Register Transfer Level Design with Verilog

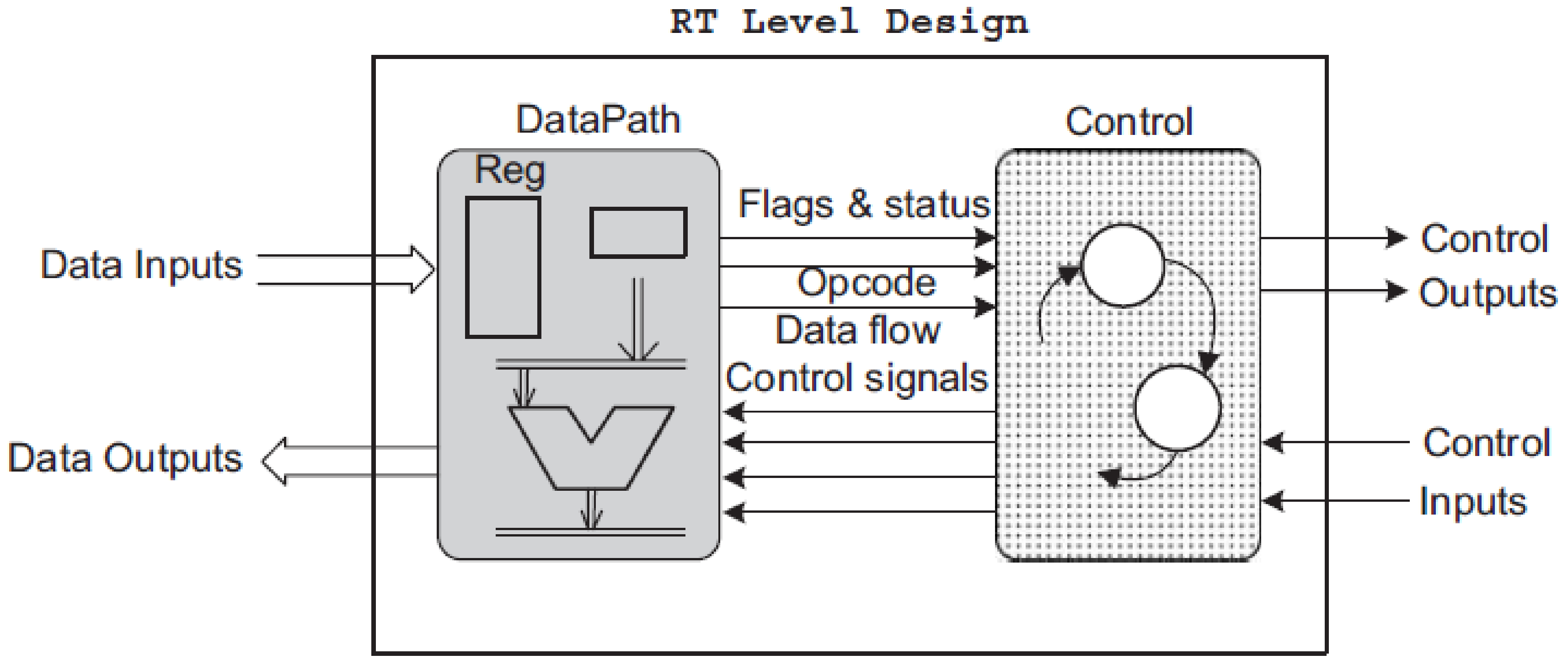
(Synchronous Digital System Design)

Introduction

- Design of small hardware components can usually be done by describing the hardware for synthesis and synthesizing and implementing the design by appropriate computer aided design tools.
- On the other hand, a large design requires proper planning, architectural design, and partitioning before its various parts can be written in Verilog for synthesis.
- Taking a high-level description of a design, partitioning it, coming up with an architecture for it (i.e., designing its bussing structure), and then describing and implementing various components of this architecture is referred to as RT level design.

Controller/datapath partitioning

The first step in an RT level design is the partitioning of the design into a data part and a control part. The data part consists of data components and the bussing structure of the design and the control part is usually a state machine generating control signals that control the flow of data in the data part.



Datapath

The data part of an RTL design consists of the interconnection of data components that are, registers, combinational logic units, register files, and busses that interconnect them. The data part, which we also refer to as the data path, has external data inputs and outputs, as well as control inputs and outputs from and to the control part.

DataPath Module

```
module DataPath
   (DataInput, DataOutput, Flags, Opcodes, ControlSignals);

   input  [15:0] DataInputs;
   output [15:0] DataOutputs;
   output Flags, ...;
   output Opcodes, ...;
   input  ControlSignals, ...;
   // instantiation of data components
   // ...
   // interconnection of data components
   // bussing specification
endmodule
```

Control part

- The control part of an RTL design takes control inputs from the data part and external control inputs and depending on its state makes decisions as to when and what control signals to issue.
- The control part, which we also refer to as the control unit, consists of one or more state machines that keep the state of the circuit, make decisions based on the current data and data status, and control how data is routed and what operations are performed on the data in the data part.

Outline of a Controller

```
module ControlUnit
        (Flags, Opcodes, ExternalControls, ControlSignals);
    input   Flags, ...;
    input   Opcodes, ...;
    input   ExternalControls, ...;
    output ControlSignals;
    // Based on inputs decide :
    // What control signals to issue,
    // and what next state to take
endmodule
```

Design of a system

- Top-down approach.

it involves understanding the specification & doing partitioning the entire design into two parts, ~~maint~~ namely ⇒ Datapath (Functional units) & Control unit.

Steps to be followed.

1. Understand the design problem
   Specification should be completely unambiguous.

2. Identify the functional Components ('off the shelve or IPs) which may be used to design the System.

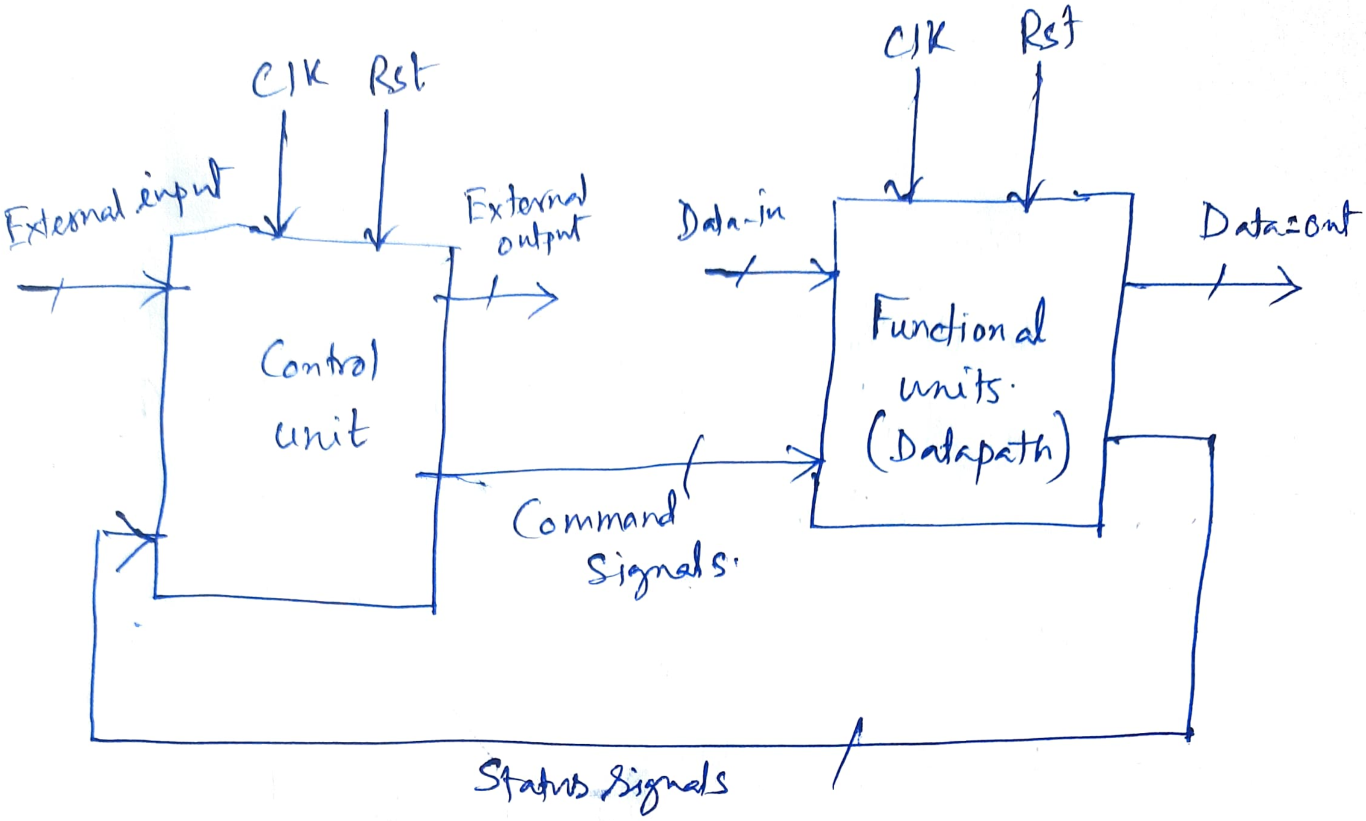
3. Identify the Signals (Ports) required ~~to~~ for proper functioning of the System.

4. Manually identify the Control unit signals which may be interfaced with the datapath (functional units). Particularly the Status Signals & Command Signals.

5. ~~Desi~~ Write the behavioural model ~~of the top modul~~. of the system in top-down fashion.

$$\begin{tikzcd}
 & \text{Datapath} \arrow[r] & \text{~~D~~ Component 1} \\
\text{Top-module} \arrow[ur] \arrow[dr] & \arrow[r] & \text{" 2} \\
 & \text{Controller unit} \arrow[r] & \text{3.} \\
\text{[Assemble the whole design]} & \text{(FSM)} &
\end{tikzcd}$$

Adder
multiplier
Subtractor
Register
MUX
Comparator.



Block-diagram of a Synchronous Digital System.

Control unit periodically check the status signals of the functional units & based on these signals it generate Commond signals. Based on these commands signals, the data flows through the datapath from one data-path component (FU) to another datapath component. The sequence of operations ~~is dec~~ in the datapath is decided by the command signals generated by the control unit.

Case-study :- I (~~Great~~ Greatest Common divisor)

Euclid's GCD Algorithm.

Now we are going to take up a specific Seqnontial algorithm & we would see how we can translate it to Hardware.

Algorithm.

```
a = read ();     }  Read two numbers given by user.
b = read ();     }        a & b ~~r~~ are positive.
while (a != b)
    if (a < b)        you should keep running the loop till a = b.
        b = b - a;
                      GCD (5, 5) = 5.
    else              GCD (10, 10) = 10.
        a = a - b;

  end if;
 end while
 print (a)  →  a or b is the GCD.
            →  print ~~t~~ GCD of a & b
```

Example.

a	b
42	16
26	16
10	16
10	6
4	6
4	2
2	2

⇒ a = b, So, GCD (42, 16) = 2.

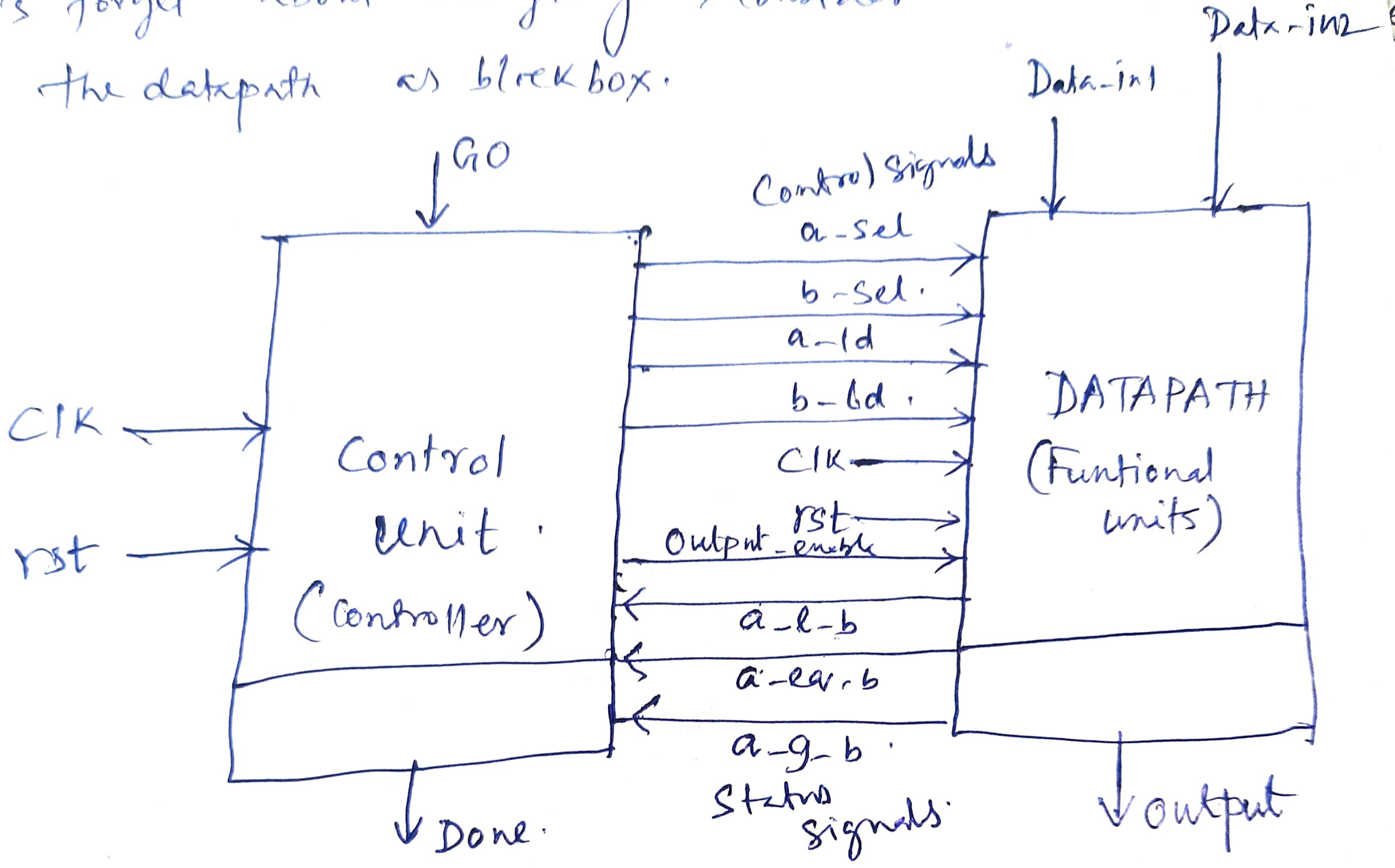
a	b
60	45.
15	45
15	30
15	15.

⇒ a = b.

So, GCD (60, 45) = 15.

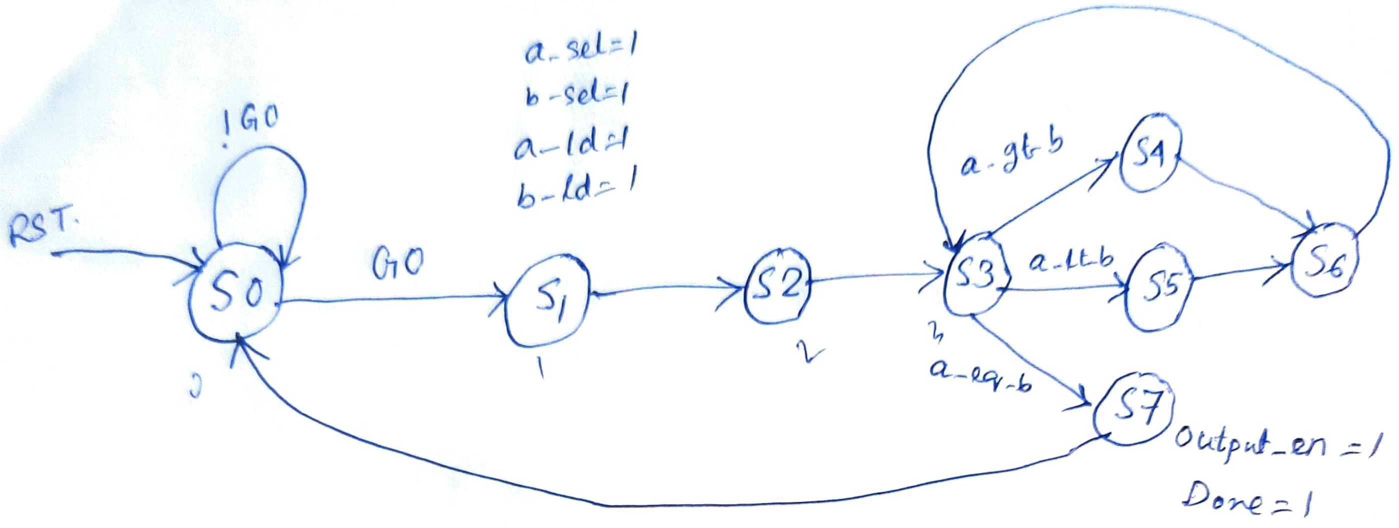
2 | 42, 16
  21, 8

let's forget about everything, consider the datapath as black box.



Controller

FSM



S0: Start/Reset state

S1 : Load values.

S2 : Wait state. for inputs to Load.

S3 : Comparison State.

S4 : a_sel=0 & a-ld=1 [a=a-b]

S5 : b-sel=0 & b_ld=1 [b=b-a]

S6 : Wait state

S7 : Done state.

0 1 2 3