

EE141 – FALL 99 – TERM PROJECT

1. Designing a Viterbi decoder - Background

The Viterbi algorithm is commonly used in a wide range of communications and data storage applications. It is used for decoding convolutional codes, in baseband detection for wireless systems, and also for detection of recorded data in magnetic disk drives. The requirements for the Viterbi decoder or Viterbi detector, which is a processor that implements the Viterbi algorithm, depend on the applications where they are used. This results in very wide range of required data throughputs and power or area requirements. Viterbi detectors are used in cellular telephones with low data rates, of the order below 1Mb/s but with very low energy dissipation requirement. They are used for trellis code demodulation in telephone line modems, where the throughput is in the range of tens of kb/s, with restrictive limits in power dissipation and the area/cost of the chip. On the opposite end, very high speed Viterbi detectors are used in magnetic disk drive read channels, with throughputs over 600Mb/s. But at these high speeds, area and power are still limited.

In this semester's project we will design a critical part of a Viterbi decoder, under different design constraints.

1.1. The Viterbi Algorithm

The Viterbi algorithm is commonly expressed in terms of a trellis diagram, which is a time indexed version of a state diagram. The simplest 2-state trellis is shown in Figure 1.

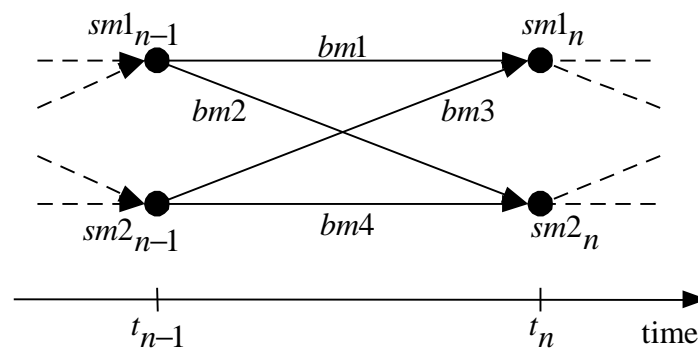


Figure 1: Two state trellis

Maximum likelihood detection of a digital stream with intersymbol interference can be described in terms of maximizing probabilities of paths through a trellis of state transitions (branches). Each state corresponds to a possible pattern of recently received data bits and each branch of the trellis corresponds to a receipt of the next (noisy) input.

The branch metrics is the cost of traversing along a specific branch, as indicated in Figure 1. In additive white Gaussian noise (AWGN), they represent the squared difference between the received sample r , and corresponding equalization target value t_k :

$$bm_k = (r - t_k)^2.$$

State metrics, or path metrics, accumulate the minimum cost of 'arriving' into a specific state. The algorithm states that the states are updated performing the add-compare-select recursion. The two branch metrics are added to state metrics from the previous time instant, and the smaller one is selected to be the new state metric for each state, as illustrated in Figure 2

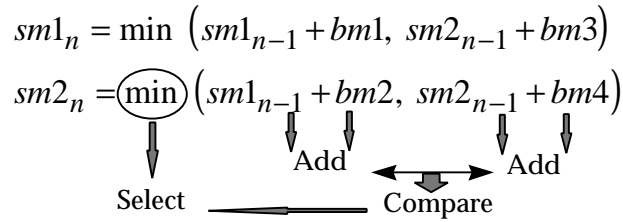


Figure 2: Add-compare-select recursion in the algorithm.

Finally taken path through the trellis represents a survivor sequence, or the most likely sequence of transmitted data.

The illustration of Viterbi algorithm using Java Applets can be seen in: <http://www.alantro.com/viterbi/viterbi.htm>

1.2. Implementation of Viterbi Decoder

The implementation of the Viterbi decoder, a processor that implements the Viterbi algorithm, consists of three major blocks: branch metrics calculation unit (BMU), add-compare-select unit (ACS), and survivor path decoding unit.

Branch metrics unit:

It performs the calculation of distances of sampled signals from targets, which are Euclidean in case of AWGN:

$$bm_{k,i} = (r_i - t_k)^2$$

(or Hamming in case of binary symmetric channel).

The k new branch metrics are computed for each incoming sample r_i , at every clock cycle.

Add-Compare-Select:

The new value of state metrics has to be computed in each time instant. In other words, the state metrics have to be updated in every clock cycle. Because of this fundamental recursion, a common approach to increasing the throughput of the system by pipelining is not applicable. This part results in largest power consumption and largest area.

In order to keep the required precision, it is necessary to keep 7 bits that are updated by incoming 5 bits of branch metrics. Since only the state metrics are a positive numbers and only positive branch metrics are added to it, the accumulated metrics would grow indefinitely without normalization. In this project we have chosen to implement modulo normalization, which requires keeping additional bit (8 instead of 7). The operation of ACS unit is shown in Figure 3. The new branch metrics are added to previous state metrics to form the candidates for the new state metrics. The comparison can be done by using the subtraction of the two candidate state metrics, and the MSB of the difference points to a larger one of two.

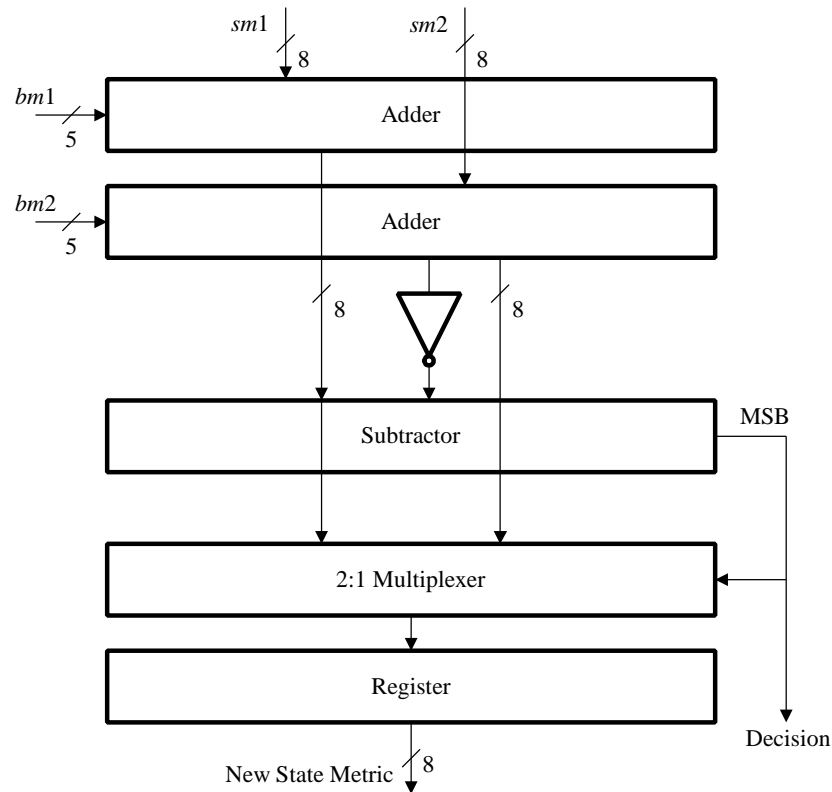


Figure 3: Block diagram of ACS unit.

Survivor sequence detection

In order to decode the input sequence, the survivor path, or shortest path through the trellis must be traced. The selected minimum metric path from the ACS output points the path from each state to its predecessor. In theory, decoding of the shortest path would require the processing of the entire input sequence. In practice the survivor paths merge

after some number of iterations, as shown in bold lines in 4-state example in Figure 4. From the point they have merged together, the decoding is unique. The trellis depth at which all the survivor paths merge with high probability is referred as the survivor path length.

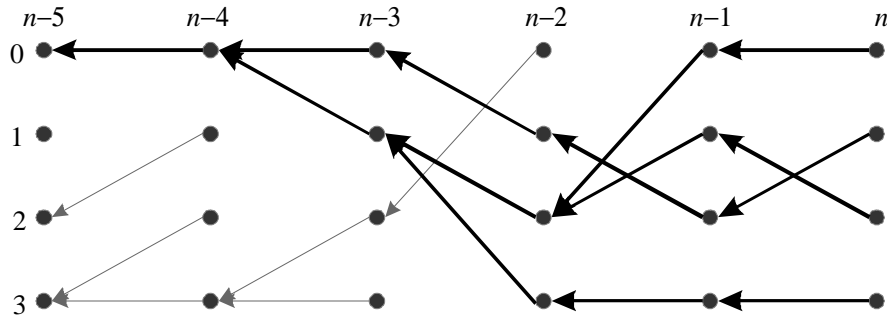


Figure 4: Survivor sequence detection.

2. Implementation and Constraints

The goal is to design an ACS unit to be used in the Viterbi decoder one of the three cases. The project will be done in THREE phases.

PHASE 1 GOALS:

The goal of the first phase is to perform the logic optimization, circuit style selection and first-order COMBINATIONAL circuit optimization to meet the stated design goals and constraints.

The fine-tuning of the design and the actual physical layout of the ACS will be performed in phase 2.

You should select one of the following design CASES:

- Low data throughput: Design a single ACS such that the average energy is minimized while still meeting the constraint that the worst-case delay is smaller than 50ns! No constraints are put on the area.**
- High data throughput: Maximize the single ACS operating speed. No constraints are put on area or power.**
- Low area decoder: Minimize the area of a single ACS, while meeting the constraint that the worst-case delay is smaller than 50ns! No constraints are put on energy.**

The project is to be done in pairs. You should sign up in teams of two students and choose design goal a) b) or c) by Tuesday, October 12 1999.

You are free to choose any logic family for the implementation of the project: complementary CMOS, pseudo-NMOS, pass-transistor logic, dynamic logic, etc.

TECHNOLOGY: The design is to be implemented in a 0.25 μm CMOS process with 4 metal layers. The SPICE technology is in the g25.mod file.

POWER SUPPLY: You are free to choose any supply voltage and logic swing up to 2.5V. Make sure that you use the appropriate model when you perform hand analysis.

PERFORMANCE METRIC: The propagation delays for static designs is defined as the time interval between the 50% transition point of the inputs and the 50% point of the worst-case output signal. Make sure you pick the worst-case condition and state **EXPLICITLY** in your report what that condition is. Note that for dynamic designs the propagation delay is defined in this case as the delay of the evaluate phase **ONLY** (at least in this phase of the process)!

AREA: The area is defined as **the smallest rectangular box** that can be drawn around the design.

NAMING CONVENTIONS: You should label the inputs and the outputs of the design as it is shown in Figure 3. The least significant bits of state metrics should be labeled as sm1[0] and sm2[0], and the most significant bits should be labeled as sm1[7] and sm2[7]. The least significant bits of branch metrics should be labeled as bm1[0] and bm2[0], and the most significant bits should be labeled as bm1[4] and bm2[4]. The newly computed state metric should be labeled as nsm[0]-nsm[7].

REGISTERS: In the first phase you don't need to design the registers. This will be a part of later phase of the project.

VOH, VOL, NOISE MARGINS: You are free to choose your logic swing. The noise margins should be at least 10% of the voltage swing. Test this by computing the VTC between one of the inputs and the output signals (with the other outputs set to the appropriate values) for a static design. For a dynamic circuit, apply an input signal with a 10% noise value added to the input and observe the outputs.

RISE AND FALL TIMES: All input signals and clocks have rise and fall times of 500ps. The rise and fall times of the output signals (10% to 90%) **should not exceed 1ns**.

LOAD CAPACITANCE: Each output bit of the ACS unit stage should have a **50 fF** load.

3. Simulation

Analyze the circuit by using either SPICE or IRSIM to simulate the design. Prove the functional operation of your circuit using either SPICE or IRSIM.

4. Report

The quality of your report is as important as the quality of your design. One must sell the design by justifying the design decisions and providing all the vital information, while eliminating the unnecessary materials. **Organization, conciseness, and completeness are of paramount importance.** Use the templates provided on the web-page (in Frame-maker, word, and pdf formats). Electronic submission of the reports is encouraged! If filing electronically, mail your report as a postscript or pdf file to icdesign@zabriskie.eecs.berkeley.edu. In case you do not have the means to create an electronic report, print out the template and deposit a paper copy of the report in the box in 558 Cory. **Make sure to fill in the cover-page; and to use the correct units.** A report has to be submitted at the end of each phase of the project.

Report 1:

Your should discuss your overall design philosophy and the important design decisions you made at the logic and circuit level. Discuss why your approach increases the operating speed or helps to reduce energy or area, while meeting the performance specs.

Provide your current estimates of the results and describe how you got them. Include schematics and highlight the important elements.

Prove that your alleged results are TRUE by providing the crucial plots (don't forget to mention the input patterns you used to obtain those plots). **The total report should not contain more than three pages.** You are **not** allowed to add any other sheets, except for important plots. It should be based on the following outlay:

Page 1: Executive summary, overall design decisions, remarks and motivations

Page 2: Logic and transistor diagram - annotated with transistor sizes and worst-case timing path. Plot showing the functional operation of the cell. Comments.

Page 3: Timing and energy simulations - derive value of worst-case path and average energy. For the latter, a set of test patterns will be provided on the web page.

Also, you are required to send by e-mail the SPICE INPUT DECK you used to analyze the energy to icdesign@zabriskie.eecs.berkeley.edu

Remember, a good report is like a good layout: it should perform its function (convey information) **in the smallest possible area with the least delay and energy (to the reader) possible.**

The quality of the report is an important (major) part of the grade!

Grade decomposition:

Part 1 will be responsible for 40% of the total grade on the project (20% for part 2 and 40% for part 3). The grade will be divided over the result (30%), approach and correctness (20%), report (40%) and creativity (10%).

Phase 1 Due-date: Friday, October 22 by 5pm.

NO EXTENSIONS ACCEPTED!