אלגברת מיתוג

<u>סדר ביצוע פעולות בוליאניות:</u> סוגריים, NOT, AND, OR

זהויות מיתוג בסיסיות:

1. <u>Idempotency</u> $x+x=x \quad x\cdot x=x$
2. <u>ערכים אדישים ושולטים</u>
   - $x+1=1$ — 1 שולט בחיבור
   - $x\cdot 0=0$ — 0 שולט בכפל
   - $x+0=x$ — 0 אדיש בחיבור
   - $x\cdot 1=x$ — 1 אדיש בכפל
3. <u>חילוף (קומוטטיביות)</u> $x\cdot y=y\cdot x \quad x+y=y+x$
4. <u>קיבוץ (אסוציאטיביות)</u> $(x\cdot y)\cdot z=x\cdot(y\cdot z) \quad (x+y)+z=x+(y+z)$
5. <u>השלמה</u> $x\cdot x'=0 \quad x+x'=1$
6. <u>פילוג</u> $x+(y\cdot z)=(x+y)\cdot(x+z) \quad x(y+z)=xy+xz$
7. <u>חוק "הבליעה"</u> $x(x+y)=x \quad x+xy=x$
8. <u>חוק הבליעה השני</u> $x(x'+y)=xy \quad x+x'y=x+y$
9. <u>חוק הקונסנזוס</u> $xy+x'z+yz=xy+x'z$

חוקי דה-מורגן:

10. <u>היפוך עצמי (involution)</u> $(x')'=x$
11. $(xy)'=x'+y' \quad (x+y)'=x'\cdot y'$

פונקציות מיתוג

צורות קנוניות:

לפונקציות בעלות אותה טבלת אמת, אותה צורה קנונית. אלו פונקציות שוות/זהות/שקולות.

ליטרל: כל משתנה מיתוג וכל משלים שלו: x, y, z'

צורה קנונית של סכום מכפלות: (DNF)

ערך הביטוי יהיה '1' אם לפחות אחת המכפלות ערכה '1'.

כל מכפלה (minterm) מקבלת '1' רק עבור צירוף אחד של ערכי המשתנים.

minterm: מכפלה של n ליטרלים המתאימים למשתנים שונים.

יש 2^n מינטרמים ב-n משתנים- מינטרם אחד לכל צירוף כניסה.

רישום: $f = \sum(0,2,3) \longrightarrow f(x,y,z) = x'y'z' + x'yz' + x'yz$

המספרים בסוגריים מציינים את מספרי השורות בטבלת האמת (בייצוג עשרוני החל מ0) בהן הפונקציה מקבלת ערך '1'.

צורה קנונית של מכפלת סכומים: (CNF)

מספיק שאחד הסכומים יהיה '0' כדי שכל הביטוי יהיה '0'.

maxterm: סכום n ליטרלים המתאימים למשתנים שונים (כפונקציה- מקבל את הערך 0 רק עובר צירוף יחיד של ערכי משתני כניסה).

רישום: $f = \Pi(1,4,5) \longrightarrow f(x,y,z) = (x+y+z')(x'+y+z)(x'+y+z')$

פונקציה ופונקציה הפוכה:

$$f(x,y,z) = \sum(0,2,3,6,7) = \Pi(1,4,5)$$
$$\overline{f(x,y,z)} = \sum(1,4,5) = \Pi(0,2,3,6,7)$$

העברה לצורה קנונית (ללא טבלת אמת):

1. לכל מכפלה בביטוי, אם היא מינטרם (מופיעים בה כל n הליטרלים). עבור.
2. אחרת:
   a. כפול את המכפלה ב- $(x_i + x_i')$ עבור כל ליטרל x_i החסר במכפלה.

      (עבור מכפלת סכומים- לחבר $x_i x_i'$ עבור כל ליטרל חסר בסכום).
   b. פתח סוגריים ובטל מכפלות חוזרות.
   c. סדר כל מכפלה לפי סדר אחיד של משתנים ואת הסכום כולו לפי הסדר בטבלת אמת.
   d. בטל מכפלות חוזרות.

הפונקציה XOR:

תכונות:

$$A \oplus B = B \oplus A$$
$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$
$$A(B \oplus C) = (AB) \oplus (AC)$$
$$A \oplus A = 0 \Rightarrow A = B \Rightarrow A \oplus B = 0$$
$$A \oplus A' = 1, \quad A \oplus 1 = A', \quad A \oplus 0 = A$$

$EQ \quad \overline{A \oplus B} = A = B = EQ(A,B)$ היא פונקצית השוויון: $EQ(A,B) = 1$ אם"ם $A = B$

החלפת ערכי המשתנים בעזרת XOR:

$$1.\ X \leftarrow X \oplus Y \quad 2.\ Y \leftarrow X \oplus Y \quad 3.\ X \leftarrow X \oplus Y$$

מערכת פעולות שלמה:

קבוצה של אופרטורים וקבועים שניתן לתאר באמצעותה כל פונקצית מיתוג.

דוגמא: NAND, NOR.

מינימיזציה של פונקציות מיתוג

ביטוי מינימאלי: מספר המכפלות יהיה מינימאלי ובכל מכפלה מספר ליטרלים מינימאלי. ביטוי מינימאלי אינו בהכרח יחיד.

ביטוי בלתי ניתן לצמצום: סכום מכפלות שמחיקת מכפלה או ליטרל יוצרת ביטוי שאינו שקול למקורי. ביטוי כזה אינו בהכרח ביטוי מינימאלי.

מפת קרנו:

משבצות שכנות: משבצות במפה שייצוגן נבדל בסיבית אחת בלבד. לכל משבצת n שכנים.

צמצום פונקציות: מבוסס על הזהות: $Ax' + Ax = A$ (כלומר שתי משבצות שכנות הנבדלות בליטרל אחד ניתנות לליכוד למכפלה שלא כוללת ליטרל זה).

קובייה m-ממדית::

הגדרה: אוסף מקסימאלי של 2^m משבצות שכולן זהות בייצוגן הבינארי ב $n-m$ מקומות.

משפט: קובייה m-ממדית של nמשתנים מתאימה למכפלה של n-m ליטרלים.

מטרת המינימיזציה: "כיסוי" ה-1-ים במפה ע"י מינימום קוביות גדולות ככל האפשר (במובן של מספר ליטרלים מינימאלי).

פונקציות מינימאליות ותכונותיהן

הגדרות:

פונקציה מכסה: f מכסה את g אם היא מכילה את g כלומר, בכל שורה בה בטבלת האמת של g יש-1 אז גם בטבלת האמת של f יש-1 (וייתכן שגם בשורות נוספות). סימון: $f \supseteq g$

גורר (implicant): ולהפך- g היא גורר של f אם f מכסה אותה וגם g היא מכפלה של ליטרלים. סימון: $g \to f$.

טענות:

1. $g \supseteq g \cdot h;\quad g + h \supseteq g$ (g מכיל כל חיתוך של עצמו עם דברים אחרים וכל חיבור שלו עם דברים אחרים מכיל אותו).
2. אם $f \supseteq g$ וגם $g \supseteq f$ אז: $f = g$
3. $f \supseteq g + h$ אם"ם $f \supseteq g$ וגם $f \supseteq h$

הגדרות נוספות:

גורר ראשוני (PI): מכפלת ליטרלים שהיא גורר של f ($p \to f$) וגם: מחיקת ליטרל כלשהו ממנה יוצרת מכפלה שאיננה מכוסה ע"י f משמעות: אין קובייה גדולה יותר במפה שמכסה אותו.

משפט: כל סכום מכפלות בלתי ניתן לצמצום השקול לפונקצית מיתוג f הינו סכום של גוררים ראשוניים של f

גורר ראשוני חיוני (EPI): גורר ראשוני של f אשר מכסה לפחות מינטרם אחד של f שאינו מכוסה ע"י אף גורר ראשוני אחר (משבצת שמכוסה רק ע"י EPI זה).

*ביטוי מינימאלי חייב להכיל את כל הEPI.

תהליך קבלת ביטוי מינימאלי לפונקציה:

1. מצא את כל הגוררים הראשוניים (PI) של f
2. מצא מתוכם את כל הגוררים הראשוניים החיוניים (EPI) והכנס אותם לביטוי.
3. הוצא מרשימת הPI את כל ה-EPI וכן את כל הPI שכבר כוסו ע"י הEPI (יכול להיות שPI מכוסה ע"י שני EPI, כי אם רק EPI מכסה אותו זה כבר לא יהיה PI).
4. אם קבוצת ה-EPI מכסה את f סיימנו. אחרת:
   4.1 בחר PI נוספים כך ש- f תכוסה כולה וכן שמספרם יהיה מינימאלי, ומבין כל הסכומים האלה בחר את בעל מספר הליטרלים הקטן ביותר.

צירופי ברירה (Don't Care):

צירופי כניסה מסוימים לא ייתכנו או ערך היציאה עבור צירופי כניסה כאלה אינו מעניין.

היציאה עבורם תסומן ב- ϕ. ניתן להחליף אותו ב-0 או ב-1 לפי הנוחות (לצורך קבלת ביטוי מינימאלי).

ביטויי-מפה:

שיטה לתאר פונקציה של יותר מ- nמשתנים במפת קרנו של n משתנים.

ביטוי מפה: ביטוי המופיע במפה שאינו 0,1 או ϕ.

פישוט פונקציה עם ביטויי מפה:

1. החלף במפה כל ביטוי-מפה ל0 ומצא ביטוי מינימאלי למפה.
2. עבור כל ביטוי מפה M:
   2.1 בכל משבת המכילה את M הצב 1
   2.2 בכל משבצת המכילה 1 או ϕ הצב ϕ
   3.2 בכל משבצת אחרת הצב 0

מודל ספרתי ושערים לוגיים

הגדרות:

שער לוגי: התקן (בד"כ רכיב אלקטרוני) המממש פונקצית מיתוג

רכיב צירופי: רכיב הממש פונקצית מיתוג בעל כניסות ויציאות שמתאים ערך מיתוג 0 או 1 לכל יציאה בעבור כל צירוף אפשרי של ערכי כניסות (למשל: טבלת אמת). קיימות לו מגבלות תזמון (פיסיות).

מעגל צירופי: מעגל הכולל רכיבים צירופיים המחוברים ביניהם, בעל כניסה ויציאה שאינו כולל מעגלים פנימיים, כלומר: מעגל שבו יש רכיב שיציאתו מחוברת לאחת מכניסותיו אינו מעגל צירופי.

מעגל לוגי/מעגל מיתוג: צירוף של שער לוגי אחד או יותר לצורך חישוב פונקצית מיתוג. יציאת המעגל הלוגי תלויה אך ורק בצירוף של הערכים הלוגיים של כניסותיו ("השמה").

תכן לוגי (Logic Design):

תכנון מעגלים ספרתיים באמצעות שערים.

כיום, המטרה היא לממש פונקצית מיתוג באמצעות מספר מינימאלי של רכיבים ולאו דווקא של שערים.

ההפשטה הספרתית:

- מתח חשמלי ייצג ערך לוגי: מתח גבוה='1', מתח נמוך='0' מתחים מסוימים אחרים יוגדרו כבלתי קבילים לייצוג ערכים לוגיים.
- בזמנים מסוימים נסכים שהמעגל האלקטרוני מייצג את פעולתה של פונקצית המיתוג, בזמנים אחרים (למשל בזמן שינוי ערכים) נסכים שהמעגל איננו מייצג את הפונקציה.

רמות לוגיות: רמות המתחים מוגבלות לתחום $[V_{MINUS}, V_{PLUS}]$ כאשר טווח המתחים של '1' לוגי ושל '0' לוגי נקבע מראש.

רעש ושולי רעש:

יתכן שהאות החשמלי "יתקלקל" במקצת תוך כדי מעבר בחוטים המקשרים את השערים. ניתן להרשות לכניסות השערים לכלול "רעש" ע"י כך שמגדילים את הטווח לקליטת '0' ו'1' לוגי מכיוון אחד (בד"כ שולי הרעש לא יחרגו מ$[V_{MINUS}, V_{PLUS}]$).

ביציאה השער יוציא ערך תקין בטווח הלוגי האמיתי שנקבע. זהו "חיזוק" האות אל הטווח הנכון.

הגדרת רמות לוגיות (כניסה ויציאה):

V_{OL} - מתח יציאה מקסימאלי לייצוג 0 לוגי

V_{IL} - מתח כניסה מקסימאלי לייצוג 0 לוגי

V_{IH} - מתח כניסה מקסימאלי לייצוג 1 לוגי

V_{OH} - מתי יציאה מינימאלי לייצוג 1 לוגי

* V_{IL}, V_{IH} מגדירים כמה שולי רעש קיימים (כאשר קיים הפרש ביניהם לבין ערכי היציאות).

המהפך- אופיין המתח (פונקצית תמסורת):

- שיפוע של יותר מ45 מעלות (מהפך אידיאלי: שיפוע של 90 מעלות).
- כל שינוי קל במתח הכניסה יביא לשינוי חזק במתח היציאה.
- המטרה: המעבר באזור האסור יתרחש בזמן קצר ככל האפשר.
- תקטן רגישות המעגל לרעש.
- הפרש מתחי הכניסה קטן הרבה יותר מהפרש מתחי היציאה $\Delta V_O > \Delta V_i$

*אם נכנס ערך לא חוקי, לא מסתכלים על היציאה.

הגדרות זמני השהייה:

t_{CD} - Contamination Delay- זמן "הזיהום"- משך הזמן מרגע שינוי הכניסה, אשר בו מובטח כי היציאה לא תשתנה עדיין.

t_{PD} - Propagation Delay- זמן ההשהיה מרגע שינוי הכניסה ועד שמובטח שהיציאה כבר השתנתה לערכה החדש.

שני סוגים של זמן השהייה:

t_{HL} - זמן ההשהיה מכניסה ליציאה כאשר היציאה 'יורדת' מ-1 ל-0 לוגי.

t_{LH} - זמן ההשהיה מכניסה ליציאה כאשר היציאה 'עולה' מ-0 ל-1 לוגי.

הבהובים סטטיים (Static Hazards):

לעיתים במהלך זמן ההשהיה (t_{PD}) עלולה היציאה לקבל ערך ביניים לא נכון (מתחילה מערך תקין, עוברת לערך לא תקין ובסוף זמן ההשהיה הערך שוב תקין).

יקרה רק במעבר מקובייה אחת לשנייה במפת קרנו ונגרם ע"י הבדלים בזמני ההתפשטות ברכיבים שונים (מכפלה אחת הפסיקה להיות תקפה לפני שהשנייה נכנסה לתוקף- היו הפרשי זמנים).

מניעת הבהובים סטטיים (Hazard Free): משנים את המעגל כך שכבר לא יהיה מצומצם. מניחים כי:

1. בו-זמנית לא משתנה יותר מכניסה אחת למעגל
2. שינויים נוספים בכניסות לא יקרו עד אשר יסתיימו כל השינויים בתוך המעגל הנובעים משינוי הכניסה האחרון.
3. מוסיפים למעגל רכיבים כך שכל זוג משבצות שכנות מכוסה ע"י אחת המכפלות (כלומר, מוסיפים קוביות נוספות למפה).

הבהובים דינמיים (Dynamic Hazards):

קורים כאשר יציאת המעגל אמורה להשתנות אבל השינוי נעשה תוך שלושה מעברים לפחות.

מסכמים, בוררים, מפענחים

מסכם בינארי מלא (Full-Adder (FA:

מחבר בין שני מספרים ובין כניסת carry-in, והוא מוציא תוצאה ו-carry out.

ב-half-adder לא נכנס carry-in, FA טוב לזכירת carry בשביל הפעולה הבאה.

Ripple Carry Adder:

שרשור n FA כדי לחבר שני מספרים בינאריים בני n סיביות. כל רכיב יחבר זוג סיביות עם הנשא מהרכיב הקודם.

זמן החישוב- כזמן התפשטות הנשא (כל הסיביות מגיעות לכל הרכיבית בזמן-0): $n\tau$.

Carry Look-Ahead Adder:

מוסיפים רכיבים ללוגיקה כדי להאיץ את זמן החישוב: מחשבים כל מה שניתן לחשב ללא הcarry-in כדי שלאחר שיגיע יידרש מספר פעולות המועט והזריז ביותר עד להוצאת carry-out לרמה הבאה.

בורר- Selector/ multiplexer/ mux:

קלט: m סיביות בקרה המייצגות מספר כלשהו בין 0 ל 2^m-1. $S_0, S_1, ..., S_{m-1}$

כניסות נתונים אשר בכל אחת מהן סיבית $D_0, D_1, ..., D_{n-1}$ ($n = 2^m$)

E כניסת enable.

פלט: אם E=0 אז '0'

אחרת: הסיבית D_i כאשר i הינו המספר המיוצג ע"י סיביות הבקרה.

*שני מהפכים: אין להם תפקיד לוגי. משמשים להגברה חשמלית של התדר (לטווח הרצוי). מיועדים להציג לכניסה "עומס" של שער יחיד, כאילו כל כניסה מזינה שער פנימי אחד בלבד ולא כמה (מקל על מלאכת התכנון של המעגל).

*בורר בלבד אינו מהווה מערכת פעולות שלמה, הוא לא יכול לממש NOT.

*בורר + ערכים קבועים בכניסות הנתונים = ROM (read-only-memory), נכתב פעם אחת ואחרי זה הוא יעשה בדיוק אותו הדבר כל-פעם.

מפענח- Decoder:

קלט: m כניסות בקרה $d_0, d_1, ..., d_{m-1}$

E כניסת enable.

פלט: אם E=0 אז בכל היציאות יש '0'. אם E=1 אז ביציאה f_i יש-1 כאשר i הינו המספר המיוצג ע"י סיביות הבקרה.

מבנים רגולריים

בניית שערים לוגיים באמצעות מתגים:

לכל מתג 3 קצוות: כניסת בקרה (C) ושני קצוות (A,B) שהמתג יכול לחבר ביניהם. שני סוגים:

מתג P (Positive): C=0 מתג: מחובר. C=1 מתג: מנותק.

מתג N (Negative): C=0 מתג: מנותק. C=1 מתג: מחובר.

בניית מהפך באמצעות מתגים:

זוג מתגים המחוברים בטור בין הקבועים '1' ו-'0' ומחוברים לאותה הכניסה-A

כאשר A=0, מתג N מנותק וP מחובר וכך עובר הקבוע-'1' ליציאה B.

כאשר A=1, מתג N מחובר ומתק P מנותק וכך עובר הקבוע '0' ליציאה B.

בניית שער NAND באמצעות מתגים:

בנוי כך שרק כאשר גם A וגם B שווים ל-'1' אז שני המתגים המובילים מה-ground מתחברים ויוצא '0' לוגי.

Three (Tri)-State Bus:

E=1: פועל כמו שער NOT רגיל

E=o: היציאה מנותקת ולא יכולה להוציא לא '0' ולא '1'.

"שער צף": לא המעגל עצמו יקבע את ערכו הלוגי אלא משהו חיצוני אחר יקבע את ערך היציאה.

כינוי: NOT עם יציאה מסוג "three-state"/ עם יציאת HIGH-Z.

- ניתן לחבר יחד יציאות של מספר שערים כאלו. היציאה המשותפת קרויה: BUS.
- לכל היותר רק שער אחד רשאי לכתוב בו-זמנית על היציאה המשותפת.
- המפענח יבטיח שלכל היותר רק אחד מן השערים יכתוב על הBUS בו-זמנית.
- אם למפענח כניסת Enable=0 אז אף שער אינו כותב על הBUS.

*המערכת כולה שקולה לבורר, אך BUS מהיר יותר כי בורר מצריך חוטים רבים ול-BUS חוט יחיד.

Wired-AND (pull-down) Bus:

BUS המורכב כולו משערי pull-down (החלק התחתון של שער NOT) פועל כלהלן:

אם יש '1' בכניסה של שער אחד או יותר, ערכו של הBUS הוא '0'. אחרת: ערכו '1'.

pull-down Bus מממש פונקציה של שער NOR.

Wired-OR (pull-up) Bus:

Bus המורכב כולו משערי pull-up (החלק העליון של שער NOT) פועל כלהלן:

אם יש '0' בכניסה של שער אחד או יותר, ערכו של הBus הוא '1'. אחרת: '0'.

pull-up Bus מממש פונקציה של שער NAND.

Programmable Logic Array (PLA):

Wired-AND Buses (המאורגנים ב"מישור AND") משמשים לחישוב המכפלות

Wired-OR Buses ("מישור OR") משמש לסיכום המכפלות ליצירת הפונקציות.

Read Only Memory (ROM):

ROM- רכיב זיכרון הכולל:

- קווי כניסה ("קווי הכתובת") לבחירת תא זיכרון מסוים.
- קו יציאה לקריאת ערכו של תא הזיכרון שנבחר.

יתרון: המידע אינו נמחק עם כיבוי המתח החשמלי לכן הROM משמש לאכסון מידע קבוע.

מימוש ROM דו-מימדי:

- בבורר של $(n+m)$ כניסות יש ס"ג של $2^m \cdot 2^n$ שערים.
- במבנה הכולל מפענח בעל n כניסות ובורר בעל m כניסות יש ס"ג של $2^m + 2^n$ שערים.
- מבחינה גיאומטרית קל יותר לממש שני התקנים (מפענח ובורר) מאשר בורר אחד גדול.
- מבנה זה מאפשר קריאת כתובות זיכרון עוקבות במהירות רבה, כאשר משנים רק את כניסות הבקרה של הבורר.
- ניתן אף לצרף מספר בוררים ולבצע קריאות בו-זמניות.

RAM (dual port: read, write):

- ע"י הוספת בורר נוסף, ניתן יהיה לבצע פעולת כתיבה ושתי פעולות קריאה בו זמנית.
- ע"י שימוש ב-n עותקים של מעגל זה, ניתן לבנות זיכרון למילים בנות n סיביות.

גילוי תקלות בזיכרונות:

שיטה-1: לכתוב לכל תא בזיכרון, לקרוא אותו, ולהשוות את מה שנכתב למה שנקרא (תהליך ארוך ויקר).

שיטה-2: מכונות מצבים המייצרות תוכן אקראי לכאורה, כותבות לזיכרון וקוראות בחזרה. מכונות אלה בנויות עם הזיכרון. השיטה נקראת: Built In Self Test (BIST).

שיטה-3: ניתן להוסיף לכל זיכרון כמות קטנה של זיכרון רזרבי: המוכנה לגילו התקלות יכולה לתכנת את הזיכרון כך שאיזור זיכרון שיש בו תקלה יוחלף בזיכרון הרזרבי- Built In Self Repair (BISR).

יחידות זיכרון

מערכות עם מהפכים:

שני מהפכים במעגל סגור. למערכת זו שני מצבים יציבים. כאשר היא תתייצב באחד מהם היא תישאר בו. זו מערכת דו-יציבה (bistable) והיא ראוייה לשמש בתור רכיב זיכרון.

מערכת רוטטת: שלושה מהפכים במעגל סגור. המערכת רוטטת (מתנדנדת) בין ערכים לוגיים בלתי יציבים.

הכללה למעגלים בעלי n מהפכים: n זוגי- המערכת דו-יציבה. n אי-זוגי- המערכת אינה יציבה.

Set-Reset Latch (SR):

מעגל עם שני מהפכים המחובר לשתי כניסות SET וRESET ששוות ל-'0'.

SET: מעלים את SET ל-'1' אז זה משתנה ל-$Q = '1', \overline{Q} = '0'$.

RESET: החלפת S ל-'0' וR ל-'1'.

גם אם SET חוזר ל-'0' היציאות לא ישתנו כי המעגל הסגור עם שני המהפכים משמש כרכיב זיכרון וממשיך להזין אתם אותם ערכים ליציאות.

*אסור להסיר את ה-'1' ב-S לפני שהמעגל התייצב.

Race: מכניסים '1' לשתי הכניסות ואז שתי היציאות נתקעות על '0'. ואז בבת-אחת: מורידים את הכניסות ל-'0' ואז נוצר מעגל אינסופי של הבהובים: עליות וירידות. במציאות: השער המהיר יותר ינצח.

Latch (מנעול): רכיבי זיכרון שכניסותיהן רשאיות להשתנות בכל עת (להוציא-race) ושהיציאות מושפעות מיידית מן הכניסות.

Set-Reset Gated Latch (SR):

ניתן לבקר מתי כניסות S, R תשפענה על ה-latch באמצעות הכניסה G=gate:

G=1: ה-latch 'פתוח' או 'שקוף'

G=0: ה-latch נעול ויציאותיו אינן מושפעות משינויים בכניסות.

אוגר הזזה:

בכל פעם ש-G עולה ל-'1' ויורד ל-'0' רוצים כי הסיביות באוגר יזוזו בדיוק מקום אחד ימינה.

אם נשתמש ב-Transparent Latch, מספר התאים דרכם יעבור הקלט תלוי באורך הזמן שבו G גבוה.

אפשר אמנם להקפיד ש-G יהיה גבוה בדיוק למשך הזמן הדרוש למעבר דרך latch אחד, אבל אז המעגל יהיה רגיש לשינויי זמן עליהם קשה לשלוט, וכל לשינויים רבים אחרים.

Edge Triggered D Flip Flop (DFF/ETDFF):

שני latch-ים מחוברים, כאשר בכל עלית שעון אחד נפתח ואחר נסגר וכאשר השעון יורד, להפך.

בכל מחזור שעון מלא, עובר הערך מ-Data ועד ליציאה.

כדי שהערך מ-Data אכן יעבור ליציאה, הוא צריך להישאר יציב פרק זמן קצר מסוים לפני שינוי השעון. ולכן, בזמן שהשעון עולה אנחנו מניחים שהכניסה כבר התייצבה ואינה עוברת שינוי בו-זמנית לעליית השעון ולכן מטילים מגבלות על זמני ההשתנות של הכניסה-Data.

*בעזרת ETDFF ניתן לבנות אוגר הזהה הפועל נכון: כל עליית שעון מקדמת את הסיביות במקום אחד.

תזמונים בפליפ-פלופים:

t_{pC-Q} - זמן התפשטות מקסימאלי מרגע עליית השעון ועד שמוצא הפליפ-פלופ מתייצב בערך החוקי.

t_{cC-Q} - הזמן המינימאלי מרגע עליית השעון בו עדיין מובטח שמוצא הרכיב יישאר יציב ברמתו הלוגית הקודמת.

t_{SETUP} - משך הזמן לפני עליית השעון בו הכניסה לפליפ-פלופ צריכה להיות יציבה.

t_{HOLD} - משך הזמן לאחר עליית השעון בו הכניסה לפליפ-פלופ צריכה להיות יציבה.

*בד"כ t_S, t_H קטנים בהרבה מן ההשהיות האחרות במערכת.

* t_{cC-Q} הוא בד"כ קטן, אולם מניחים שאיננו, אלא מקיים $t_{cC-Q} > t_H$ (כדי שנוכל לחבר פליפ-פלופים אחד לשני).

משטרי תזמון:

המשטרים מבוססים על **<u>שני סוגי רכיבים:</u>**

- רכיב צירופי (רכיב המתואר ע"י פונקצית מיתוג).
- רכיב זיכרון מתוזמן (כלומר שיש לו כניסת שעון).

המשטר הסטטי:

כל רכיב צירופי מתוכנן כל שבהינתן לו ערכי כניסה לוגיים תקינים ויציבים למשך זמן מספיק, יתקבל ביציאתו ערך לוגי תקין לאחר זמן ההשהיה המתאים.

המשטר הדינמי:

- מעגל לוגי מורכב מרכיבים צירופיים ורכיבי זיכרון מתוזמנים בלבד.
- מעגל לוגי אינו מכיל לולאות של רכיבים צירופיים. כל לולאה חייבת להכיל לפחות רכיב זיכרון מתוזמן אחד.
- מחזור השעון ארוך מספיק על-מנת לספק את דרישות הזמן של כל הרכיבים.
- בכל הכניסות, הרמות הלוגיות תהיינה יציבות למשך זמן מספיק.

מטה-סטביליות (Meta-Stability):

קיימות שתי נקודות **שיווי-משקל יציב** המייצגות ערכים חוקיים. אבל שינויים קטנים בערכי V_{OUT}, V_{IN} (רעש למשל) יוחזרו ע"י ההתקן אל נקודת שיווי המשקל התקינה.

הנקודה השלישית מערבת מתחים שאינם ערכים לוגיים חוקיים. נקודת **שיווי-משקל רופף**- שינוי קטן בערכי V_{OUT} או V_{IN} יגרום למעבר המערכת לאחת משתי הנקודות היציבות.

- המערכת עלולה להיכנס למצב על-יציב כתוצאה מאותות כניסה שאינם בעלי ערכים חוקיים, או שמשכם אינו מספיק.
- אי היציבות של נקודה זו, בשילוב עם הרעשים הטבעיים, יבאו את המערכת בסופו של דבר, לאחת משתי נקודות המשקל היציבות שלה. אולם הזמן שיעבור עד שזה יקרה הוא בלתי חסום והערך שיבחר (0 או 1) בלתי ניתן לחיזוי.

מערכות עקיבה

- נמצאות בכל רגע נתון במצב מסוים.
- המצב מיוצג ע"י ערכי הזיכרון של מערכת העקיבה.
- המערכת יכולה לעבור ממצב אחד למצב אחר, בתלות במצב ובכניסות.
- מערכת עקיבה סינכרונית יכולה לעבור ממצב אחד למצב אחר רק בזמנים מסוימים.
- המערכת מקבלת אות שעון, והשעון קובע מתי יתבצעו מעברי המצב.

FSM (Finite State-Machine):

מערכת העקיבה ממומשת ע"י מכונת מצבים סופית שרכיביה הם:

- קבוצה סופית של מצבים, אחד מהם נקבע כמצב ההתחלתי.
- קבוצה סופית של כניסות בינאריות וקבוצה סופית של יציאות בינאריות.
- פונקצית מעבר המגדירה לכל צירוף של מצב נוכחי וערכי הכניסות, את המצב הבא.
- פונקצית יציאה המגדירה לכל צירוף של מצב נוכחי וערכי הכניסות, את ערכי היציאות.
- תזמוני כניסה t_S, t_H ותזמוני יציאה t_{pC-Q}, t_{cC-Q}.

Mealy Vs Moore:

במכונת Moore היציאות הן פונקציה של **המצב הנוכחי בלבד** ולא מושפעות מהכניסות באופן ישיר. מושפעות מהכניסות אבל רק בזמן קבוע, ע"פ השעון.

במכונת Mealy היציאות יכולות להיות משופעות גם מהכניסה הנוכחית.

כללי תזמון במכונת מצבים:

משך זמן מחזור השעון T: $\boxed{T_{Clk} \geq t_{pC-Q}(FF) + t_{PD}(CL) + t_{SETUP}(FF)}$.

הוספת הלוגיקה יכולה לחבל בתנאי שהיה תקין קודם.

זמן Setup: ייקח זמן $t_{PD}(CL)$ עד שהלוגיקה הצירופית תשפיע על הNext State.

המעברים: $\begin{matrix} Present\ State \rightarrow Next\ State \\ Inputs \rightarrow Next\ State \end{matrix}$ קורים במקביל.

הכניסות למערכת הצירופית צריכות להיות תקפות בערכים הנכונים במשך $t_S(input)$

$$\boxed{t_{SETUP}(input) \geq t_{PD}(CL) + t_{SETUP}}$$

זמן Hold: צריך להיות קטן מסכום t_{CD} של הזיכרונות וה-logic: $\boxed{t_{Hold} < t_{cC-Q}(FF) + t_{CD}(CL)}$

תכנון מערכות עקיבה

שלבי התכנון של מערכת עקיבה:

1. תיאור מילולי של המערכת הנדרשת (הבנת הדרישות)
2. בנית טבלת מצבים או דיאגראמת מצבים
3. צמצום טבלת המצבים
4. בחירת הקצאת מצבים ובחירת רכיבי זיכרון (נדון בעיקר בDFF).
5. רשימת טבלת המעברים וטבלת היציאה.
6. מציאת פונקציות העירור ופונקציות היציאה
7. שרטוט המעגל המממש את המערכת.

*כאשר אחד מערכי הכניסה לא מוגדר חלק גדול מהזמן, אז גם היציאה תקפה רק בזמנים קצרים ביותר. אפשר להוסיף FF בכניסה או ביציאה על-מנת לסנכרן אותן. בדרך זו "מתקנים" את מכונת Mealy למכונת Moore מוזרה. ואז היציאה כן תהיה מוגדרת כל הזמן.

Toggle Flip-Flop:

כאשר הכניסה $T=1$, מחליף ה-FF את ערכו.

$$T=0 \quad 0\to 0,\ 1\to 1$$
$$T=1 \quad 0\to 1,\ 1\to 0$$

*כדי לממש מערכת עקיבה עם TFF: למשל כאשר יש 3 רכיבי זיכרון (3 ביטים לכניסה ול-Next State) אז בכל מקום שבו רוצים להפוך סיבית אז הT המתאים (מתוך שלושה) צריך להיות-1.

*כל מה שניתן לממש אם TFF או עם SRFF (Set-Reset FF) ניתן גם עם DFF.

תכונות ומגבלות של מערכות עקיבה

מספר המצבים הסופי מטיל מגבלה על יכולת החישוב.

ע"פ גודל המחזור של הפלט, ניתן לדעת כמה מצבים שונים יש למכונה.

משפט: מכונה בעלת מספר מצבים סופי מפיקה יציאה מחזורית תחת קלט מחזורי.

*כאשר הפלט אינו מחזורי, אז לא ניתן לבנות מכונה סופית כזו.

דוגמאות שלא ניתן לבצע ב-FSM:

- מכונה המוציאה 1 אם ורק אם מספר ה-'1' שווה למספר ה-אפסים.
- מכונה המוציאה 1 אם ורק אם מספר ה-'1' בכניסה הוא ריבוע שלם.

מערכות איטרטיביות:

מעגל בו מחוברים מספר רכיבים (צירופיים) זהים בטור. הרעיון הוא "לפרוש" מערכת עקיבה המתקדמת בציר הזמן למערכת איטרטיבית הנפרשת במרחב.

כאשר סדרת הכניסה ארוכה יותר ממספר הרכיבים אז קולטים את הקלט בשלבים, כל פעם נכנסים ביטים לכל הכניסות, קוראים את היציאות ואז מאפסים את המערכת שיכולה לקבל קלט נוסף.

שקילות מצבים וצמצום מכונות

המטרה בצמצום: בהינתן מכונה סופית, מצא מכונה המבצעת אותה משימה בדיוק בעלת מינימום מספר מצבים.

הגדרות:

בני הפרדה: שני מצבים S_i ו-S_j של מכונה M נקראים בני הפרדה (distinguishable) אם קיימת סדרת כניסה אחת לפחות (סדרת-הפרדה) של M, המספקת יציאות שונות עבור המצבים ההתחלתיים S_i ו-S_j

מצבים בני k הפרדה: שני מצבים S_i ו-S_j ייקראו k-בני-הפרדה עם קיימת עבורם סדרת הפרדה באורך-k (קיימות 2^k סדרות, ואם אחת מהן לפחות גורמת להפרדה, כאשר כל סדרה באורך $k-1$ לא גורמת להפרדה).

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.

סימון: $S_i \equiv S_j$.

יחס שקליות מקיים את התכונות הבאות: 1. רפלקסיביות- $S_i \equiv S_i$. 2. סימטריות- $S_i \equiv S_j \rightarrow S_j \equiv S_i$. 3. טרנזיטיביות- $S_i \equiv S_j$, $S_j \equiv S_k \rightarrow S_i \equiv S_k$.

מחלקות שקילות: 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 קובצה ריקה (המחלקות זרות הדדית).

*כל מחלקת שקילות "נחליף" במצב אחד.

האלגוריתם של Moore לצמצום מכונה:

1. נפתח בקבוצת כל המצבים שהם 0-שקולים: $P_0 = (ABCDEF)$.
2. נבצע חלוקה על P_0 למצבים 1-שקולים: $P_1 = (ACE)(BDF)$.

מצב 0-עוקב (0-successor) של S_i - המצב שעוברים אליו מ-S_i בגין כניסה 0.

מצב X-עוקב (X-successor) של S_i - מצב שמגיעים אליו מ-S_i תחת סדרת הכניסות-X.

3. נמצא את מחלקות המצבים ה-k-שקולים שהם:

$(k-1)$ שקולים. גם המצבים העוקבים שלהם הינם $(k-1)$ שקולים.

נקבל חלוקה P_2 שהיא עידון של P_1 - חלוקות נוספות של המחלקות הקיימות.

4. תנאי עצירה: $P_{K+1} = P_K$. P_K תיקרא <u>חלוקת השקילות</u>.

<u>משפט:</u> חלוקת השקילות P_K היא יחידה.

<u>משפט (תכונת העצירה של אלגוריתם Moore):</u> 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.

<u>הוכחה:</u> אם $i < j$ אז P_j מכילה (לפחות) מחלקה אחת יותר מ-P_i ומספר המחלקות המקסימאלי האפשרי הינו-n, כמספר מצבי המכונה.

<u>שקילות בין מכונות:</u>

<u>הגדרה:</u> שתי מכונות M ו-M' תיקראנה שקולות אם לכל מצב ב-M קיים מצב שקול מתאים ב-M' ולהפך.

*בהינתן מכונה M, נמצא מכונה M^* השקולה ל-M ובעלת מספר מצבים מינימאלי. M^* תיקרא הצורה מהמינימאלית/המצומצמת של M. כל מצב ב-M^* יתאים למחלקת שקילות אחת ויחידה בחלוקת השקילות של-M.

<u>מכונות איזומורפיות:</u> שתי מכונות זהות הנבדלות רק בשמות המצבים נקראות איזומורפיות (שוות צורה).

גילוי תקלות

סוגי בדיקות:

1. בדיקה לגילוי תקלות- מטרתה לקבוע האם המעגל תקין או לא.
2. בדיקה לאיתור התקלות- מטרתה לקבוע היכן בדיוק בתוך המעגל נמצאת התקלה.

סוגי תקלות:

1. "צומת תקוע ב-0" (s-a-0, stuck-at-0)- תקלה לפיה חוט מסוים במעגל תקוע במצב לוגי 0.
2. "צומת תקוע ב-1" (s-a-1, stuck-at-1)- תקלה לפיה חוט מסוים במעגל תקוע במצב לוגי 1.

סוגים נוספים:

- "צומת מנותק" (stuck open)- חוט מנותק
- "גישור" (bridging)- שני חוטים שונים מחברים זה אל זה ואחד מהם קובע את הערך הלוגי של שניהם.
- "השהייה"- שער מגיב הרבה יותר לאט מהצפוי.

ניסוי ובדיקה לגילוי תקלות:

הניסוי מורכב ממספר בדיקות. בכל בדיקה מגישים למעגל צירוף אחד בלבד של הכניסות ובודקים את היציאות. מטרת הניסוי לבחון האם המעגל עובד נכון עבור כל קבוצת הבדיקות המוכלות בניסוי.

טבלת תקלות:

שורה לכל בדיקה אפשרית ועמודה לכל תקלה אפשרית, x במשבצת המתאימה אם בדיקה מסוימת תעזור לגלות תקלה מסוימת. מציאת קבוצה מינימאלית של שורות בטבלה כל שבכל עמודה יהיה x אחד לפחות. נאמר שקבוצה זו (השורות=בדיקות) "מכסה" את טבלת התקלות.

בדיקות חיוניות: שורות שבהן יש X בעמודה כלשהי שלא מופיע עבור שורות אחרות.

מגבלות השיטה:

הטבלה עלולה להיות גדולה מידי עבור מעגלים גדולים (אין הבטחה שלא נזדקק לכל 2^n השורות בטבלה).

אם נגביל את עצמנו למספר מסוים של בדיקות, צריך לבחור בניסוי שיגלה את המספר הגבוה ביותר של תקלות.

"כיסוי תקלות": אחוז התקלות שהניסוי מגלה מתוך סך כל התקלות האפשריות.

גילוי תקלות במערכות עקיבה:

כדאי לפרק מערכות עקיבה למעגלים צירופיים ולרכיבי זיכרון ולבדוק כל אחד לחוד.

שיטת הסריקה (Scanned FF): מחליפי כל FF ב-Scanned FF ומחברים אותם בשרשרת אחת, ההופכת אותם לרגיסטר הזזה אחד. מחברים כניסת בקרה SCAN CONTROL לכל ה-FF-ים.

<u>מהלך הסריקה:</u> 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 <u>למשך מחזור שעון אחד</u>, ובו 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, וכן הלאה.

תוספות מתרגולים:

ייצוג מספרים בעלי סימן

	גודל וסימן	1's complement	2's complement
טווח ייצוג	$-(2^{N-1}-1) \le A \le 2^{N-1}-1$	$-(2^{N-1}-1) \le A \le 2^{N-1}-1$	$-2^{N-1} \le A \le 2^{N-1}-1$
יתרונות	קלה לביצוע	חיבור/חיסור פשוטים	חיבור/חיסור פשוטים. ייצוג בודד לאפס.
חסרונות	סיבוך פעולות חשבון. ייצוג כפול לאפס.	ייצוג כפול לאפס	

קודים לקידוד ספרות:

קודים המשלימים את עצמם: "המשלים ל-9" של כל ספרה מתקבל ע"י הפיכת אחדים לאפסים.

תנאי הכרחי: סכום משקלי הקוד הוא:9.

קודים לגילוי ותיקון שגיאות:

קוד: אוסף של מילים מתוך אוסף כל המילים האפשריות.

מרחק בין שתי מילות קוד: מספר הסיביות שיש לשנות על-מנת לקבל מילה אחת מהשנייה.

מרחק קוד: המרחק הקטן ביותר שבין שתי מילות קוד כלשהן.

גלוי ותיקון: אם המרחק המינימאלי הוא K אז ניתן לגלות עד k-1 שגיאות.

אם מניחים שנפלו לכל היותר (k-1)/2 שגיאות אז ניתן גם לתקן אותן (נוכל לתקן על המילה הלא תקנית קרובה יותר לאחת המילים החוקיות ולא קרובה לשתי מילים חוקיות באותה המידה).

*הוספת סיבית זוגיות: מגדילה את מרחק הקוד ב-1.

מערכת פעולות שלמה:

הגדרה: קבוצת פעולות נקראת שלמה אם ניתן להציג בעזרתה את כל הפונקציות הלוגיות.

כלל אצבע: כדי לשלול שלמות של פונקציה נציב משתנה אחד בכל הכניסות של הפונקציה. אם התוצאה אינה היפוכו של המשתנה אזי בהכרח הפונקציה אינה שלמה (לא עובד בכיוון ההפוך).

חצי שלמה: מערכת שהינה שלמה רק בתוספת קבועים (0 או 1 או שניהם).

פונקציה שומרת אפסים: ערך הפונקציה הוא אפס כאשר כל ערכי הכניסה של הם אפס.

פונקציה שומרת אחדים: ערך הפונקציה הוא אחד כאשר כל ערכי הכניסה שלה הם אחד.

*כל הרכבה של פונקציות שומרות אפסים/אחדים היא גם כזו.

פונקציה ליניארית: אם ורק אם ניתן לבטאה כך: $f(x_1,\ldots,x_n) = a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n$

בדיקת ליניאריות: כל משתנה x_i מקיים אחת משתי אפשרויות:

1. אם $a_i = 0$ אז x_i לא משפיע על ערך הפונקציה ללא תלות בערכי המשתנים האחרים.
2. אם $a_i = 1$ אז שינוי בערכו של x_i תמיד ישנה את ערך הפונקציה, ללא תלות בערכי המשתנים האחרים (בגלל התכונות של XOR).

אנליזה של מערכות סינכרוניות:

1. מציאת פונקציות הכניסה לרכיבי הזיכרון, ופונקציות המוצא.
2. טבלת מצבים (מעברים).
3. טבלת מצבים סימבולית- כינוי המצבים "שמות" (A, B, ...).
4. דיאגראמת מצבים (מעברים).
5. בעזרת סדרת בוחן עבור הכניסה מנסים להבין, מה המערכת עושה.

סינתזה של מערכות סינכרוניות:

1. דיאגראמת מצבים (+ צמצום מצבי יתר).
2. טבלת מצבים (+ צמצום).
3. הקצאת מצבים (קוד לכל מצב).
4. טבלת מעבר ופלט (הצבת הקודים בטבלת המצבים).
5. מציאת הפונקציות לכניסות הFF ולמוצא Z.
6. שרטוט המעגל.

מגבלות של מערכות עקיבה:

הגדרה: נאמר שמערכת עקיבה סינכרונית מקבלת סדרה, אם כאשר היא מופעלת מן המצב ההתחלתי שלה ומוזנת בסדרה זו, היא מייצרת פלט 1 עם קבלת התו האחרון של הסדרה בקלט.

דוגמא: מכונה שמקבלת פלינדרום.