## C Based VLSI Design – ISA 1

Name: Rajath J

SRN: PES1201801707

**Aim**: Write a C-program for a Fibonacci series to n-terms and apply minimum latency scheduling under resource constraints.

Note:

Functions working on recursions is not used. Since recursion can't be implemented in HLS.

Since a program for a Fibonacci series to n terms is asked. The array version of the Fibonacci series is implemented and scheduling will be done for the same. The methods to calculate the n-th term like the formula method ( $F(n) = \varphi^n/sqrt(5)$ , where $\varphi$ is the golden ratio ) and the Matrix multiplicative method cannot be used as the time complexity and resource requirement would increase.

Driver Code:

```c
// Driver code
int main()
{
    long int n = 0;
    int ret;

    while(1)
    {
        printf("Enter the value of \"n\" upto to which you want the Fibonacci series to be\
generated\n");
        printf("n : ");
        ret = scanf("%ld",&n);
        if( ret != 1 ) // Checks if the input was an integer
            {
                printf("\nOnly natural numbers are allowed !!\n\n");
                getchar(); // Get the floating input
            }
        else if( n <= 0 ) // Checks if the num is greater than 0
            printf("\nOnly values of n>0 will be accepted\n\n",n);
        else
            break;
    }

    unsigned long long int arr[n];

    // This is the main function
    Fibonacci(arr, n);

    //print the sequence to verify
    printf("*************************\
        \nThe Fibonacci series is\
        \n*************************\n");
    for( int i = 0; i<n; i++)
    {
        printf("%llu \n",arr[i]);
    }

    return 0;
}
```

This is the driver code for our C-program. This makes sure that values of n>0 will only be sent to the Fibonacci function. Since the Fibonacci values blow up after just a few iterations and are always positive, the **unsigned long long int** type is used.

Minimum Latency under Resource Constraints ( MLRC ):

The Fibonacci series is given by $T_n = T_{n-1} + T_{n-2}$. This means that it will always have a maximum time complexity of O(n). Where n is the number of terms. Hence the maximum allowable latency is "n". Since the first two terms are always known the total latency would be "n-2".

Furthermore, the present operations are dependent on the previous operations. This gives a delay constraint of "$x_{i,1} = 1$, $x_{i,2} = 1$ where i = {0,1,2,3...n}". Meaning the n-th term can always be scheduled at only the n-th time step.

Solution for MLRC. The ASAP and ALAP scheduling for the problem would be exactly the same based on the above delay constraints. Hence, we will be solving the Minimum Latency under Unconstrained resources and then apply the resource constraints. The code would be changed appropriately.

The Fibonacci function:

V1. Minimum Latency under Unconstrained Resources.

```c
// The function accepts an array
// to fill the series
void Fibonacci(unsigned long long int *arr, unsigned long int n)
{
    // T(n) = T(n-1) + T(n-2)
    arr[0] = 0;
    arr[1] = 1;
    unsigned long int j = 2;

    while(j<n)
    {
        arr[j] = arr[j-1] + arr[j-2];
        j = j + 1;
    }
    return;
}
```
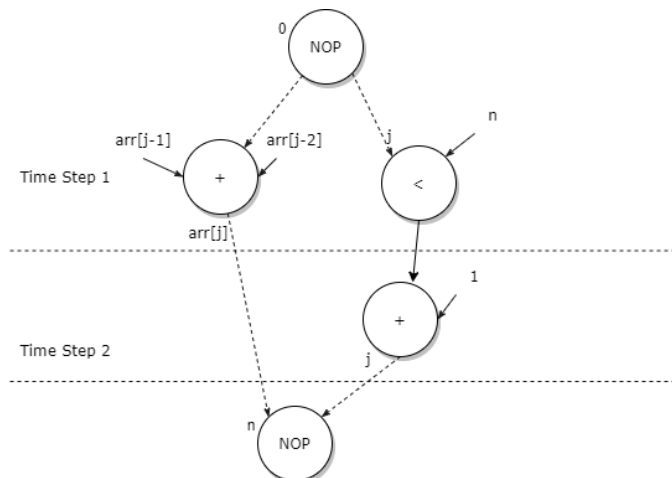
Resources Used:

2 ALU, Memory with a write port and with at least 2 active read ports which allow for simultaneous read operations, register j.

Working:

The values of 0 and 1 would be stored in the array at the $0^{th}$ and $1^{st}$ index locations respectively. For each successive value, the value in the previous 2 index locations would be summed and result would be stored at the current index. The sequence graph is given below.
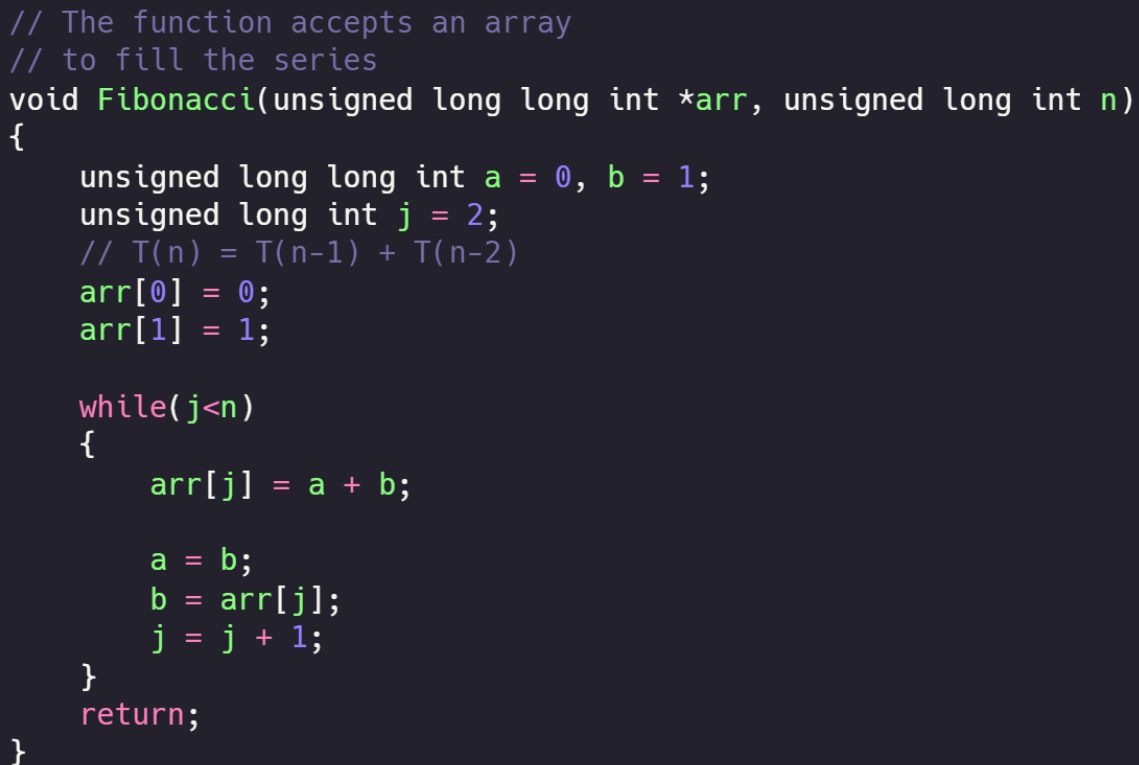
Sequence Graph:
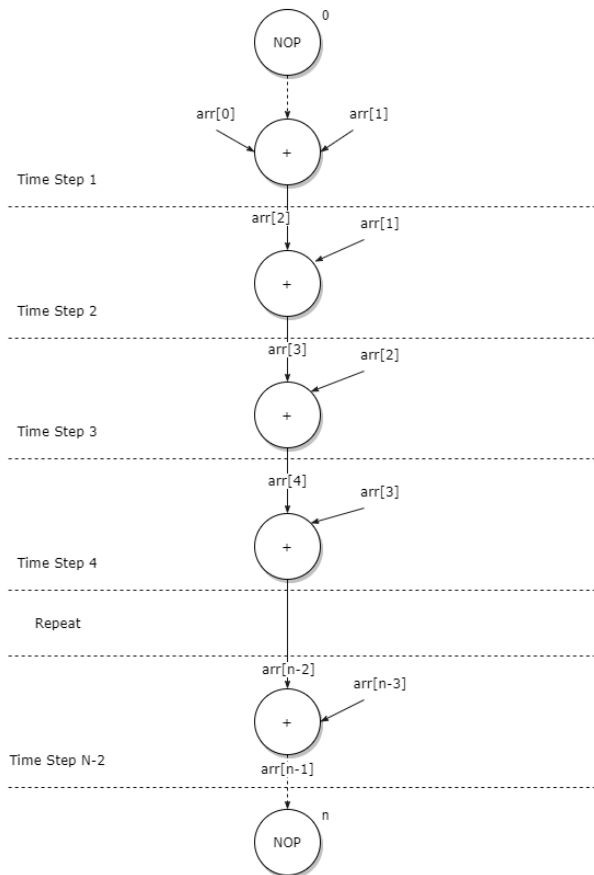


Discussion:

The graph gives us the Minimum latency ( The array is unrolled ). But for large values of n this is not feasible. Hence the general version given below is used.

But at each iteration we have two read cycles of the array values. Parallel access of memory locations has to be minimized. Optimization of resources is now needed. By using 2 temporary registers, we can use an Array with only one active read port. Meaning, have a single read cycle in one iteration.

The Fibonacci function:

V2. Minimum Latency under Resources Constraints.

```c
// The function accepts an array
// to fill the series
void Fibonacci(unsigned long long int *arr, unsigned long int n)
{
    unsigned long long int a = 0, b = 1;
    unsigned long int j = 2;
    // T(n) = T(n-1) + T(n-2)
    arr[0] = 0;
    arr[1] = 1;

    while(j<n)
    {
        arr[j] = a + b;

        a = b;
        b = arr[j];
        j = j + 1;
    }
    return;
}
```

Resources Used:

2 ALU, Memory with a write port and read port and 3 registers ( register j, a and b ).

Working:

The values of 0 and 1 would be stored in the register a, b and in the array at the $0^{th}$ and $1^{st}$ index locations respectively. For each successive value, the register values would be summed up and written to the current index of the array. The registers would then be updated with "a = b" and "b = arr[j]" respectively. This makes sure that only one read cycle occurs in an iteration. The sequence graph is given below.

Sequence Graph:

Initial step: a = arr[0] = 0, b = arr[1] = 1

After the value is loaded to arr[j]. We need to update j to j + 1, move b into a and arr[j] into b, all of which happen in the same step.
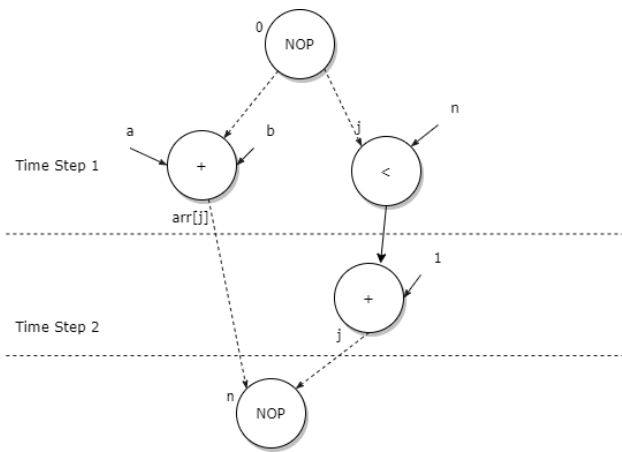
Discussion:

The above graph gives the minimum latency under resource constraints.

Note: We can't update "j" before the values of a+b is loaded into arr[j] and before checking if j<n. We may use only the register a. But that would cause 2 read cycles to happen in a loop.

Code Execution:

```
PS C:\Users\rajath\Downloads\C-based VLSI> .\fibonacci.exe
Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : 1
************************
The Fibonacci series is
************************
0
PS C:\Users\rajath\Downloads\C-based VLSI> .\fibonacci.exe
Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : 2
************************
The Fibonacci series is
************************
0
1
PS C:\Users\rajath\Downloads\C-based VLSI> .\fibonacci.exe
Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : 3
************************
The Fibonacci series is
************************
0
1
1
```

```
Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : -5

Only values of n>0 will be accepted

Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : l

Only natural numbers are allowed !!

Enter the value of "n" upto to which you want the Fibonacci series to be generated
n : 6
************************
The Fibonacci series is
************************
0
1
1
2
3
5
```
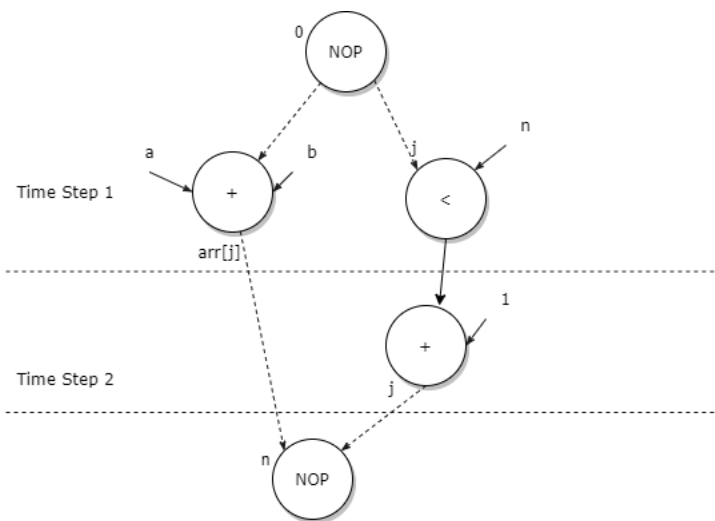
Result:

C-Program to generate a Fibonacci series to n-terms was written and MLRC was successfully applied. The sequence graph is given below.



The program will have a time step of 2 in each iteration.

The resources used were 2 ALU, Memory of n-locations and 3 registers.

The code can be found at https://github.com/rajathjn/C-Based-VLSI-ISA-1.