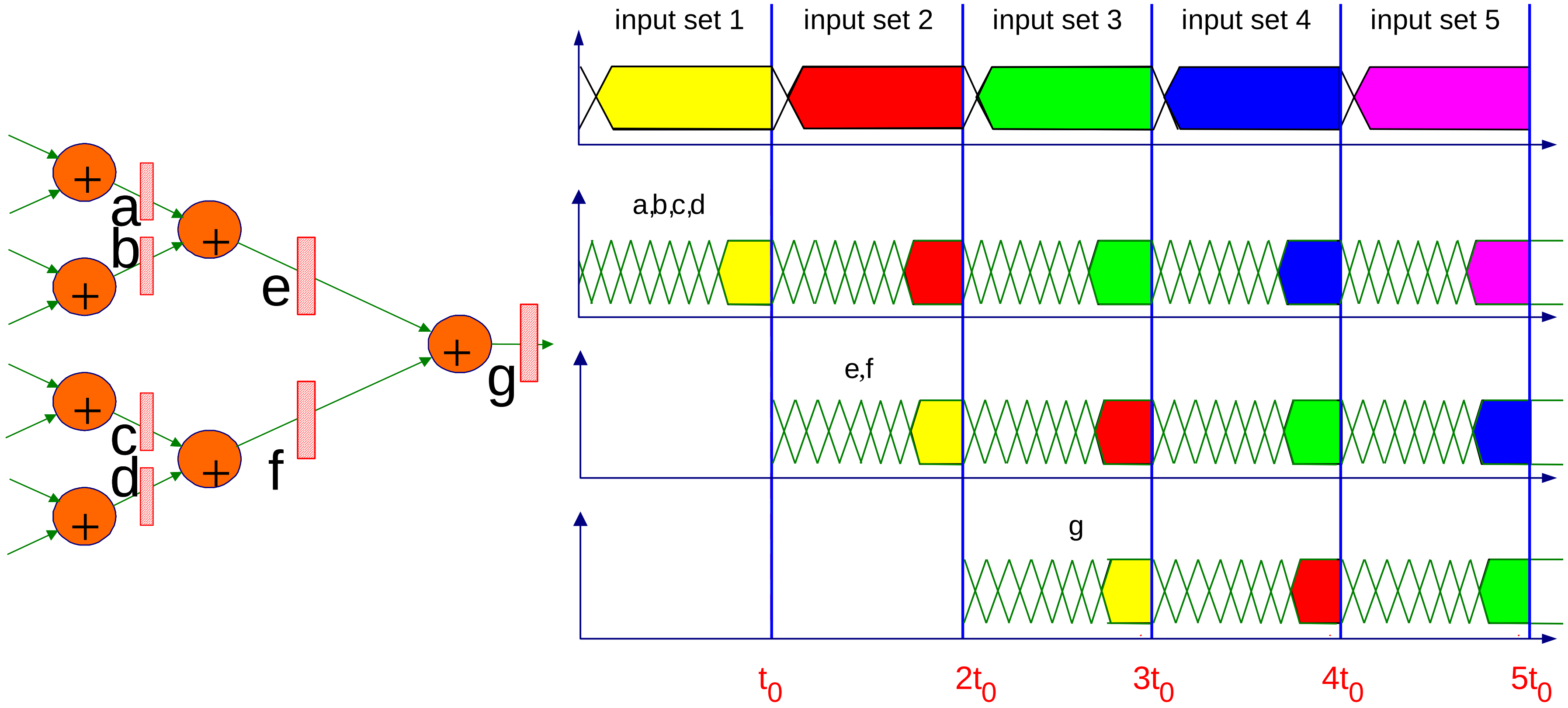


- לתהליך בו חישוב טורי עובד בו זמנית על מספר חישובים קוראים Pipeline (צינור)

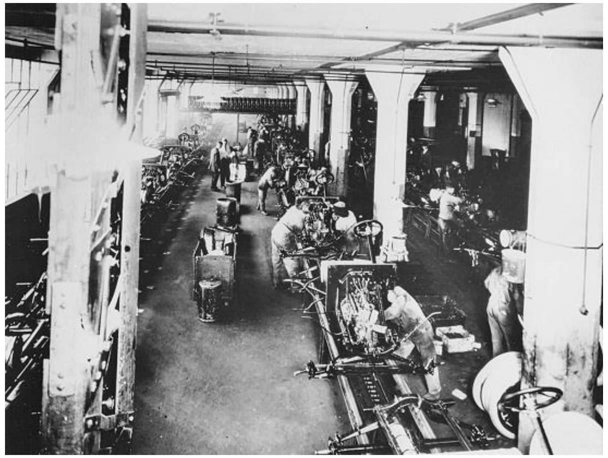
Pipeline תזמון

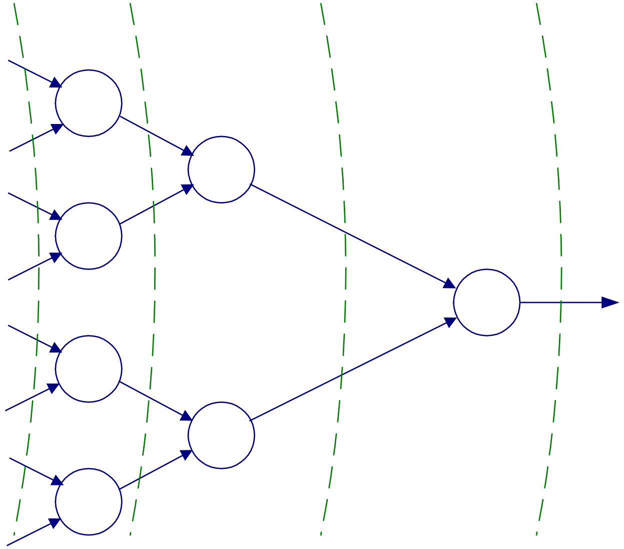
$$\underset{Pipeline}{\text{Throughput}} = \frac{1}{t_0}$$

- נבחר מחזור שעון t_0
- תוצאה חדשה יוצאת בכל מחזור
- ניצולת מכסימלית של המשאבים



אנלוגיות מועילות

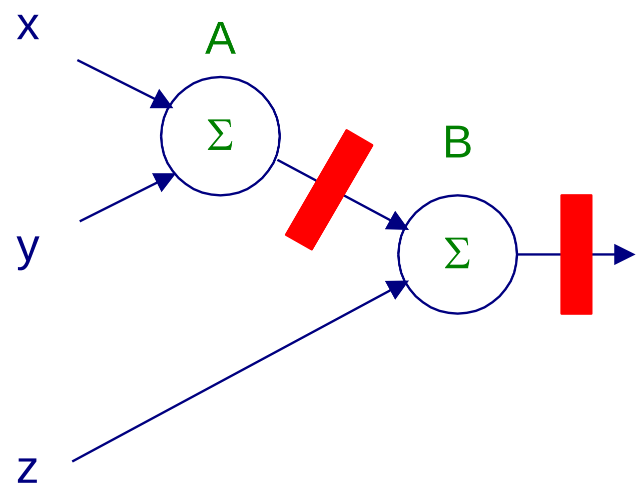

www.corporate.ford.com



- קו ייצור (הומצא ע"י הנרי פורד לפני 100 שנים)
  - קיימות מספר תחנות בקו
  - בכל תחנה עושים פעולה שונה, שהיא חלק מהעיבוד
  - הקו עובד בו-זמנית על מוצרים רבים
- גלי ים
  - תנועה מתמדת
  - הגל מתחיל לפני שהקודם לו הגיע לחוף
  - הגלים בים יכולים לעלות אחד על השני, בלוגיקה לא כדאי שזה יקרה...
- לא כל בעיה ניתנת לפתרון יעיל יותר בעזרת Pipeline
  - האם תמיד ניתן למקבל חישוב?

תכנון יחידת Pipeline

- תכנון אינטואיטיבי:
  - תכנן מעגל צירופי
  - הוסף אוגרים במקומות הדרושים עד להשגת ספיקה מכסימלית
- בעיה אפשרית: חוסר איזון
- דוגמה: חיבור 3 מספרים
  - נקבל $x_1+y_1+z_2$

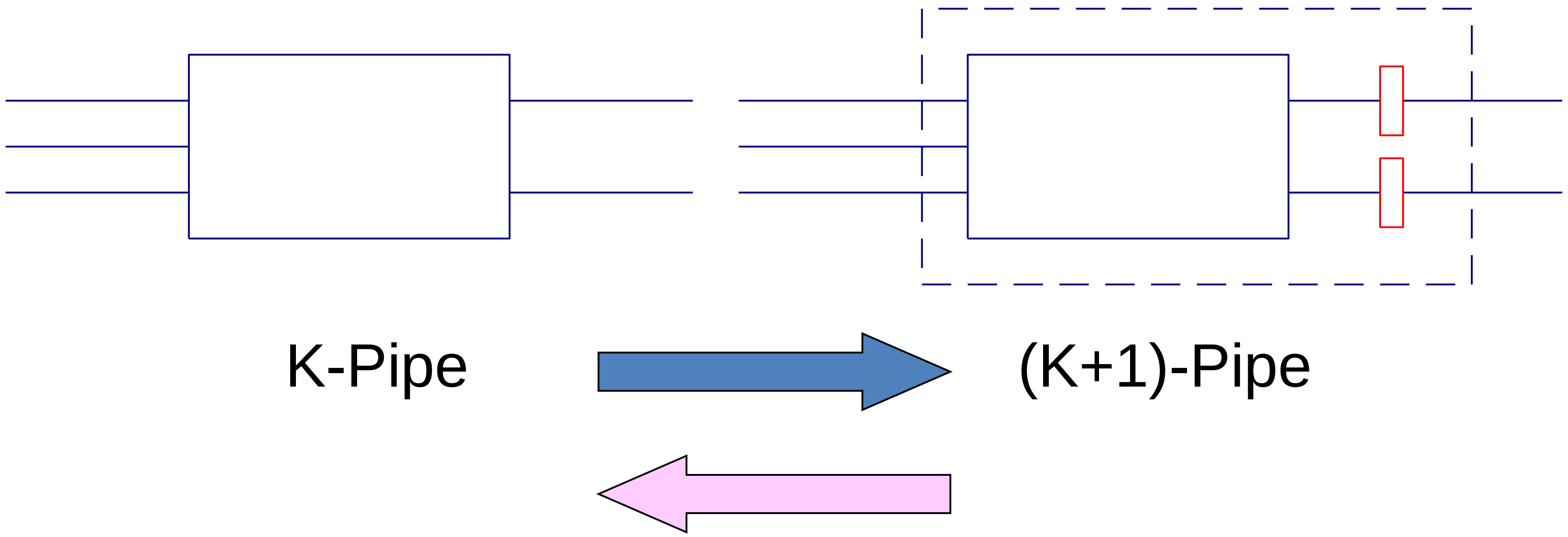


תכנון שיטתי של Pipeline

- הגדרה: K - Pipeline
  - מעגל לוגי ללא משוב
  - כולל רכיבים צירופיים ואוגרים
  - כל מסלול מכניסה ליציאה כולל בדיוק K אוגרים
- מיועד למנוע את חוסר האיזון שראינו בדוגמה
- דוגמה: מעגל צירופי הוא 0-Pipeline
- תהליך התכנון השיטתי:
  - נתחיל ממעגל צירופי (0-Pipeline)
  - נוסיף אוגרים לפי שני החוקים להלן, שישמרו את ה-Pipeline נכון תמיד

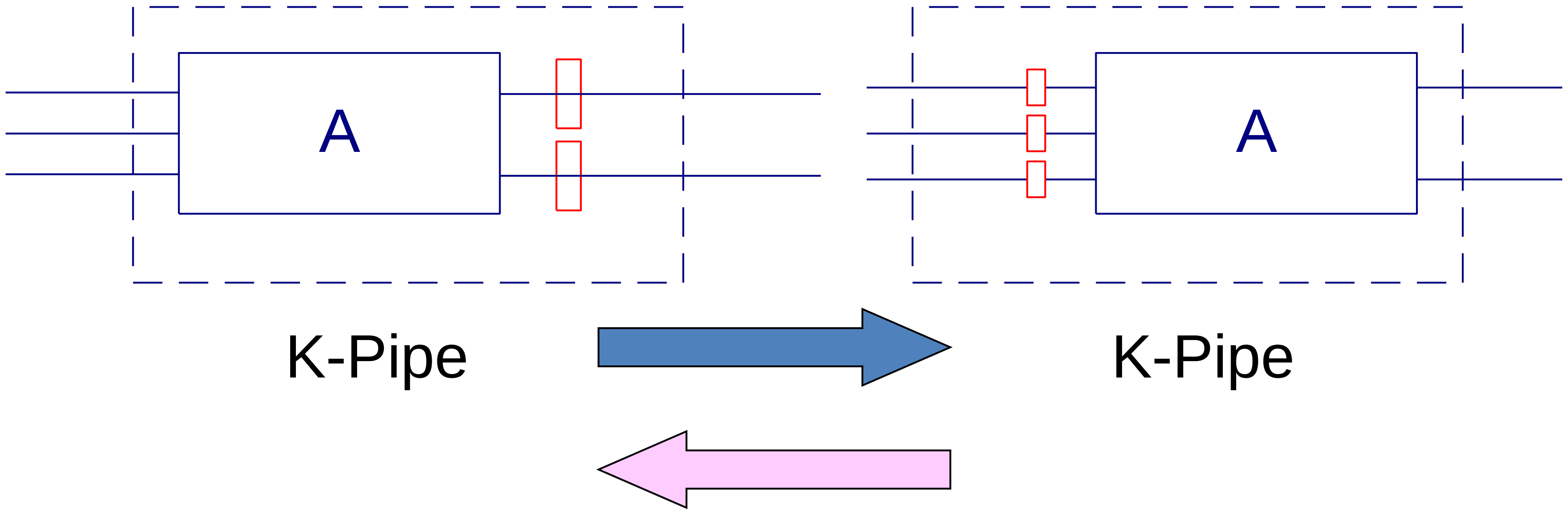
חוק ראשון : הוספת אוגרים (רגיסטרים)

- הוסף אוגר **<u>לכל יציאה</u>** של המערכת
- החישוב אינו משתנה
- התוצאה מתעכבת במחזור שעון אחד



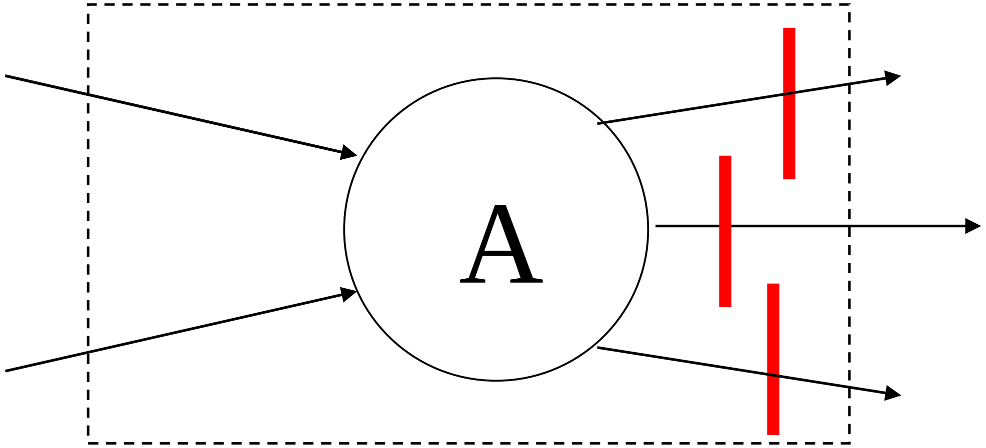
חוק שני: Retiming

- הורד אוגר **<u>מכל יציאה</u>** והוסף אוגר **<u>לכל כניסה</u>**
- ניתן ליישום למערכת שלמה או לרכיב אחד בתוכה
- תוצאה:
  - הרכיב A עושה בדיוק אותו חישוב על אותם נתונים, אבל מאוחר יותר
  - ההשהיה (מדודה במחזורי שעון) מכל כניסה לכל יציאה של המערכת נשארת זהה

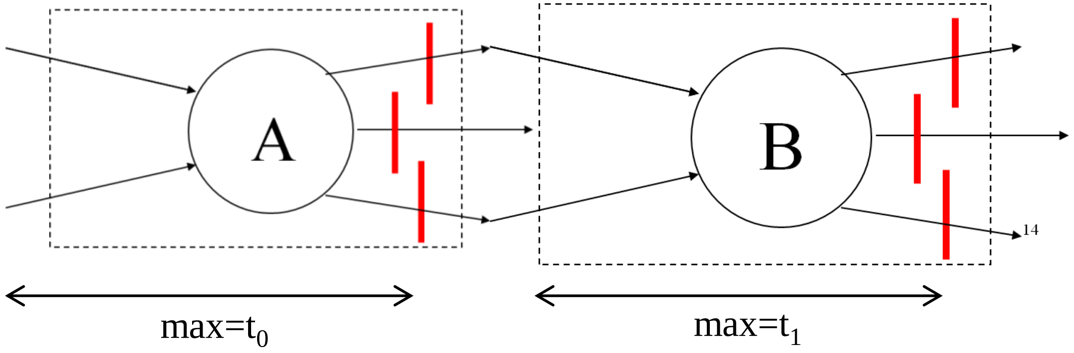


תקני Pipeline

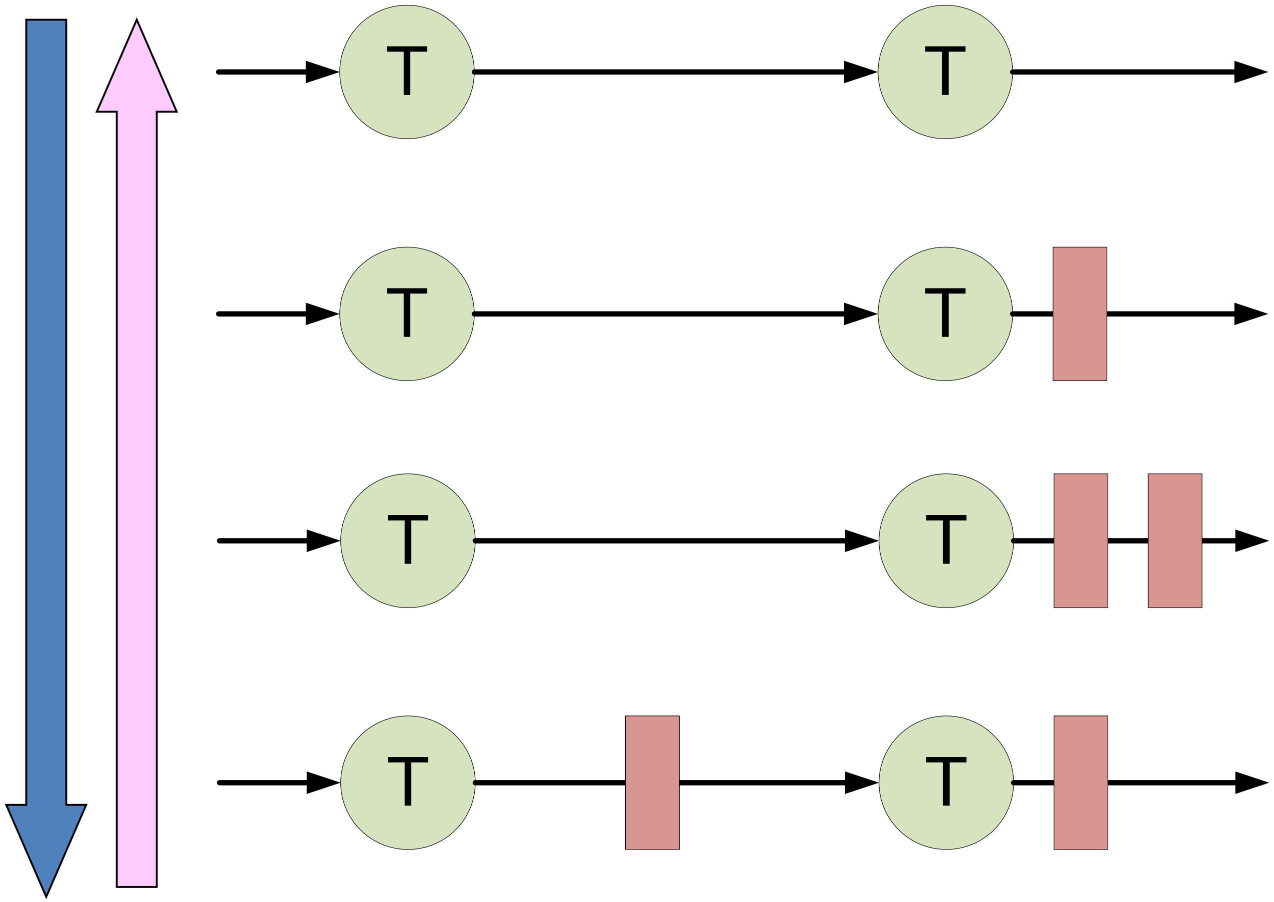
- נשתמש רק ביחידות בסיסיות הכוללות :
  - רכיבים צירופיים
  - אוגרים



מודולריות:



ממעגל צירופי ל-Pipeline



Latency = 2T
Throughput = 1/(2T)

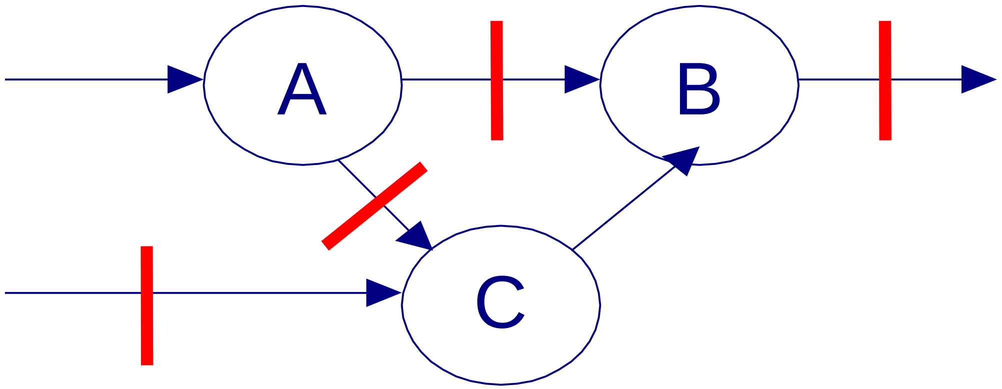
Latency = 2T
Throughput = 1/(2T)

Latency = 4T
Throughput = 1/(2T)

Latency = 2T
Throughput = 1/T

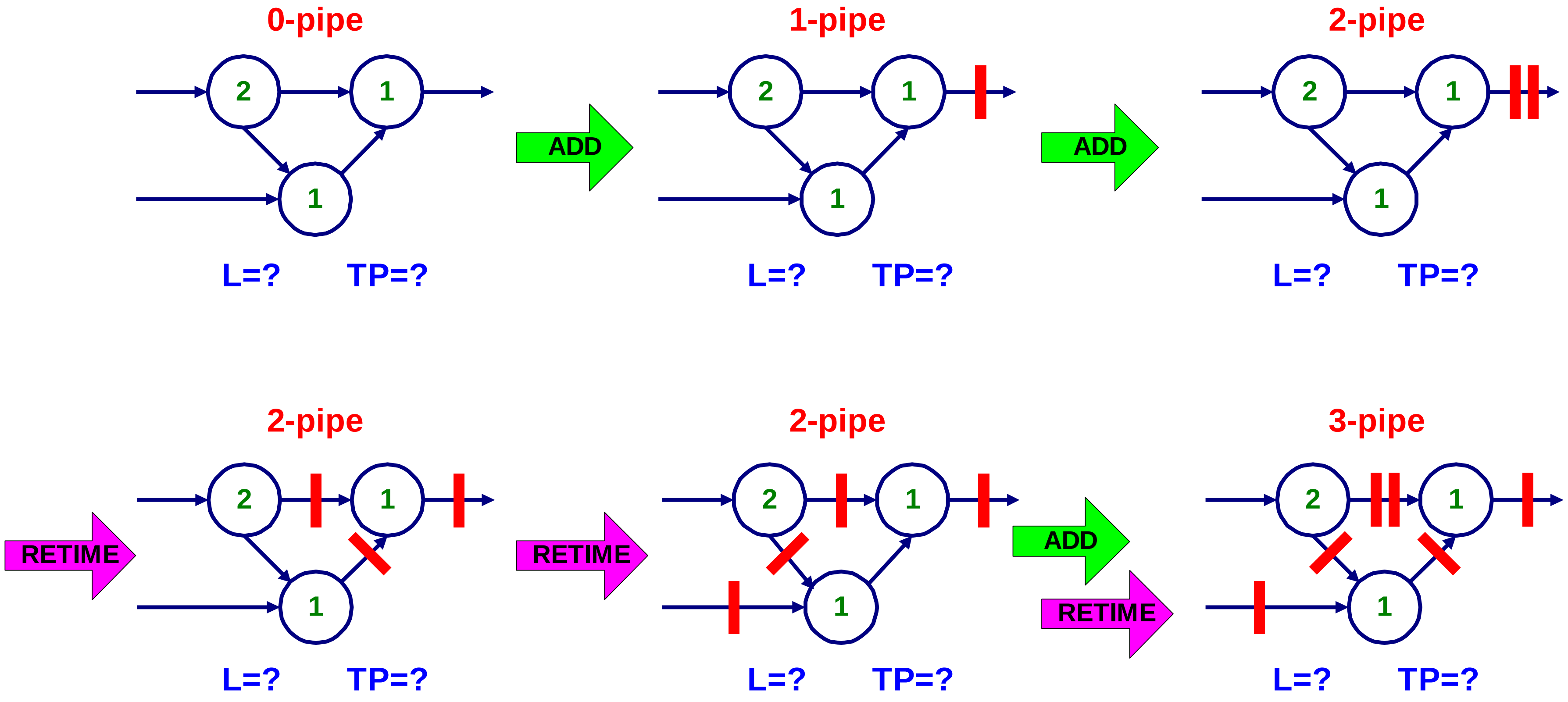
תזמון Pipeline

- Pipeline הוא מערכת סדרתית לכל דבר
- מחזור השעון נקבע לפי התנאי שבכל מסלול מאוגר לאוגר יתקיים
  $T_C \geq T_{PD}(DFF)+T_{PD}(CL)+T_{SU}(DFF)$
- צריך גם לוודא שמתקיימים תנאי HOLD



דוגמה

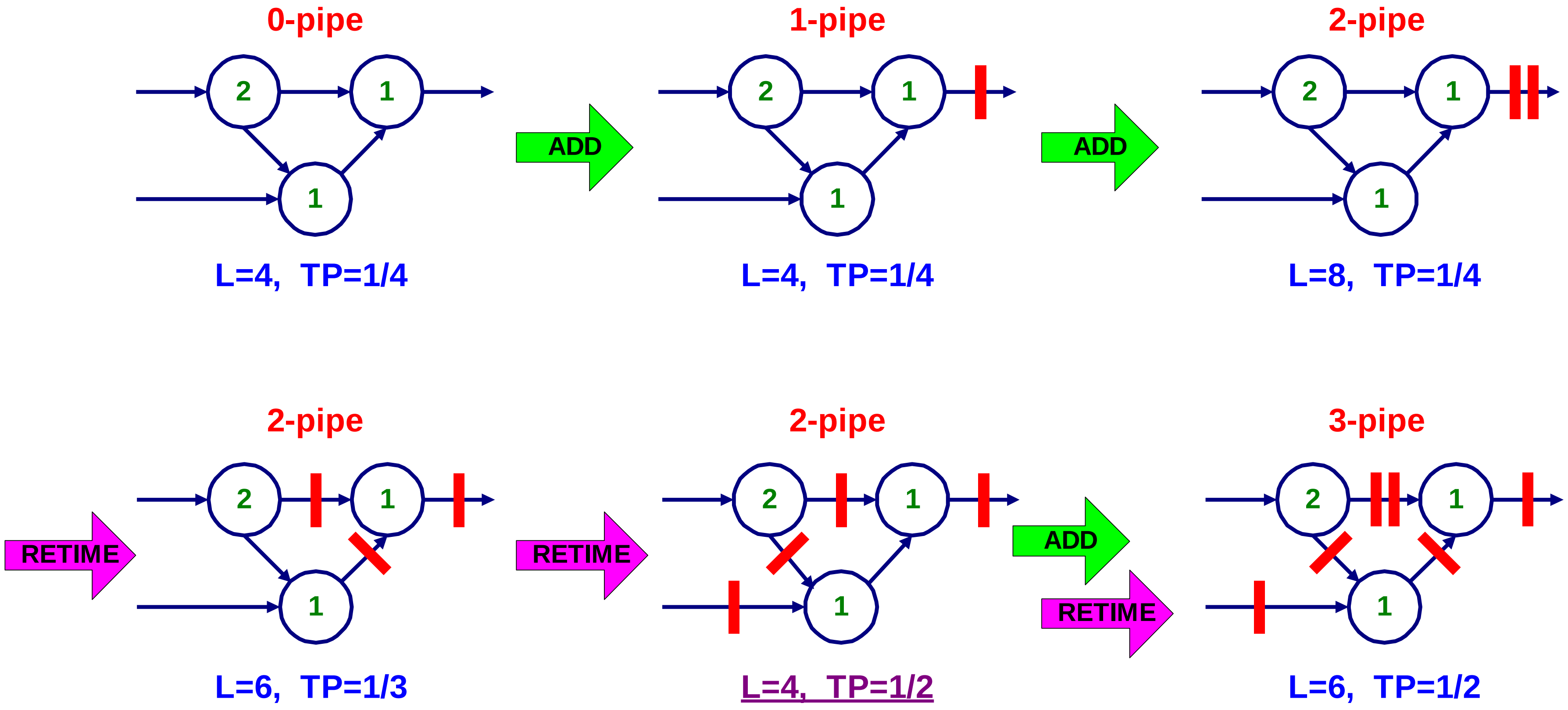
- ננסה למצוא Throughput מכסימלי ו- Latency מינימלי



- שימוש ב- pipeline משפר Throughput אבל לא Latency !
  - אם מוסיפים את t_{SETUP}, t_{pCQ} של האוגרים מסתבר שההשהיה אפילו גדלה במקצת

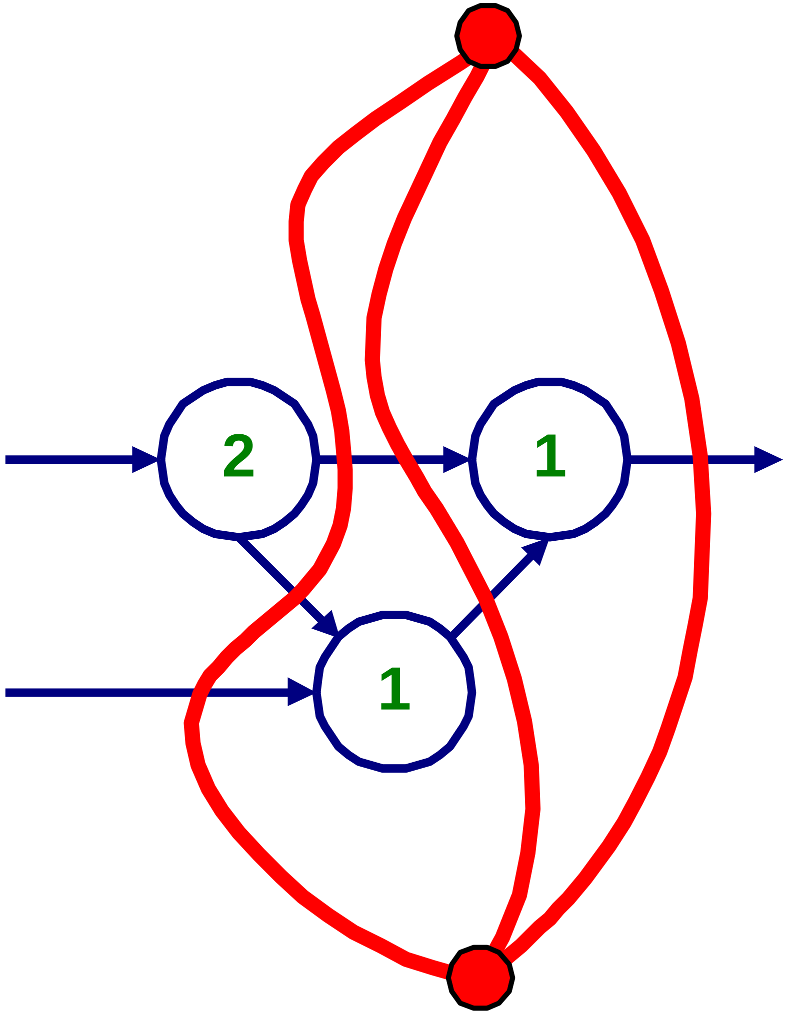
דוגמה

- ננסה למצוא Throughput מכסימלי ו- Latency מינימלי



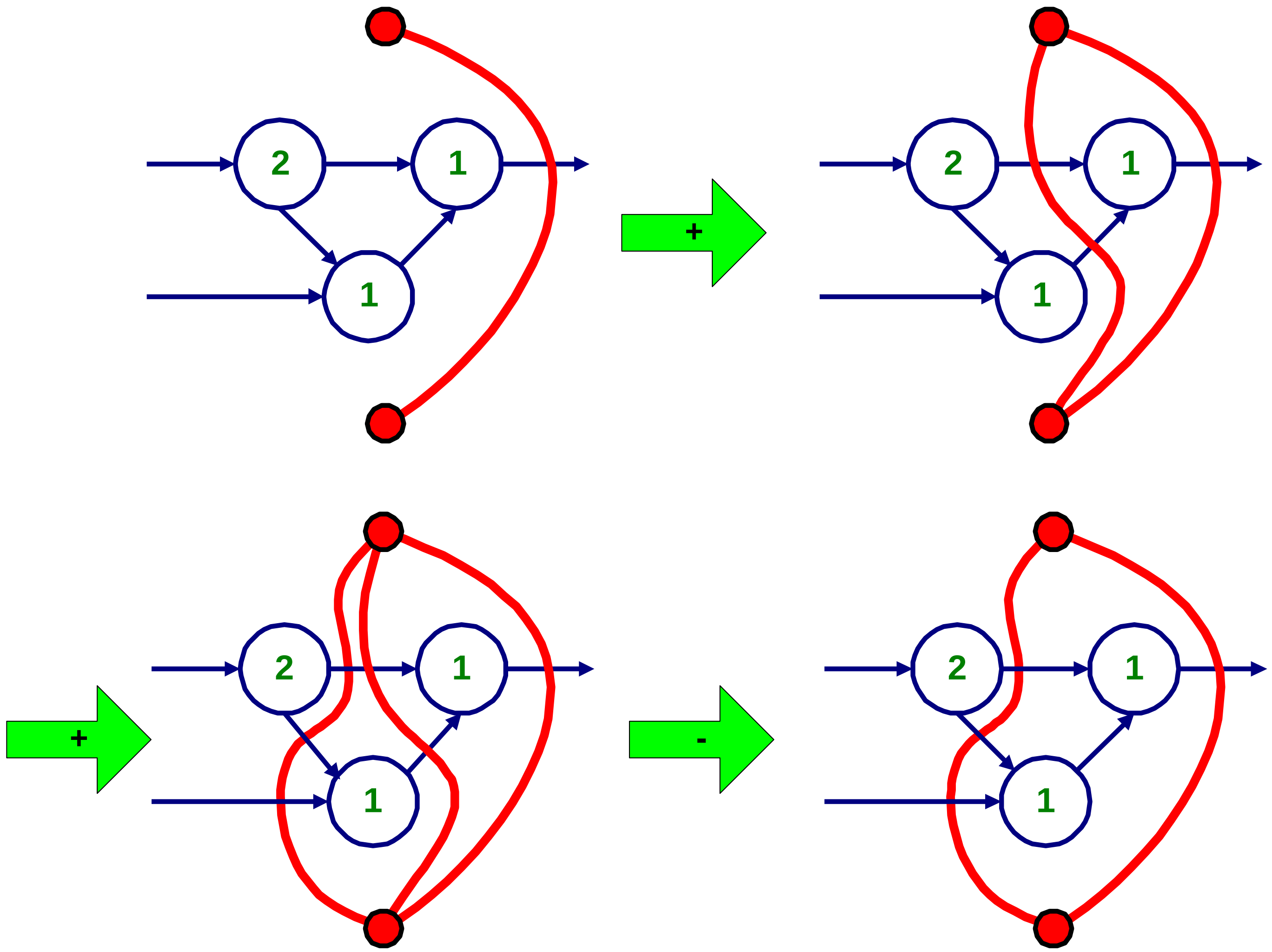
- שימוש ב- pipeline משפר Throughput אבל לא Latency !
  - אם מוסיפים את t_{SETUP}, t_{pCQ} של האוגרים מסתבר שההשהיה אפילו גדלה במקצת

דרך חליפית ליישום החוקים



- חוק ראשון: צייר קו שחוצה את כל היציאות מהמערכת, וסמן את שתי נקודות הקצה שלו
- חוק שני: הוסף קוים בין נקודות הקצה כך שיחצו חיצים שונים, ושכל החיצים שנחצים הם באותו כיוון
- הצב רגיסטר בכל חציית חץ — ה-pipeline המתקבל יהיה תמיד חוקי

ביקור נוסף בדוגמה



Summary

- Equivalence of States and Machines
- Fault Detection in Sequential Systems
- Pipeline