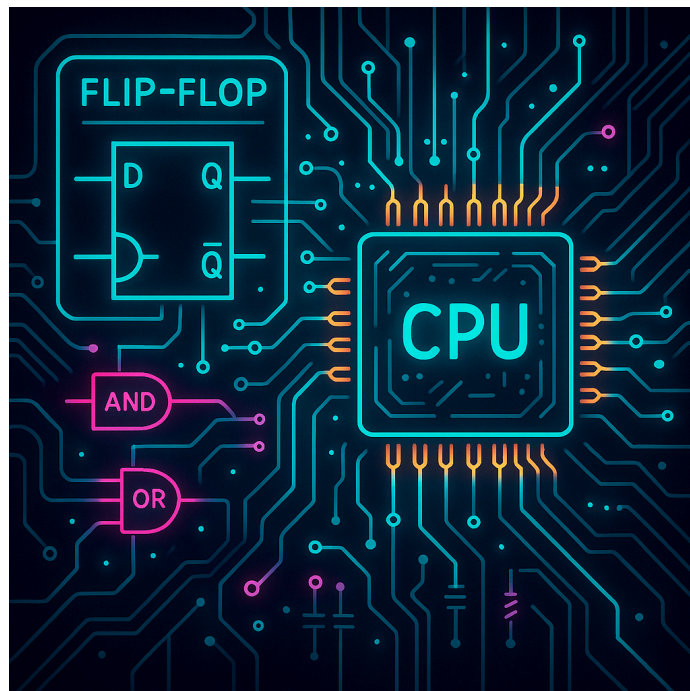


LAB 2

ADVANCED CPU ARCHITECTURE AND HARDWARE ACCELERATORS LAB

IDO RON : 322384330

BENYAMIN OUMANSKY: 322688946



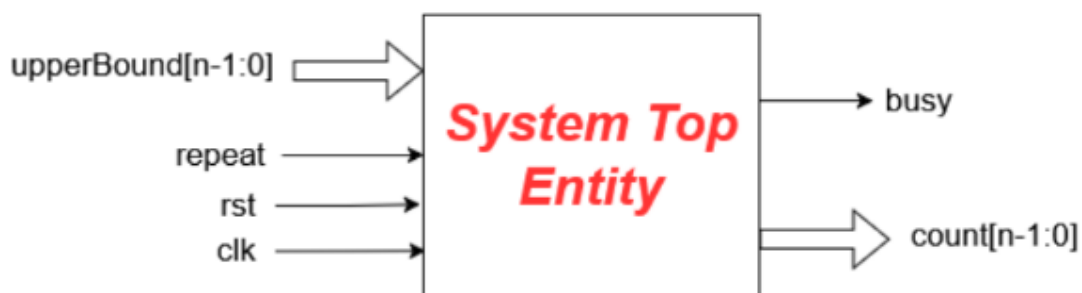
מבוא

במעבדה זו המטרה היא התמקדות בקוד רציף (Sequential logic) לעומת מעבדה 1 בה בוצע שימוש בקוד מקבילי (Combinational logic).

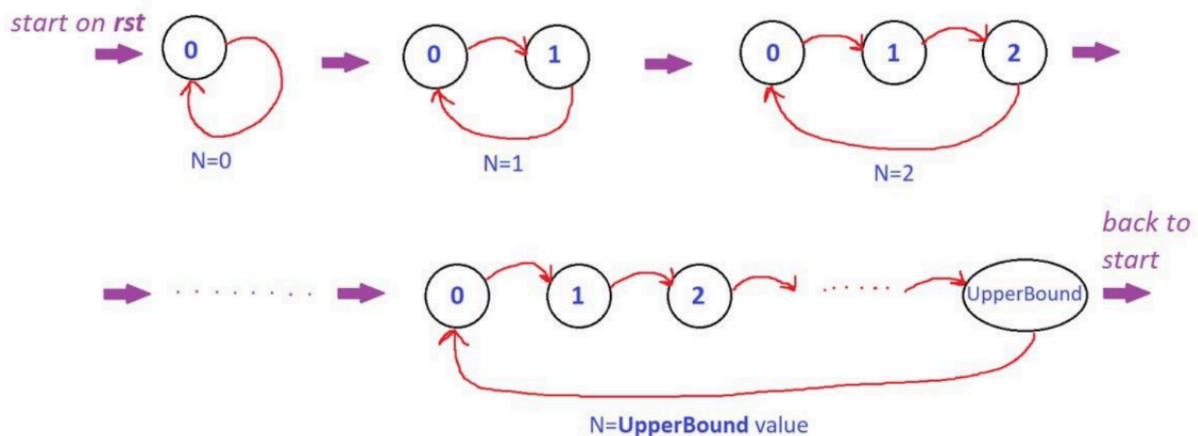
במשימה זו נממש מערכת דיגיטלית סינכרונית המקבלת 3 קלטים: וקטור כניסה של ערך מקסימלי: "Upperbound [n-1:0]", אות שאחראי על איפוס "rst", אות כניסה של שעון "clk" ואות שאחראי על חזרה "repeat".

המערכת תוציא ביט שמסמל אם היא עסוקה "busy" ומונה "Count[n-1,0]" אופן הפעולה באופן כללי: המערכת מיישמת מונה שגדל באופן דינמי (כתלות במונה שני) בכל סבב עד שהוא מגיע לערך UpperBound, כאשר הקלט repeat שווה ל 1 אז המונה מתאפס וחוזר שוב על הפעולה של הספירה.

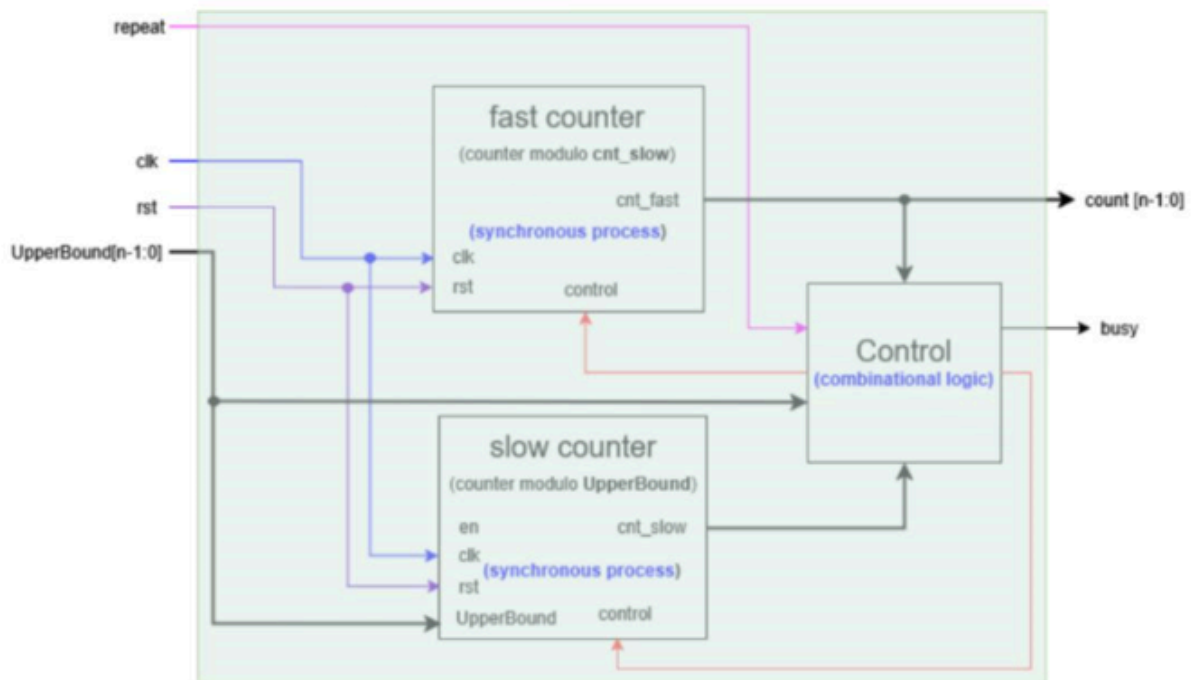
המערכת:



תיאור של המערכת על ידי דיאגרמת מצבים:



תרשים בלוקים של מודול מערכת:



System top entity:

Inputs:

- Repeat
- Rst
- Clk
- Upperbound

Outputs:

- Count
- Busy

זהו הרכיב היחיד במערכת, כלומר לא מימשנו עוד קומפוננטות בניגוד למעבדה הראשונה בה ה- TOP היה מורכב מרכיבים. כאן מימשנו מודולים: שני מונים באמצעות Process כלומר Sequential logic ואת Control באמצעות Combinational logic.

Fast counter (module 1):

Input :

- Clk
- Rst
- Control

Output :

- Count

המונה המהיר : החסם שלו הוא מוצא המונה האיטי (שאותו הוא מקבל מה control), המונה המהיר סופר כל פעם עד שמגיע לערך המונה האיטי, כך עד שהמונה האיטי מגיע לחסם העליון של המערכת כולה. המונה המהיר מקבל מידע מבלוק ה - Control אשר קובע את פעולתו (הסבר בהמשך). המוצא של המונה המהיר הוא גם מוצא המערכת כולה.

Slow Counter (module 2):

Input :

- Cnt_slow
- Cnt_fast
- Upperbound
- Repeat

Output:

- Cnt_slow

המונה האיטי:

המונה האיטי סופר עד ה - Upperbound ומתקדם ב - 1 בכל פעם שהמונה המהיר מגיע אליו. במידה והגיע אל ה- Upperbound (או עבר אותו) הוא מתאפס כתלות בביט ה- repeat ובמידע מבלוק ה - Control אשר קובע את פעולתו (הסבר בהמשך).

Control (module 3):

Inputs:

- Repeat
- Rst
- Clk
- Upperbound

Outputs:

- Busy

התפקיד של Control:

הוא לגשר בין שני המונים: שני המונים שולחים בכל עליית שעון את ערכם כסיגנל ל-control.

control שולח למונה המהיר את ערכו של המונה האיטי כך שהמונה המהיר ידע עד לאן עליו לספור. בנוסף הוא שולח למונה המהיר עוד ביט שאחראי להגדיר לו שאם ערכו ('1') עליו להמשיך לספור ברגע שמגיע למונה האיטי, כלומר לאפס את עצמו ולהמשיך לספור שוב או במקרה שערכו ('0') לעצור ברגע שיגיע לערכו של המונה האיטי.

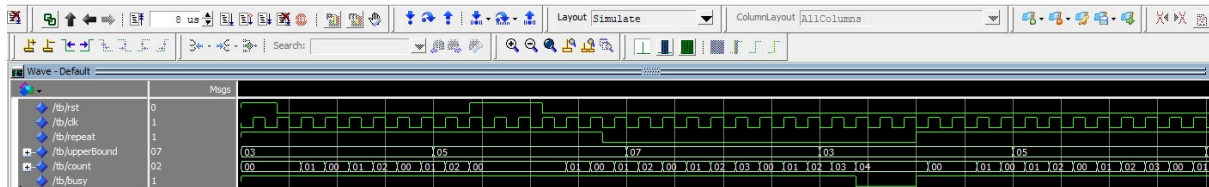
ה-control שולח סיגנל של ביט בודד למונה האיטי שתפקידו להגדיר לו אם להמשיך ('1') בספירה ברגע שהמונה האיטי הגיע אליו או אם לעצור ('0'). ערכו של סיגנל זה תלוי באם המונה המהיר הגיע לערכו של המונה האיטי, בחסם העליון וב-repeat. אם המונה המהיר הגיע למונה האיטי והמונה האיטי הגיע לחסם (או אפילו עבר אותו במידה והחסם קטן תוך כדי הריצה) ו-repeat כבוי אז עליו לשדר למונה האיטי 0 אבל אם repeat דלוק אז עליו לשדר 1 והמונה האיטי צריך להתחיל את הספירה מחדש.

בנוסף, ה-control אחראי על מוצא המערכת busy כך שכל עוד המונה המהיר סופר אז ביט זה הוא '1' וכאשר הוא עוצר אז ביט זה הינו '0'.

תוצאות הסימולציה:

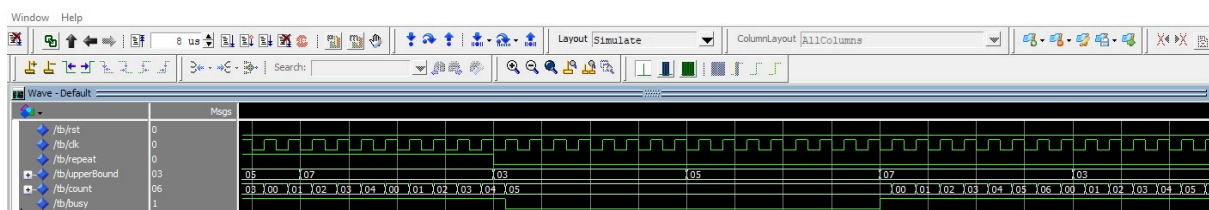
ביצענו בדיקות תרחישי קצה שונים על מנת לאמת את תקינות המערכת:

חלק א':



ניתן להבחין כי בתחילת הפעולה ביט ה- rst דלוק ולכן מוצא המערכת הינו 1. ה Upperbound מוגדר בהתחלה להיות 03 ולכן ברגע שביט ה- rst מתאפס המערכת מתחילה במנייה. לאחר זמן ה- Upperbound עולה ב-2 ל- 05 אך זה לא משפיע על המנייה כי היא עוד לא הגיע ל- 03. נשים לב כי ביט ה- RST עולה ולכן מוצא המערכת מתאפס שוב. לאחר מכן ברגע שה- rst מתאפס המערכת מתחילה לספור שוב וה- Upperbound עולה ב-2 ל- 07. מקרה קיצון: ברגע מסוים החסם העליון יורד ל-03 אבל המונה האיטי נמצא כבר על 04 (ניתן להבחין זאת מכיוון שהמוצא של המערכת שהוא המוצא של המונה המהיר כבר ספר עד ל- 03 וכעת אמור לספור עד 04) בגלל שהחסם ירד אז המונה המהיר ממשיך לספור עד שיגיע למונה האיטי שכרגע עומד על 04 ולאחר מכן תעצר עד שידלוק ביט ה- repeat ולכן המערכת תתחיל את הספירה מחדש מכיוון שהיא עברה את החסם העליון הנוכחי שלה.

חלק ב' (המשך):



כעת המערכת סופרת והחסם העליון שלה הוא 05 ולאחר מכן 07. וברגע מסוים הוא יורד ל- 03 אך כמו מקודם בגלל שהמונה האיטי עומד על 05 אז המונה המהיר (המוצא) ימשיך לספור עד שיגיע ל- 05 ואז המערכת תפסיק לעבוד וביט ה- BUSY יתאפס. עד שהחסם העליון יעלה ויגיע ל- 07 והמערכת תמשיך לספור.