|  |  |
| --- | --- |
| [Logo](http://sites.google.com/site/simple086/) | ראש הטופס      חיפוש באתר זה  תחתית הטופס |
|  | |
| **ספציפיקציות וישבון**   |  |  |  | | --- | --- | --- | | |  | | --- | | [פרוייקט חומרה פתוחה](http://sites.google.com/site/simple086/Home) | | **ספציפיקציות וישבון**  ארכיטקטורת וישבון למערכות על שבב (The WISHBONE SoC Interconnect Architecture) היא ספציפיקציות ליצירת ממשק חיבור בין ליבות IP. הספציפיקציות נכתבו במקור על ידי .Silicore Corp, וכיום התקן מתוחזק על ידי [OpenCores](http://www.opencores.org/opencores,wishbone).   **יתרונות**  תקן וישבון לא מוגן בזכויות יוצרים, והוא שוחרר לרשות הציבור (public domain). הפצתו והעתקתו מותרת בכל אמצעי. מעבר לכך, ניתן להשתמש בו לעיצוב וייצור של מעגלים משולבים בלי תמלוגים או התחייבויות כספיות אחרות.  על פי וישבון, כל הליבות מחוברות באמצעות אותו ממשק סטנדרטי, מה שמפחית בעיות אינטגרציה. עם זאת, אפשר לממש באותה מערכת שני ממשקי וישבון נפרדים, אחד לליבות בעלות ביצועים גבוהים והשני לליבות עם ביצועים נמוכים כדי למצות יותר טוב את היכולות של הליבות.  **הספציפיקציות** ערוכות בשיטה של מיון לחוקים, רשויות (דרכי פעולה אופציונליות), הצעות ותצפיות. המידע שמופיע כאן הוא תרגום של עיקרי הדברים שחובה להתייחס אליהם בעת תיכנון של ליבה עם ממשק וישבון. התרגום לא מלא, ובמקום להעתיק שוב מידע שמופיע בפעם השנייה בשינוי קל, פשוט כתבתי מה השינוי. את הספציפיקציות המלאות אפשר למצוא בקישור [WISHBONE, Revision B.3 Specification](http://www.opencores.org/downloads/wbspec_b3.pdf) (קובץ PDF). הציורים שמופיעים בהמשך הדף נלקחו משם.  **תוכן** [פרק 3 - מחזור באס קלאסי של וישבון](http://sites.google.com/site/simple086/wishbone-spec#3)     [3.1 פעולה כללית](http://sites.google.com/site/simple086/wishbone-spec#3.1)         [3.1.1 פעולת איתחול](http://sites.google.com/site/simple086/wishbone-spec#3.1.1)         [3.1.2 איתחול מחזור העברה](http://sites.google.com/site/simple086/wishbone-spec#3.1.2)         [3.1.3 פרוטוקול לחיצת היד](http://sites.google.com/site/simple086/wishbone-spec#3.1.3)         [[3.1.3] שימוש ב- STB\_O](http://sites.google.com/site/simple086/wishbone-spec#%5B3.1.3%5D)    י         [3.1.4 שימוש ב- ACK\_O, ERR\_O וב- RTY\_O](http://sites.google.com/site/simple086/wishbone-spec#3.1.4)   י         [3.1.5 שימוש בתגים](http://sites.google.com/site/simple086/wishbone-spec#3.1.5)     [3.2 מחזורים מסוג מחזור קריאה\כתיבה יחיד](http://sites.google.com/site/simple086/wishbone-spec#3.2)         [3.2.1 מחזור קריאה יחיד](http://sites.google.com/site/simple086/wishbone-spec#3.2.1)         [3.2.2 מחזור כתיבה יחיד](http://sites.google.com/site/simple086/wishbone-spec#3.2.2)     [3.3 מחזור קריאת\כתיבת בלוק](http://sites.google.com/site/simple086/wishbone-spec#3.3)         [3.3.1 מחזור קריאת בלוק](http://sites.google.com/site/simple086/wishbone-spec#3.3.1)         [3.3.2 מחזור כתיבת בלוק](http://sites.google.com/site/simple086/wishbone-spec#3.3.2)     [3.4 מחזור קרא-עדכן-כתוב](http://sites.google.com/site/simple086/wishbone-spec#3.4) [פרק 5 - ספציפיקציות תזמון](http://sites.google.com/site/simple086/wishbone-spec#5)   **פרק 3 - מחזור באס קלאסי של וישבון**  **3.1 פעולה כללית** המאסטר והעבד מחוברים בסט של אותות שמאפשרים להם לחליף בינהם מידע. הסט מכונה "באס" והוא חלק ממודל הנקרא INTERCON.  **3.1.1 פעולת איתחול (reset)** כל ממשקי החומרה מאותחלים למצב מוגדר מראש על ידי האות RST\_O (אותו אפשר להעלות בכל זמן, אם כי הפעולה שלו למעשה סינכרונית- מכונות מצבים ומונים מאתחלים את עצמם בעליית השעון הבאה). הוא משמש גם בשביל סימולציות בדיקה ע"י איתחול כל מכונות המצבים והמונים של העיצוב (design). אות האיתחול RST\_O מונע ע"י מודול SYSCON. הוא מחובר לכניסת RST\_I של כל ממשקי המאסטרים והעבדים. מחזור איתחול:  [http://sites.google.com/site/simple086/_/rsrc/1255943660339/wishbone-spec/3.1.gif](http://sites.google.com/site/simple086/wishbone-spec/3.1.gif?attredirects=0)  חוק 3.00: כל ממשקי וישבון חייבים לאתחל את עצמם בעליית השעון CLK\_I שלאחר עליית RST\_I. הם חייבים להשאר במצב המאותחל עד העלייה של CLK\_I שלאחר הירידה של RST\_I.  חוק 3.05: האות RST\_I חייב להיות בגבוה במשך לפחות מחזור שעון שלם של כל ממשקי הוישבון.  רשות 3.00: האות RST\_I יכול להיות גבוה למשך יותר ממחזור שעון אחד, ואפילו לנצח.  חוק 3.10: כל ממשקי וישבון חייבים להיות מסוגלים להגיב לאות RST\_I בכל זמן.  חוק 3.15: כל מכונות המצבים והמונים בממשקים של וישבון חייבים לקיים את חוק 3.00  תצפית 3.00: באופן כללי, מכונות מצבים לא חייבות לעבור איתחול, אלא שעבודה ללא איתחול יכולה לגרום לבעיות מכיוון שלחלק מכלי הסימולציה קשה למצוא מצב התחלתי למכונות האלו. מעבר לכך, מכונות מצבים כאלו יכולות לעבור מספר לא מוגדר של מצבי איתחול לפני שהן מגיעות לשלב ההתחלתי (starting state) שלהן, מה שמקשה על צפיית ההתנהגות שלהן בזמן ההפעלה. חוק האיתחול מונע את שתי הבעיות הללו ע"י הכרחת המכונות להיות במצב ידוע בזמן ידוע יחד עם שאר המערכת.  חוק 3.20: אותות המאסטר הבאים חייבים לרדת בעליית השעון שאחרי עליית RST\_I, וחייבים להישאר בנמוך עד עליית CLK\_I הבאה אחרי ירידת RST\_I: האות STB\_O והאות CYC\_O. המצב של כל שאר אותות המאסטר לא מוגדר ביחס לאות האיתחול.  תצפית 3.05: בממשק המאסטר STB\_O ו- CYC\_O יכולים להיות מועלים בעליית CLK\_I שאחרי ירידת RST\_I.  תצפית 3.10: ממשק העבד מוריד באופן אוטומטי את ACK\_O, ERR\_O ו- RTY\_O כאשר STB\_I שלהם יורד.  המלצה 3.00: עצב את מודולי SYSCON כך שהם יעלו את RST\_O בזמן מצב power-up. האות RST\_O צריך להישאר בגבוה עד שכל רמות המתח ותדירויות השעון במערכת יציבים. כאשר אתה מוריד את RST\_O , עשה זאת באופן סינכרוני שבהתאמה לספציפיקציות האלו.  תצפית 3.15: אם שעון ממושער (gated) נמצא בשימוש, ואם השעון נעצר, אז ממשק וישבון לא מסוגל להגיב לאות RST\_I שלו.  הצעה 3.00: מעגלים מסויימים דורשים יכולת איתחול א-סינכרונית. אם ליבה, או רכיב SOC אחר דורשים איתחול א-סינכרוני, הגדר את האות כ- nonWishbon. הדבר ימנע בלבול עם אות האיתחול RST\_I של וישבון, שמשתמש בפרוטוקול סינכרוני "טהור", ושצריך להיות ממומש לממשק וישבון בלבד.  תצפית 3.20: כל ממשקי וישבון מגיבים לאות האיתחול. למרות זאת, הליבה המחוברת לממשק וישבון לא חייבת להגיב לאות האיתחול.   **3.1.2 איתחול מחזור העברה** ממשק המאסטר מתחיל מחזור העברה על ידי העלאת CYC\_O. כאשר CYC\_O מורד, כל אותות המאסטר האחרים הם לא ולידיים. ממשק עבד מגיב לאותות של עבד אחר רק כאשר CYC\_I בגבוה. אותות SYSCON ותגובות לאותות SYSCON לא מושפעים.  חוק 3.25: ממשקי מאסטר חייבים להעלות את CYC\_O, במשך מחזורי קריאה\כתיבה יחידים (single read/write), מחזורי העברת בלוק או מחזורי קרא-עדכן-כתוב (RMW). האות CYC\_O חייב להיות בגבוה לא יאוחר מהשפה העולה של האות CLK\_I שמבשרת את עליית STB\_O. האות CYC\_O חייב לרדת לא לפני השפה העולה של CLK\_I שמבשרת את הורדת STB\_O.  רשות 3.05: ממשקי מאסטר רשאים להשאיר את CYC\_O בגבוה לנצח.  המלצה 3.05: לוגיקת ארביטרציה משתמשת לעיתים ב- CYC\_I כדי לבחור בין ממשקי מאסטר. השארת CYC\_O בגבוה יכולה לגרום לבעיות ארביטרציה, ולכן הרעיון לא מומלץ.  חוק 3.30: לממשקי עבד אסור להגיב לאותות עבד כלשהם כאשר CYC\_I בנמוך. לעומת זאת, ממשקי עבד חייבים להגיב תמיד לאותות SYSCON   **3.1.3 פרוטוקול לחיצת היד** כל מחזורי הבאס משתמשים בפרוטוקול לחיצת היד בין ממשקי המאסטר והעבד. כפי שמוצג בציור 3-2, המאסטר מעלה את STB\_O כאשר הוא מוכן להעביר מידע. STB\_O נשאר בגבוה עד שהעבד מעלה את אחד מאותות סיום המחזור ACK\_I, ERR\_I או RTY\_I. בכל שפה עולה של CLK\_I אות הסיום נדגם, ואם הוא בגבוה האות STB\_O מורד. הדבר נותן לשני הממשקים, המאסטר והעבד, את האפשרות לשלוט על קצב העברת המידע.  [http://sites.google.com/site/simple086/_/rsrc/1255943797106/wishbone-spec/3.2.gif](http://sites.google.com/site/simple086/wishbone-spec/3.2.gif?attredirects=0)  רשות 3.10: אם ידוע שהעבד מסוגל לעמוד בקצב של כל ממשקי המאסטר ואם האותות ERR\_I ו- RTY\_I לא בשימוש, אז האות ACK\_O של העבד יכול להיקשר ל AND הלוגי של כניסות העבד STB\_I ו- CYC\_I. הממשק יתפקד באופן נורמלי בנסיבות אלו.  תצפית 3.25: ממשקי עבד מעלים אות סיום בתגובה ל- STB\_I. עם זאת, STB\_I ולידי רק כאשר CYC\_I ולידי.  חוק 3.35: אותות סיום המחזור ACK\_O, ERR\_O ו- RTY\_O חייבים להיות מיוצרים בתגובה ל-AND לוגי של CYC\_I ו- STB\_I.  רשות 3.15: אותות נוספים, מלבד CYC\_I ו- STB\_I, יכולים להיות כלולים בייצור של אותות סיום מחזור.  תצפית 3.30: גם אותות עבד פנימיים קובעים איזה אות סיום מחזור יועלה- ומתי.   רוב הדוגמאות במסמך הספציפיקציות הזה מתארות את השימוש באות ACK\_I לסיום מחזור הבאס המקומי. עם זאת, העבד יכול לסיים את המחזור גם עם שני האותות האחרים.  כל ממשקי המאסטר כוללים את ACK\_I. העלאת האות בזמן מחזור באס גורמת לסיומו הנורמלי.  העלאת ERR\_I בזמן המחזור תסיים אותו, ומשמש כהודעה למאסטר שארעה בעיה בזמן המחזור. האות משמש בדרך כלל כאשר הלוגיקה של העבד זיהתה שגיאה. למשל, אם העבד הוא זכרון מוגן סיבית זוגיות, אז ERR\_I יכול לעלות לגבוה אם שגיאת זוגיות ארעה. הספציפיקציות של וישבון לא מכתיבות מה המאסטר צריך לעשות במקרה שגיאה (בתגובה ל ERR\_I).  העלאת האות RTY\_I בזמן מחזור באס תסיים אותו, ותשמש כהודעה למאסטר שיש לנטוש את המחזור הנוכחי ולנסות שנית מאוחר יותר. האות משמש בעיקרון עבור זכרונות משותפים וגשרי באס. במקרים אלו, מעגל העבד מעלה את RTY\_I אם משאב מקומי תפוס. הספציפיקציות של וישבון לא מכתיבות מה המאסטר צריך לעשות במקרה של צורך לנסות שנית מאוחר יותר (בתגובה ל RTY\_I).   חוק 3.40: כמינימום, ממשק המאסטר חייב לכלול את האותות הבאים: ACK\_I, CLK\_I, CYC\_O, RST\_I ואת STB\_O. כמינימום, ממשק העבד חייב לכלול את האותות הבאים: ACK\_O, CLK\_I, CYC\_I, STB\_I ואת RST\_I. כל האותות האחרים אופציונליים.  רשות 3.20: ממשקי מאסטר ועבד יכולים לתמוך באותות ERR\_I (מ) וב- ERR\_O (ע) בהתאמה.  רשות 3.25: ממשקי מאסטר ועבד יכולים לתמוך באותות RTY\_I (מ) וב- RTY\_O (ע) בהתאמה.  חוק 3.45: אם עבד תומך באותות ERR\_O או RTY\_O, אז אסור לעבד להעלות יותר מאחד האותות הבאים בכל זמן נתון: ACK\_O, ERR\_O או RTY\_O.  תצפית 3.35: אם עבד תומך ב- ERR\_O או RTY\_O, אבל המאסטר לא, יכול להיווצר קיפאון (deadlock).  המלצה 3.10: תכנן מודול INTERCON שימנע מצב של קיפאון. פתרון אחד לבעיה הוא לכלול כלב שמירה (watchdog), פונקציית טיימר שמנטרת את אות STB\_O של המאסטר, ומעלה את ERR\_I או RTY\_I אם מחזור עבר את מגבלת הזמן המוקצבת (לא פותר את תצפית 3.35....). אפשר לתכנן מודולי INTERCON שינתקו ממשקים מבאס וישבון אם הם מייצרים שגיאות באס ו\או חריגות זמן של כלב השמירה.  המלצה 3.15: תכנן את ממשקי המאסטר של וישבון כך שאין שערים לוגיים בין פליפ-פלופים (registered flip-flop) ואותות המוצא STB\_O ו- CYC\_O. השהיית זמן עבור שני אלה היא בדר"כ המסלול הקריטי של המערכת. הדבר מונע מתכנון מרושל מלהאט את החיבור בגלל השהיות נוספות על שני האותות הללו.  חוק 3.50: ממשקי עבד חייבים להיות מתוכננים כך שהאותות ACK\_O, ERR\_O ו- RTY\_O מועלים ומורדים בתגובה לעלייה וירידה של STB\_I.  רשות 3.30: העלאה של ACK\_O, ERR\_O ו- RTY\_O רשאית להיות א-סינכרונית יחסית לאות CLK\_I (כלומר יש מסלול לוגיקה קומבינטורית בין STB\_I לבין ACK\_O).  תצפית 3.40: העלאה א-סינכרונית של ACK\_O, ERR\_O או RTY\_O מבטיחה שהממשק יכול להשלים מעבר מידע אחד פר מחזור שעון. יתרה מכך, היא מפשטת את התכנון של ארביטרים (arbiters) ביישומים רב-מאסטריים.  תצפית 3.45: העלאה א-סינכרונית מהתצפית הקודמת יכולה להיות בלתי אפשרית ליישום. למשל, מצבי המתנה של עבד הכי קל לממש באמצעות אות ACK\_O ברגיסטר (registered).  תצפית 3.50: בעיצובים גדולים במהירות גבוהה, העלאה א-סינכרונית כזאת יכולה לגרום להשהיות בלתי מתקבלות, שיגרמו על ידי השהיות חוזרות מהמאסטר אל העבד ובחזרה. שימוש באותות מרגוסטררים מקטין באופן משמעותי את ההשהיות המוחזרות במחיר של מצב המתנה נוסף בכל העברה. (לספציפיקציות וישבון יש פרק ספציפי שמתייחס לנושא ומגדיר מחזורים מסוג לא קלאסי, wishbone registered feedback bus cycles).  רשות 3.35: בנסיבות מסויימות, ממשקי עבד יכולים להיות מתוכננים כך שיחזיקו את ACK\_O בגבוה. למשל בחיבור נקודה-לנקודה (point-to-point) כשיש עבד אחד במערכת והוא מתפקד ללא מצבי המתנה.  חוק 3.55: ממשקי מאסטר חייבים להיות מתוכננים כדי לתפקד באופן נורמלי כאשר ממשק העבד מחזיק את ACK\_I בגבוה.   **3.1.3 שימוש ב- STB\_O** (טעות המספור במקור) חוק 3.60: ממשקי מאסטר חייבים לתאם עם STB\_O, את האותות הבאים: ADR\_O DAT\_O SEL\_O WE\_O TAGN\_O  רשות 3.40: אם מאסטר לא מייצר מצבי המתנה, STB\_O ו- CYC\_O יכולים להיות אותו האות.  תצפית 3.55: CYC\_O צריך להיות בגבוה בכל זמן מחזור ההעברה. מאסטר שלא מייצר מצבי המתנה לא מוריד את STB\_O בזמן מחזור ההעברה, כלומר גם הוא בגבוה במשך כל מחזור ההעברה. מכאן שאפשר ששניהם יהיו אות אחד. עם זאת, שניהם חייבים להיות חלק מהממשק.   **3.1.4 שימוש ב- ACK\_O, ERR\_O וב- RTY\_O** חוק 3.65: ממשקי עבד חייבים להתאים את האות DAT\_O עם ACK\_O, ERR\_O או RTY\_O   **3.1.5 שימוש בתגים TAG TYPES** ממשק וישבון יכול להיות מותאם אישית באמצעות טכניקה שנקראת תיוג (tagging), קונספט מוכר היטב בתעשיית הבאסים למיקרו מחשבים. התגים מאפשרים לקשר (associate) מידע המוגדר על ידי המשתמש עם כתובת, מילת מידע או מחזור באס.  כל אותות התגים חייבים להתאים לסט של קווים מנחים הידועים כטיפוסי תגים (TAG TYPEs). טבלה 3-1 מפרטת את כל הטיפוסים עם המידע המקושר אליהם ואת צורת האות (waveform). כאשר תג מוסף לממשק, מוקצה לו טיפוס מהטבלה. הדבר מגדיר חד-ערכית איך התג פועל. המידע חייב להיכלל בדף המידע (wishbone datasheet).  [http://sites.google.com/site/simple086/_/rsrc/1255943937935/wishbone-spec/table.3.1.gif](http://sites.google.com/site/simple086/wishbone-spec/table.3.1.gif?attredirects=0)  לדוגמה, ממשק מאסטר עם הגנת אות זוגיות בשם PAR\_O המיוצר ממילת המידע ביציאת (DAT\_O(15..0. אם האות מוסף לממשק, המידע הבא נדרש (בדף המידע של וישבון) כדי להגדיר לחלוטין את התזמון של PAR\_O:  SIGNAL NAME:             PAR\_O DESCRIPTION:              Even parity bit MASTER TAG TYPE:    TGD\_O()  חוק 3.70: כל התגים מוגדרי המשתמש חייבים להיות מטיפוס תגים כלשהו. יתרה מכך, הם חייבים להיצמד לספציפיקציות הזמן שנתונות במסמך- לטיפוס הנתונים שלהם.  רשות 3.45: תג לא חייב להיות מערך, למרות שכל טיפוסי הנתונים בטבלה מסומנים כך [עם סוגריים ()].  המלצה 3.15: אם ממשק מאסטר תומך ביותר ממחזור באס מוגדר אחד על סט משותף של קוי אות, הוסף תג מחזור (cycle tag) כדי לזהות כל סוג של מחזור באס. הדבר מאפשר ל- INTERCON ולממשק העבד להבדיל בין מחזורי הבאס האלו (במידה ויש צורך). הגדר את האותות כתגים מסוג ()TGC\_O, והשתמש בשמות אות SGL\_O, BLK\_O ו- RMW\_O כדי לציין מחזור יחיד (SGL), בלוק (BLK) וקרא-עדכן-כתוב.   **3.2 מחזורים מסוג מחזור קריאה\כתיבה יחיד** מחזורים מסוג קריאה\כתיבה יחידה מבצעים העברת מידע אחת בכל פעם. אלה המחזורים הבסיסיים שמבצעים מעבר מידע בחיבור וישבון.   חוק 3.75: כל ממשקי המאסטר והעבד שתומכים במחזור קריאה או כתיבה יחיד, חייבים להתאים לדרישות הזמן המופיעות בחלקים 3.2.1 ו- 3.2.2.  רשות 3.50: ממשקי מאסטר ועבד יכולים להיות מתוכננים כך שהם לא יתמכו במחזור קריאה\כתיבה יחיד.  **3.2.1 מחזור קריאה יחיד** ציור 3-3 (בהמשך) מראה מחזור קריאה יחיד. פרוטוקול הבאס עובד כך:  שפת שעון 0:     המאסטר מציג כתובת ולידית ב- ()ADR\_O ו- ()TGA\_O.     המאסטר מוריד את WE\_O כדי לציין שמדובר במחזור קריאה.     המאסטר מציג bank select באות ()SEL\_O כדי לציין היכן הוא מצפה למידע.     המאסטר מעלה את CYC\_O ואת ()TGC\_O כדי לציין שזהו תחילת המחזור.     המאסטר מעלה את STB\_O כדי לציין את תחילת הפאזה.  Setup, שפה 1:     העבד מפענח את הכניסות, והעבד המגיב מעלה את ACK\_I.     העבד מציג מידע ולידי על ()DAT\_I ועל ()TAD\_I.     העבד מעלה את ACK\_I בתגובה ל- STB\_O כדי לציין שהמידע ולידי.     המאסטר מנטר את ACK\_I, ומתכונן לשמור (to latch) את המידע שעל ()DAT\_I ועל ()TGD\_I.      הערה: העבד יכול להכניס מצבי המתנה (-wait states -WSS) לפני העלאת ACK\_I, מה שמאפשר לו לעקב את מהירות המחזור. אפשר להכניס כל מספר של מצבי המתנה.  שפת שעון 1:     המאסטר שומר מידע את המידע שעל ()DAT\_I ועל ()TGD\_I.     המאסטר מוריד את STB\_O ואת CYC\_O כדי לציין שזהו סוף המחזור.     העבד מוריד את ACK\_I בתגובה לירידת STB\_O  [http://sites.google.com/site/simple086/_/rsrc/1255944105612/wishbone-spec/3.3.gif](http://sites.google.com/site/simple086/wishbone-spec/3.3.gif?attredirects=0)  **3.2.2 מחזור כתיבה יחיד** כנ"ל אלא שהאות WE\_O מועלה ולא מורד, ושהמידע נשלח מהמאסטר אל העבד (ולא ההיפך) על ()DAT\_O ועל ()TGD\_O בנוסף לשאר האותות שהמאסטר שולח בשפה 0 (כתובת, סלקט וכו').  [http://sites.google.com/site/simple086/_/rsrc/1255944226955/wishbone-spec/3.4.gif](http://sites.google.com/site/simple086/wishbone-spec/3.4.gif?attredirects=0)  **3.3 מחזור קריאת\כתיבת בלוק** מחזורי העברת בלוק מבצעים מעברי מידע מרובים. הם דומים למחזורי קריאה וכתיבה, אבל יש בהם שינויים מסויימים כדי לתמוך במעברים המרובים של מידע.  במהלך מחזורי בלוק הממשק למעשה מבצע מחזור קריאה\כתיבה יחיד כפי שתואר לעיל, אך המחזורים הבודדים הללו (שנקראים פאזות) משולבים יחד כדי ליצור מחזור בלוק בודד. הפונקציה הזאת שימושית במיוחד כאשר מאסטרים מרובים משתמשים בחיבור הוישבון. לדוגמה, אם העבד הוא זכרון בעל שני פורטים (dual port) ומשותף, אז ארביטר עבור הזכרון יכול לקבוע מתי מאסטר אחד סיים איתו כך שאחר יכול לקבל גישה לזכרון.  כפי שאפשר לראות בציור 3-5, האות CYC\_O מועלה במשך כל מחזור הבלוק. האות יכול לשמש כדי לבקש רשות לגישה למשאב משותף מארביטר מקומי. כדי להחזיק את הגישה עד סוף המחזור, האות LOCK\_O חייב להיות בגבוה כפי שמוצג. במהלך כל פאזות מעבר המידע (בתוך מעבר הבלוק), פרוטוקול לחיצת היד הרגיל בין STB\_O לבין ACK\_I נשמר.  [http://sites.google.com/site/simple086/_/rsrc/1255944333137/wishbone-spec/3.5.gif](http://sites.google.com/site/simple086/wishbone-spec/3.5.gif?attredirects=0)  חוק 3.80: כל ממשקי המאסטר והעבד שתומכים במחזורי בלוק חייבים להתאים לדרישות הזמן המוצגות בחלקים 3.3.1 ו- 3.3.2  רשות 3.55: ממשקי מאסטר ועבד יכולים להיות מתוכננים כך שהם לא יתמכו במחזורי בלוק.   **3.3.1 מחזור קריאת בלוק** ציור 3-6 מראה מחזור קריאת בלוק. מחזור הבלוק מסוגל להעביר מידע בכל מחזור שעון. עם זאת, הדוגמה הזאת מראה גם איך גם ממשק המאסטר וגם ממשק העבד יכולים לעכב את קצב ההעברה של הבאס באמצעות הכנסת מצבי המתנה. אחרי ההעברה הרביעית העבד הכניס מצב המתנה, והמחזור הסתיים אחרי ההעברה החמישית. הפרוטוקול להעברת בלוק עובד כך:  שפת שעון 0:     המאסטר מציג כתובת ב- ()ADR\_O וב- ()TGA\_O.     המאסטר מוריד את WE\_O כדי לציין שזהו מחזור קריאה.     המאסטר מציג את בחירת הבנק ב- ()SEL\_O כדי לציין היכן הוא מצפה למידע.     המאסטר מעלה את CYC\_O ואת ()TGC\_O כדי לציין שזוהי תחילת המחזור.     המאסטר מעלה את STB\_O כדי לציין שזוהי תחילת הפאזה.      הערה: המאסטר מעלה את CYC\_O ו\או ()TGC\_O בשפת השעון או בכל זמן לפניו.  Setup, שפה 1:     העבד מפענח את הכניסות והעבד המגיב מעלה את ACK\_I.     העבד מציג מידע ולידי ב- ()DAT\_I וב- ()TGD\_I.     המאסטר מנטר את ACK\_I ומתכונן כדי לשמור את המידע שמעביר העבד על קוי המידע.  שפת השעון 1:     המאסטר שומר את המידע שעל קווי מידע.     המאסטר מציג כתובת חדשה בקווי הכתובת.     המאסטר מציג בחירת בנק חדשה ב- SEL\_O()  Setup, שפה 2:     העבד מפענח את הכניסות ומגיב על ידי העלאת ACK\_I. (בעצם, אלא אם הוא רוצה להכניס מצבי המתנה, הוא פשוט לא מוריד אותו. כלומר, מצב המתנה בא במצב שבו העבד לא מסוגל לעמוד בקצב של המאסטר)     העבד מציג מידע ולידי על קווי המידע.     המאסטר מנטר את ACK\_I ומתכונן לשמור את המידע מהעבד.  שפת השעון 2:     המאסטר שומר את המידע שעל קווי המידע.     המאסטר מוריד את STB\_O כדי ליצור מצב המתנה WSM  setup, שפה 3:     העבד מוריד את ACK\_I בתגובה ל- STB\_O.      הערה: כל מספר של מצבי המתנה יכול להיות מוכנס ע"י המאסטר.  שפת שעון 3:     המאסטר מציג כתובת חדשה על קווי הכתובת.     המאסטר מציג בחירה חדשה ב- ()SEL\_O.     המאסטר מעלה את STB\_O  Setup, שפה 4:     העבד מפענח את הכניסות ומגיב בהעלאת ACK\_I.     העבד מציג מידע על קווי המידע.     המאסטר מנטר את ACK\_I ומתכונן לשמור את המידע.  שפת השעון 4:     המאסטר שומר את המידע שעל הקווים.     המאסטר מציג כתובת חדשה.     המאסטר מציג בחירת בנק חדשה.  Setup, שפה 5:     העבד מפענח את הכניסות ומגיב בהעלאת ACK\_I     העבד מציג את המידע החדש בקווים.     המאסטר מנטר את ACK\_I ומתכונן לשמירת המידע.  שפת השעון 5:     המאסטר שומר את המידע.     העבד מוריד את ACK\_I כדי להכניס מצב המתנה.      הערה: כל מספר של מצבי המתנה יכול להיות מוכנס ע"י העבד.  Setup, שפה 6:     העבד מפענח את הכניסות ומגיב בהעלאת ACK\_I.     העבד מעלה מידע לקווים.     המאסטר מנטר ACK\_I  שפת השעון 6:     המאסטר שומר את המידע.     המאסטר מסיים את המחזור על ידי הורדת STB\_O ו- CYC\_O  [http://sites.google.com/site/simple086/_/rsrc/1255944473534/wishbone-spec/3.6.gif](http://sites.google.com/site/simple086/wishbone-spec/3.6.gif?attredirects=0)  **3.3.2 מחזור כתיבת בלוק** כנ"ל, בהבדלים הבאים:     המידע עובר מהמאסטר לעבד ו-וילדי לכל אורך הפאזה.     WE\_O בגבוה.     SEL\_O מציין איפה המאסטר שולח את המידע.  [http://sites.google.com/site/simple086/_/rsrc/1255944635484/wishbone-spec/3.7.gif](http://sites.google.com/site/simple086/wishbone-spec/3.7.gif?attredirects=0)  **3.4 מחזור קרא-עדכן-כתוב RMW** מחזור קרא-עדכן-כתוב (read-modify-write) משמש לפעולת סמפור בלתי ניתנת לחלוקה. במהלך החצי הראשון של המחזור, מתבצעת פעולת קריאה יחידה, ובמהלך החצי שני של המחזור מתבצעת פעולת כתיבה יחידה. האות CYC\_O נשאר בגבוה בזמן שני החצאים.  חוק 3.85: כל ממשקי המאסטר והעבד שתומכים בפעולת RMW חייבים להתאים לדרישות הזמן שנתונות בפרק 3.4 (הפרק הנוכחי..).  רשות 3.60: ממשקי המאסטר והעבד יכולים להיות מתוכננים כך שלא יתמכו במעבר RMW.  ציור 3-8 מראה מחזור RMW.  המאסטר והעבד רשאים להכניס מצבי המתנה באמצע, וכך נעשה בדוגמה.  [http://sites.google.com/site/simple086/_/rsrc/1255944756924/wishbone-spec/3.8.gif](http://sites.google.com/site/simple086/wishbone-spec/3.8.gif?attredirects=0)  **פרק 5 – ספציפיקציות תזמון** ממשק וישבון מתוכנן כך שמשתמש הקצה יצטרך לעמוד באילוצי זמן פשוטים. הממשק מתוכנן כך שהוא יוכל לעבוד ללא צורך בספציפיקציות תזמון מפורטות. בכל המקרים, מידע הזמן היחיד שנדרש עבור משתמש הקצה הוא תדירות השעון המקסימלית (עבור CLK\_I) שמועברת לכלי ה- P&R. תדירות השעון המקסימלית מוכתבת על ידי השהיית הזמן בין שפת השעון החיובית ב- CLK\_I לבין הגעת האות למבנים שנמצאים בשלבים שבמורד מסלול אותות הלוגיקה. ההשהייה מוצגת באיור 5-1, ומוגדרת כ-  Tpd,clk-su.  [http://sites.google.com/site/simple086/_/rsrc/1255944880644/wishbone-spec/5.1.gif](http://sites.google.com/site/simple086/wishbone-spec/5.1.gif?attredirects=0)  חוק 5.00: כניסת השעון CLK\_I לכל ליבת IP חייבת לתאם את כל הפעילויות לכל הלוגיקה הפנימית של ממשק וישבון. כל אותות המוצא של וישבון מרגוסטררים בשפה העולה של CLK\_I. כל אותות הכניסה של וישבון חייבים להיות יציבים לפני השפה העולה של CLK\_I.  רשות 5.00: כלי ה- P&R של המשתמש רשאי לכפות את חוק 5.00.  תצפית 5.00: אפשר לקנפג (configure) בקלות את רוב כלי ה- P&R כדי לכפות את חוק 5.00. באופן כללי, הדבר דורש רק ספציפיקצית זמן אחת עבור Tpd,clk-su.  חוק 5.05: ממשק הוישבון חייב להשתמש במתודולוגיות עיצוב RTL סינכרוניות, כך שבהנתן שערים מהירים באופן "כמעט אינסופי", הוא יפעל בטווח "כמעט אינסופי" של תדירויות שעון עבור CLK\_I.  תצפית 5.05: מציאותית, לעולם לא יצופה מממשק וישבון לעבוד ב"טווח כמעט אינסופי" של תדירויות שעון. עם זאת, הדרישה הזאת מחסלת את הצורך בדרישות זמן "לא ניידות", כלומר שאינן ניתנות להעברה בין פלטפורמות שונות, שהיו גורמות ליכולת לעבוד רק עם מכשירי יעד ספציפיים.  תצפית 5.10: הלוגיקה של ממשק וישבון מניחה שסכימת חלוקת clock-skew נמוך נמצאת בשימוש במכשיר היעד, ושה- clock-skew ישאר נמוך מספיק כדי להרשות פעולה אמינה בתנאי הסביבה (הקיצור, ממשק וישבון יוצא מנקודת הנחה שאין בעיות תזמון כתוצאה מ- clock skew).  רשות 5.05: ליבת ה- IP שמחוברת לממשק וישבון רשאית לכלול דרישות תזמון ספציפיות לאפליקציה.  חוק 5.10: כניסת השעון CLK\_I חייבת להיות בעלת duty cycle שאינו נמוך מ- 40%, ואינו גבוה מ- 60%.  רשות 5.10: מודול ה- SYSCON רשאי להשתמש במחולל שעון משתנה (variable clock generator). במקרים אלו תדירות השעון יכולה להשתנות ע"י מודול SYSCON כל עוד שפות השעון נשארות נקיות ומונוטוניות, והשעון לא מפר את דרישות ה- duty cycle.  רשות 5.15: מודול ה- SYSCON רשאי להשתמש במחולל שעון ממושער (gated..). במקרים אלו השעון יעצר במצב נמוך. כאשר השעון המשוערר נעצר ומתחיל, השפות נדרשות להישאר נקיות ומונוטוניות.  הצעה 5.00: כאשר משתמשים במחולל שעון ממושער (gated), כבה את השעון כשהחיבור WISHBONE interconnection לא עסוק. דרך אחת לעשות זאת היא ליצור ממשק מאסטר שמטרתו היחידה היא לרכוש את החיבור ולכבות את השעון. הדבר יוודא שהחיבור לא עסוק כשהשעון מכובה על ידי השערים. כאשר אות השעון מוחזר, המאסטר משחרר את החיבור.  תצפית 5.15: הספציפיקציות של וישבון לא מתיימרות לנהל את פעולת השעון הממושער או המשתנה.  הצעה 5.10: תכנן את ליבת ה- IP כך שכל המעגלים (כולל WISHBONE interconnect) מקיימים את החוקים הנ"ל, כך שהם יהפכו את הליבה ל"ניידת" (portable) בטווח רחב של מכשירי מטרה וטכנולוגיות.  [אל הדף הראשי של פרוייקט תכנון מעבד פשוט](http://sites.google.com/site/simple086/Home) | |   **\_displayNameOrEmail\_** - \_time\_ - [הסר](javascript:;)  \_text\_ | |

[כניסה](https://www.google.com/a/UniversalLogin?service=jotspot&continue=http://sites.google.com/site/simple086/wishbone-spec)   [הפעילות שבוצעה לאחרונה באתר](http://sites.google.com/site/simple086/system/app/pages/recentChanges)   [תנאים](javascript:void(window.open('http://www.google.com/sites/help/intl/iw/terms.html')))   [דווח על שימוש לרעה](http://sites.google.com/site/simple086/system/app/pages/reportAbuse?src=/wishbone-spec)   [הדפס דף](javascript:;)  |    **מופעל על ידי** [**Google Sites**](http://sites.google.com)