Subject explanation (in short):

So, in a nutshell, UART is a simple and cost-effective way for devices to chat using only two wires. One device says, "I'm starting to send data," then sends 8 pieces of information, checks if everything looks right, and finally says, "I'm done now." The other device listens, puts the pieces together, and understands the message.

בקצרה, UART היא דרך פשוטה וחסכונית עבור מכשירים לשוחח בצ'אט באמצעות שני חוטים בלבד. מכשיר אחד אומר "אני מתחיל לשלוח נתונים", ואז שולח 8 פיסות מידע, בודק אם הכל נראה תקין, ולבסוף אומר, "סיימתי עכשיו". המכשיר השני מקשיב, מחבר את החלקים ומבין את המסר.

Subject explanation (in long):

Start Bit '0': When one device wants to send data to the other, it begins with a special signal, which we can think of as a "start signal." This tells the receiving device, "Hey, pay attention, some data is coming."  
8-Bits of Data: After the start signal, the first device sends 8 pieces of information (bits) one by one. These bits can be 1s or 0s and represent the actual data being sent.   
Even Parity Bit: To make sure the data is correct and hasn't been tampered with during transmission, an extra bit is added. This extra bit can be a 1 or a 0, and it's chosen so that the total number of 1s in all the bits (including this extra one) is either even or odd, depending on the chosen parity. This helps catch errors.   
Stop Bit: To signal that the data transmission is complete, there's another special signal called a "stop signal." It's like saying, "Okay, I'm done sending data now."   
Transmitter and Receiver: The device sending the data (transmitter) sends one bit at a time, while the device receiving the data (receiver) waits for the bits to become stable and then stores them. Once all the bits are received, the receiver puts them together like a puzzle to understand the complete message.  
Baud Rate: This is like how fast two people talk on the phone. Some people talk fast, and some talk slowly. In UART, different devices can talk at different speeds. So, in the code (the instructions for the devices to communicate), you can change this speed. The default speed is like 9600 words per second.   
Clock Rate: This is like how often a device checks for new messages. Imagine you're waiting for a message on your phone. You can check every second, every minute, or every hour. Devices also have their own checking speeds, and this is called the clock rate. In the code, you can adjust this too. The default checking speed is like 50 million times per second.   
in simple terms, the code allows you to change how fast the devices talk (baud rate) and how often they listen for messages (clock rate). By default, they talk at 9600 words per second and listen 50 million times per second, but you can make them talk faster or slower and listen more or less frequently if needed.

בד – הקצב שבו הם מדברים, שבו מועברים נתונים  
שעון – כל כמה זמן בודקים מידע חדש

Transmitter:

1 – 10:  
כניסות שהם שעון, ריסט, המידע שאני רוצה להעביר, והמשדר שכמובן אם פועל זה סימן שאנחנו רוצים להעביר מידע  
יציאות שהם אקטיב שפועל במידה והמשדר פועל, והביטים שאנו מוציאים

12 – 16:

רשימת רגיסטרים (משתנים שיכולים לשמור\לאחסן מידע), הפרטי ביט שעוזר לנו לוודא שאין שגיאות, שיפט רג שמאותחל למצב התחלתי ידוע ואורכו 11 ביט (מוסבר בקוד למה) ושעון

18 – 25:

מגדיר את מצב המערכת, אידיל מצב חוסר פעילות כלומר מצב בו אנחנו לא משדרים כלום, שידור כלומר המצב בו אנו מעוניינים לשדר.  
פרמטר נוסף נועד לייצג את מספר מחזורי השעון (תקופות שעון) הנדרשות להעברת סיבית נתונים אחת זה עוזר להבטיח שהשולח והמקבל מסונכרנים בהעברת הנתונים שלהם.  
הסבר על הפרמטר האחרון באופן פשוט יותר, תאר לעצמך שיש לך שעון מתקתק, ואתה רוצה להשתמש בשעון הזה כדי לשלוח נתונים. השעון מתקתק פעמים רבות בשנייה, ואתה רוצה לדעת כמה תקתוק שעון צריך כדי לשלוח רק פיסת מידע אחת, כמו מכתב בהודעת טקסט. ה"תדירות של Clock" היא כמה מהר השעון שלך מתקתק. "תדירות bd" היא כמה מהר אתה רוצה לשלוח את הנתונים שלך. לכן, כאשר אתה מחלק את המהירות שבה השעון שלך מתקתק כמה מהר אתה רוצה לשלוח נתונים, אתה מקבל את מספר תקתוקי השעון הדרושים כדי לשלוח פיסת מידע אחת. במונחים פשוטים עוד יותר, זה כמו לשאול, "כמה פעמים השעון שלי צריך לתקתק כדי לשלוח דבר אחד?"

26 – 36:

כזכור הבלוק פועל כאשר יש שינוי ברשימת הרגישויות, במקרה שלנו הוא רגיש לשינויים בעליית השעון או בעליית של הריסט, במידה ואנחנו במצב ריסט השיפט רג מוגדר להיות המצב ההתחלתי להעברת נתונים.

39 – 61:

במצב idel כאשר המשדר פועל אנו נכנסים לתוך לולאה, בתוך הלולאה קורים הדברים הבאים, הלולאה עוברת על כל אחד מהביטים של הדאטה ומבצעת xor עם הערך הקודם שהרגע הכנסנו כך שלבסוף לתוך המשתנה reg יכנס הפרטי ביט של כל שמונת הביטים, החזקה מסמלת xor

סיבית הזוגיות:  
סיביות הזוגיות היא סיבית נוספת שמתווספת לנתונים המועברים. ערכו מחושב על סמך סיביות הנתונים, ומטרתו לזהות שגיאות במהלך השידור.

-במקרה שלנו אם סיבית הזוגיות אכן זוגית ושווה 0 כאשר נעביר את המידע המקבל ידע שהכל תקין, במידה והסיבית תהייה אי זוגית ושווה 1 סימן שיש בעיה בדרך וכשהמקבל יראה את זה הוא יבין שהייתה בעיה ויחשוב על פתרון-

62 – 83:

התנאי הראשון משמש כדי להבטיח שהשידור של ביטים בודדים מתרחש בקצב הנכון ומסונכרן עם מחזורי השעון. אם תנאי זה נכון, זה אומר שמספר מחזורי השעון הדרוש לשידור ביט אחד עבר, זה מסמל שהמשדר מוכן לשדר את סיבית הנתונים הבאה.  
אנו מחסירים 1 מאחר וברוב המערכות הדיגיטליות, הספירה מתחילה מ-0. לכן, אם אתה רוצה להפעיל פעולה במחזור השעון ה-10, אתה בודק מתי המונה שלך שווה ל-9 או גדול מ-9. זה מבטיח שהפעולה מתרחשת ממש בקצה של מחזור השעון ה-10.  
ensuring that the action is taken at the precise moment needed for proper synchronization in digital systems  
מעבירים את כל הביטים של שיפט ריג אחד ימינה כדי שנוכל להעביר את הביט הכי פחות משמעותי LSB בתאכלס לקחת את הביט הכי ימני.  
לבסוף בודק האם העברנו את כל המידע אם כן מאפס את כל הנתונים אחרת מוסיף 1 לשעון ובודק שוב

Receiver:

1 – 10:

הגדרת משתנים, המשתנה של הסיריאל זה בעצם חוט שעל גביו הועבר בכל פעם ביט אחד מן המשדר  
מוגדר הפרטי ביט שאותו נשווה למידע שיש לנו ונדע אם המידע עבר באופן תקין.  
משתנה הווליד שהוא בודק שכל המידע עבר במלואו, והמשתנה ה RX מכיל את שמונת הביטים שעברו.

12 – 18:

הרגיסטר של הביט קאונטר נועד לוודא שכל הביטים של המידע עברו, גודלו 4 מאחר ובגודל 4 נוכל לספור עד כ -15 ביטים שבפועל צריך בסך הכל 8 (אם נקח במקום 3 ביטים זה יתן לנו 7 כלומר 111 וזה לא מספיק לכן 4)

אני צריך שני רגיסט של פרטי ביט כי אחד אני משווה עם הקודם ואז בהתאם להשוואה הדגל של השגיאה פועל

הרגי של הסטייט הוא בגודל 3 כי יש לנו סך הכל 5 מצבים כזכור 3 ביטים מאפשר לנו להגיע ל 5 כי 101

20 – 30:

באופן כללי תמיד האאוט פוטים שלנו יקבלו רגי באסיין

49 – 59:

במצב IDEL אנו בודקים האם הביט של הסיריאל שווה 0 כלומר האם מדובר בביט ההתחלה, במידה וכן אנו עוברים למצב הבא מצב ההתחלה אחרת נשארים באותו מצב.  
  
61 – 73:  
במצב START BIT אנחנו בודקים שהאות המתקבל יציב ואינו משתנה, אנחנו מחלקים ב – 2 מאחר וזה מאפשר לנו דיוק מירבי בין המשדר למקלט ומקטין את הסיכוי לשגיאה.  
ה"אמצע" של סיבית הנתונים הוא הזמן הבטוח ביותר לבדוק אם זה '0' או '1'. בדרך זו, אתה נמנע משגיאות הנגרמות מכך שהביט מגיע קצת מוקדם או מאוחר בגלל הבדלי תזמון בין השולח למקבל. לכן, דגימה באמצע עוזרת להבטיח שתקבל את הנתונים הנכונים גם אם התזמון אינו מושלם.  
בנוסף עוד שני תנאים, אם התנאי הראשון לא מתקיים נעלה את הקלוק כאונטר באחד, אם הביט הראשון לא 0 נחזור למצב .IDEL  
התנאי clk\_counter >= (POSEDGES\_FOR\_BIT-1)/2 בודק אם מספר מחזורי השעון שנספרו על ידי clk\_counter הגיע או עלה על מחצית הדרך של משך הביט. במילים אחרות, הוא בודק אם עבר מספיק זמן שהמעגל נמצא כעת באמצע קבלת הביט. הגעה לאמצע משך הסיביות מעידה על כך שלאות היה מספיק זמן להתייצב. על ידי דגימת הנתונים באמצע הסיביות, סביר פחות שתושפע משינויי תזמון קלים, רעש או ריצוד שעלולים להתרחש בתחילת או בסוף הסיביות. אם תנאי זה מתקיים, זה מצביע על כך שהאות יציב יחסית, והקוד יכול להמשיך לבדוק את הערך של serial\_in כדי לקבוע אם זה ביט התחלה חוקי או לא

75 – 88:  
במצב DATA\_BITS התנאי הראשון הוא בודק אם עבר מספיק זמן (מחזורי שעון) כדי לקרוא בבטחה את סיבית הנתונים הנכנסת. כאשר clk\_counter (מונה עבור מחזורי שעון) הופך להיות גדול או שווה למספר מחזורי השעון הנדרשים עבור ביט אחד (מצוין על ידי POSEDGES\_FOR\_BIT) פחות אחד, זה אומר שהמקלט מאמין שהוא קרוב לסוף הסיביות הנכנסות ויכול לקרוא בצורה מהימנה הנתונים. זה נעשה בדרך כלל כדי להבטיח שהנתונים יהיו יציבים ואינם משתנים לפני שהם מעובדים.   
הרגיסט reg\_RX\_byte אליו נכניס את הדאטה שלנו נעשה זאת באופן הבא, הוא מקבל הצמדה של ביט הסיריאל לצד שמאל על ידי הזזת כל הרגיסט לימין כלומר מאבדים את הביט הימני של הרגיסט, ככה עושים לכל ביטי הסיריאל שמתעדכנים בכל עליית שעון לכן לא צריך לולאה, הסיריאל מתעדכן ומוצמד לרגיסט. כאשר כל הביטים הועברו נעבור לשלב הבא

90 – 105:  
במצב זה נבדוק האם ביט המידע הנכנס שווה לפרטי ביט שחושב במידה והם שווים סימן שהמידע עבר בצורה תקינה ולכן עוברים למצב האחרון של הסטופ, אחרת דגל השגיאה פועל וחוזרים למצב IDEL

107 – 120:  
המצב האחרון, בו בודקים אם הביט הסופי הוא 1 במידה וכן דגל התקינות עולה ל 1 אחרת חוזרים למצב IDEL

Top:

בחלק זה אנו שולטים במשדר והמקלט בעזרת מודל אחד, זה עוזר לשלוט ולתאם את חילופי הנתונים בין המשדר למקלט, למשל בחלק הראשון:  
הסוגריים מכילים את החיבורים (הידועים גם כחיבורי יציאה) בין מודול UART\_TOP למודול UART\_TX.  
 .clk(clk): מחבר את כניסת clk של מודול UART\_TX לכניסת clk של מודול UART\_TOP, ומבטיח ששני המודולים חולקים את אותו אות שעון.   
reset(reset): מחבר את כניסת האיפוס של מודול UART\_TX לכניסת האיפוס של מודול UART\_TOP, ומאפשר לאפס את שני המודולים יחד.   
TX\_data\_in(data\_in): מחבר את קלט TX\_data\_in של מודול UART\_TX לכניסת data\_in של מודול UART\_TOP, מעביר את הנתונים שישודרו.

בחלק השני אותו עיקרון הפעם עם המקלט.

Test bench:

7 – 15:  
ב-testbench הקלט הוא reg והפלט הוא wire.  
"אירוע" הוא כמו אות או מתג שאתה יכול להשתמש בו כדי לגרום לחלקים שונים בקוד שלך לעשות דברים מתי שאתה רוצה שהם יעשו זאת, על ידי הפעלת המתג בזמן הנכון.

17 – 18:  
מקשר בין Top לבין הטסט בנץ

20 – 32:  
בחלק זה אנו מדליקים את המשדר מאפסים את כל שאר הנתונים ומציגים את קטגוריות הגלים כמו add wave

שורות קוד אלו משמשות ליצירת תצוגה ידידותית למשתמש בקונסולת הסימולציה. שורת $display מגדירה את הכותרת עבור הנתונים שברצונך לנטר, ושורת $monitor מתעדכנת ומציגה באופן רציף את הערכים של האותות שצוינו יחד עם זמן הסימולציה הנוכחי כלומר $display: משמש להצגת הודעות טקסט מעוצבות וערכים משתנים במהלך סימולציה. הוא משמש לעתים קרובות לניפוי באגים ולניטור. $monitor: דומה ל-$display, אך הוא עוקב ברציפות אחר משתנים או ביטויים שצוינו ומציג את הערכים שלהם בכל פעם שהם משתנים.m נשים לב כי $time מאחר וזוהי פקודת מערכת כלומר משימה מוגדרת ומובנית מראש

35 – 40:  
בחלק זה אנו בעצם רוצים להגדיר את הדאטה שלנו, בעיקרון אנו מייצרים דאטה רנדמולי (את השמונה ביטים שלנו) וכמובן מציבים במשתנה הדאטה שלנו, הדרך לעשות זאת היא קצת מסורבלת

42 – 43:  
שורת הקוד הזו אומרת, "ברציפות, כל 2 יחידות זמן, שנה את הערך של clk למצב ההפוך שלו." זוהי דרך נפוצה ליצור גל מרובע או אות שעון עם מחזור עבודה של 50% (זמן שווה במצבים גבוהים ונמוכים) בסימולציית Verilog

.

45 – 56:  
הקוד הזה הוא כמו סימולציה של הפעלת המכשיר הדיגיטלי שלך. הוא ממתין לאות (reset\_trigger) כדי להפעיל את המכשיר, ממתין מעט, מאפס את המכשיר, ממתין שוב, ואז נותן למכשיר להפעיל כרגיל. זה עוזר להבטיח שהמכשיר שלך מופעל ופועל כראוי, בדיוק כמו שצריך כשאתה מפעיל אותו בחיים האמיתיים.  
אתחול: כאשר אתה מפעיל את המכשיר, הוא צריך להתחיל בצורה ידועה ואמינה. קוד זה עוזר לדמות תהליך זה.   
אירוע reset\_trigger: תאר לעצמך שיש אות מיוחד בשם reset\_trigger שאומר למכשיר להתחיל בתהליך האתחול. כאשר אתה מגדיר את האות הזה, זה כמו לחיצה על כפתור "התחל" במכשיר שלך.   
מחכה לשעון: בתוך המכשיר שלך, יש שעון שמתקתק בקביעות. קוד זה ממתין עד שהשעון יתקתק 10 פעמים (כל סימון הוא "קצה שלילי" של אות השעון). תקופת המתנה זו היא כמו לתת למכשיר שלך קצת זמן להתכונן.   
אות איפוס: לאחר מכן, הוא מגדיר אות שנקרא איפוס ל-1. אות זה אומר למכשיר לאפס את עצמו. זה כמו להגיד למכשיר שלך לנקות את הזיכרון שלו ולהתחיל מחדש.   
מחכה שוב: לאחר הגדרת האיפוס ל-1, הקוד ממתין 15 יחידות זמן. במהלך הזמן הזה, המכשיר נמצא במצב איפוס, כמו כשאתה מפעיל את המחשב שלך והוא מציג את לוגו האתחול.   
כיבוי איפוס: לבסוף, הוא מחזיר את האיפוס ל-0. זה אומר למכשיר שתהליך האיפוס נעשה, והוא יכול להתחיל את פעולתו הרגילה.

58 – 69  
בקטע קוד זה אנחנו בעצם מפעילים את כל הקוד.  
יצירת רנדום דאטה ראשון:  
מבחן אתחול: שידור הנתונים האקראי הראשון משמש כשלב אתחול. זה כמו לספק נתונים ראשוניים למכשיר כאשר הוא מופעל או מאופס. זה עוזר להבטיח שהמכשיר מאתחל כהלכה ויוכל לטפל בנתונים ברגע שהוא הופך לפעיל.  
יצירת רנדום דאטה שני:  
מבחן פונקציונלי: שידור הנתונים האקראי השני דומה יותר למבחן פונקציונלי. זוהי דרך לוודא שהמכשיר לא רק מאתחל אלא גם ממשיך לפעול כהלכה לאחר הפעלה או איפוס. שלב זה מסייע להבטיח שהמכשיר פועל כמצופה במהלך פעולה רגילה, לא רק במהלך האתחול.

כללי:  
נשתמש ב begin ו end כמו { } כאשר יש לנו יותר משתי שורות קוד בבלוק הספציפי