Design of Arithmetic Processors

SKE 2263 Panel of Lecturers

June 5, 2013

1 Introduction

A processor is a digital device that implements either general or specialised algorithm described by a multi-step procedure.

Structurally, a processor are partitioned into two units:

- **Datapath Unit**: It processes input data into output data. A typical datapath consists of
  - registers to store variable of the algorithm
  - ALU and other combinational blocks that implement operation (operators) of the algorithm
  - Step counters that implement the loop counters.
- **Control Unit**: This is the "controller" which controls the operations performed by the data path and the proper sequencing of these operations. The controller is implemented as an FSM that may be designed in the conventional manner. The control unit receives two type of signals: control signals like "start of the processing, or " input data ready", etc, from the outside world and conditions from the datapath. The control unit generates two types of signals: opcodes to blocks of datapath, and status signals, e.g. "ready" to the outside world

Both parts of the processor are synchronised with the same clock signal.

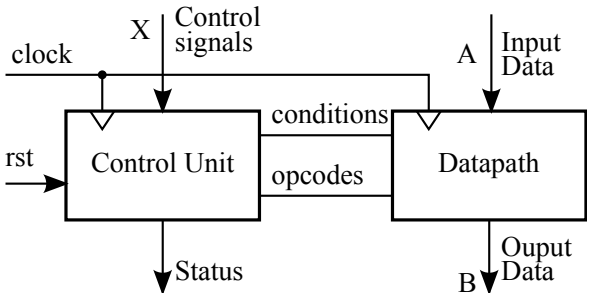


Figure 1: General structure of arithmetic processor.

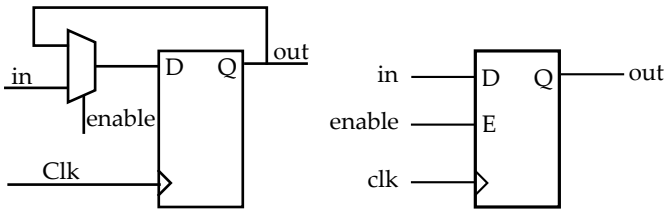
When designing an arithmetic processor, we build the datapath first and test it "manually". When the datapath unit is working, we then design the control unit to automate the datapath operation.

1.1 Building Blocks

Before we design anything, we list here a few circuits that are useful in digital system design

1.1.1 DFF with enable

A Data Enabled flip-flop or D flip-flop with enable (DFFE) has a mux at the data input, which is controlled by the enable signal. The enable input allows selective loading of the FFs.



(a) Schematic. (b) Logic symbol.

Figure 2: DFF with enable.

1.1.2 Shift Register with Parallel Load

The shift register based on the DFF with enable allows up selectively enable and disable shifting. We can also load all FFs in parallel.

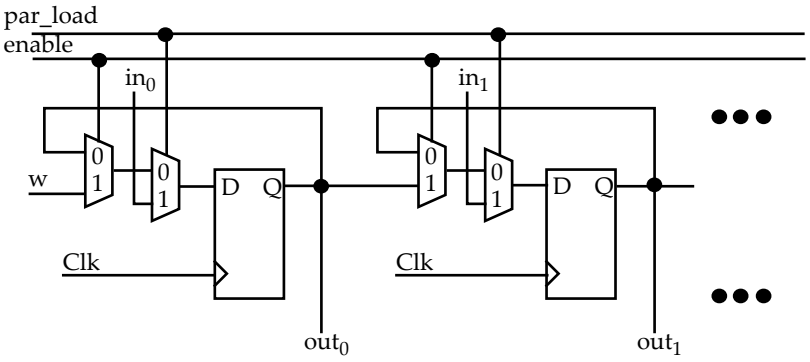


Figure 3: PIPO.

1.1.3 Loop control counter

The loop control counter is used to control how many times a task is repeated. The counter is a downcounter which is initialized with the number of loops required. For example, if a process is to be repeated 4 times, a 2-bit counter is initialized with 11. Each time a task is performed, the counter is decremented by 1, until it reaches zero. A zero detector consisting of NOR gates outputs a ZERO signal which is used by the FSM to stop further operations.

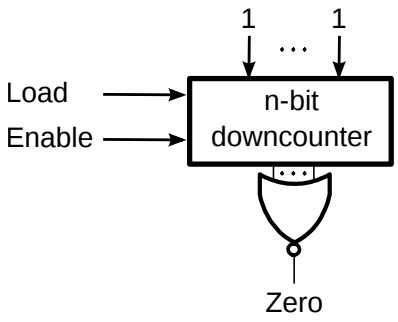


Figure 4: n-bit counter for loop control.

2 Bit-Serial Adders

A bit-serial adder is a digital circuit that can add any two arbitrarily large numbers using a single full adder. Just as humans, the serial adder operates on one pair of bits/digits at a time. Sequential serial adders are economically efficient and simple to build. Bit-serial architectures have been used successfully in many applications that are dealing with a bit stream such as signal processing, audio, video etc.

The price of this logic reduction is that the serial hardware takes n clock cycles to execute, while the equivalent parallel structure executes in one clock cycle.

2.1 Specification

We start with specification of the top level component (entity) and the algorithm to be performed.

Inputs

- **A**: First n-bit Operand (Addend)
- **B**: Second n-bit Operand (Augend)
- **st**: Start signal which initiates the addition operation
- **rst**: Reset signal which puts the controller into the initial state

Outputs

- **S**: The (n+1)-bit sum and carry ($S = A + B$)
- **done**: Active when the operation is complete

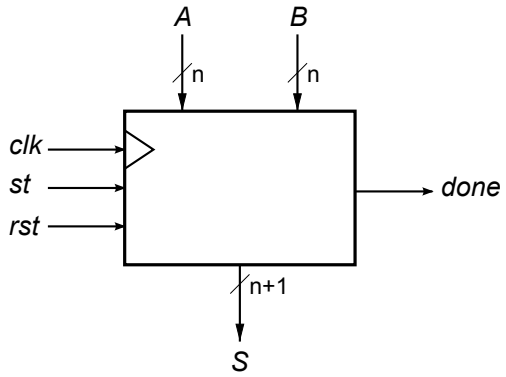


Figure 5: Serial Adder entity.

Subsequently we specify the concept of the algorithm in a top-level pseudo-language that can be of the following form

```
Load A, Load B, c = 0;
counter = 0
while (counter < n)
  c,S[k] = A[k] + B[k] + c
done = 1
```

Variables on the left-hand side of the equations are normally need to be stored in registers, hence, we infer that two n-bit registers A and B, and a 1-bit carry register, c are needed. Since the addition is performed in a serial fashion, a 1-bit adder is needed. Finally, a step counter is needed to count the number of speps.

2.2 Basic Circuit

A serial adder consists of a 1-bit full-adder and several shift registers. In serial adders, pairs of bits are added simultaneously during each clock cycle. Two right-shift registers are used to hold the numbers (A and B) to be added, while one left-shift register is used to hold the sum (S). A "first-cut" block diagram of a serial adder is shown in Figure 6.

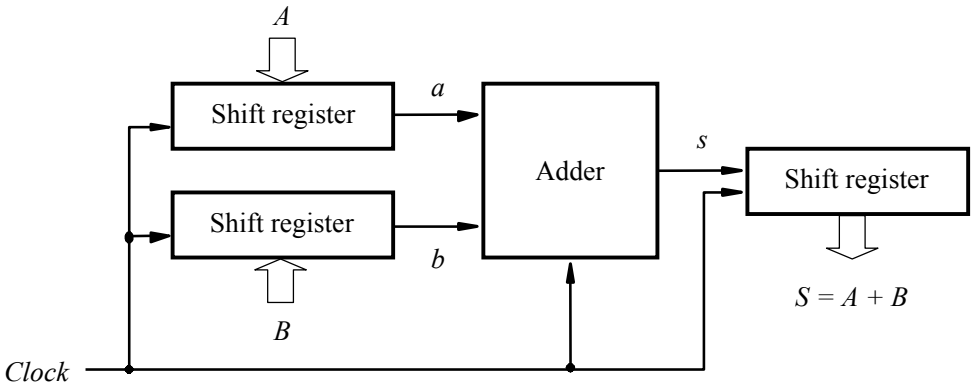


Figure 6: Basic idea of the serial adder circuit.

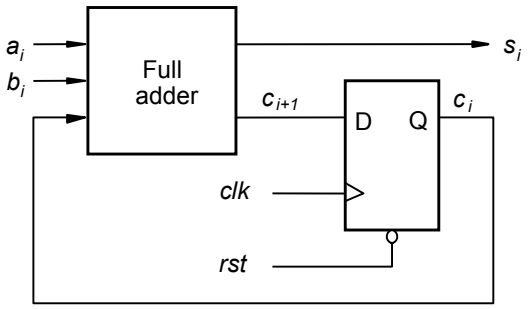


Figure 7: Full adder component.

An full-adder performs the addition operation on the values stored in the input shift registers and stores the sum in a separate shift register during several clock cycles. During each clock cycle, two input bits a_i and b_i are shifted from the two input right-shift registers into the 1-bit full-adder, which adds the two bits and evaluates the sum bit s_i and the carryout bit c_{i+1}. The sum bit s_i, is shifted out to the left-shift register and the carryout bit c_{i+1} is stored in the state memory of the serial adder for the next two bits. The time sequence of the operation of a 4-bit serial adder is illustrated in Figure 8.

A	*B*	*S*	s_i	c_{i+1}
1011	0011	0000	0	1
0101	0001	1000	1	1
0010	0000	1100	1	0
0001	0000	1110	1	0

Figure 8: Time Sequence of the Operation of a 4-bit Serial Adder

As you can see, the serial adder described here will complete the addition after 4 clock pulses. The clock pulses must be disabled after completion. Otherwise, the result that was calculated will be shifted out and disappear.

In an actual digital system, the clock is a continuous signal. To hold the results the S and C registers, extra controls must be added.

2.3 Serial adder datapath

Figure 9 shows the complete top level circuit diagram for the serial adder. Notice that a DFFE and 2-bit counter is now part of the datapath. The signals needed for controlling this DPU are:

- LA, LB, LC: load the register/counter
- EA, EB, EC, ED, ES: enable register/counter operation
- rst: reset DFFE

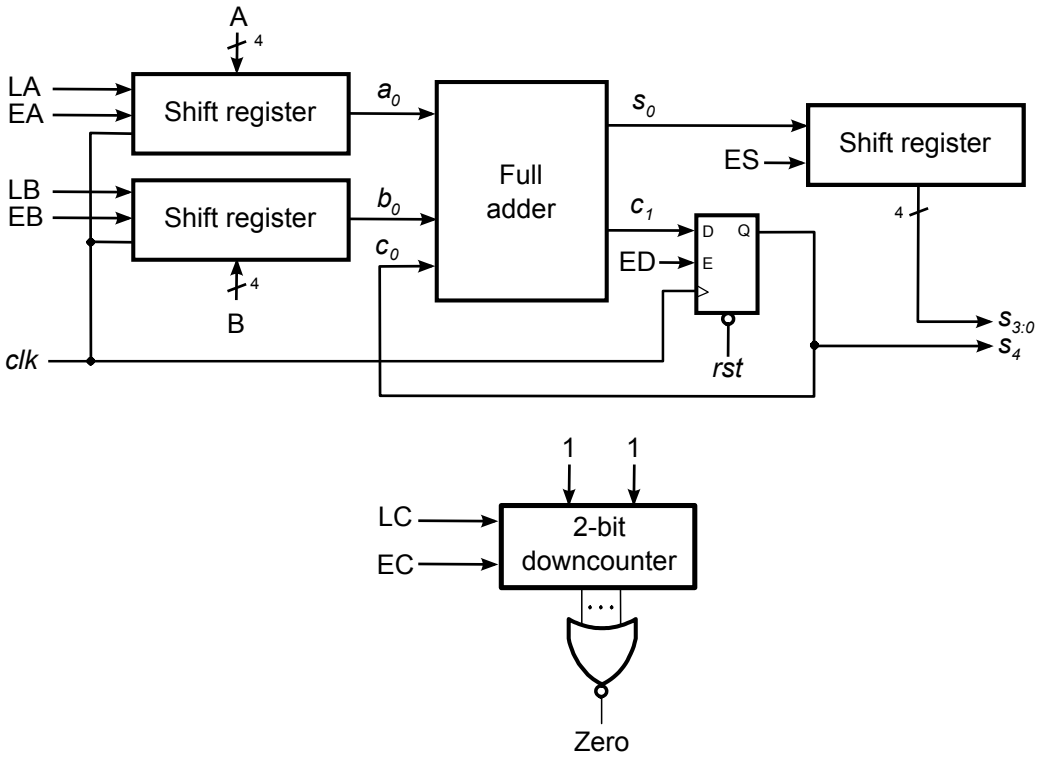


Figure 9: Complete datapath.

Here's how it operates:

- On reset, flip-flop D is cleared
- Idle state, waiting for *st* (start) signal: A, B registers and counter C are continously loaded. Flip-flop D remains cleared as it is not enabled.
- When start signal is received, repeat the following 4 times:
  - Shift registers A, B, S and flip-flop D
  - Decrement counter C
- After the bits have been shifted 4 times (signaled by Zero signal from downcounter), stop shifting and output the *done* signal. The 5-bit sum is now stable and can be used by other circuits.

This datapath can be simulated manually in Quartus by giving the desired inputs and clocking the datapath for at least 4 clock cycles.

2.4 The controller

Next we design the controller to automate the operation.

The high-level ASM describes the tasks the circuit is doing.

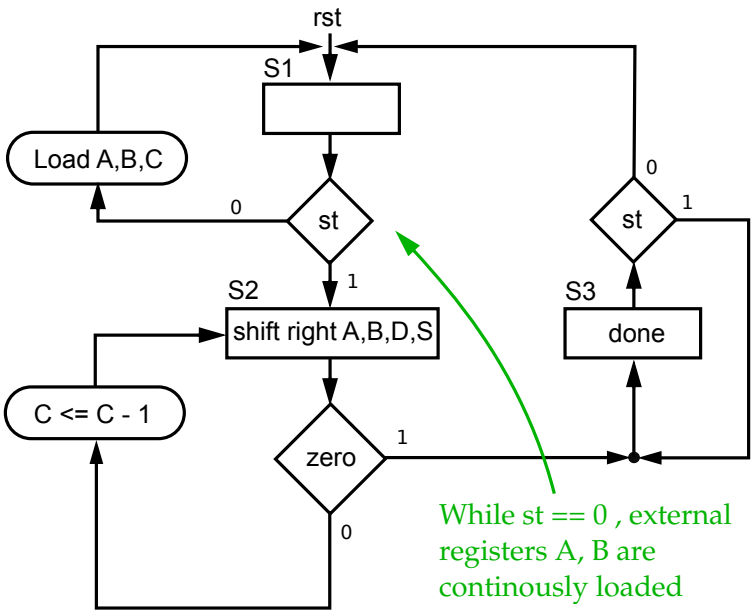


Figure 10: High-level ASM for serial adder.

The low-level ASM lists the status and control signals required for control unit operation.

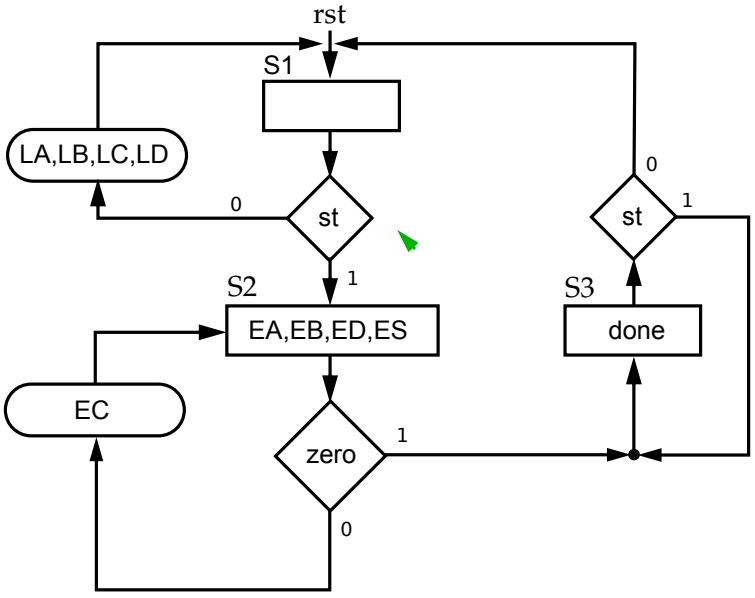


Figure 11: Low-level ASM for serial adder.

As the circuit is relatively simple and there are only three states, the FSM can still be designed using a state diagram. Conversion from ASM to state diagram is left as an exercise.