תכן לוגי
תרגיל רטוב #1

תאריך ההגשה נמצא באתר הקורס בלשונית "תרגילי בית".
המתרגל האחראי על התרגיל: משה ליכטנשטיין.
שאלותיכם במייל (smosesli@campus **ולא** @cs), (כולל עניינים מנהלתיים) יופנו רק אליו.
כתבו בתיבת subject: רטוב תכן לוגי.
שאלות בעל-פה ייענו על ידי כל מתרגל.

שימו לב! תרגיל זה "ארוך" ביחס לאחרים ומורכב משני חלקים:

- חלק א – תרגיל בverilog.
- חלק ב – תרגיל על רכיב CLA (ללימוד עצמי)

ניתן (וכדאי) להתחיל את חלק א (התקנת סביבת העבודה של Verilog) ואת חלק ב עוד לפני שמועבר התרגול על Verilog.

הוראות הגשה:

- את החלק הרטוב מגישים דרך אתר הקורס, את החלק היבש מגישים בתא הקורס.
- ההגשה בזוגות.
- **לחלק היבש יש לצרף דף שער הכולל ברקודים של ת"ז המגישים.**
- תמונת ברקוד אפשר לייצר לדוגמה בקישור :
https://barcode.tec-it.com/en/Code128
- בחלק הרטוב עליכם להגיש 3 קבצים, הורידו אותם מאתר הקורס והשלימו את הקוד
- אין לשנות את שמות הקבצים, המודולים או את שמות הכניסות והיציאות.
- שאלות ותשובות הנוגעות לתרגיל יפורסמו באתר הקורס תחת הלשונית "שאלות ותשובות".

חלק א - Verilog

התקנה

התקנת iverilog

- כדי לקמפל את הקוד נשתמש בכלי Icarus Verilog, גם תקינות הקוד תימדד לפיו.
- התקינו את הקומפיילר לפי ההוראות בקישור:
http://iverilog.wikia.com/wiki/Installation_Guide
(גללו למטה עד לחלק של "Installation From Premade Packages").
משתמשי windows: יש להפעיל את ההתקנה כמנהל (לחצן ימני ובחירת "הפעל כמנהל") כדי למנוע בעיות בהמשך.

התקנת GTKWave

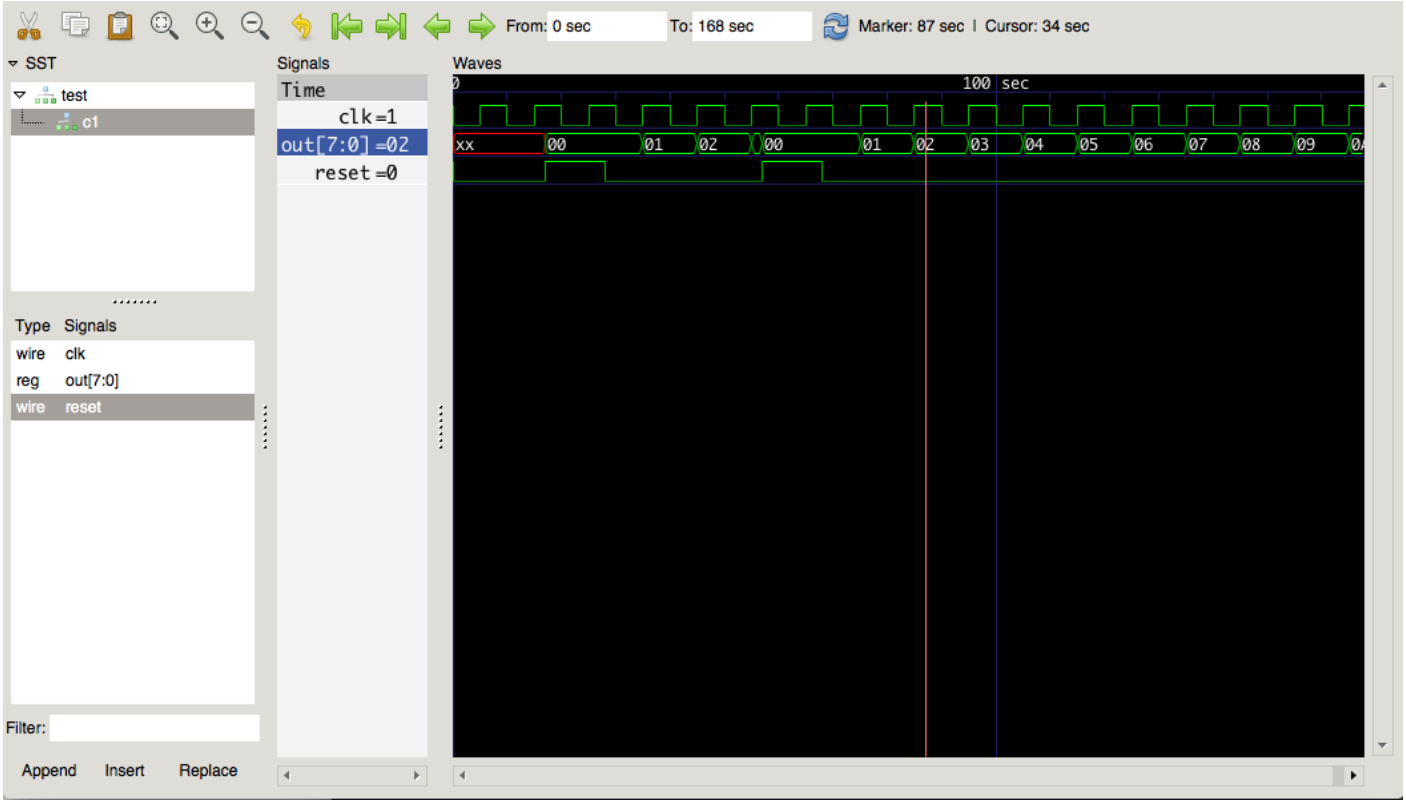
- כדי לדבג את הקוד נשתמש בכלי GTKWave שמאפשר לראות waveform. מי שהתקין על windows יכול לדלג על שלב זה. (ההתקנה הקודמת התקינה גם את GTKWave).
- הוראות התקנה ושימוש נמצאות בקישור:
http://gtkwave.sourceforge.net/gtkwave.pdf

Getting Started

- קראו את המדריך הבא, ופעלו על פי ההוראות:
http://iverilog.wikia.com/wiki/Getting_Started
- מדריך זה יסביר לכם איך להתחיל לכתוב ב-Verilog, איך לקמפל את הקוד שלכם ואיך להריץ עליו בדיקות (testbench).
בפרט, צרו את הקבצים counter_tb.v, counter.v לפי ההוראות בקישור (אנו נצטרך אותם בשלב הבא).
- כעת נלמד איך להציג waveform מהרצת סימולציה של Verilog.
בקובץ counter_tb.v החליפו את $stop ב $finish והוסיפו את הבלוק.

```
initial
 begin
   $dumpfile("test.vcd");
   $dumpvars(0,test);
 end
```

- הפקודות הנ"ל יוצרות קובץ בשם test.vcd שאליו יפלטו ערכי הקווים השונים. להרחבה על שימוש בפקודות אלה ניתן לעיין בקישור:
http://referencedesigner.com/tutorials/verilog/verilog_62.php
- וודאו שנוצר קובץ מתאים עבור הגלים test.vcd ופתחו אותו בעזרת GTKWave. אתם תקבלו את הפלט הבא: (וודאו שאתם מבינים אותו)


From: 0 sec
To: 168 sec
Marker: 87 sec | Cursor: 34 sec
SST
test
c1
Type Signals
wire clk
reg out[7:0]
wire reset
Filter:
Append
Insert
Replace
Signals
Time
clk =1
out[7:0] =02
reset =0
Waves
0
100 sec
xx
00
01
02
00
01
02
03
04
05
06
07
08
09

חלק יבש

שאלה 1

הסבירו את ההבדל בין X ו Z

מלאו את טבלאות האמת הבאות

שער and

	0	1	x	z
0				
1				
x				
z				

שער or

	0	1	x	z
0				
1				
x				
z				

שאלה 2

נניח כי a = reg[3:0] , אילו ערכי a יחשבו כערך אמת (if (a)) ?

הסבירו את ההבדל בין האופרטור == לבין האופרטור ===

הסבירו את ההבדל בין האופרטור reduction OR לבין האופרטור bit-wise OR

הסבירו את ההבדל בין האופרטור הבינארי & לבין האופרטור הבינארי &&

שאלה 3

בכל אחת מהשורות הבאות מלאו בהתאמה reg / wire / both

left-hand side of an assign statement	
right-hand side of an assign statement	
left-hand side of an = or <= sign in an always@ block	
right-hand side of an = or <= sign in an always@ block	
can be connected to the input port of a module instantiation	
can be connected to the output port of a module instantiation	
can be used as outputs within an actual module declaration	
can be used as inputs within an actual module declaration	

שאלה 4

הסבירו את ההבדל בין Blocking לבין Nonblocking assignment ותנו דוגמה בה שימוש בכל אחת יביא לתוצאה שונה

שאלה 5 (testbench)

כתבו לולאת for שמדפיסה את המספרים 0-9

הסבירו את ההבדל בין בלוק initial לבלוק always

הסבירו את ההבדל בין $display לבין $monitor

הסבירו את ההבדל בין $finish לבין $stop

חלק רטוב

1) ALU

בשאלה זו עליכם לממש alu (Arithmetic logic unit) פשוט. הורידו את הקובץ ALU.v מאתר הקורס, כתבו בו מודול שמקבל שתי כניסות של 32 ביט, ומוציא בהתאם לכניסה שלישית תוצאה של פעולה אריתמטית על שתי הכניסות, בקובץ תמצאו בהערות את הקידודים המתאימים ואת הפעולות האריתמטיות שצריכות להתבצע.

2) parameterized mux

בשאלה זו עליכם לממש בורר (mux), עליו למדתם בקורס "מערכות ספרתיות" (במקרה ששכחתם - זה הזמן לחזור על החומר..). הורידו את הקובץ mux3.v מאתר הקורס, כתבו בו מודול שמקבל שלוש כניסות ופרמטר שמציין את רוחבן של הכניסות. בקובץ תמצאו בהערות את הקידודים המתאימים.

3) FSM

הקוד של מכונת המצבים שהנכם מתבקשים לממש צריך להתאים לטמפלט המופיע בקישור:
https://verilogguide.readthedocs.io/en/latest/verilog/fsm.html#mealy-architecture-and-verilog-templates

קראו את השאלה המופיעה בעמוד הבא, פתרו את השאלה (אין צורך להגיש) וממשו את מכונת המצבים המתאימה לסעיף 1 של השאלה בקובץ simon1.v שתורידו מאתר הקורס. במימוש זה עליכם להוסיף אות כניסה reset מתאים, התנהגות המכונה היא כמו בטמפלט מהקישור – כאשר האות עולה על מכונת המצבים לחזור למצבה ההתחלתי.

4) כתיבת טסט (לא להגשה)

כתבו טסטים מתאימים למודולים שכתבתם, חשבו מהם מקרי הקצה האפשריים ובדקו אותם.

לכל תרגיל, מצורף קובץ template בסיסי שיעזור לכם להתחיל לכתוב טסט (בעל אותו שם, עם תוספת _tb). ניתן (ואף מומלץ) לכתוב טסטים מסובכים יותר משלכם.

שימו לב! טסטים אלה לא משקפים את הבדיקות שיעשו לתרגילים על-ידי צוות הקורס!

שאלת FSM (לא להגשה)

המשחק "סימון (simon) זוגי" הוא משחק זיכרון לשני שחקנים. המשחק עצמו הוא קונסולה שמכילה ארבעה מקשים: אדום, ירוק, כחול וצהוב.

מהלך המשחק: שחקן א' ("התוקף") מייצר רצפים של המקשים באורך הולך וגדל בכל שלב.
המטרה של שחקן ב' ("הזוכר") היא לזכור את הרצף שהתוקף ייצר.
לאחר זכירה נכונה של רצף, התוקף מייצר (בשלב הבא) רצף חדש שגדול ב-1 מהרצף הקודם.

המשחק נגמר כאשר הזוכר מתבלבל ברצף או כאשר התוקף מתייאש.

מהלך משחק לדוגמא:

שלב 1:	**שלב 2:**	**שלב 3:**
התוקף: לוחץ על: אדום.	התוקף: לוחץ על: כחול, צהוב.	התוקף: לוחץ על: ירוק, אדום, צהוב.
הזוכר: לוחץ על: אדום.	הזוכר: לוחץ על: כחול, צהוב.	הזוכר: לוחץ על: ירוק, כחול – ונפסל.

(שימו לב: המשחק נגמר לאחר הטעות הראשונה של הזוכר, ולא לאחר כל המהלך).

בתרגיל זה נרצה לבנות מכונת מצבים מסוג מילי שתשמש כקונסולת המשחק.
המכונה תקבל (בקו הקלט In) אחת מארבע אפשרויות: R, G, B, Y.
ותייצר שלושה פלטים (בקו פלט Out): X,0,1, באופן הבא: כל זמן שהתוקף טוען את הרצף שלו למכונה – המכונה פולטת X.
כאשר מגיע תורו של הזוכר לשחק: על כל זיהוי נכון – המכונה תפלוט 1. כאשר הזוכר טועה – המכונה תפלוט 0 ותעבור למצב GameOver (שממנו לא ניתן לעבור למצבים אחרים).

כמו כן, נגדיר תת משחק של המשחק המקורי באופן הבא:
"סימון שלב i" הוא משחק שבו התוקף והזוכר משחקים את שלב i פעם אחת.
במקרה הזה, אם הזוכר זכר את הרצף – המכונה פולטת 1 ועוברת למצב Win (ונשארת בו).
אם לא, המכונה עוברת למצב GameOver כמו שפורט לעיל.

ענו על הסעיפים הבאים. אם לא ניתן לבנות מכונת מצבים כנדרש – יש לסמן X.

1. שרטטו מכונת מצבים מצומצמת של "סימון שלב 1".

2. כמה מצבים יש במכונת מצבים מצומצמת שתממש קונסולה למשחק "סימון שלב i"?

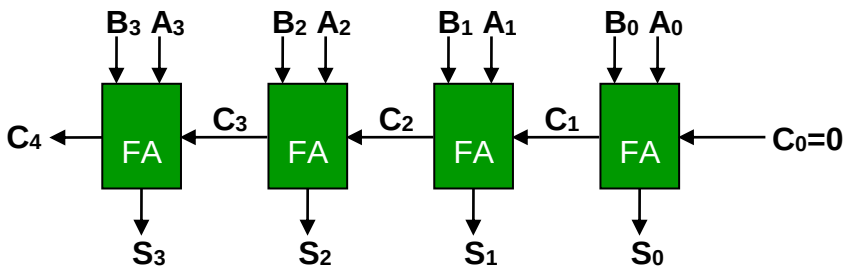
3. כמה מצבים יש במכונת מצבים מצומצמת שתממש קונסולה למשחק "סימון זוגי"?

חלק ב -שאלת CLA (כן להגשה)

בשאלה נציג דרך להאיץ את פעולת החיבור של שני מספרים A,B בני n ביטים באמצעות לוגיקה צירופית. כיוון שפעולת החיבור נפוצה, האצת החיבור היא קריטית לביצועי המחשב. שני המספרים A,B שבקלט הם:

$A = A_{n-1} A_{n-2} \dots A_2 A_1 A_0$
$B = B_{n-1} B_{n-2} \dots B_2 B_1 B_0$

א. בקורס "מערכות ספרתיות" למדתם על Ripple-Carry Adder.
לדוגמה, עבור **n=4**, Ripple-Carry Adder ממומש באופן הבא: (FA = Full Adder)

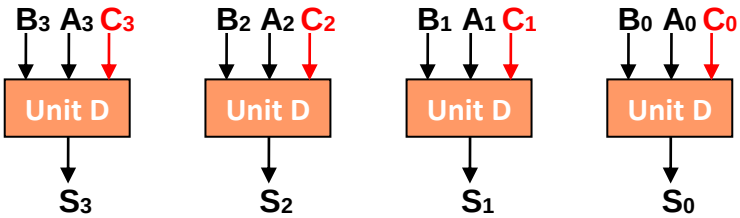


עבור **n=32**, כמה רכיבי FA יידרשו למימוש Ripple-Carry Adder? _____
עבור **n=32**, אם נסמן את השהיית FA (כלומר, את $T_{PD}(FA)$) ב-$\mathbf{T_{FA}}$, מה תהיה ההשהייה (ה-T_{PD}) של Ripple-Carry Adder כתלות בפרמטר T_{FA}? _____

ב. בשאלה זו נבנה Carry-Lookahead Adder (CLA), שמבצע חיבור בהשהייה נמוכה יותר מהשהיית Ripple-Carry Adder עבור ערכי n בינוניים וגדולים. השאלה כוללת את כל הפרטים הדרושים לצורך הפתרון, אך ניתן למצוא עזרה ומידע נוסף בתרגול הזה.

החישוב ב-CLA מורכב מארבעה שלבים (**שלבים 1-4**). הרעיון הוא שמחשבים מראש את כל ביטי הנשא (carry) $C_0, C_1, C_2, \dots, C_{n-1}, C_n$ **בשלבים 1-3**, ואז **בשלב 4** מחשבים ביעילות את כל ביטי הסכום $S_0, S_1, S_2, \dots, S_{n-1}, S_n$.

להלן המימוש של **שלב 4** עבור **n=4**:



מלאו בטבלה הבאה את **הפלט** של Unit D עבור ערכי הקלט A,B,C: (שימו לב: Unit D מקבלת כקלטים את שני ביטי הקלט A_i,B_i ואת ביט ה-carry הנכנס C_i, ומוציאה כפלט את ביט הסכום S_i)

A_i	B_i	C_i	S_i
0	1	0	
1	0	1	
1	1	1	

עבור **n=32**, כמה יחידות Unit D יידרשו למימוש **שלב 4**? _____
עבור **n=32**, אם נסמן את השהיית Unit D (כלומר, את $T_{PD}(\text{Unit D})$) ב-$\mathbf{T_D}$, מה תהיה ההשהייה (ה-T_{PD}) של **שלב 4** כתלות בפרמטר T_D? _____

ג. כעת נבנה את **שלבים 1-3**, שמטרתם לחשב מתוך ביטי הקלט $\{A_i,B_i\}$ את ביטי ה-carry $\{C_i\}$.
ב**שלב 1**, נחקור את התנהגות ה-carry היוצא C_{i+1} כתלות בביטי הקלט A_i,B_i וב-carry הנכנס C_i. נסווג את הביט

ה-i לאחת משלוש התנהגויות אפשריות של C_{i+1} כתלות ב-C_i:
(הסיווג מתבצע לפי זוג ביטי הקלט A_i,B_i: כלומר, הביטים A_i,B_i קובעים את התנהגות הביט ה-i.)

a. לביט **Generate** מתקיים $\mathbf{C_{i+1} = 1}$: ה-carry היוצא הוא 1, ללא תלות ב-carry הנכנס.

b. לביט **Propagate** מתקיים $\mathbf{C_{i+1} = C_i}$: ה-carry היוצא שווה ל-carry הנכנס.

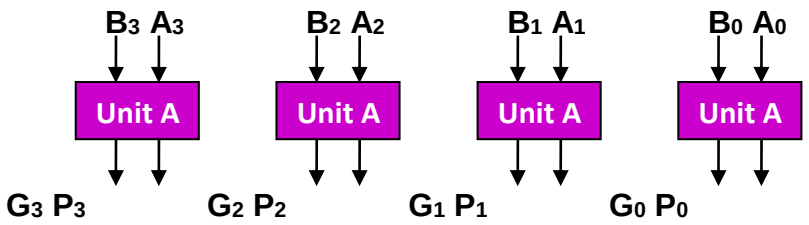
c. לביט **Kill** מתקיים $\mathbf{C_{i+1} = 0}$: ה-carry היוצא הוא 0, ללא תלות ב-carry הנכנס.

בנוסף, נגדיר שני ביטים G_i,P_i באופן הבא: $G_i = 1$ אם ורק אם התנהגות ביט i היא Generate;
$P_i = 1$ אם ורק אם התנהגות ביט i היא Propagate; לכן $G_i = P_i = 0$ אם התנהגות ביט i היא Kill.

מלאו בטבלה הבאה את **ההתנהגות** (Generate/Propagate/Kill) ואת **ערכי G_i,P_i**, כתלות בביטי הקלט A_i,B_i:

A_i	B_i	ההתנהגות	G_i	P_i
0	0			
1	0			
0	1			
1	1			

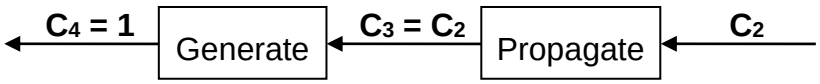
הטבלה לעיל היא למעשה טבלת האמת של היחידה Unit A. להלן **שלב 1** של ה-CLA עבור **n=4**:



עבור **n=32**, כמה יחידות Unit A יידרשו למימוש **שלב 1**? _____
עבור **n=32**, אם נסמן את השהיית Unit A (כלומר, את T_{PD}(Unit A)) ב-$\mathbf{T_A}$, מה תהיה ההשהייה (ה-T_{PD}) של **שלב 1** כתלות בפרמטר T_A? _____

ד. ב**שלב 2**, נמזג ביטים בודדים לבלוקים של מספר ביטים, ונחקור את התנהגות הבלוקים (ה-carry היוצא מהבלוק כתלות ב-carry הנכנס לבלוק): האם היא Generate, Propagate, או Kill?

נניח לדוגמה שנתונים לנו שני ביטים: ביט ימני (מספר 2) וביט שמאלי (מספר 3).
נניח שההתנהגות של ביט מספר 2 היא Propagate וההתנהגות של ביט מספר 3 היא Generate. ניתן להבחין שההתנהגות של **בלוק הביטים 3-2** היא **Generate**, כמתואר בציור הבא:

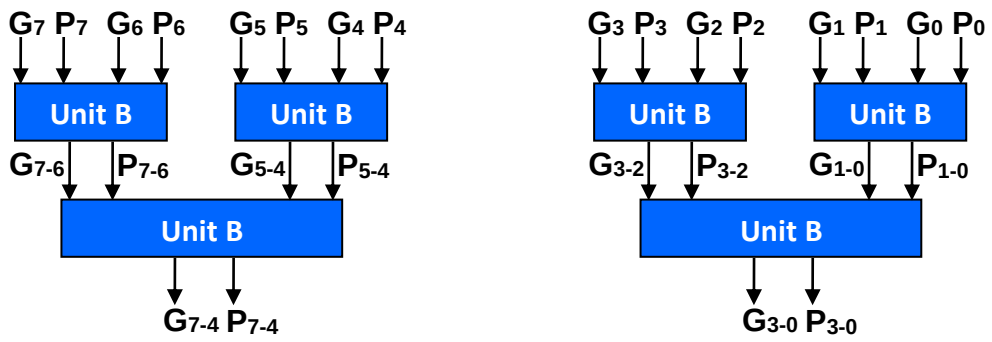


מלאו בטבלה הבאה את יתר התנהגויות **בלוק הביטים i-(i+1)** (המורכב מהביט הימני i ומהביט השמאלי i+1), Generate/Propagate/Kill, כתלות בהתנהגות הביטים i ו-i+1:

	(מלאו בתאים המרכזיים את התנהגות בלוק הביטים i-(i+1))	התנהגות הביט השמאלי i+1		
		Generate	**Propagate**	**Kill**
התנהגות הביט הימני i	**Generate**			
	Propagate	Generate (כבר מצאנו)		
	Kill			

הטבלה הנ"ל מתארת את התנהגות היחידה Unit B, שבהינתן ערכי G,P של שני ביטים (או של שני בלוקים) **סמוכים** מחשבת את ערכי G,P של הבלוק הממוזג.

שלב 2 של ה-CLA מחשב את ערכי G,P של כל הבלוקים הרלוונטיים, במבנה של **עץ**, כמתואר באיור. להלן **שלב 2** עבור **n=8**:



עבור **n=32**, כמה יחידות Unit B יידרשו למימוש **שלב 2**? _____
עבור **n=32**, אם נסמן את השהיית Unit B (כלומר, את T_{PD}(Unit B)) ב-$\mathbf{T_B}$, מה תהיה ההשהייה (ה-T_{PD}) של **שלב 2** כתלות בפרמטר T_B? _____

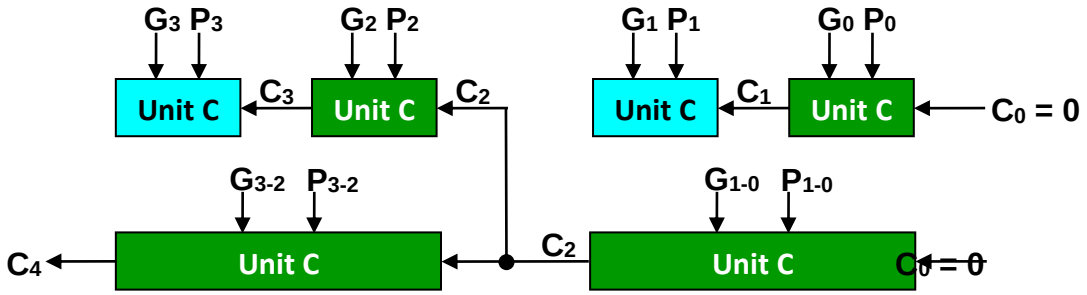
ה. ב**שלב 3**, נשתמש בערכי G,P שחישבנו בשלב 2 כדי לחשב את כל ה-carries ביעילות, תוך שימוש במבנה של עץ. כלומר, נשתמש בהתנהגות Generate/Propagate/Kill של בלוק וב-carry הנכנס כדי לחשב את ה-carry היוצא.

נניח שידועה לנו ההתנהגות של בלוק מסוים של ביטים j-i (מביט i עד ביט j, כולל) וגם ה-carry הנכנס C_i. ברצוננו לחשב את ה-carry היוצא C_{j+1}. מלאו את הטבלה הבאה:

(מלאו בתאים המרכזיים את ערך ה-carry היוצא C_{j+1})		ערך ה-carry הנכנס C_i	
		$C_i = 0$	$C_i = 1$
התנהגות הבלוק j-i	**Generate**	C_{j+1} = ___	C_{j+1} = ___
	Propagate	C_{j+1} = ___	C_{j+1} = ___
	Kill	C_{j+1} = ___	C_{j+1} = ___

הטבלה הנ"ל מתארת את התנהגות היחידה Unit C, שבהינתן ערכי G,P של בלוק וה-carry הנכנס לבלוק מחשבת את ה-carry היוצא מהבלוק.

שלב 3 של ה-CLA מחשב את כל ביטי ה-carry: C_0, C_1, C_2, ..., C_{n-2}, C_{n-1}, C_n, במבנה של **עץ**, כמתואר באיור. להלן **שלב 3** עבור **n=4**:



שימו לב: כל היחידות באיור זהות (Unit C). היחידות המסומנות בצבע בהיר (תכלת) **אינן מוצאות פלט** ולמעשה אינן נחוצות למימוש, **ואין לכלול אותן בחישובים שבשורות הבאות**.

עבור **n=32**, כמה יחידות Unit C יידרשו למימוש **שלב 3**? _____
עבור **n=32**, אם נסמן את השהיית Unit C (כלומר, את T_{PD}(Unit C)) ב-$\mathbf{T_C}$, מה תהיה ההשהייה (ה-T_{PD}) של **שלב 3** כתלות בפרמטר T_C? _____

ו. בסעיף זה נשתמש בתוצאות מסעיפים א'-ה' כדי למצוא את כמות הציוד ואת ההשהיות לכל שלב.

נחשיב כל אחת מהיחידות Full Adder, Unit A, Unit B, Unit C, Unit D כיחידה בסיסית אחת.
הניחו שכל השהיות הרכיבים שבהם דנו בסעיפים הקודמים **שוות**: $T_{FA} = T_A = T_B = T_C = T_D = T$.

מלאו את הטבלה הבאה: (נדרשות תוצאות מדויקות, לא אסימפטוטיות)

המערכת הלוגית	כמות הציוד עבור **n=32** (נמדדת בכמות יחידות בסיסיות)	כמות הציוד עבור **$n=2^k$ כאשר k כלשהו** (נמדדת בכמות יחידות בסיסיות; **תלויה ב-n**)	ההשהייה עבור **n=32** (**תלויה ב-T**)	ההשהייה עבור **$n=2^k$ כאשר k כלשהו** (**תלויה ב-T וב-n**)
Ripple-Carry Adder				
שלב 1 של CLA				
שלב 2 של CLA				
שלב 3 של CLA				
שלב 4 של CLA				
CLA (שרשור ארבעת השלבים זה לזה)				

החל מאיזה ערך של **$n=2^k$**, ההשהייה של CLA נמוכה מהשהיית Ripple-Carry Adder? _____
(**התעלמו** מערכים של n שאינם חזקות של 2.)