HLS LAB#A Presentation

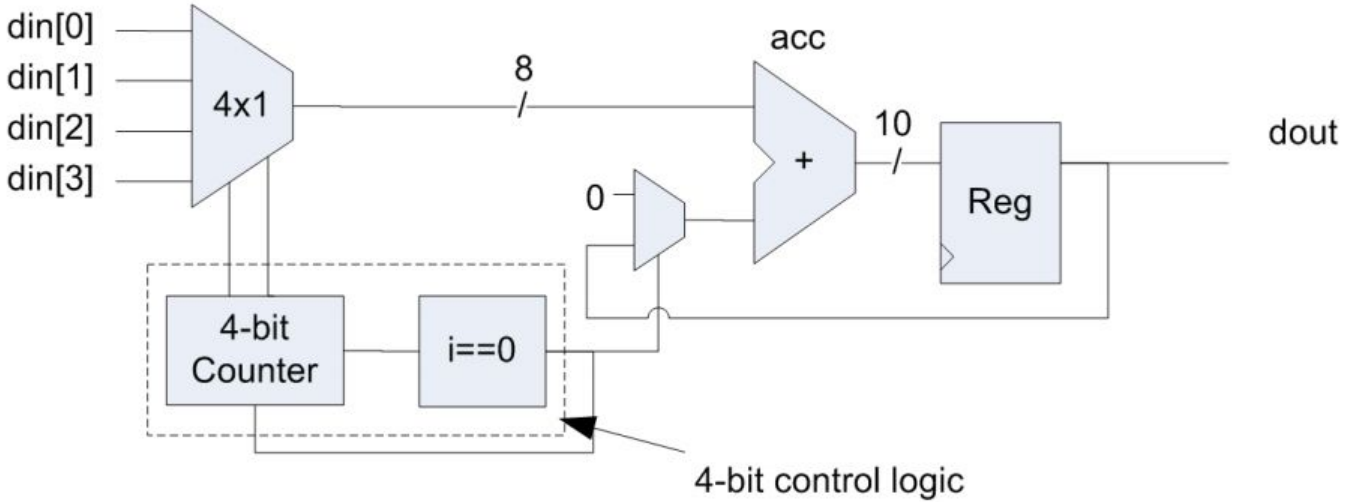
Team 8
陳昱銓 陳冠豪 楊焌佑
https://github.com/ian861226/ACA21-HLS-LAB-A-team-8

Outline

- 6.5. Accumulator
- 6.6. Shifters
- 6.7. Adder Trees
- 6.8. Lookup Tables(LUT)

Accumulator

- Hardware



Accumulator

- Design a templatized accumulator with 3 arguments
- W(width): bit width of the input
- S(sign): use signed or unsigned data type
- N(length of array)
- The number of extra bits needed to avoid overflow is log2ceil(N) bits.
  - Bit width of output

Accumulator

- Header

```
const int Width = 8;
const int Num_reg = 4;
const int log2_Num_reg = 2;

template<int W, int N, int l>
ap_int<W + l> acc_tmpl(ap_int<W> din[N]) {
    ap_int< W + l> acc;
    acc = 0;
    ACCUM: for (int i = 0; i < N; ++i)
        acc += din[i];
    return acc;
}

void accumulator(ap_int<Width> din[Num_reg], ap_int<Width + log2_Num_reg>* dout);
```

- Note: Variable used as template argument must be constant in compile time, or synthesis will be failed.

Accumulator

- Top function

```
void accumulator(ap_int<Width> din[Num_reg], ap_int<Width + log2_Num_reg>* dout) {
    *dout = acc_tmpl<Width, Num_reg, log2_Num_reg>(din);
}
```

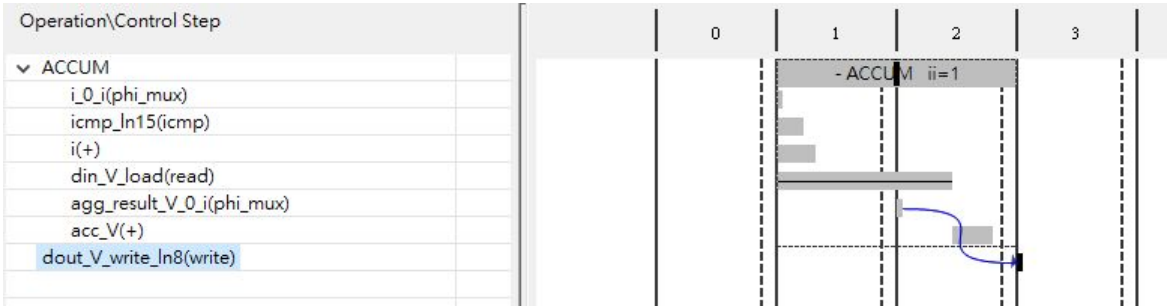
- Test bench
  - Randomly generate 1000 cases, and compare the results.

Accumulator: Utilization

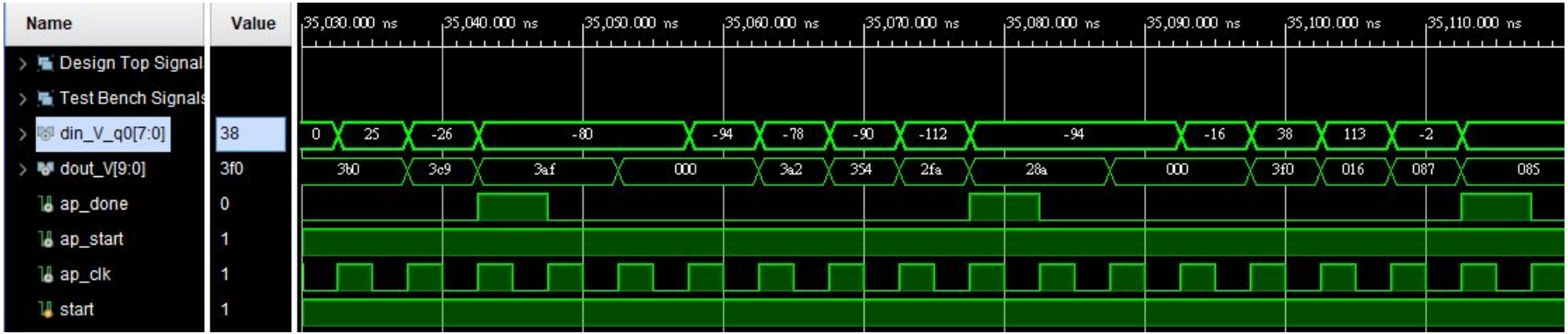
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	39	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	54	-
Register	-	-	19	-	-
Total	0	0	19	93	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0



Accumulator: Waveform



Extension of Accumulator

- To design a signed value templatized accumulator

```
template<int W, template<int W2> class dType >
class CData
{
public:
    dType<W>  Content;
};

template<int W>
class signed_ap
{
public:
    ap_int<W> val;
};

template<int W>
class unsigned_ap
{
public:
    ap_uint<W> val;
};
```

```
template<template<int W2> class T, int W, int N, int l>
CData<W + l, T> acc_tmpl(CData<W, T> din[N]) {
    CData<W + l, T> acc;
    acc.Content.val = 0;
    ACCUM: for (int i = 0; i < N; ++i)
        acc.Content.val += din[i].Content.val;
    return acc;
}
```

Extension of Accumulator

- Pass csim and csynth, but something strange in cosim…

```
template<int W, template<int W2> class dType >
class CData
{
public:
    dType<W>  Content;
};
```

In accumulator.h

```
#pragma pack()
         template< int W > struct CData;
         template<> struct CData<8, signed_ap> {
             signed_ap<8 > Content;
            } ;
```

In apatb_accumulator.cpp

```
extern void AESL_WRAP_accumulator (
struct CData<8 > din[4],
ap_int<10>* dout);
```

In apatb_accumulator.h

```
apatb_accumulator.cpp:37:47: error: wrong number of template arguments (2, should be 1)
          template<> struct CData<8, signed_ap> {
                                              ^
apatb_accumulator.cpp:36:36: note: provided for 'template<int W> struct CData'
          template< int W > struct CData;
                                   ^~~~~
```

Error message

Shifters

- Arithmetic barrel shift : use ap_int

```
ap_int<NUM_BITS> barrel_shift_al(ap_int<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    return din << s;
}

ap_int<NUM_BITS> barrel_shift_ar(ap_int<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    return din >> s;
}
```

- Logical barrel shift : use ap_uint

```
ap_uint<NUM_BITS> barrel_shift_ll(ap_uint<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    return din << s;
}

ap_uint<NUM_BITS> barrel_shift_lr(ap_uint<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    return din >> s;
}
```

Shifters

- Rotating

```
ap_uint<NUM_BITS> barrel_shift_rr(ap_uint<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    ap_uint<4> stmp =  s % NUM_BITS;
    return (din >> stmp) | (din << (NUM_BITS-stmp));
}
```

- Constant shifter

```
ap_uint<NUM_BITS> barrel_shift_lr_const(ap_uint<NUM_BITS> din, ap_uint<CTRL_BITS> s) {
    ap_uint<NUM_BITS> tmp = din;
    if(s == 1)
        tmp >>= 1;
    else if(s == 5)
        tmp >>= 5;

    return tmp;
}
```

Shifters

- Bi-directional

```
ap_uint<NUM_BITS> bi_shift(ap_uint<NUM_BITS> din, ap_int<CTRL_BITS> s) {
    return din >> s;
}
```

- Example:
  - input: 1000 1001 (137)
  - s: -1
  - output: 0001 0010 (18)

Shifters

- Dynamic bit masking: use more operation resources

```
ap_uint<NUM_BITS> shift_mask_dynamic(ap_uint<NUM_BITS> din) {
    ap_uint<NUM_BITS> acc = 0;
    LOOP: for(int i = 0; i < NUM_BITS; ++i)
        acc += (din >> i) & 1;
    return acc;
}
```

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	286	-
FIFO	-	-	-	-	-
Instance	0	-	78	88	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	103	-
Register	-	-	42	-	-
Total	0	0	120	477	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

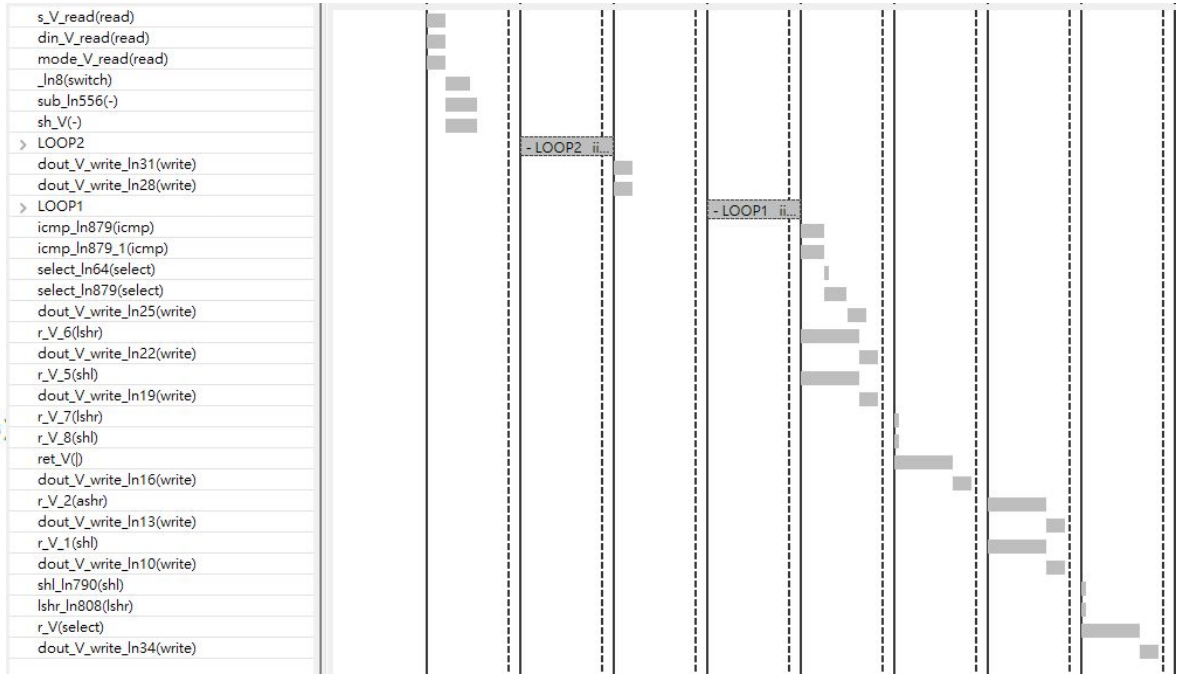
- Static bit masking: use more registers

```
ap_uint<NUM_BITS> shift_mask_static(ap_uint<NUM_BITS> din) {
    ap_uint<NUM_BITS> acc = 0;
    ap_uint<NUM_BITS> tmp = din;

    LOOP: for(int i = 0; i < NUM_BITS; ++i) {
        acc += tmp & 1;
        tmp >>= 1;
    }
    return acc;
}
```

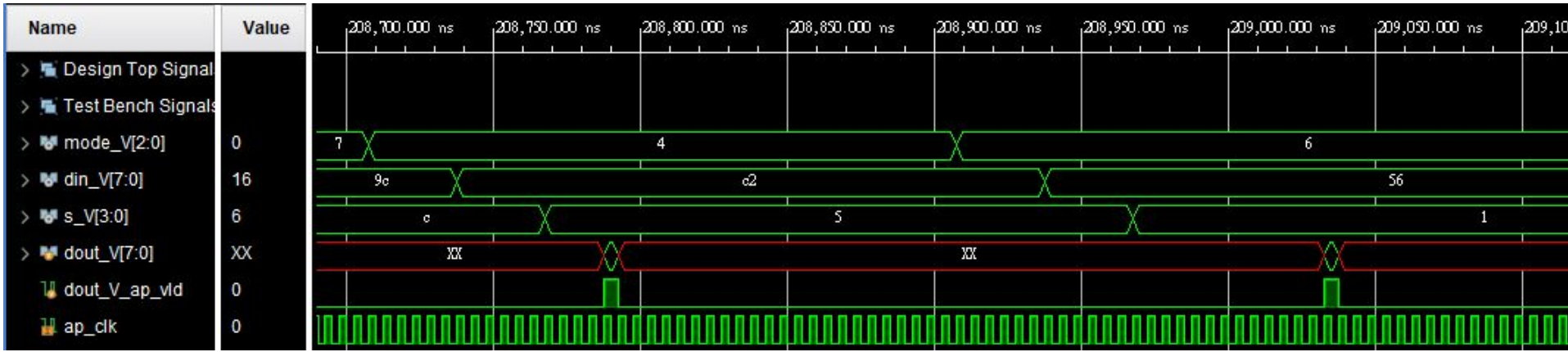
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	267	-
FIFO	-	-	-	-	-
Instance	0	-	78	88	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	112	-
Register	-	-	50	-	-
Total	0	0	128	467	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

Shifters: Top Function

```
switch(mode) {
case 0:
    *dout = barrel_shift_al(din, s);
    break;
case 1:
    *dout = barrel_shift_ar(din, s);
    break;
case 2:
    *dout = barrel_shift_rr(din, s);
    break;
case 3:
    *dout = barrel_shift_ll(din, s);
    break;
case 4:
    *dout = barrel_shift_lr(din, s);
    break;
case 5:
    *dout = barrel_shift_lr_const(din, s
    break;
case 6:
    *dout = shift_mask_dynamic(din);
    break;
case 7:
    *dout = shift_mask_static(din);
    break;
default:
    *dout = bi_shift(din, s);
    break;
}
```



Shifters: Waveform

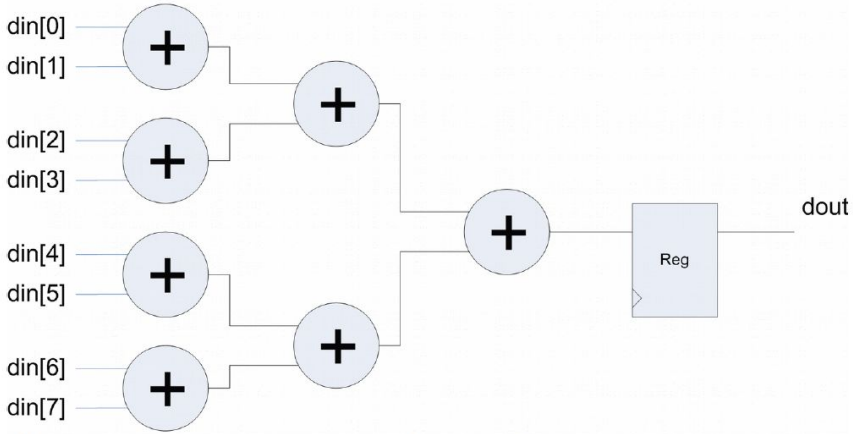


Adder Trees

- Types of adder trees
  - Automatic tree balancing
  - Preventing tree balancing
  - Forcing tree balancing

Adder Trees - automatic tree balancing

- Hardware illustration

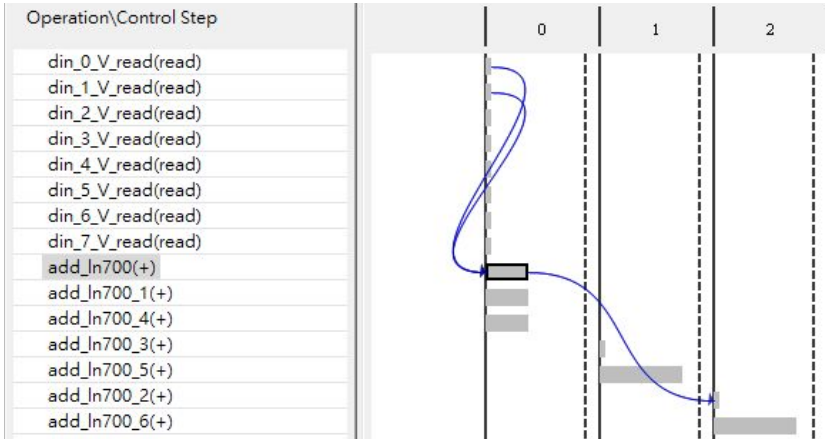


- Code

```
#include "Automatic_tree_balancing.h"

ap_int<WIDTH_OUT> adder_tree_balanced (ap_int<WIDTH> din[NUM_REGS] ) {
    #pragma HLS ARRAY_PARTITION variable=din complete dim=1
    #pragma HLS PIPELINE

    ap_int<WIDTH_OUT> acc = 0;
    adder_loop:for (int i=0; i!=NUM_REGS; i++) {
        #pragma HLS UNROLL
        acc += din[i];
    }
    return acc;
}
```

Adder Trees - automatic tree balancing

- Analysis timeline



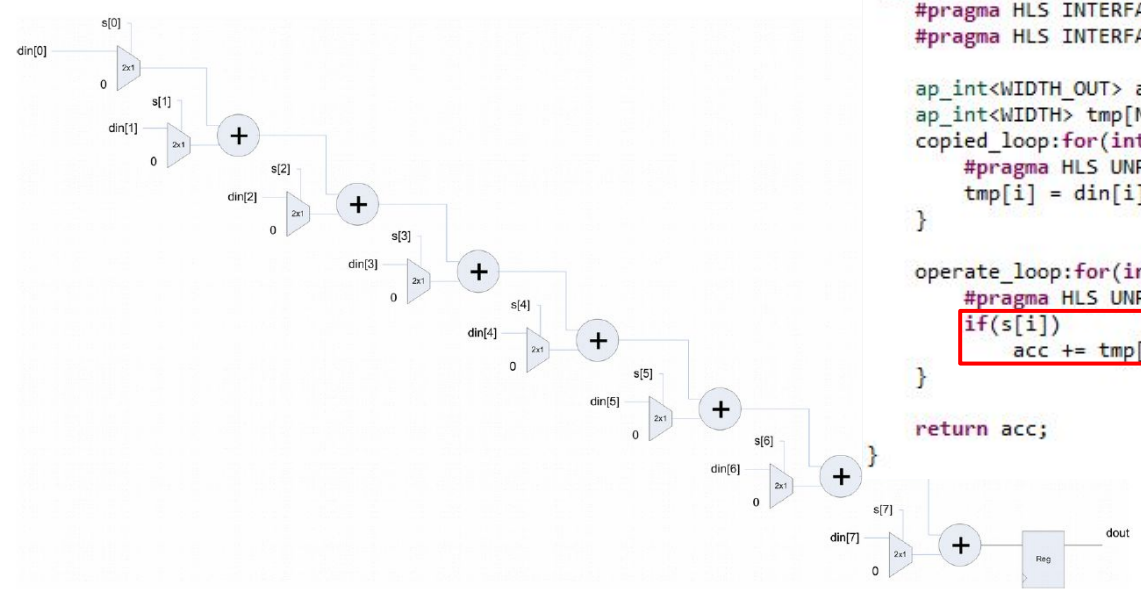
- Interface summary

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_rst	in	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_start	in	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_done	out	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_idle	out	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_ready	out	1	ap_ctrl_hs	adder_tree_balanced	return value
ap_return	out	8	ap_ctrl_hs	adder_tree_balanced	return value
din_0_V	in	8	ap_none	din_0_V	pointer
din_1_V	in	8	ap_none	din_1_V	pointer
din_2_V	in	8	ap_none	din_2_V	pointer
din_3_V	in	8	ap_none	din_3_V	pointer
din_4_V	in	8	ap_none	din_4_V	pointer
din_5_V	in	8	ap_none	din_5_V	pointer
din_6_V	in	8	ap_none	din_6_V	pointer
din_7_V	in	8	ap_none	din_7_V	pointer

Adder Trees - preventing tree balancing

- Hardware illustration

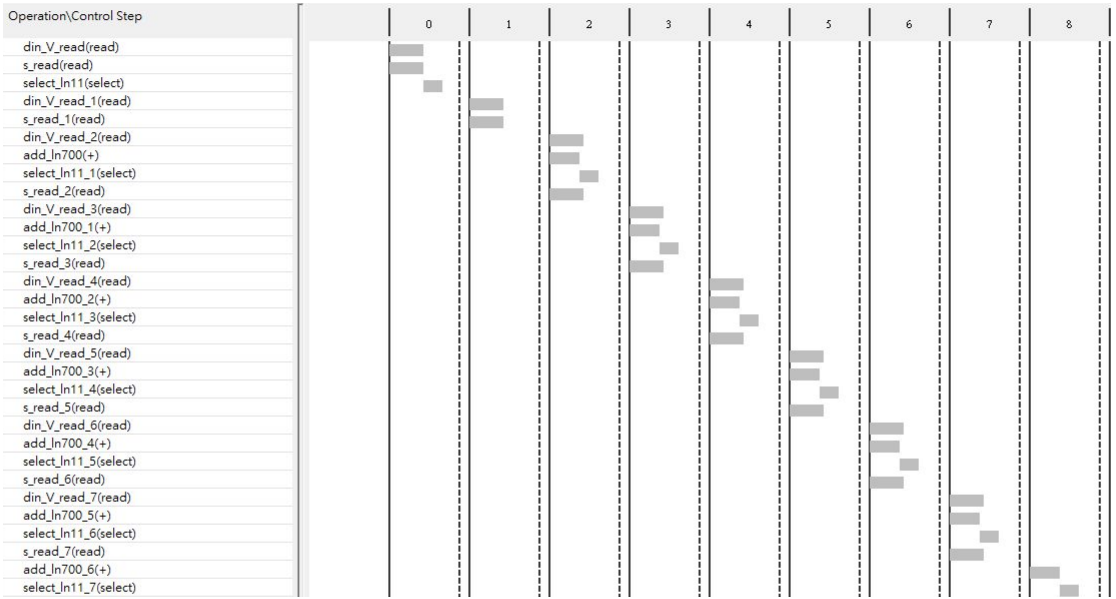


- Code

```
#include "Preventing_Automatic_tree_balancing.h"

ap_int<WIDTH_OUT> adder_tree_unbalanced(ap_int<WIDTH> din[NUM_REGS], bool s[NUM_REGS])
    #pragma HLS INTERFACE ap_fifo port=s
    #pragma HLS INTERFACE ap_fifo port=din

    ap_int<WIDTH_OUT> acc = 0;
    ap_int<WIDTH> tmp[NUM_REGS];
    copied_loop:for(int i=0;i!=NUM_REGS;i++) {
        #pragma HLS UNROLL
        tmp[i] = din[i];
    }

    operate_loop:for(int i=0;i!=NUM_REGS;i++) {
        #pragma HLS UNROLL
        if(s[i])
            acc += tmp[i];
    }

    return acc;
}
```

Adder Trees - preventing tree balancing

- Analysis timeline



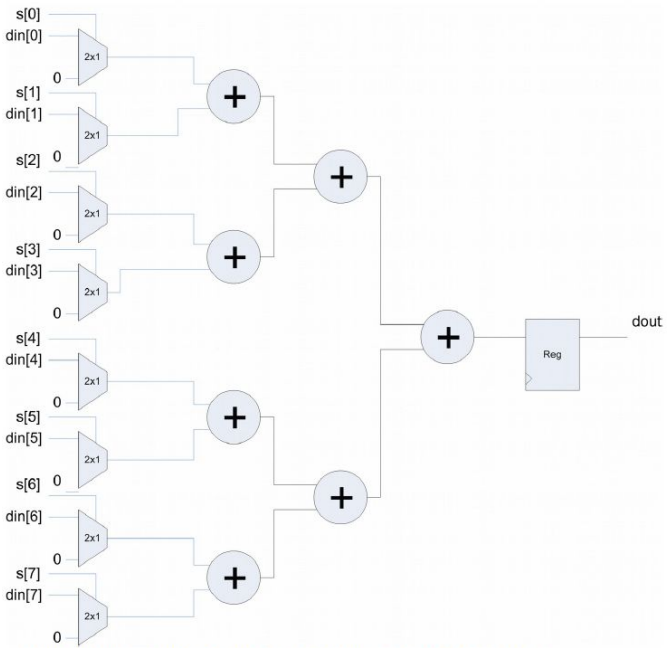
- Interface summary

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_rst	in	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_start	in	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_done	out	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_idle	out	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_ready	out	1	ap_ctrl_hs	adder_tree_unbalanced	return value
ap_return	out	8	ap_ctrl_hs	adder_tree_unbalanced	return value
din_V_dout	in	8	ap_fifo	din_V	pointer
din_V_empty_n	in	1	ap_fifo	din_V	pointer
din_V_read	out	1	ap_fifo	din_V	pointer
s_dout	in	1	ap_fifo	s	pointer
s_empty_n	in	1	ap_fifo	s	pointer
s_read	out	1	ap_fifo	s	pointer

Adder Trees - forcing tree balancing

- Hardware illustration

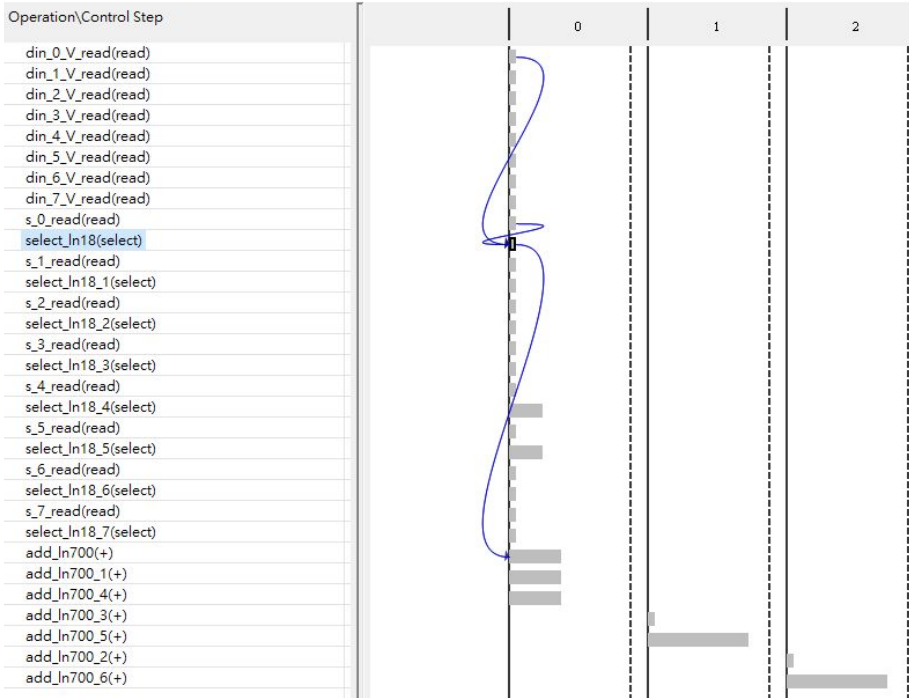


- Code

```
#include "Forcing_tree_balancing.h"

ap_int<WIDTH_OUT> adder_tree_rebalanced(ap_int<WIDTH> din[NUM_REGS], bool s[NUM_REGS]){
    #pragma HLS ARRAY_PARTITION variable=s complete dim=1
    #pragma HLS ARRAY_PARTITION variable=din complete dim=1
    #pragma HLS PIPELINE II=1

    ap_int<WIDTH_OUT> acc = 0;
    ap_int<WIDTH> tmp[NUM_REGS];

    copied_loop:for(int i=0;i!=NUM_REGS;i++) {
        #pragma HLS UNROLL
        tmp[i] = din[i];
    }
    operate_loop:for(int i=0;i!=NUM_REGS;i++) {
        #pragma HLS UNROLL
        acc += s[i] ? tmp[i] : (ap_int<WIDTH>)0;
    }

    return acc;
}
```

Adder Trees - forcing tree balancing

- Analysis timeline



- Interface summary

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_rst	in	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_start	in	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_done	out	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_idle	out	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_ready	out	1	ap_ctrl_hs	adder_tree_rebalanced	return value
ap_return	out	8	ap_ctrl_hs	adder_tree_rebalanced	return value
din_0_V	in	8	ap_none	din_0_V	pointer
din_1_V	in	8	ap_none	din_1_V	pointer
din_2_V	in	8	ap_none	din_2_V	pointer
din_3_V	in	8	ap_none	din_3_V	pointer
din_4_V	in	8	ap_none	din_4_V	pointer
din_5_V	in	8	ap_none	din_5_V	pointer
din_6_V	in	8	ap_none	din_6_V	pointer
din_7_V	in	8	ap_none	din_7_V	pointer
s_0	in	1	ap_none	s_0	pointer
s_1	in	1	ap_none	s_1	pointer
s_2	in	1	ap_none	s_2	pointer
s_3	in	1	ap_none	s_3	pointer
s_4	in	1	ap_none	s_4	pointer
s_5	in	1	ap_none	s_5	pointer
s_6	in	1	ap_none	s_6	pointer
s_7	in	1	ap_none	s_7	pointer

Look Up Table (LUT)

- Data

```
0,
.375,
.703125,
.921875,
1,
.921875,
.703125,
.375,
0,
-.390625,
-.71875,
-.9375,
-1,
-.9375,
-.71875,
-.390625
```

- Code

```
#include "lut.h"

ap_fixed<WIDTH,2> lut(ap_uint<ADDR_WIDTH> i) {
  const ap_fixed<WIDTH,2> sin_table[NUM_REGS] = {
    #include "data.inc"
  };
  return sin_table[i];
}


for(unsigned int i=0;i<NUM_REGS;i++) {
    cout << "-------test " << i << " -------" << endl;
    golden = sin(2*pi*i/(double)NUM_REGS);
    output = lut(i);
    cout << "your output: " << output << endl;

    if(output != golden) {
        cout << "golden: " << golden << endl;
        pass = 0;
        //break;
    }
}
```

LUT - RTL code

```
`timescale 1 ns / 1 ps
(* rom_style = "distributed" *) module lut_sin_table_rom (
addr0, ce0, q0, clk);

parameter DWIDTH = 8;
parameter AWIDTH = 4;
parameter MEM_SIZE = 16;

input[AWIDTH-1:0] addr0;
input ce0;
output reg[DWIDTH-1:0] q0;
input clk;

(* ram_style = "distributed" *) reg [DWIDTH-1:0] ram[0:MEM_SIZE-1];

initial begin
    $readmemh("./lut_sin_table_rom.dat", ram);
end


always @(posedge clk)
begin
    if (ce0)
    begin
        q0 <= ram[addr0];
    end
end
```

LUT - Performance & Interfaces

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.254 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	1	5.000 ns	5.000 ns	1	1	none

Detail

Instance

Loop

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	lut	return value
ap_rst	in	1	ap_ctrl_hs	lut	return value
ap_start	in	1	ap_ctrl_hs	lut	return value
ap_done	out	1	ap_ctrl_hs	lut	return value
ap_idle	out	1	ap_ctrl_hs	lut	return value
ap_ready	out	1	ap_ctrl_hs	lut	return value
ap_return	out	8	ap_ctrl_hs	lut	return value
i_V	in	4	ap_none	i_V	scalar

THANKS