**LZRW3 –DATA FLOW**

**המסמך מתייחס ל**: מעבר המידע בין היחידות השונות, מידע על קווי בקרה בין היחידות השונות, קצה אל קצה.

**מקרא**: שמות יחידות באנגלית בכתב גדול, שמות יחידות פנימיות (קומפוננטות) בכתב קטן.

**תיאור מעבר המידע מהמחשב אל עבר הבורד:**

תהליך זרימת המידע מתחיל ונגמר בשלב ה-GUI, באמצעותו נוכל לשלוח קובץ אל הדוחס לקבלו בחזרה ולהשוותו לקובץ הדחוס שהתקבל דרך ה-Golden Model. ברגע שנבחר קובץ(input file) ונלחץ ב-GUI את הפקודה RUN, מידע יתחיל לזרום באופן טורי מהמחשב אל עבר הדוחס באמצעות קו ה-UART המחבר בין המחשב לבורד עליו מותקן ה- FPGA. פרוטוקול UART שולח מסגרות מידע העטופות במידע המשמש לפרוטוקול לבקר על ערוץ התקשורת. בכניסה ל- FPGA נמצאת יחידה בשם Rx PATH שבתוכה קומפוננטה בשם uart\_rx שתפקידה לקלוט את המסגרות המתקבלות. בנוסף יחידה זו גם בודקת האם המידע שהגיע אליה הגיע בצורה תקינה (בדיקת שדה Checksum או CRC תלוי מה התוכנה שלחה) במידה והמידע שהתקבל איננו תקין היא תבקש שליחה חוזרת ובמידה והמידע אכן תקין היא תעביר את המידע ליחידה בשם mp\_dec  אשר תסיר את תקורת המידע המתקבל (מידע הבקרה) ותשמור את חלק המידע (ה- DATA) המתקבל בזיכרון.RAM לאחר שהיחידה סיימה לשמור על ה-RAM היא תסמן ליחידה הבאה באמצעות הקוmp\_done  שתקרא את המידע השמור ב- RAM המידע יקרא ליחידה הבאה בשם input memory . קריאת המידע נעשית באופן מקבילי.

הדוחס הינו מהיר ואילו קו ה-UART איטי בהרבה ממנו, לכן, אנו נאגור בזיכרון input memory מספר תוכני מסגרות (DATA) עד אשר נקבל גודל של בלוק רצוי אותו נשלח לדוחס באופן מקבילי בקצב 125MHZ.

**מעבר המידע לליבת הדוחס LZRW3 ועד יציאתו ממנו:**

לאחר שנצבור גודל בלוק רצוי (בין 2K ל- K32) נשלח לדוחס על קו ברוחב 8 ביט בקצב 125MHZ את הבלוק שיגיע אל היחידה הראשונה בדוחס.

היחידה הראשונה בדוחס הינה INPUT FILE BLOCK אשר שומרת בתוכה את כל קובץ הקלט ומארגנת אותו בצורה רציפה בתוך זיכרון בגודל 1MBYTE . אין צורך להמתין עד לקבלת קובץ הקלט השלם אלה ניתן להתחיל לדחוס החל מקבלת הבלוק הראשון ובקבלת כל בלוק נוסף.

מיחידה זו נקראים שלושה בתים(BYTES) אל תוך יחידה בשם HASH FUNCTION אשר מפעילה רצף פעולות על שלושת הבתים המתקבלים ומחזירה מספר הנע בין 0 ל- 4095.

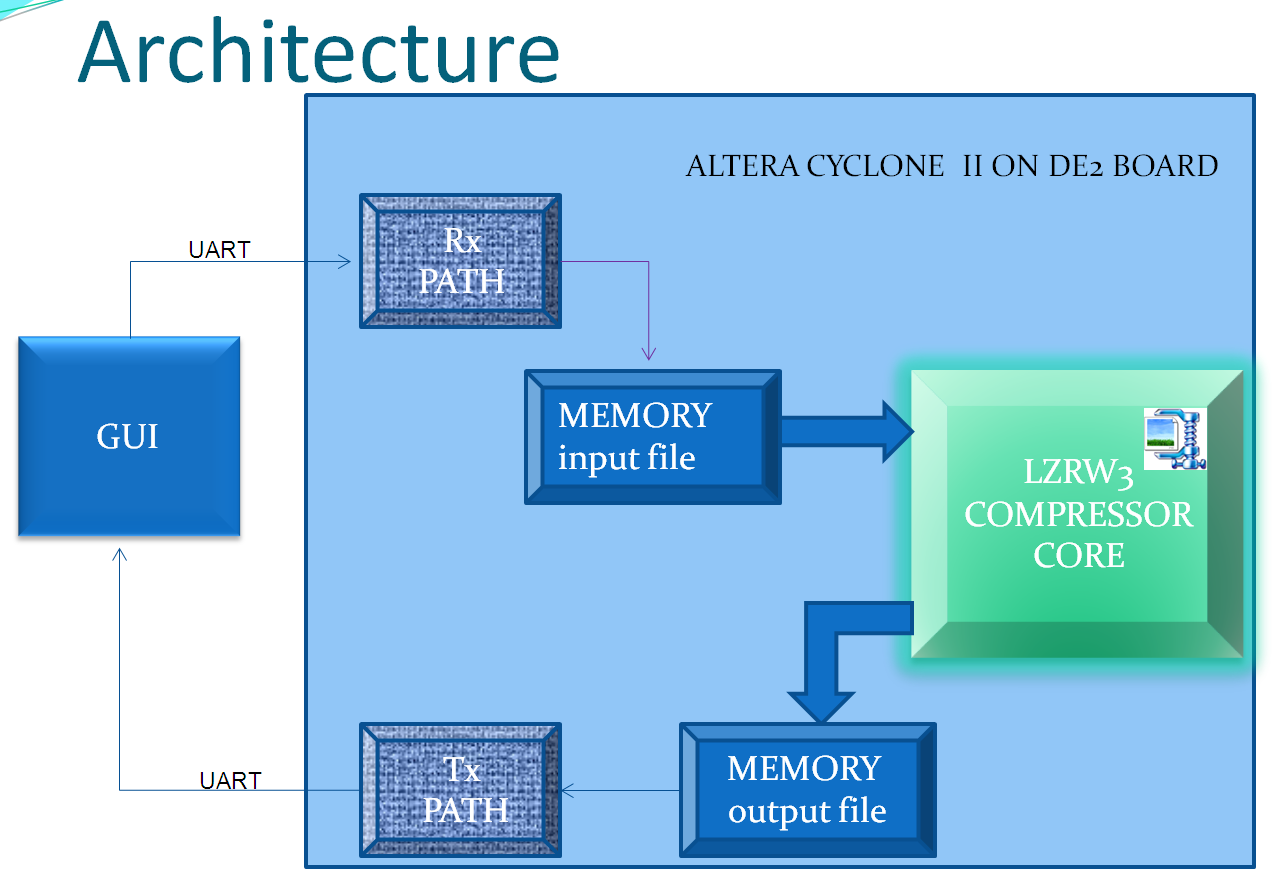
המספר שמתקבל מתוך יחידת ה- HASH FUNCTION מסמל את הכתובת (ה-INDEX) אותה צריך לקרוא מתוך יחידת ה HASH TABLE. בתלות בחישובים של האלגוריתם אנו נכתוב מידע אל הכתובת הזו ונעביר את ה- INDEX שלה ליחידה הבאה בשם OUTPUT GROUP BLOCK.

יחידת ה- OUTPUT GROUP BLOCK הינה יחידת זיכרון אשר תכיל בתוכה חלק מהבלוק הדחוס. יחידה זו תעביר את המידע הדחוס אל עבר זיכרון בשם OUTPUT MEMORY בגודל MHZ 1 בקירוב הנמצא מחוץ לליבת הדוחס.

**תיאור מעבר המידע מהבורד אל עבר המחשב:**

יחידת ה- OUTPUT MEMORY ,תאגור מידע עד אשר היא תקבל סימון מליבת הדוחס ואז היא תחל להעביר מידע אל עבר יחידת ה- Tx PATH אשר תקבל את המידע בקצב איטי בהתאם ליכולות ה- UART. יחידת ה- Tx PATH תקבל מנות קטנות של מידע אותן היא תשמור בזכרון RAM המתאימות בגודלן לגודל המידע (DATA) במסגרת המקיימת פרוטוקול UART. לאחר שהתקבלה כמות מתאימה קומפוננטת ה- mp\_enc שבתוך ה- Tx PATH תקבל סיגנל בשם reg\_ready ותתחיל לעטוף את המידע במידע בקרה ותכניס אותו לתור FIFO בצורה מקבילית המוביל ליחידת השידורuart\_tx  אשר תשדר את המידע בצורה טורית על הקו אל עבר המחשב. המחשב יקבל את החבילות ויבדוק לפי מידע הבקרה כי הן התקבלו באופן תקין ויצור קובץ אותו נשווה עם הקובץ המקבל ע"י ה- Golden Model באמצעות ה-GUI.

**תיאור מפורט של יחידות הדוחס:**



בתרשים ניתן לראות את ערוצי הכניסה והיציאה, וליבת הדוחס

**ערוץ הכניסה:** פירוט על Tx PATH ומבנה MEMORY Input file

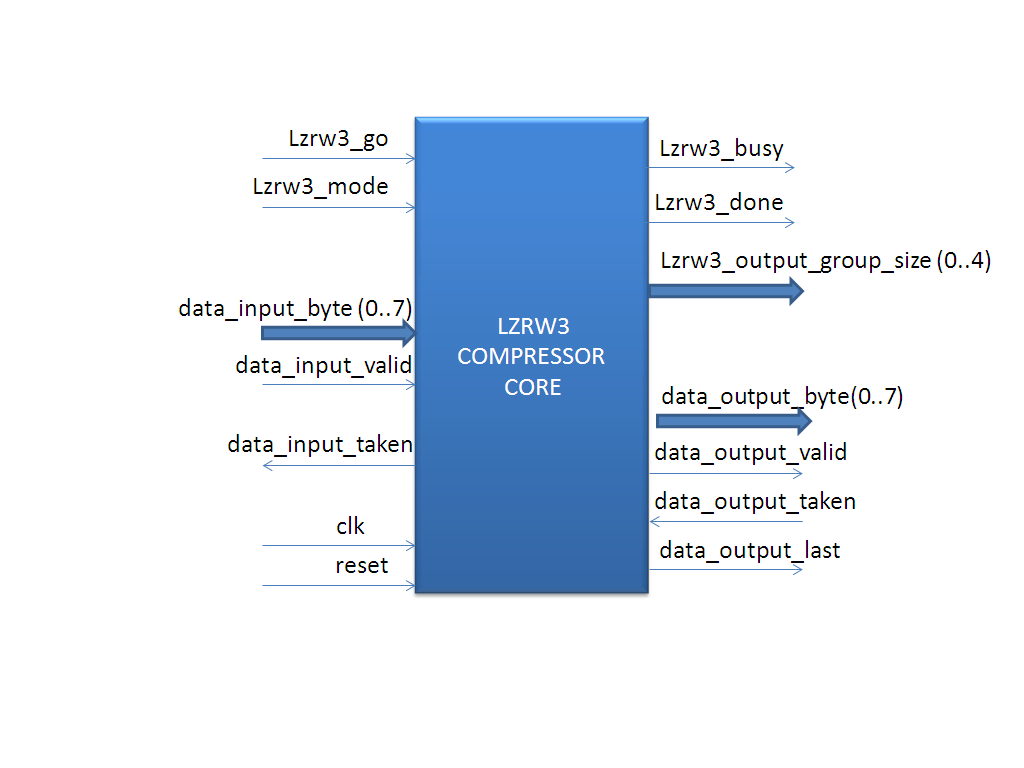
**ערוץ היציאה:** פירוט על Rx PATH ומבנה MEMORY output file

**ליבת הדוחס:**

**מטרה:** הליבה תקבל בלוק בגודל הנע בין 2kbytes עד 32kbytes בקצב של 125Mhz ותוציא מידע המחולק ל- groups בגודל הנע בין 18-34 bytes בהתאם לדחיסה המתאפשרת.

**פונקציונאליות:** הליבה מבצעת את אלגוריתם הדחיסה Lzrw3 עצמו כפי שתואר.

**אותות ממשק:**



**Inputs:**

**Lzrw3\_go:** התחל פעולת דחיסה במידה ו- Lzrw3\_busy איננו 1.

**Lzrw3\_mode:** דוחס(1)/פורס (0) (עבור אפשרות להרחבת הפרויקט).

**data\_input\_byte(0..7) :**מקבל 8 ביט מ input memory – במידה ו-data\_input\_valid הוא 1.

**data\_output\_taken:** אינדיקציה שה- group התקבל בהצלחה, וזמין לקבלת בלוק נוסף מגיע מה - output memory.

**Outputs:**

**Busy-** מבצע דחיסה כעת.

**Lzrw3\_done:** פעולת הדחיסה על בלוק הסתיימה, נותן אינדיקציה ליחידה הבאה (output memory) שה-group\_size עדכני.

**Lzrw3\_output\_group\_size (0..4):** גודל ה group היוצא, הערך נע בין 0 ל 31 byte, ביחידה הבאה (output memory ) גודל ה- group האמיתי יחושב לפי הערך המתקבל בתוספת 3 .

**data\_output\_byte(0..7)** : מוצא בלוק דחוס (groups format)

**data\_output\_valid:** המידע הדחוס עדכני.

**data\_input\_taken:** בלוק קודם התקבל, זמין לקבלת בלוק נוסף.

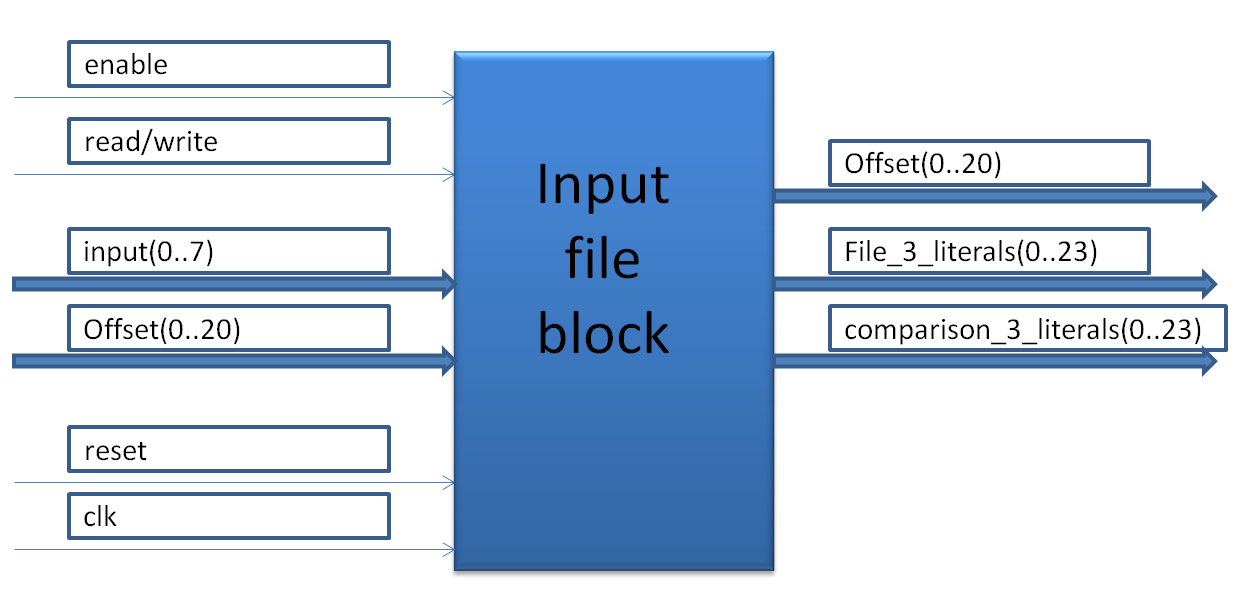
**data\_output\_last:** אינדיקציה ל output memory שזה הבלוק האחרון שמשודר. סיום תהליך דחיסה.

**Input file block:**

**מטרה:** היחידה מקבלת את הקובץ לדחיסה ושומרת אותו בזיכרון שבה עד לסיום תהליך הדחיסה

**פונקציונאליות:** היחידה מכילה זכרון בגודל של 1MByte ויכולה לקרוא ולכתוב אליו.

**אותות ממשק:**



**Inputs:**

**Enable** – controller

**Read/write** – controller

**Input(0..7)** – RX path

**Offset(0..20)** – hash table

**Outputs:**

**Offset(0..20)** – hash table

**File\_3\_literals(0..23)** – hash function/comparator

**comparison\_3\_literals(0..23)** - comparator

**hash function:**

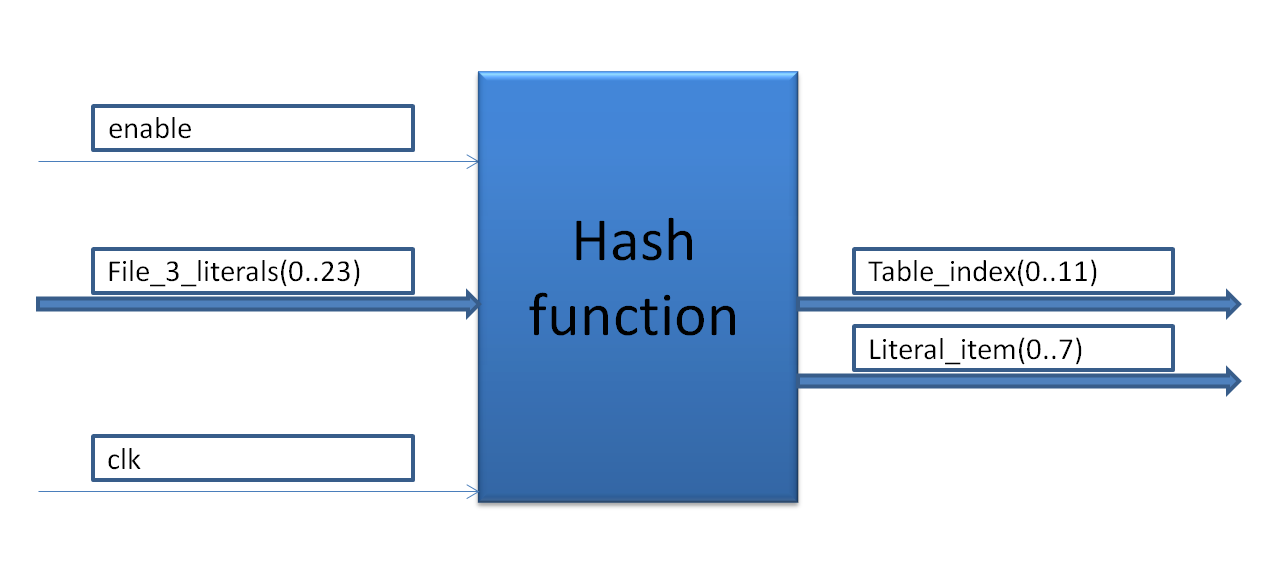
**מטרה:** להשיג עבור צירופים שונים של 3bytes פיזור אחיד כמה שניתן על טבלת ה hash.

**פונקציונאליות:** פונקציה ידועה מראש: PTR מייצג מצביע ל BYTE הראשון מבין השלשה.

(((40543\*(((\*(PTR))<<8)^((\*((PTR)+1))<<4)^(\*((PTR)+2))))>>4) & 0xFFF)

הפונ' ... (הסבר של התנהגות הפונ')

**אותות ממשק:**



**Inputs:**

**Enable** – controller

**File\_3\_literals(0..23)** –input file block

**Outputs:**

**Table\_index(0..11)** – hash table

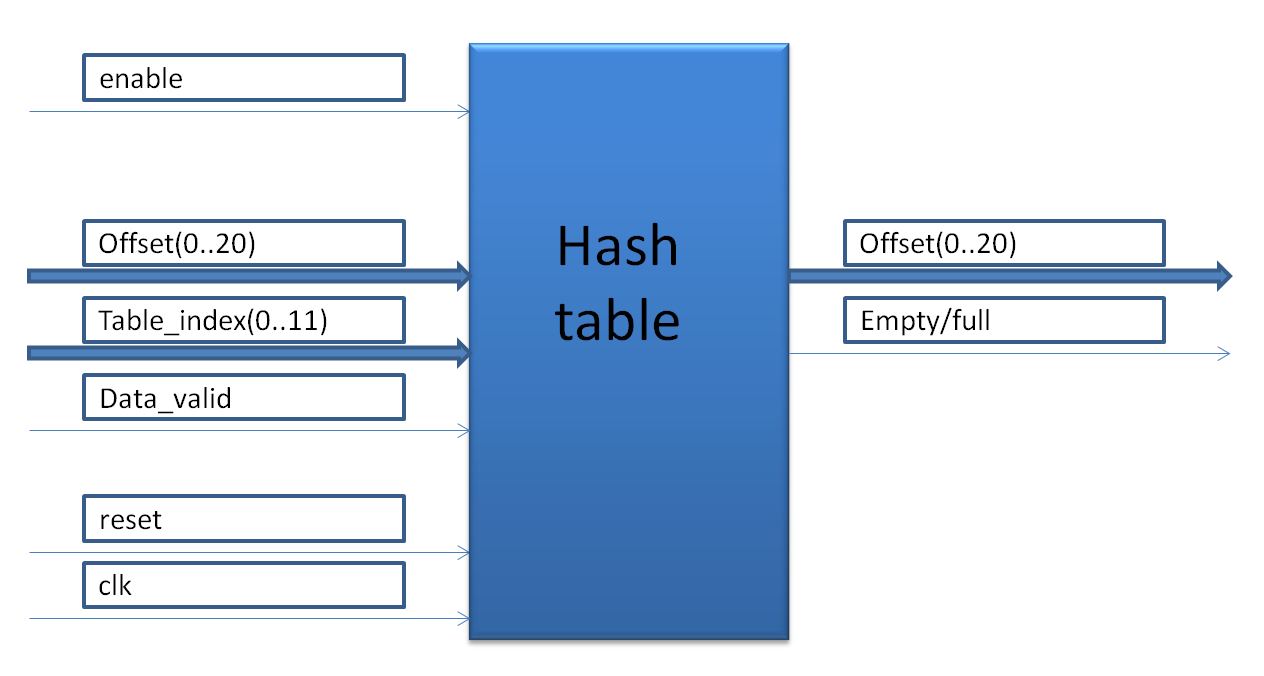
**Literal\_item(0..7) = File\_3\_literals(0..7)** –output file block (first literal of the 3 in the input is transferred to the ouput group block, will be used in case of literal item)

**Hash table:**

**מטרה:** להחזיק את רשימת ה offset של מופעים קודמים של צירופי הליטרלים.

**פונקציונאליות:** היחידה מכילה טבלה בגודל של 4096 מקומות כאשר כל מקום ברוחב 20bit.

**אותות ממשק:**



**Inputs:**

**Enable** – controller

**Read/write** – controller

**Offset(0..20)** – input file block

**Table\_index(0..11)** – hash function

**Data\_valid** – hash function

**Outputs:**

**Offset(0..20)** – input file block

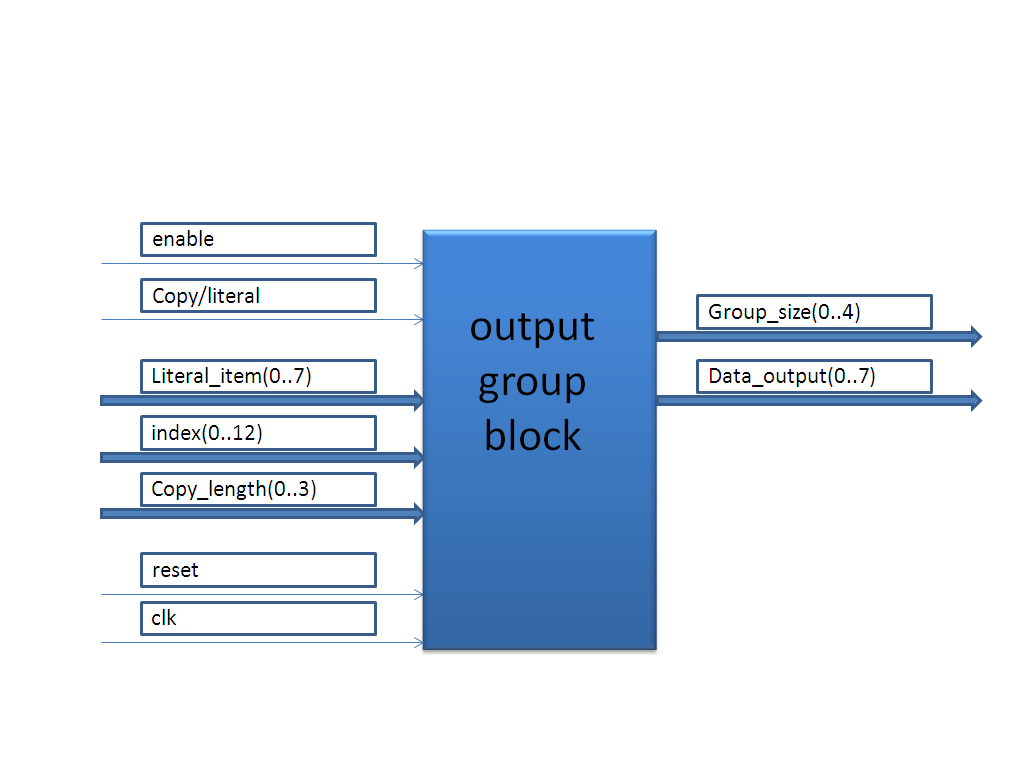
**Empty/full** - controller

**Output group block:**

**מטרה:** להחזיק בזיכרון RAM את ה-ITEMS (סה"כ 16) אשר יתקבלו מפעולת הדחיסה ולקלטת כל ITEM כ- COPY ITEM או כ LITERAL ITEM על ידי יצירת CONTROL BYTES.

**פונקציונאליות:** היחידה תסמן ב- CONTROL BYTES 1 עבור COPY ITEM ו-0 עבור LITERAL ITEM בהתאמה למיקום ה-ITEM המתאים ב- GROUP.

**אותות ממשק:**



**Inputs:**

Enable – controller

Copy/literal – controller

Literal\_item(0..7) – hash function

Copy\_length(0..3) – comparators\_counter

**Outputs:**

Group\_size(0..4) – output memory

Data\_output(0..7) – output memory

(both are represented in the core diagram)