

ESE535: Electronic Design Automation

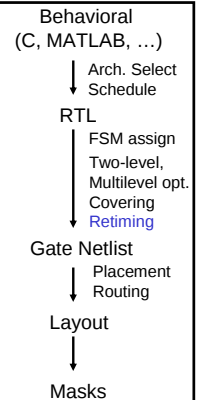
Day 18: April 1, 2009
Retiming



Penn ESE535 Spring 2009 -- DeHon

Today

- Retiming
 - Cycle time (clock period)
 - C-slow
 - Initial states
 - Register minimization



Penn ESE535 Spring 2009 -- DeHon

2

Task

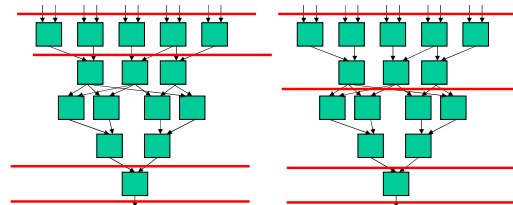
- Move registers to:
 - Preserve semantics
 - Minimize path length between registers
 - Reduce cycle time
 - ...while minimizing number of registers required

Penn ESE535 Spring 2009 -- DeHon

3

Example: Same Semantics

- Externally: no observable difference



Penn ESE535 Spring 2009 -- DeHon

4

Problem

- **Given:** clocked circuit
- **Goal:** minimize clock period without changing (observable) behavior
- *i.e.* minimize maximum delay between any pair of registers
- **Freedom:** move placement of internal registers

Penn ESE535 Spring 2009 -- DeHon

5

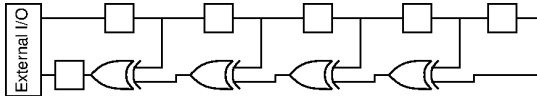
Other Goals

- Minimize number of registers in circuit
- Achieve target cycle time
- Minimize number of registers while achieving target cycle time
- ...start talking about minimizing cycle...

Penn ESE535 Spring 2009 -- DeHon

6

Simple Example



Path Length (L) = 4

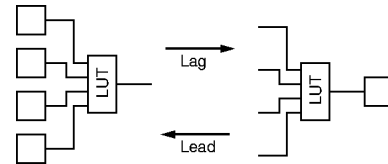
Can we do better?

Penn ESE535 Spring 2009 -- DeHon

7

Legal Register Moves

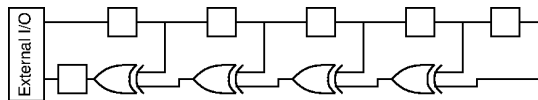
- Retiming Lag/Lead



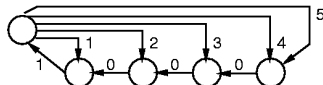
Penn ESE535 Spring 2009 -- DeHon

8

Canonical Graph Representation



Observable I/O



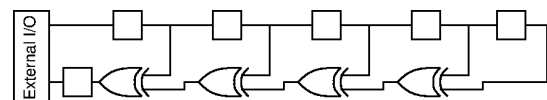
Separate arc for each path

Weight edges by number of registers
(weight nodes by delay through node)

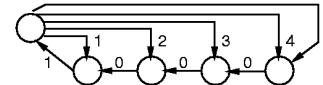
Penn ESE535 Spring 2009 -- DeHon

9

Critical Path Length



Observable I/O



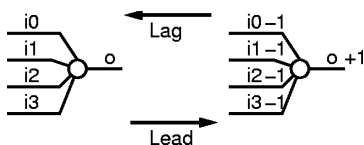
Critical Path: Length of longest path of zero weight nodes

Compute in $O(|E|)$ time by levelizing network:

Topological sort, push path lengths forward until find register.

Penn ESE535 Spring 2009 -- DeHon

Retiming Lag/Lead



Retiming: Assign a lag to every vertex

$$\text{weight}(e') = \text{weight}(e) + \text{lag}(\text{head}(e)) - \text{lag}(\text{tail}(e))$$

Penn ESE535 Spring 2009 -- DeHon

11

Valid Retiming

- Retiming is valid as long as:
 - $\forall e$ in graph
 - $\text{weight}(e') = \text{weight}(e) + \text{lag}(\text{head}(e)) - \text{lag}(\text{tail}(e)) \geq 0$
- Assuming original circuit was a valid synchronous circuit, this guarantees:
 - non-negative register weights on all edges
 - no travel backward in time :-)
 - all cycles have strictly positive register counts
 - propagation delay on each vertex is non-negative (assumed 1 for today)

Penn ESE535 Spring 2009 -- DeHon

12

Retiming Task

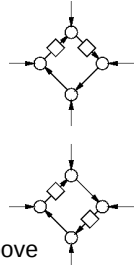
- Move registers $\equiv$ assign lags to nodes
 - lags define all locally legal moves
- Preserving non-negative edge weights
 - (previous slide)
 - guarantees collection of lags remains consistent globally

Penn ESE535 Spring 2009 -- DeHon

13

Retiming Transformation

- Properties invariant to retiming
 1. number of registers around a cycle
 2. delay along a cycle
- Cycle of length P must have
 - at least P/c registers on it to be retimeable to cycle c
 - Can be computed from invariant above



Penn ESE535 Spring 2009 -- DeHon

14

Optimal Retiming

- There is a retiming of
 - graph G
 - w/ clock cycle c
 - iff $G-1/c$ has no cycles with negative edge weights
- $G-\alpha \equiv$ subtract α from each edge weight

Penn ESE535 Spring 2009 -- DeHon

15

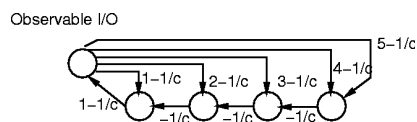
$1/c$ Intuition

- Want to place a register every c delay units
- Each register adds one
- Each delay subtracts $1/c$
- As long as remains more positives than negatives around all cycles
 - can move registers to accommodate
 - Captures the $\text{regs}=P/c$ constraints

Penn ESE535 Spring 2009 -- DeHon

16

$G-1/c$



Penn ESE535 Spring 2009 -- DeHon

17

Compute Retiming

- $\text{Lag}(v) =$ shortest path to I/O in $G-1/c$
- Compute shortest paths in $O(|V||E|)$
 - Bellman-Ford
 - also use to detect negative weight cycles when c too small

Penn ESE535 Spring 2009 -- DeHon

18

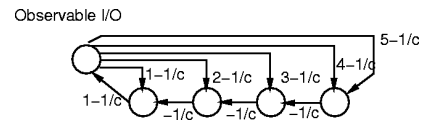
Bellman Ford

- For $l \leftarrow 0$ to N
 - $u_i \leftarrow \infty$ (except $u_i=0$ for IO)
- For $k \leftarrow 0$ to N
 - for $e_{ij} \in E$
 - $u_i \leftarrow \min(u_i, u_j + w(e_{ij}))$
- For $e_{ij} \in E$ *//still update $\rightarrow$ negative cycle*
 - if $u_i > u_j + w(e_{ij})$
 - cycles detected

Penn ESE535 Spring 2009 -- DeHon

19

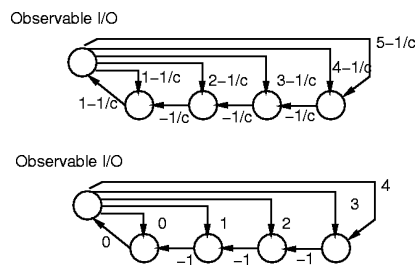
Apply to Example



Penn ESE535 Spring 2009 -- DeHon

20

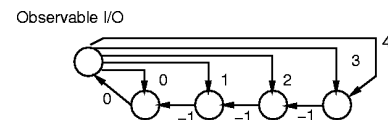
Try $c=1$



Penn ESE535 Spring 2009 -- DeHon

21

Apply: Find Lags

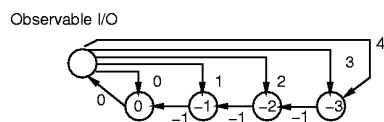


Negative weight cycles?
Shortest paths?

Penn ESE535 Spring 2009 -- DeHon

22

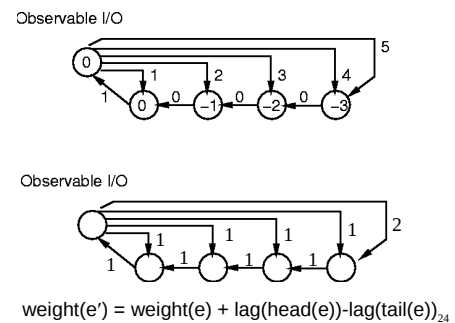
Apply: Lags



Penn ESE535 Spring 2009 -- DeHon

23

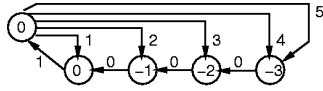
Apply: Move Registers



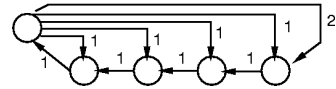
Penn ESE535 Spring 2009 -- DeHon

Apply: Retimed

Observable I/O



Observable I/O

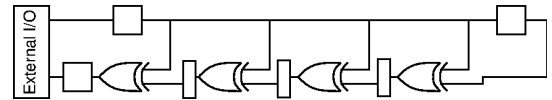
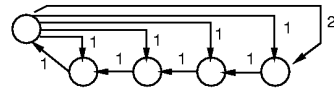


Penn ESE535 Spring 2009 -- DeHon

25

Apply: Retimed Design

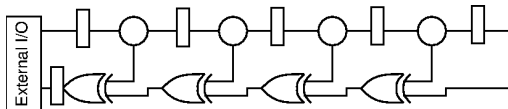
Observable I/O



Penn ESE535 Spring 2009 -- DeHon

26

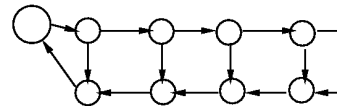
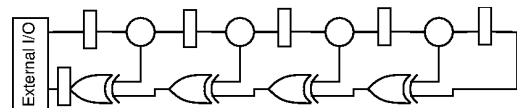
Revise Example (fanout delay)



Penn ESE535 Spring 2009 -- DeHon

27

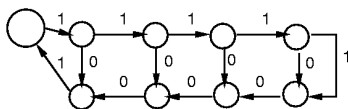
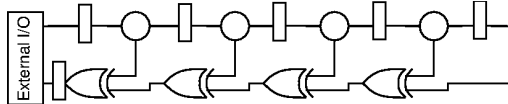
Revised: Graph



Penn ESE535 Spring 2009 -- DeHon

28

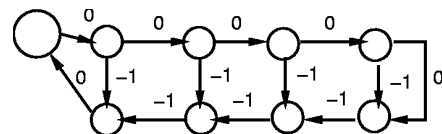
Revised: Graph



Penn ESE535 Spring 2009 -- DeHon

29

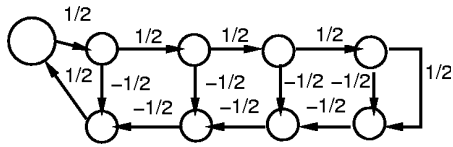
Revised: C=1?



Penn ESE535 Spring 2009 -- DeHon

30

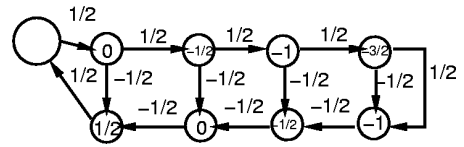
Revised: C=2?



Penn ESE535 Spring 2009 -- DeHon

31

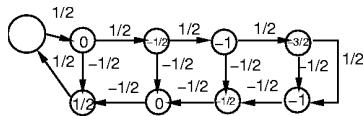
Revised: Lag



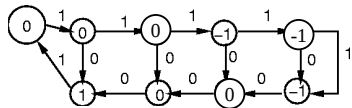
Penn ESE535 Spring 2009 -- DeHon

32

Revised: Lag



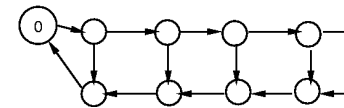
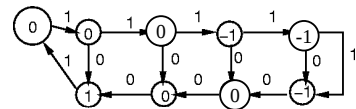
Take ceiling to convert to integer lags:



Penn ESE535 Spring 2009 -- DeHon

33

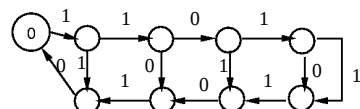
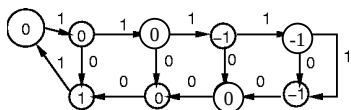
Revised: Apply Lag



Penn ESE535 Spring 2009 -- DeHon

34

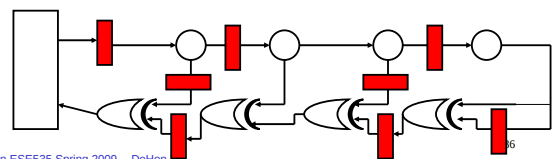
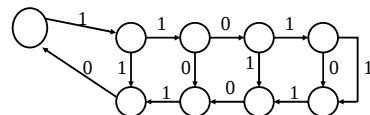
Revised: Apply Lag



Penn ESE535 Spring 2009 -- DeHon

35

Revised: Retimed



Penn ESE535 Spring 2009 -- DeHon

36

- We can use this retiming to pipeline
- Assume we have enough (infinite supply) registers at edge of circuit
- Retime them into circuit

Penn ESE535 Spring 2009 -- DeHon

Penn ESE535 Spring 2009 -- DeHon

Penn ESE535 Spring 2009 -- DeHon

The figure consists of three parts illustrating the construction of a linear feedback shift register (LFSR) for a 4-bit register.

- Top Diagram:** Shows a sequence of 4 bits in a register, labeled "n-regs". An "External I/O" block is connected to the first bit. The register is represented by a horizontal line with four circles (bits) and four rectangles (shifts).
- Middle Diagram (G):** Shows the first step of the construction. The register now has 5 bits. The first bit is connected to the last bit (feedback). The feedback is labeled "1". The shift operation is labeled "0".
- Bottom Diagram (G-1/1):** Shows the second step of the construction. The register now has 6 bits. The first bit is connected to the last bit (feedback) and the second bit (feedback). The feedbacks are labeled "0" and "1". The shift operation is labeled "0".

Penn ESE535 Spring 2009 -- DeHon

$n=5$

```
graph LR; S(( )) -- 4 --> N1(-4); N1 -- 0 --> N2(-4); N2 -- 0 --> N3(-4); N3 -- 0 --> N4(-4); N4 -- 0 --> N5(0); N5 -- 0 --> N1; N1 -- -1 --> N5; N2 -- -1 --> N5; N3 -- -1 --> N5; N4 -- -1 --> N5; N5 -- -1 --> N4;
```

Penn ESE535 Spring 2009 -- DeHon

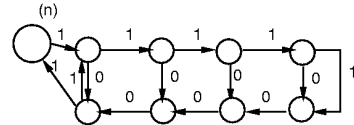
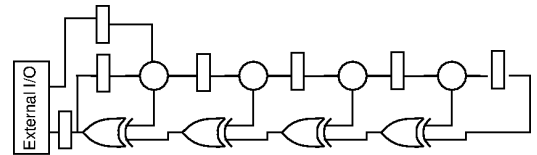
Penn ESE535 Spring 2009 -- DeHon

Real Cycle

The diagram illustrates a single clock cycle for a digital circuit. It features a horizontal timeline with several events marked by vertical lines. The components and their connections are as follows:

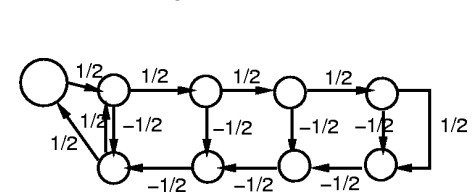
- External I/O:** A block on the left with three output lines. The top line connects to the top input of the first AND gate. The middle line connects to the top input of the first OR gate. The bottom line connects to the bottom input of the first AND gate.
- AND Gates:** There are four AND gates represented by circles with a dot. Each has two inputs and one output. The first AND gate's inputs are from the External I/O. Its output goes to the top input of the first OR gate. The second AND gate's inputs are from the top and bottom outputs of the first OR gate. Its output goes to the top input of the second OR gate. This pattern repeats for the third and fourth AND gates.
- OR Gates:** There are four OR gates represented by circles with a horizontal line. Each has two inputs and one output. The first OR gate's inputs are from the first AND gate and the External I/O. Its output goes to the top input of the second AND gate. The second OR gate's inputs are from the second AND gate and the External I/O. Its output goes to the top input of the third AND gate. This pattern repeats for the third and fourth OR gates.
- Flip-Flops:** There are four flip-flops represented by rectangles. Each has a clock input (triangle), a data input, and a data output. The clock inputs are connected to a common clock line. The data inputs are connected to the outputs of the OR gates. The data outputs are connected to the top inputs of the AND gates, forming a feedback loop.

The diagram shows the state of these components at various points in time, with the clock signal being a periodic square wave.



Cycle C=1?

The diagram illustrates a state transition graph. It begins with a node labeled $(n-1)$. An arrow labeled 0 leads from $(n-1)$ to the first node of a 2×4 grid. Within this grid, horizontal transitions are labeled 0 and vertical transitions are labeled -1 . A final arrow labeled 0 connects the last node in the bottom row back to the first node in the top row, forming a cycle.

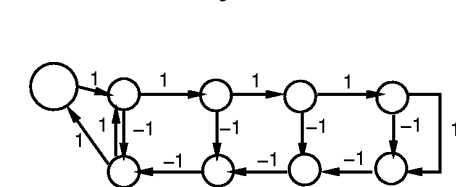


Cycle: C-slow

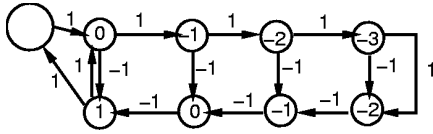
Cycle = $c \Rightarrow$ C-slow network has Cycle = 1

Penn

Cycle=c $\Rightarrow$ C-slow network has Cycle=1



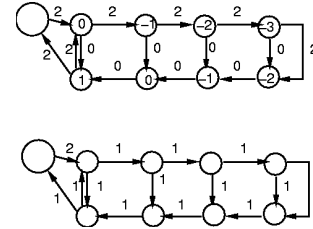
2-Slow Lags



Penn ESE535 Spring 2009 -- DeHon

49

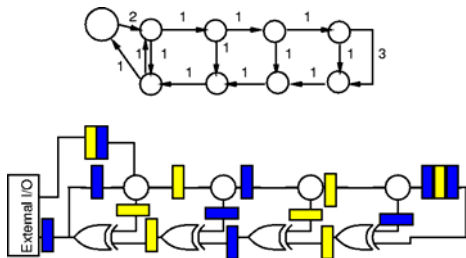
2-Slow Retime



Penn ESE535 Spring 2009 -- DeHon

50

Retimed 2-Slow Cycle



Penn ESE535 Spring 2009 -- DeHon

51

C-Slow applicable?

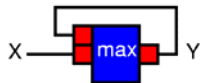
- Available parallelism
 - solve C identical, independent problems
 - e.g. process packets (blocks) separately
 - e.g. independent regions in images
- Commutative operators
 - e.g. max example

Penn ESE535 Spring 2009 -- DeHon

52

Max Example

2-Slow design:



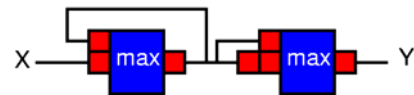
X2 X2 X1 X1 X0 X0 → Y2 ? Y1 ? Y0 ?

B2 A2 B1 A1 B0 A0 → YA2 YB1 YA1 YB0 YA0 ?

Penn ESE535 Spring 2009 -- DeHon

53

Max Example



Computes two interleaved streams: even max, odd max

Computes final max of even and odd pairs

Penn ESE535 Spring 2009 -- DeHon

54

Note

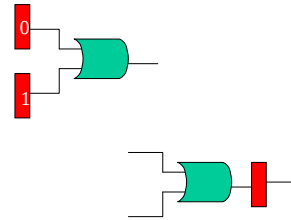
- Algorithm/examples shown
 - for special case of unit-delay nodes
- For general delay,
 - a bit more complicated
 - still polynomial

Penn ESE535 Spring 2009 -- DeHon

55

Initial State

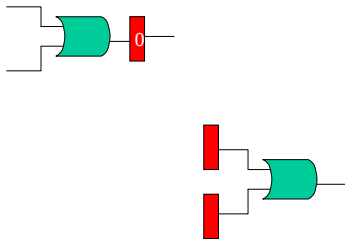
- What about initial state?



Penn ESE535 Spring 2009 -- DeHon

56

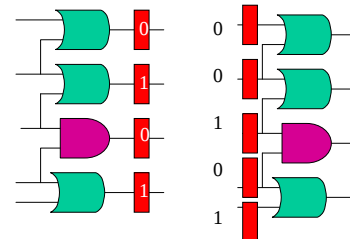
Initial State



Penn ESE535 Spring 2009 -- DeHon

57

Initial State

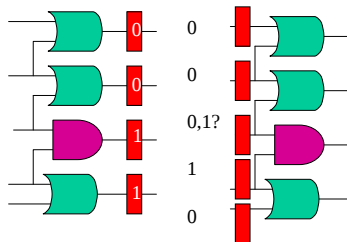


In general, constraints $\rightarrow$ satisfiable?

Penn ESE535 Spring 2009 -- DeHon

58

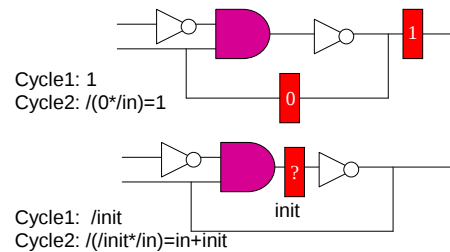
Initial State



Penn ESE535 Spring 2009 -- DeHon

59

Initial State



Cycle1: 1
Cycle2: $!(0*in)=1$

Cycle1: $/init$
Cycle2: $!(/init*in)=in+init$

init=0

Cycle1: 1

Cycle2: $!(/init*in)=in$

init=1

Cycle1: 0

Cycle2: $!(/init*in)=1$

Penn ESE535 Spring 2009 -- DeHon

60

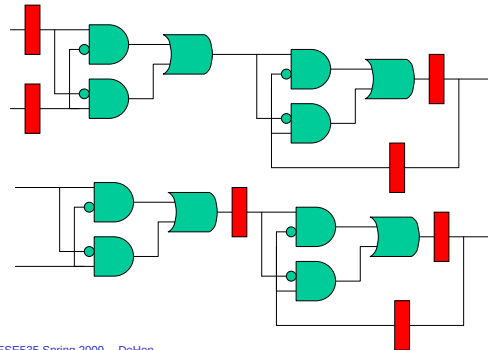
Initial State

- Cannot always get exactly the same initial state behavior on the retimed circuit
 - without additional care in the retiming transformation
 - sometimes have to modify structure of retiming to preserve initial behavior
- Only a problem for startup transient
 - if you're willing to clock to get into initial state, not a limitation

Penn ESE535 Spring 2009 -- DeHon

61

Minimize Registers



Penn ESE535 Spring 2009 -- DeHon

62

Minimize Registers

- Number of registers: $\sum w(e)$
- After retime: $\sum w(e) + \sum (FI(v) - FO(v)) \text{lag}(v)$
- delta only in lags
- So want to minimize: $\sum (FI(v) - FO(v)) \text{lag}(v)$
 - subject to earlier constraints
 - non-negative register weights, delays
 - positive cycle counts

Penn ESE535 Spring 2009 -- DeHon

63

Minimize Registers

- Can be formulated as flow problem
- Can add cycle time constraints to flow problem
- Time: $O(|V||E|\log(|V|)\log(|V|^2/|E|))$

Penn ESE535 Spring 2009 -- DeHon

64

Summary

- Can move registers to minimize cycle time
- Formulate as a lag assignment to every node
- Optimally solve cycle time in $O(|V||E|)$ time
- Also
 - Compute multithreaded computations
 - Minimize registers
- Watch out for initial values

Penn ESE535 Spring 2009 -- DeHon

65

Admin

- No Class Monday
- Assignment 4 due Wednesday
- Reading for Wednesday online

Penn ESE535 Spring 2009 -- DeHon

66

Big Ideas

- Exploit freedom
- Formulate transformations (lag assignment)
- Express legality constraints
- Technique:
 - graph algorithms
 - network flow