8-bit Parallel Prefix Adders Using Brent Kung Tree with BIST

Cory Merkel David Brenner

cem1103@rit.edu dab2714@rit.edu

EECC 730

Instructor: Dr. Kudithipudi

Lab Assistant: Dimitri Yudanov

Layout Path: /shared/dabcem/project/PPA8BIST

Table of Contents

Abstract

Binary adders are on of the most important components in many modern digital circuits, especially processors. To further investigate the design and implementation of efficient arithmetic circuitry, one such adder was created and at both the schematic and transistor levels using Mentor Graphics software. A Built-In-Self-Test implementation was also created to automatically test the functionality of the adder. Using TSMC 0.35 micron N-Well process technology, the final adder layout had an area of 28,757 square microns, with a maximum input frequency of 682 MHz.

Design Methodology

8-bit Brent-Kung Adder (Static CMOS)

An adder is one of the basic building blocks of common datapath components. As such, they are of great importance to designers being so commonly used and such a critical part of the data path. For smaller adders, carry-lookahead, carry-skip, or carry-select will suffice, but as the width of the adder grows, the delay of passing the carry through the stages begins to dominate. Parallel prefix adders address this by constructing a multilevel tree of lookahead blocks that grow with log N.

The Brent-Kung adder is a parallel prefix adder that requires $2(\log_2 N)-1$ stages. It was originally proposed as a simple and regular design of a parallel adder that addresses the problems of connecting gates in a way to minimize chip area. Accordingly, it is considered one of the better tree adders for minimizing wiring tracks, fanout, and gate count and is used as a basis for many other networks. The tree diagram for the computation of all carries for a 16-bit adder, as was shown in the original paper, is given below.

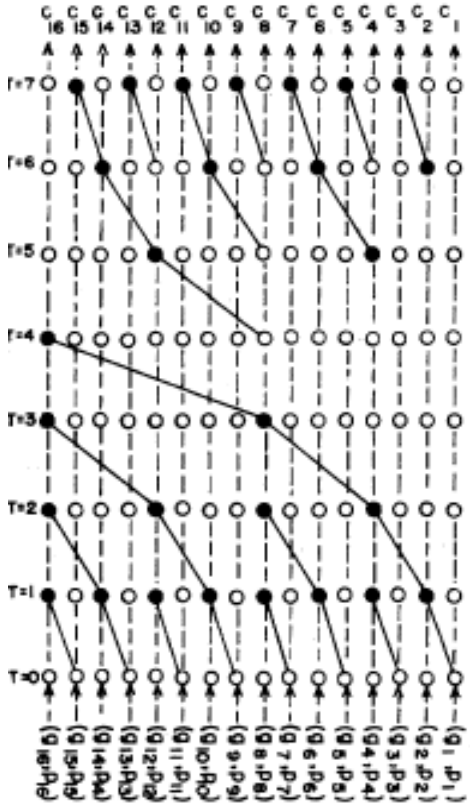

Figure 1 -- Brent-Kung Adder Tree Diagram

The operation of parallel prefix adders depends on the notion of group carry propagate and generate signals. In essence, a group of bits generate a carry when its carry is true independent of the carry-in and propagate a carry when its carry-out is true when there is a carry-in. The generate and propagate signals can be defined as follows:

$$G_{i:j} = G_{i:k} + P_{i:k} * G_{k-1:j} \quad (1)$$

$$P_{i:j} = P_{i:k} * P_{k-1:j} \quad (2)$$

with a base case for the least significant bits defined as:

$$G_{i:i} = G_i = A_i * B_i \quad (3)$$

$$P_{i:i} = P_i = A_i \oplus B_i \quad (4)$$

The sum can therefore be computes using the following equation:

$$S_i = P_i \oplus G_{i-1:0} \quad (5)$$

The 8-bit Brent-Kung adder has a tree diagram as appears below. Each dot represents the “dot operator” for the group generate and propagate signals as defined in the equations above.

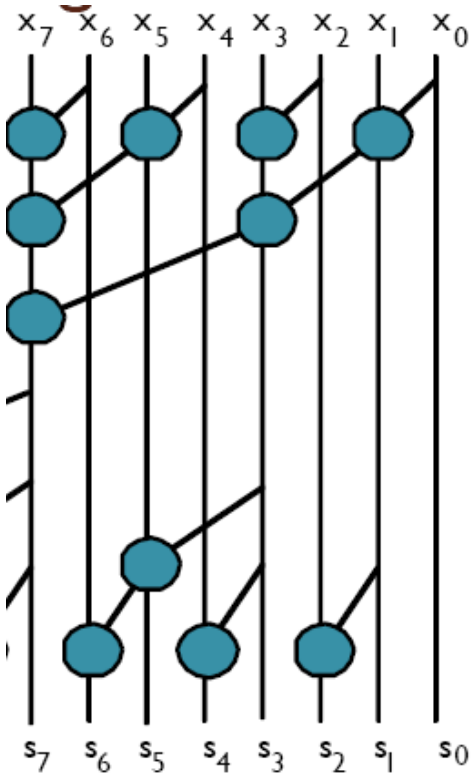

Figure 2 -- 8-bit Brent-Kung Adder Tree Diagram

The tree is not actually complete, however, as it ignores the carry-in bit. An additional generate operation is necessary to account for the carry-in. Once the design of the adder itself was fully understood and realized, the individual gates that make up the design were implemented. The adder consists of the following gates: AND, XOR, and AND-OR. The XOR gate was implemented using the “tiny-xor” design. The AND and AND-OR gates were implemented using static CMOS.

D Flip-Flop Implementation Using Pass Transistors

Conventional D Flip-Flops are implemented using transmission gates and clock-enabled/disabled gates. To reduce layout area, a D Flip-Flop was designed using pass transistors and tri-state inverters:

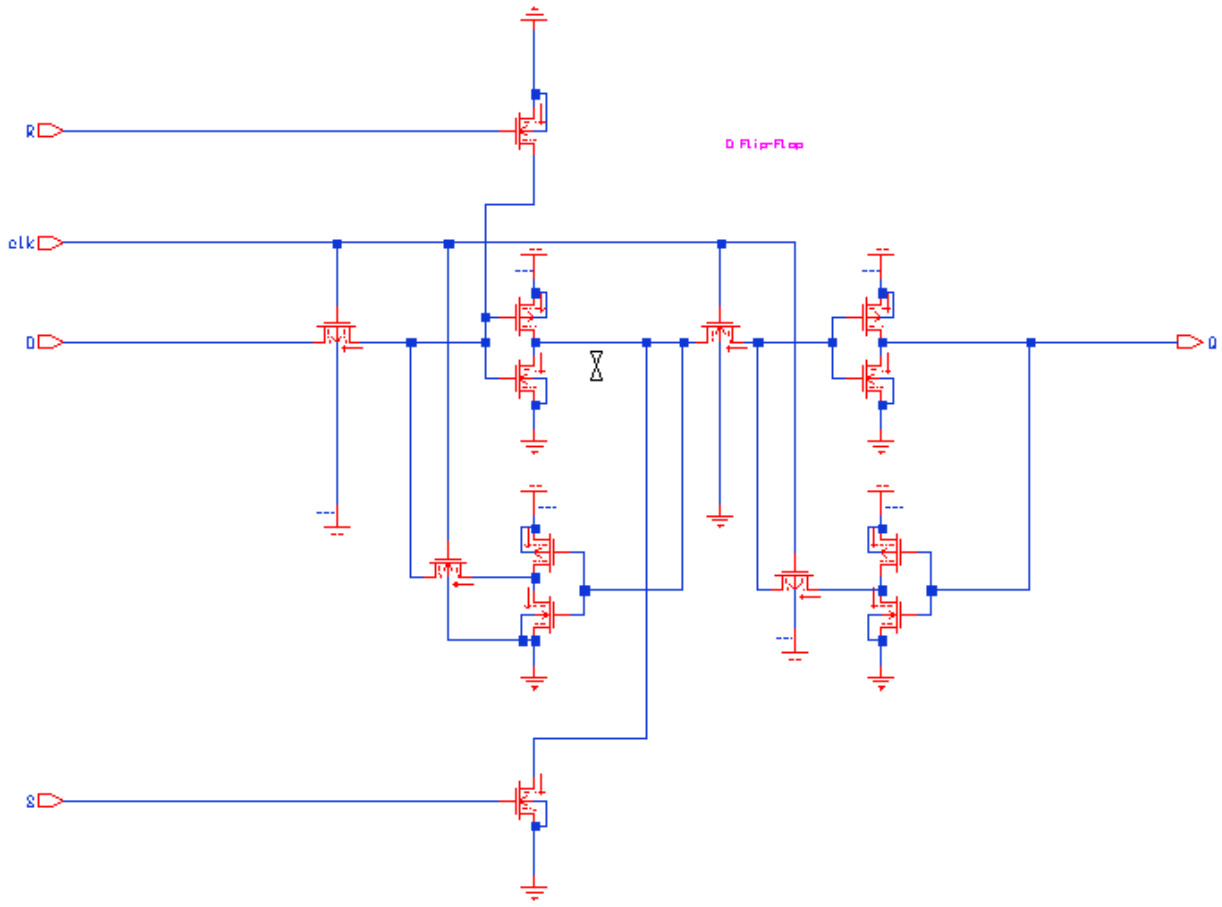

Figure 3 -- D Flip-Flop Implementation with Pass Transistors

The flip-flop is implemented with two back-to-back latches that utilize the same clock. The first latch, however, is enabled with a PMOS transistor, and the second with an NMOS transistor. By that method, the only one latch will be enabled at a time, allowing the flip-flop to operate like conventional, non-overlapping clock implementations. In general, pass transistor logic is avoided because of threshold voltage drops and noise propagation problems. In the above design, however, any threshold voltage drop is remedied with each latch's tri-state feedback inverter. The noise propagation problem can also be considered negligible as long as the tri-state inverters have reasonably large noise margins.

8-bit Built-In Logic Block Observer (Static CMOS)

Figure 1 shows the schematic of the Built-In Logic Block Observer (BILBO) used to implement Built-In Self-Test (BIST) in the 8-bit Brent Kung Adder design.

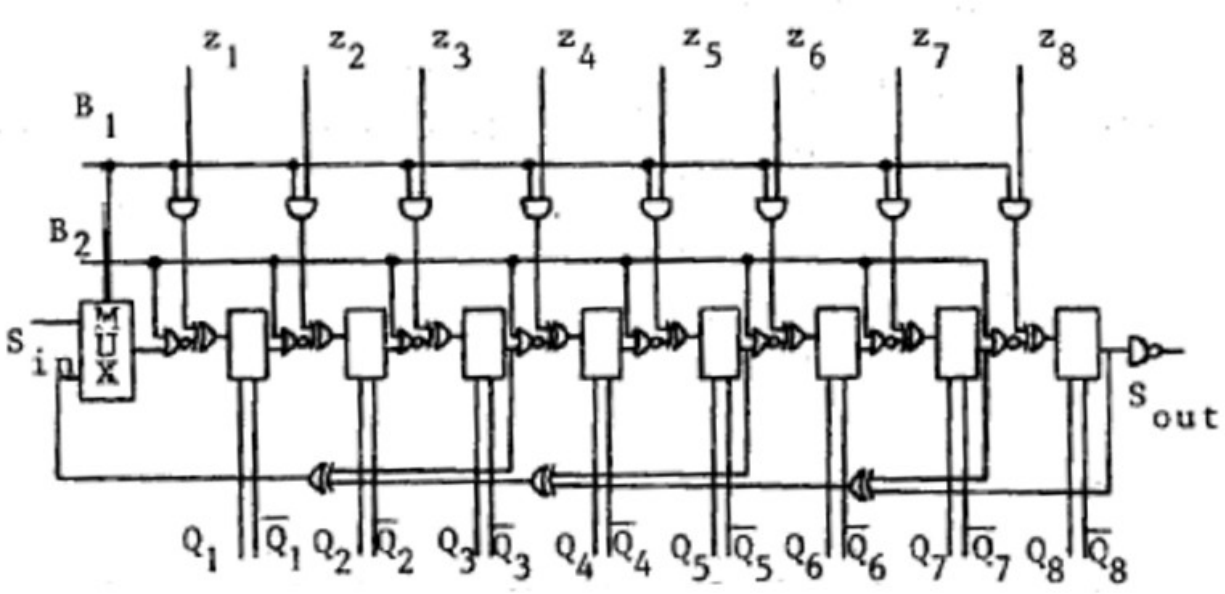

Figure 4 -- 8-bit Built-In Logic Block Observer

BILBO circuits are very useful for implementing circuits with BIST. The above circuit has four modes of operation depending on the value of B2B1:

B2	B1	Function
0	0	Scan Mode
0	1	Test Mode
1	0	Reset Mode
1	1	Normal Mode

Table 1 -- BILBO Operational Modes

Scan Mode:

When the BILBO operates in scan mode, the circuit acts like a shift register, shifting S_{in} to S_{out}. One problem with automated test circuits is that it is sometimes difficult to tell whether or not the test circuitry is working correctly. The BILBO's scan mode allows one to serially shift bits into S_{in} and verify that the circuit is functioning properly by asserting that the same values are serially read out from S_{out}. S_{in} is routed through all of the BILBO block to S_{out} in order to ensure all of the hardware is working properly.

Test Mode:

In test mode, the BILBO behaves as a linear feedback shift register with taps (in the case of **Figure 1**) at *Q8*, *Q7*, *Q4*, and *Q2*. The BILBO can perform two different functions in test mode:

1) Pseudo-random Number Generator (PRNG)

   Pseudo-random number generators provide random outputs for various applications. In the case of Built-In Logic Block Observers, pseudo-random binary sequences are used to test a circuit-under-test (CUT) with random inputs. For the BILBO in **Figure 1** to perform pseudo-random number generation the control bits must be set correctly *and* the parallel *z*inputs must all be '0'. With those conditions met, the PRNG can be given an 8-bit seed (by setting or resetting individual flip-flops), and it will produce a new value each clock cycle, repeating only after 2^8 cycles. Linear feedback shift registers are characterized by a polynomial in *x*. For the above BILBO, the characteristic polynomial is:

   $$P(x)=x^8+x^7+x^5+x^3+1 \qquad (6)$$

2) Multiple-input Signature Register (MISR)

   To test each output of a circuit for 2^ndifferent input combinations (where n is the number of inputs) would not be feasible for a large circuit. Instead, most Built-In Self-Test implementations compress the outputs of the CUT into a single n-bit word and compare that word (or signature) with a known-good signature called the golden signature. The process is called signature analysis, and it is easily performed using the MISR capabilities of a BILBO. For the BILBO to operate as an MISR, the control bits must be set correctly *and* the *z* inputs must be connected to the outputs of the CUT.

Reset Mode:

In reset mode, the each flip-flop in the BILBO is synchronously reset on the rising edge of the clock. Reset mode is used to clear the MISR BILBO (signature register) before the beginning of a test.

Normal Mode:

When the BILBO is in normal mode, it behaves as a parallel-load register, latching its *z* inputs on the rising edge of the clock cycle. Normal mode allows the CUT to behave normally.

A test of the BILBO in MISR mode is attached in the Appendices.

Results/Data Analysis

Component-Level Timing Analysis

The following table summarizes the results of timing analysis (worst-case) for each component used in the 8-bit PPA adder with Built-In Self-Test. The components were simulated with a 100 fF capacitive load at each output. (*See Appendices for schematics and simulation waveforms*).

	t_{rise} (ns)	t_{fall} (ns)	$t_{p,HL}$ (ns)	$t_{p,LH}$ (ns)
D Flip-Flop	2.2614	3.4439	2.9280	2.6165
And-Nor-XOR	2.6574	2.1558	1.9470	2.6523
2-Input MUX	1.0291	0.5314	0.2766	0.3175
XOR	1.0262	1.4320	0.9277	0.2861
AND	0.55243	0.22255	0.80084	1.0227
PG (And-Or)	1.9738	1.2472	2.2824	2.9012

Table 2 -- 8-bit PPA with B.I.S.T. Component Timing Analysis

Adder Timing Analysis

	t_{rise} (ns)	t_{fall} (ns)	$t_{p,HL}$ (ns)	$t_{p,LH}$ (ns)
8-bit Brent-Kung Adder	1.1370	0.3299	10.3140	24.1530

Using the data in the above table, the theoretical maximum input frequency for the 8-bit Brent-Kung adder is:

f[max] = 1 / (t[rise] + t[fall]) = 1 /(1.1370ns + 0.3299ns) = 681.7 MHz

The the theoretical maximum frequency of throughput is:

throughput[max] = 1/ (t[p,HL] + t[p,LH)])

= 1 / (10.3140ns + 24.1530ns) = 29MHz

Component-Level Area and Number of Transistors

The following table describes the area of each component and the number of transistors used:

	Area (λ x λ)	Area (μm²)	# NMOS Transistors	# PMOS Transistors	Total # Transistors
D Flip-Flop	75 x 103	309	8	6	14
And-Nor-XOR	75 x 88	264	7	7	14
2-Input MUX	75 x 51	153	3	3	6
XOR	75 x 51	153	3	3	6
AND	42 x 64	107.5	3	3	6
PG (And-Or)	59 x 79	186.4	4	4	8

Table -- 8-bit PPA with B.I.S.T. Component Areas and # of Transistors

BILBO, Adder, and Final Design Area and Number of Transistors

	Area (λ x λ)	Area (μm²)	# NMOS Transistors	# PMOS Transistors	Total # Transistors
8-bit BILBO	370 x 450	6660	132	116	248
8-Bit Brent-Kung Adder	318 x 483	6143.76	84	96	180
8-bit Brent-Kung Adder with BIST	745 x 965	28757	492	544	1036

Conclusions

The Brent-Kung parallel prefix adder was successfully designed and implemented in schematics and physical layout. In addition, a Built in Logic Block Observer was additionally designed for the adder in order to add a Built-in-self-test. Though there were some problems with the layout, through the use of the error messages and some patience they were overcome. The area of the adder with BIST was 28,757 μm^2. and the maximum throughput of the adder was found to be approximately 29MHz.