邏輯設計

邏輯設計期末報告

報告題目:可變換多功能存取器(自定義名稱)

學生:曹豈銘

學號:E94084066

授課教授姓名:賴槿峰

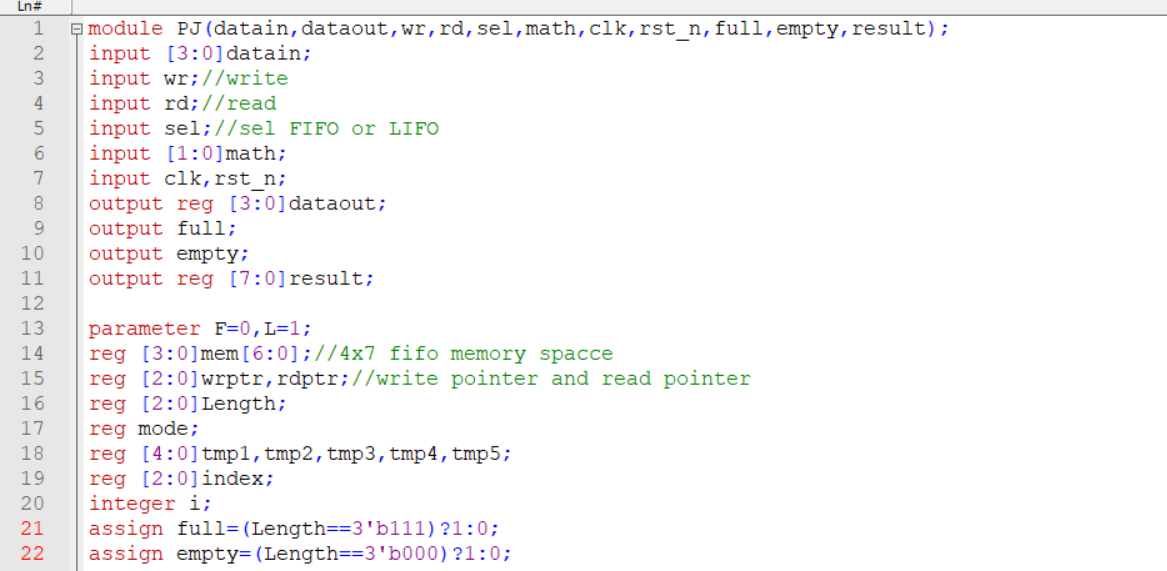
**專題構想**:

本學期第一次接觸verilog這種硬體描述語言，和以前學用來開發軟體的程式語言有著很大的不同，包括語言的描述方式及設計時的邏輯概念，這些都令人印象深刻，而上了這學期邏輯設計的課程後，更是學到了不少邏輯電路的架構及原理，這當中序向電路才有的儲存功能最讓我感到特別與好奇，因此有了想對此部分去做一些運用和延伸的想法，在經過深思熟慮後，我決定基於課堂中學到的一些概念及電路去實現一個可以選擇讓數據以先進先出(first in first out)或後進先出(Last in first out)這兩種方式進出的存取器，並在數據存滿後，還能選擇對這些數據進行不同的處理和分析，如:全部相加(數據總和)，比大小及統計出數據中的偶數個數(獲取特定數據)等，以上是我設計「可變換多功能存取器(此名稱為自創)」的動機以及目的。

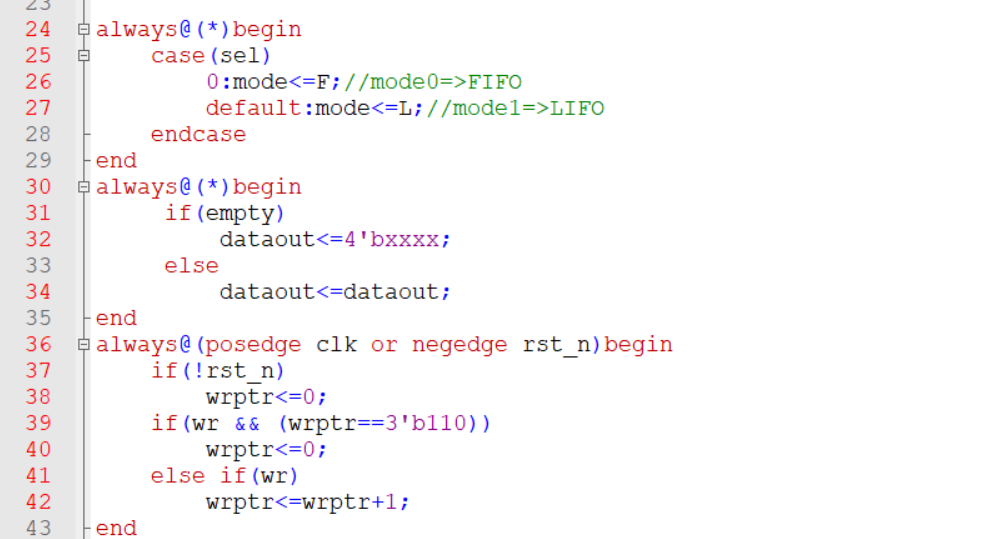
**預期目標**:

基於flip flop的功能和概念去實現存取器，並結合multiplexer來選擇該存取器為FIFO(first in first out)還是LIFO(Last in first out)，且輸入訊號有寫入(wr)及讀出(rd)這兩個訊號，當寫入為1，讀出為0，則輸入數據進入存取器中，反之將存取器內的數據輸出，一旦數據存滿整個存取器後，輸出訊號full為1，表示存取器已滿，不可再寫入數據;而若資料全部被讀出後，則輸出訊號empty為1，表示存取器已空，不可再讀出數據，最後是若數據存滿，透過multiplexer的概念和結合一小部分ALU的功能，選擇要將這些數據相加，還是找出最大或最小值又或者統計出數據中的偶數個數。

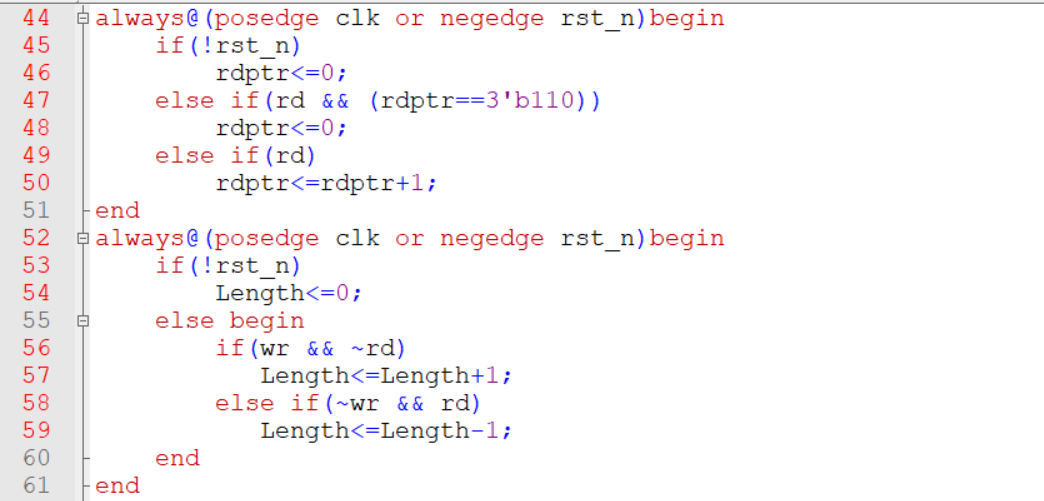
**實作方式**: **程式碼介紹與說明**



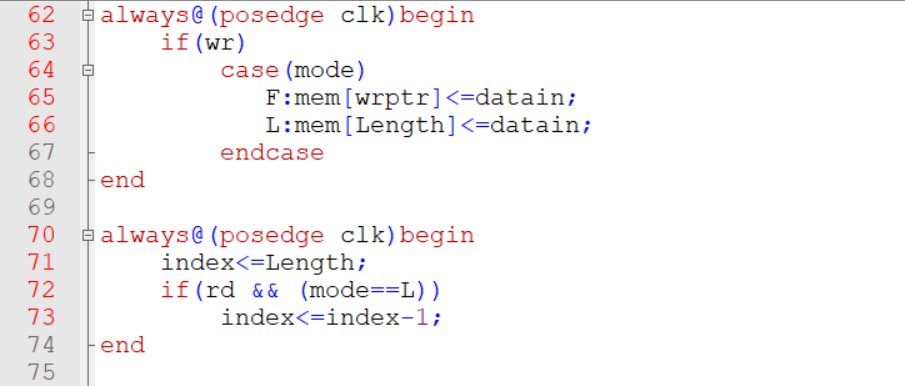
上圖主程式**line\_1**宣告模組名稱為PJ以及各個port，**line\_2~line\_7**宣告input port包含4bit的輸入數據(datain)，寫入(wr)、讀出(rd)、選擇mode的selection bit(sel)、選擇最後要做哪種數據處理2bit的math，最後是clock(clk)和reset(rst\_n)，**line\_8~line\_11**宣告output port分別有輸出數據(dataout)、存取器滿(full)、存取器空(empty)，以及數據處理後結果(result)，**line\_13~line\_20**分別是代表mode的參數F=0、L=1、存取空間(mem)為4x7、3bts的wrptr和rdptr去模擬寫入指標和讀出指標，由於此存取式記憶體的讀寫並非取決於空間位置，因此這部分需要以電路的思維去著手，接著是空間的長度Length、模式mode、運算用到的暫存空間(tmp1~5)、實現LIFO模式所需的index和for迴圈用到的整數i，**line\_21**表示當記憶體長度=7時為滿，**line\_22**表示當記憶體長度=0時為空。



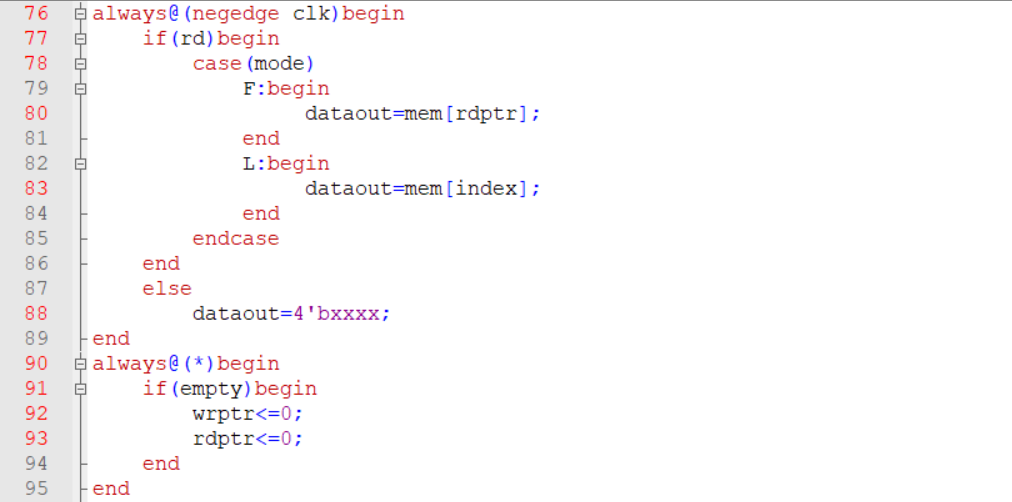
上圖**Line24~29**:多工器(multiplexer)概念的運用，藉由sel去決定模式(mode)為F(FIFO)還是L(LIFO)，**line\_30~line\_35**:若mem已空，dataout不保有上一個輸出值(模擬將資料取出)因此令其為未知，反之不影響dataout，**line\_36~43**:用到flip-flop序向電路概念，時脈正緣觸發或重置負緣觸發，若rst\_n為0則wrptr重置;若寫入信號(wr)為1且目前wrptr值為6也將其重置，因為mem[0]到mem[6]都已寫入;而若為wr則wrptr加1。



上圖**Line\_44~51**和line\_36~43相似，只是rdptr值改變的依據為讀出訊號(rd)，**line\_52~61**:一樣用flip-flop序向電路的概念，若rst\_n為0，重置Length;反之rst\_n為1，且寫入不讀出，記憶體長度加1，而若為讀出不寫入長度減1。

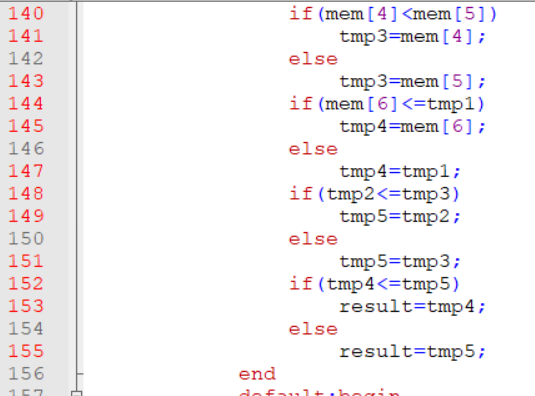


上圖**Line\_62~68**:此部分為寫入數據的運作，表示每次遇到時脈正緣時，若此時寫入訊號為1，先判定目前的使用模式為FIFO還是LIFO，若為FIFO，則數據depend on wrptr的值輸入到存取器中，而若為LIFO，因為其讀取數據是從後面開始，所以輸入數據則是depend on 當前存取器長度。**Line\_70~74**:index的定義是用來實現LIFO後進先出的特性。



上圖Line76~89:此部分為數據輸出的運作，因為一般初始的clk為0，而寫入訊號(wr)和讀出訊號(rd)的變換在測資中為一整個時脈週期，會剛好與時脈的正緣差半個週期，所以數據的輸出是在每次遇到時脈負緣時，若讀出訊號為1，一樣判斷當前的模式，若為FIFO，則數據的輸出depend on rdptr的值，而若為LIFO，則數據的輸出depend on index的值，反之若此時並非讀出狀態，則輸出數據不保有上一個輸出值(數據已被拿出)，值得注意的是，只有數據的讀出是在時脈負緣發生時，而數據的寫入及full、empty訊號的轉變皆是發生在時脈正緣。**Line\_90~95**:為一般組合電路邏輯，表示當存取器為空，將rdptr和wrptr歸0，避免影響下一輪的工作狀態。

一張含有 文字 的圖片

自動產生的描述

一張含有 文字 的圖片

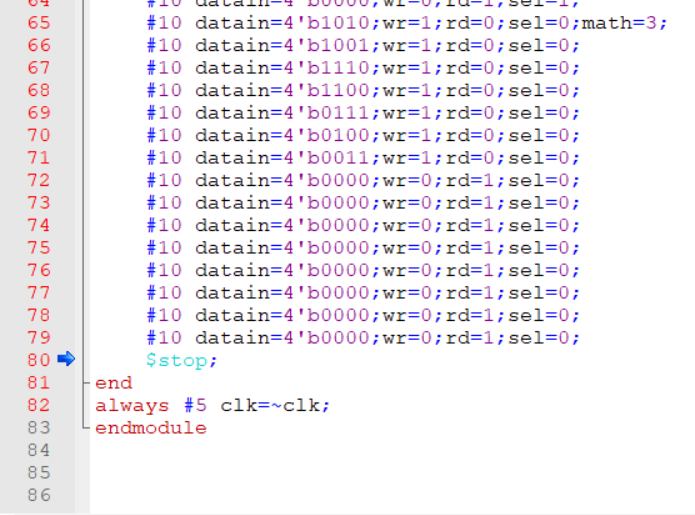
自動產生的描述

上圖**Line\_96**後開始為數據存滿後的處理及分析，若math為0則將存取器內的所有數據做相加，若math為1做找出數據中的最大值，math為2做找出數據中的最小值，math為3(default)則是找出存取器內共存了幾個偶數，若數據沒滿，則不對數據做處理和分析，即result為未知，且其不保有上一個result的值。

**成果展示**: **Testbench及模擬波形的說明**



一張含有 桌 的圖片

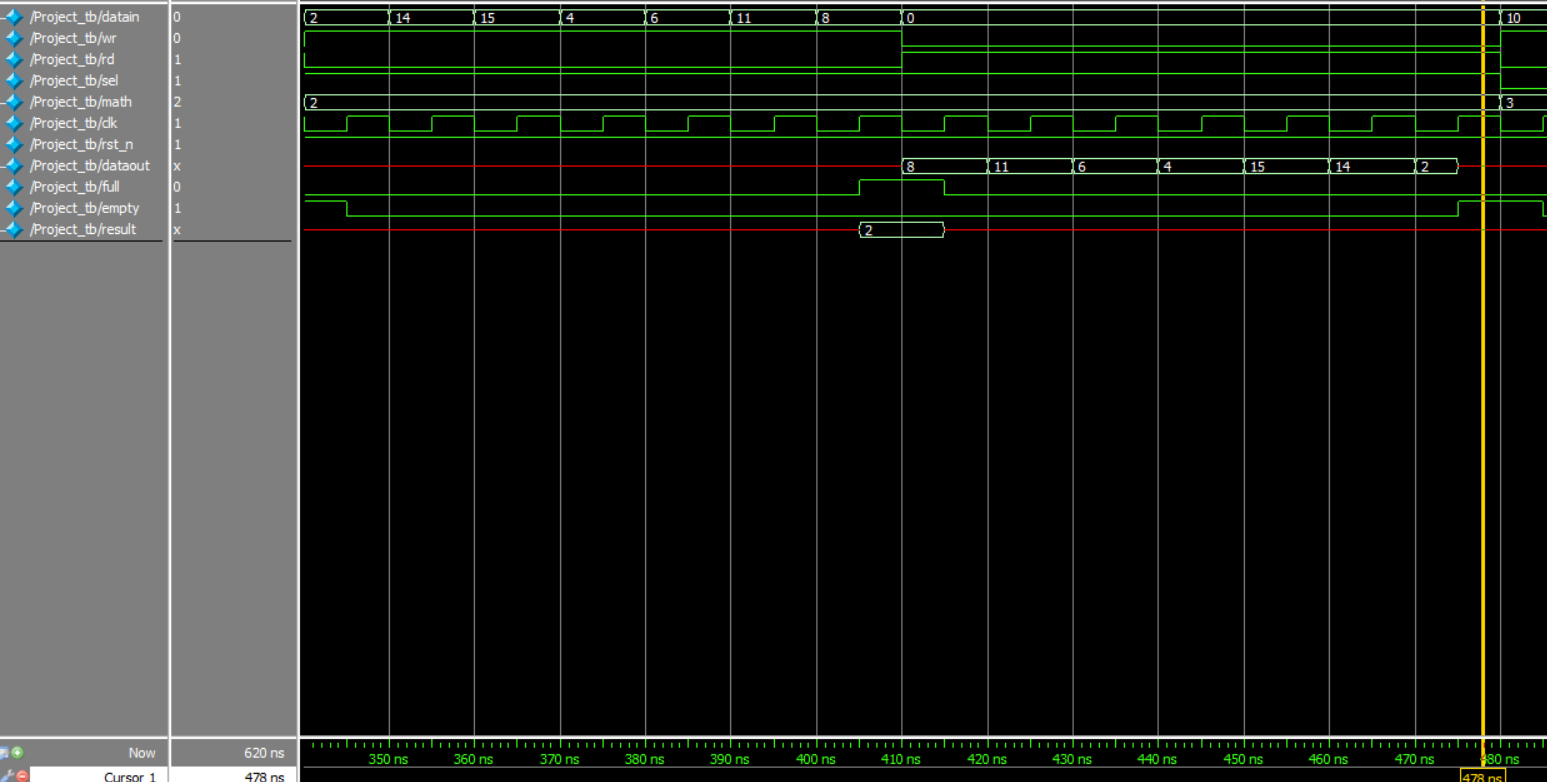
自動產生的描述

上圖**line\_1~14**:testbench的基本宣告，包括時間的單位和精度、testbench名稱、訊號線、和宣告要被測試的module PJ(名稱取為p1)。**line\_15**後開始為主要將訊號傳入的驗證區，大致可以依存滿數據後執行的功能(math)分成4大部分，**line\_16**:給所有input初始值，**line\_17~18**:利用負緣觸發的rst\_n對電路進行重置，之後開始輸入input值，**line\_19~34**:此為第一部份，sel=0，表示選用FIFO先進先出模式，透過模擬波形可看出，起初存取器為空(empty)當寫入(wr)為1，每次時脈正緣來時將數據寫入存取器中，8寫入後，存取器同時不為空，empty由1轉0，**Line\_19~24**皆是將數據寫入，**Line\_25**轉為輸出，輸出時datain的值不影響整體工作狀態，讓其為0方便觀察波形，又因為讀出發生在時脈負緣，因此數據得以在讀出(rd)訊號為1時，立刻反應輸出的動作，將最先輸入的8拿出存取器，隨後再次輸入數據，當數據3遇到時脈正緣，將其輸入存取器中，時此時存取器已存滿，full由0轉1，當前math功能選擇0，result則為4、14、5、3、6、2、3的總和等於37 ，隨後再次輸出數據，遇到時脈負緣率先輸出最前頭的4，到下一個時脈正緣時檢查到4已被輸出，因此full由1轉0，同時未滿足result的輸出條件，因此其再次為未知，照先進先出規則輸出完最後一筆數據3後，遇到下一個時脈正緣，檢查到存取器為空，empty由0轉1，同時dataout的值不做保留。

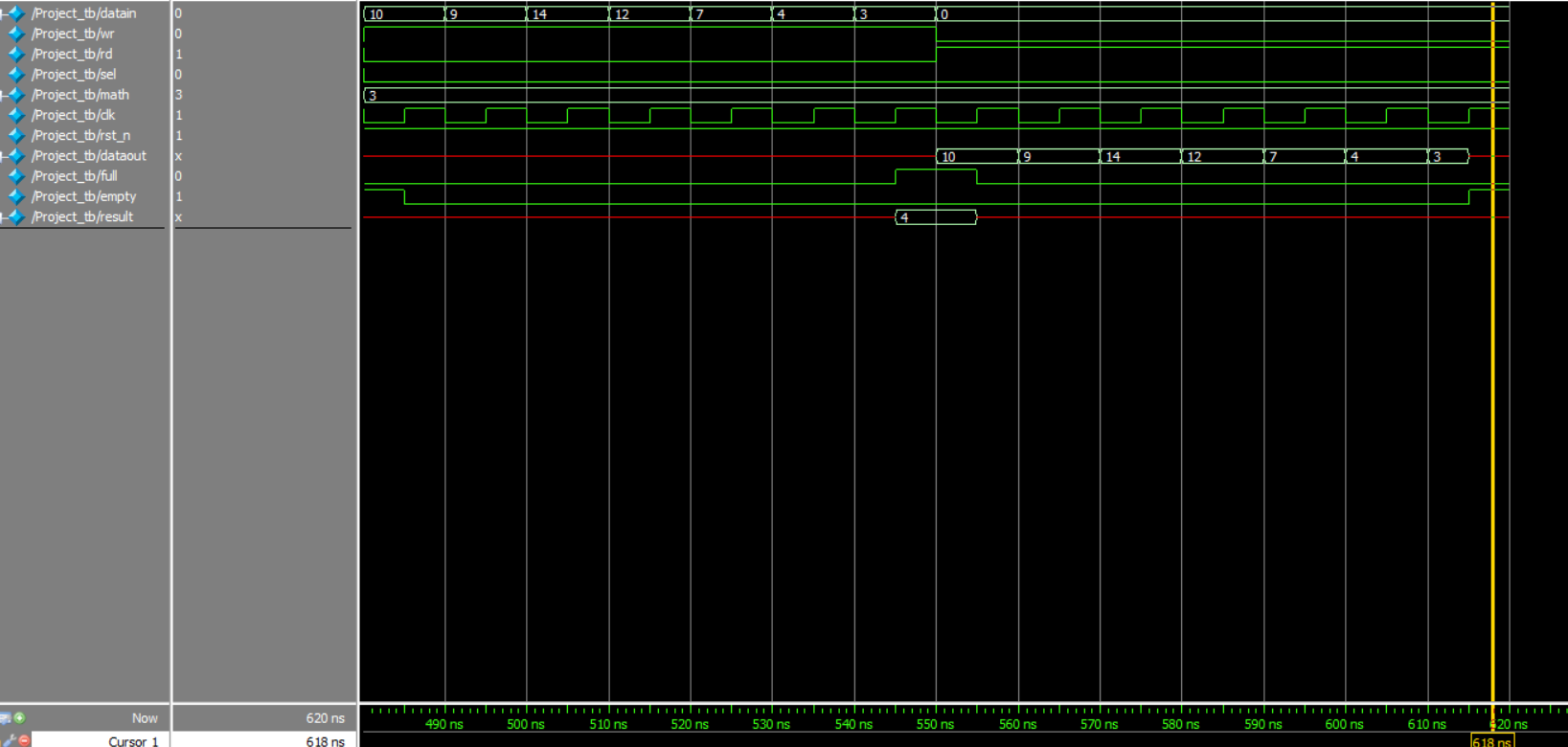
第一部份的波形

**Line\_35~50**:為第二大部份，sel為1，此時的模式改為後進先出(LIFO)，並且math選擇1，表示數據存滿後找出最大值，一樣配合testbench及模擬波形，可看出一開始因為數據已被全部讀出，empty為1，**Line\_35~40**:一樣當寫入(wr)為1、讀出(rd)為0時，輸入數據遇到時脈正緣就將數據寫入，寫入第一筆的同時empty由1轉0，**Line\_41**轉為輸出，將最後輸入存取器內的數據15取出，而後再次輸入數據，直到數據11遇到時脈正緣被輸入存取器內，此時存取器已滿，full由0轉1，result則為11、1、10、12、7、9、6中的最大值等於12，然後再次輸出數據，依後進先出原則首先輸出6，下一個時脈正緣來時檢查到6已被輸出，full由1轉0，同時result變回未知，輸出完最後一筆數據11後，下一個時脈正緣檢查到存取器為空，empty由0轉1，同時dataout的值不做保留。

第二部份的波形

**Line\_51~64**:為第三大部分，此時模式依然維持後進先出(LIFO)，但math變為2，表示數據存滿後找出最小值，由於第二部分已經展示過中途拿出並再輸入可以正常運行，此部分單就以另一組數據展示math為2，存滿後找出數據中最小值的功能，一樣起初為empty，然後開始寫入數據，存取器滿後，比較所有數據，得出result為最小值2;接著以後進先出方式清空存取器，最後empty再次由0轉1，dataout的值不做保留。

第三部份的波形

**Line\_65~79**:第四大部分，此時模式為先進先出(FIFO)，math為3，表示數據存滿後統計出數據中有多少個偶數，第一部分已經展示中途拿出再輸入的正常運行，所以此部分也是單就另一組數據展示math為3的功能，照相同步驟寫滿存取器，得出result為4，代表數據中有4個偶數，而後依先進先出規則輸出數據，直到存取器為空。**Line\_75**:simulation停止並彈出，**Line\_77**:表示clk每5單位時間變換一次。

第四部份的波形

**未來展望**:

目前此project的存取空間大小算是基本小型，且math功能也是以目前能力去實現的一些運用，以下將列出未來可調整及擴充應用的部分。

1. **存取器大小的擴充**:擴大存取器的大小使其能有效地對於大數據做存取，並配合math的功能，可以有效提高使用效率。
2. **math功能的擴充應用**:首先是存滿後加總這個部分，一旦擴大存取器的容量後，將可用於許多面向，如物件使用量的計算，像電費、車輛里程數等等。再來是比大小的部分，擴充後可用於篩出符合特定目標的最佳理想值，最後是得出數據中所含偶數個數這部分，可將其擴充為分類的工具。當然也可加入更多的math功能，使此存取器能有更靈活的運用。