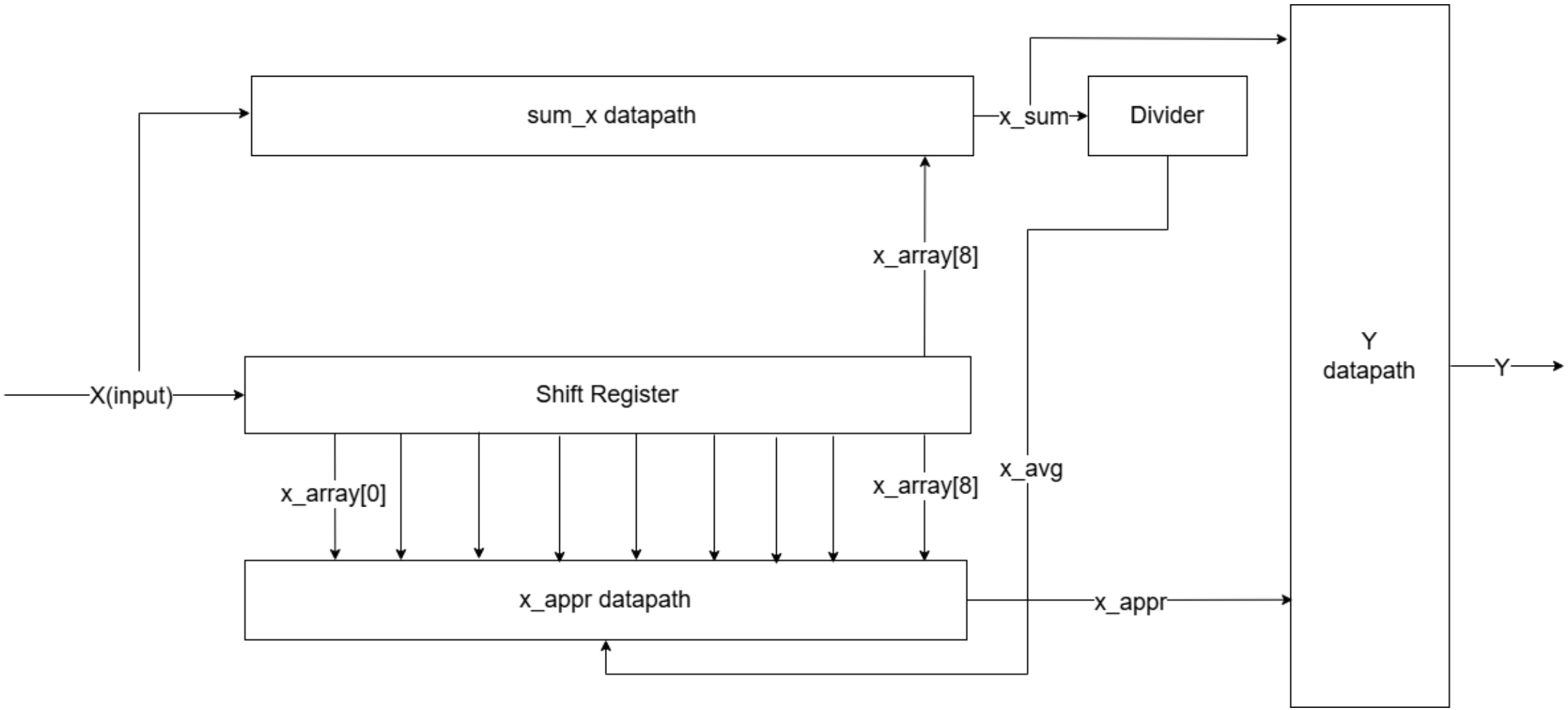
HW1: Approximate Average

黃偉峰 E34106010

Architecture Diagram


sum_x datapath
x_sum
Divider
x_array[8]
Y
datapath
Y
X(input)
Shift Register
x_array[0]
x_array[8]
x_avg
x_appr datapath
x_appr

Circuit Design Description

題目特點

2.4 Timing Diagrams

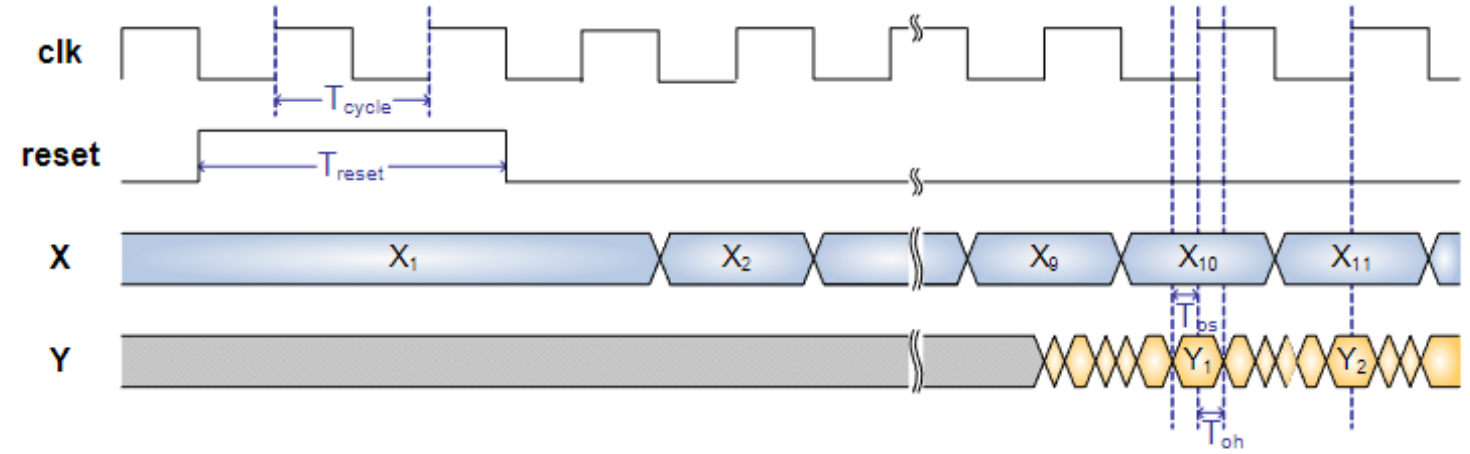
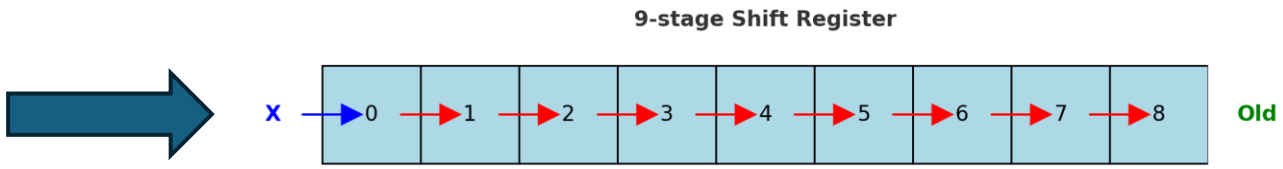


Fig. 2 I/O Timing Diagram

Y output再經過9個值後，第10個值時就必須計算完成，並且持續跟著X input計算出正確的Y，故需將計算過程透過合理安排能夠在一個clock內做完題目要的計算。

Shifter Register

```
always @(posedge clk or posedge reset) begin //x_array
    if(reset)begin
        for(i = 0; i <= 8; i = i + 1)begin
            x_array[i] <= 0;
        end
    end else begin
        x_array[0] <= X;
        for(i = 1; i <= 8; i = i + 1)begin
            x_array[i] <= x_array[i-1];
        end
    end
end
```



透過一個Shifter Register，每次有新的x進來時存入x_array[0]，並把所有位置上的值往下一個位置移動。

```
always @(posedge clk or posedge reset) begin //sum_x
    if(reset) sum_x <= 0;
    else begin
        sum_x <= sum_x + X - x_array[8];
    end
end
```

在上述過程中每次存入新值就將sum_x加上新值並減掉即將被覆蓋掉的值，以此可以在每次新值進來時將Shift Register當中的總和計算出來。

X_avg, X_appr

$$Xappr_j = \begin{cases} Xappr_j = Xavg_j \\ \quad \text{if } Xavg_j \in XS \\ X_i \mid (X_i \in XS) \text{ and } (X_i < Xavg_j) \text{ and } (Xavg_j - X_i \text{ is minimal}) \\ \quad \text{if } Xavg_j \notin XS \end{cases} \quad \cdots\cdots\cdots\cdots (3)$$

where $Xappr_j$ is the value of the jth approximate average.

```
always @(*) begin //x_appr
    x_appr = 0;
    for(i = 0; i <= 8; i = i + 1)begin
        if((x_array[i] <= x_avg) && (x_appr < x_array[i])) x_appr = x_array[i];
    end
end
```

X_appr根據題目定義為在shift register當中最大且小於等於x平均值的數值，因需要在一個cycle內完成，所以透過組合電路方式一次判斷完shift register內的8個element。

```
assign x_avg = sum_x / 9;
```

x_avg直接使用組合電路除法器除9完成

Output Y

$$Y_j = \left\lfloor \frac{\sum_{i=j}^{j+n-1} (X_i + Xappr_j)}{n-1} \right\rfloor \quad \text{.....(4)}$$

```
assign Y = (sum_x + (x_appr << 3) + x_appr) >> 3; //Y
```

透過先前已經算好的x總和，再額外加上9倍的X_appr，最後除上8即為Y值，
實作上加上9倍的X_appr，改寫成左移位3位獲得8倍的X_appr並加上X_appr獲得9倍X_appr，
除8則以向右移3位代替。

Functional Simulation Report

```
VSIM 29> run -all
# ------------------------------------------------
#
# ------------------------------------------------
#
# All data have been generated successfully!
#
# -------------------PASS-------------------
#
# ------------------------------------------------
#
# ** Note: $finish    : C:/Users/kartg/Desktop/IC_2025/2005/testfixture.v(124)
#    Time: 20040 ns  Iteration: 2  Instance: /test
# 1
```

Learn from this homework

Learn from this homework

- 這次作業學到最多的應該是verilog語法當中的for loop，在過去寫verilog時很少會用到它，但剛好這次作業中我的寫法有很多重複的部分，於是使用了for loop去展開電路，因為很少用所以還有特別去查過相關的語法，順便也學習了其他類似的語法e.g.generate等等，並且在這次作業中嘗試把自己的設計用Block diagram的方式表示出來，這也是以前很少做的。