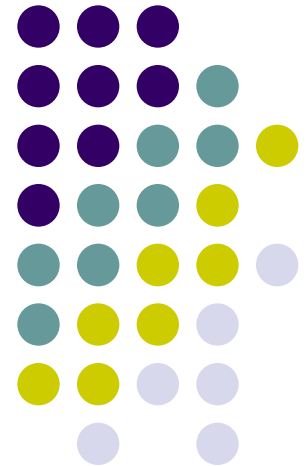
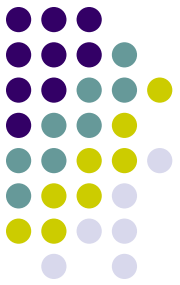


SIS: A System for Sequential Circuit Synthesis

A Tutorial of Usage and
Programming of SIS





Introduction

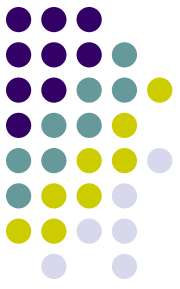
- Berkeley: MIS II -> SIS
- SIS: A System for Sequential Circuit Synthesis
 - Combinational logic synthesis
 - Two level & multi-level logic synthesis
 - Technology mapping
 - Sequential Circuit Synthesis
 - Verification

Interactive Interface: Example



```
$ /u/class/CS258G/sis-1.3/bin/sis
sis> read_blif C1908.blif2
sis> print_stats
C1908.iscas   pi=33   po=25   nodes=400   latches= 0
lits(sop)= 800
sis> source script.rugged
sis> print_stats
C1908.iscas   pi=33   po=25   nodes=152   latches= 0
lits(sop)= 553
sis> write_blif C1908_r.blif
sis> quit
```

SIS Commands used in the example



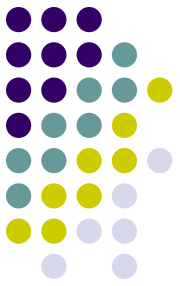
- *read_blif* circuit1.blif
 - Read blif-format circuit
- *source script.rugged*
 - Do optimization using a set of commands
 - many scripts in
/u/class/CS258G/sis-1.3/sis/sis_lib
- *write_blif* circuit2.blif
 - Write blif-format circuit

Some Other SIS Commands

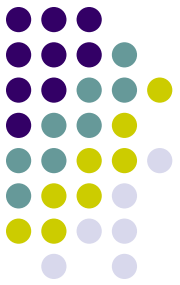


- *tech_decomp*
 - Decompose all the nodes in the current network into AND gates or OR gates or both
- *read_eqn eqn_circuit*
 - Read eqn-format circuit
- *verify blif_circuit*
 - Verify whether the *blif_circuit* is identical with the one read in before, which is *eqn_circuit*.
- *print_stats*
 - Print stats of the current circuit

SIS Programming

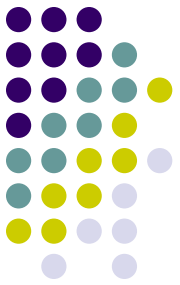


- Two styles:
 - Integrated programming style (being able to add commands into existing SIS interface)
 - Take advantage of the SIS commands online
 - Running and debugging need to be carried out under SIS's interface.
 - *[/u/class/CS258G/project_template/sis_guide.txt](#)*
 - *[/u/class/CS258G/project_template/readme.txt](#)*
 - Stand-alone programming style
 - Only link with SIS library
 - Easier to run and debug

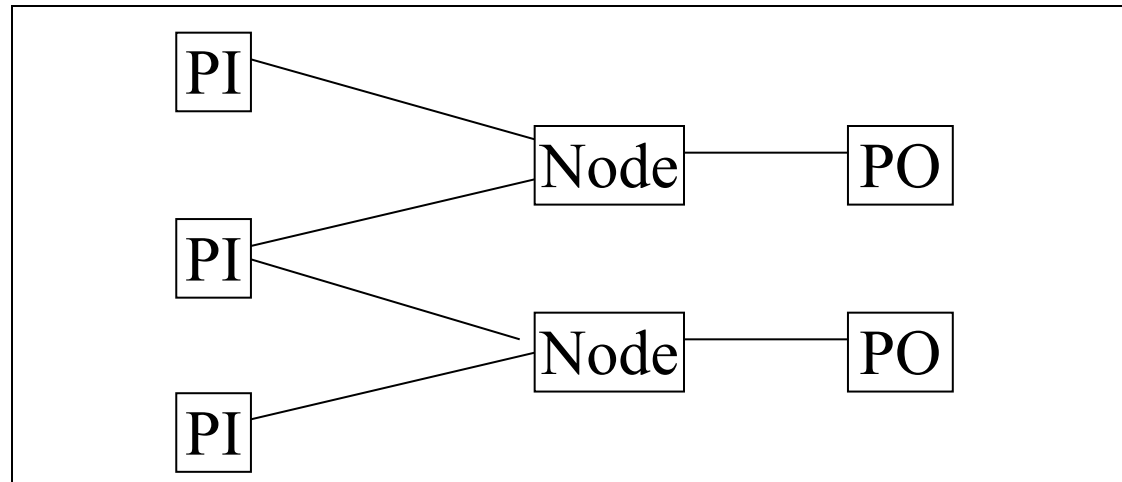


Directory Structure

- `/u/class/CS258G/sis-1.3/`
 - SIS package location
- `/u/class/CS258G/sis-1.3/bin/`
 - Executable SIS program
 - Add it to your path settings
- `/u/class/CS258G/proj_2004/circuits/`
 - Benchmark circuits
- `/u/class/CS258G/project_template/`
 - programming template for integrated style
- `/u/class/CS258G/proj_2004/rasp/`
 - UCLA RASP package executable
- `/u/class/CS258G/tutorial/`
 - Contains this tutorial



SIS Network

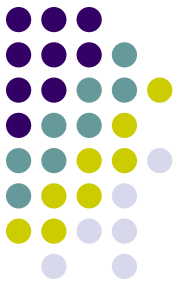


PI: Primary Input

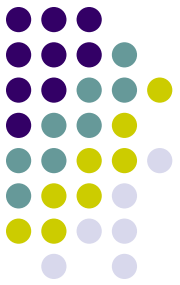
PO: Primary Output, a special node with one input and no output. Its function is the same as a buffer.

Node: Internal node

Input and Output: BLIF format



- BLIF format:
 - Berkeley Logic Interchange Format
 - Used widely in academic research work
- Function: Cubic form (sum of product)
- Each node has multi-inputs and single output
- Benchmark circuit for this project
 - `/u/class/CS258G/proj_2004/circuits/`
 - optimized by `script.algebraic`
 - Decomposed into 2-input AND gates and OR gates

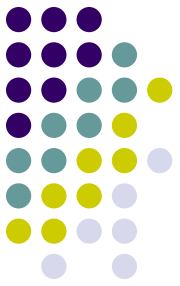


Blif Example: C17.blif

```
.model C17.iscas
.inputs 1GAT[0] 2GAT[1] 3GAT[2] 6GAT[3] 7GAT[4]
.outputs 22GAT[10] 23GAT[9]
.names 1GAT[0] 2GAT[1] 3GAT[2] [2] 22GAT[10]
1-1- 1
-1-1 1
.names 2GAT[1] 7GAT[4] [2] 23GAT[9]
1-1 1
-11 1
.names 3GAT[2] 6GAT[3] [2]
0- 1
-0 1
.end
```

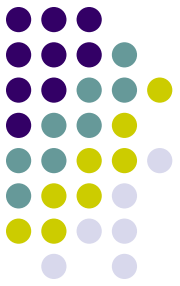
Programming Example

(Makefile of Stand-alone Style)



```
CC = gcc
SIS = /u/class/CS258G/sis-1.3/sis
PSRC  = my_main.c my_util.c
POBJ  = $(PSRC:.c=.o)
PHDR  = my_main.h
TARGET = my.x
LIBS  = $(SIS)/lib/libsis.a
INCLUDE = -I$(SIS)/include
CFLAGS = -g $(INCLUDE) -DSIS
LDFLAGS = -g
$(TARGET): $(POBJ) $(LIBS)
    $(CC) $(LDFLAGS) -o $(TARGET) $(POBJ)
    $(LIBS) -lm
```

Programming Example (C codes)

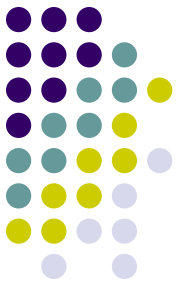


```
#include "my_main.h"
int blif_to_network ( char *filename, network_t **net )
{
    FILE      *fp;
    fp=fopen( filename, "r");
    read_blif( fp, net ) ;
    fclose( fp );
}
int write_result( char *filename, network_t *net )
{
    FILE      *fp;
    fp=fopen( filename, "w");
    write_blif(fp, net, 0, 0 );
    fclose( fp );
}
```

Programming Example (Con't)



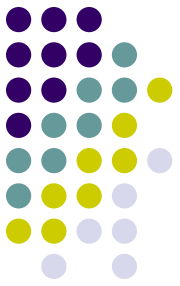
```
main(int argc, char **argv) {
    init_sis(0);
    option(argc, argv);
    blif_to_network(mid_filename,&net1);
    dfs = network_dfs(net1);
    for(i1=0;i1<array_n(dfs);i1++){
        node1 = array_fetch(node_t*, dfs, i1);
        if(node_type(node1)==PRIMARY_INPUT)
            continue;
        if(node_type(node1)==PRIMARY_OUTPUT)
            continue;
        if(node_num_fanin(node1)==1)
            continue;
        simplify_node(node1, SIM_METHOD_NOCOMP, SIM_DCTYPE_ALL,
            SIM_ACCEPT_SOP_LITS, SIM_FILTER_NONE);
    }
    network_sweep(net1);
    write_result(final_filename, net1);
    network_free(net1);
}
```



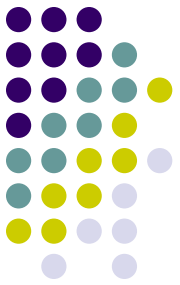
SIS Documents

- SIS paper
 - */u/class/CS258G/sis-1.3/SIS_paper.ps*
 - The detailed introduction of SIS system, including BLIF format description
- Description of functions
 - */u/class/CS258G/sis-1.3/sis/doc/*

Commonly Used Documents

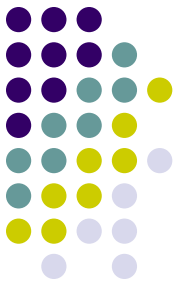


- network.doc
 - describe network operations.
- node.doc
 - describe node operations.
- array.doc
 - describe the operations for array data structure.
- st.doc
 - describe the operations for st_table (hash table)
- io.doc
 - describe the operations for I/O of SIS.
- util.doc
 - describe some supporting operations for programming.



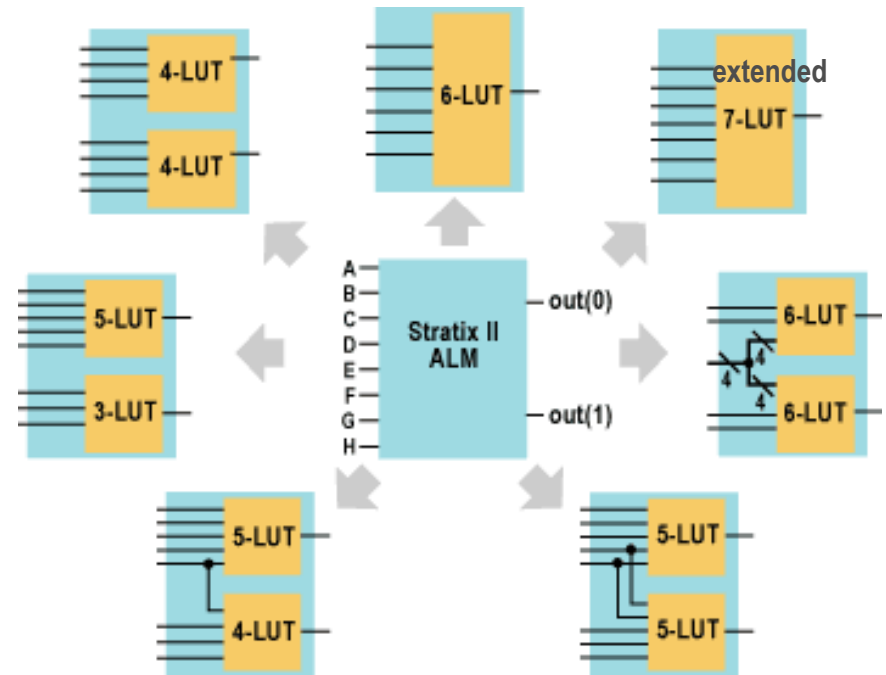
Questions To

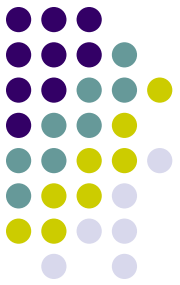
- Deming Chen
 - demingc@cs.ucla.edu
- Check FAQ once in a while
 - *</u/class/CS258G/FAQ>*



Stratix II ALM Overview

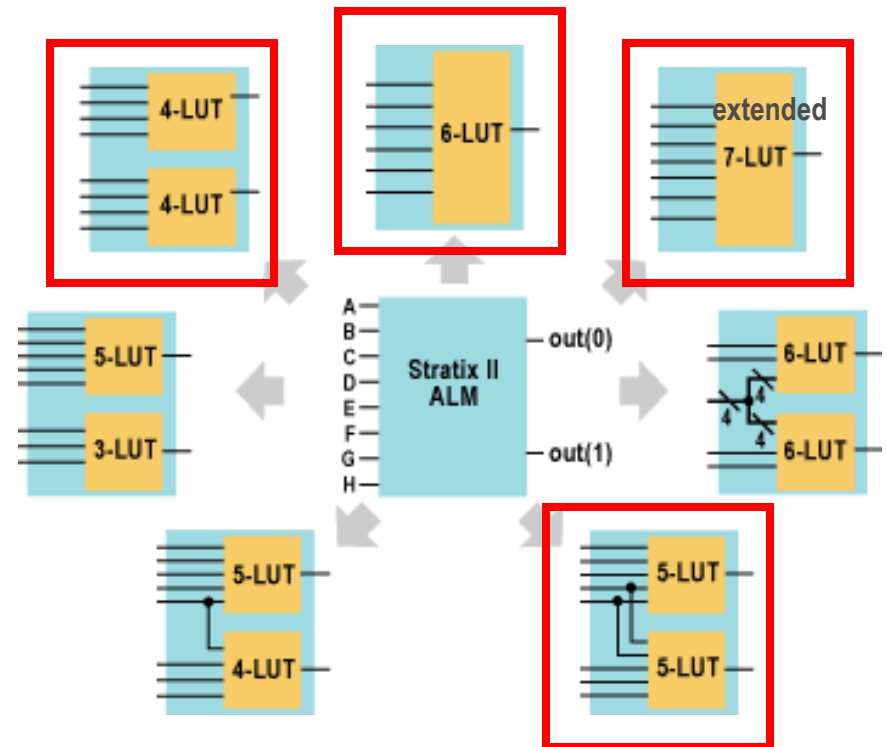
- High flexibility
- Implement complex logic functions with large input number
 - Reduce logic level
 - Reduce routing utilization





Stratix II ALM Overview

- High flexibility
- Implement complex logic functions with large input number
 - Reduce logic level
 - Reduce routing utilization

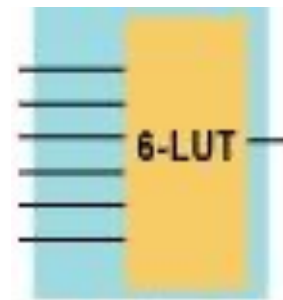
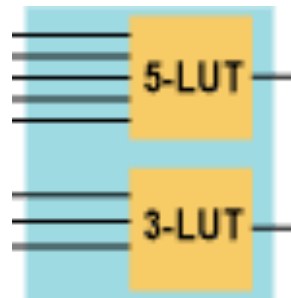
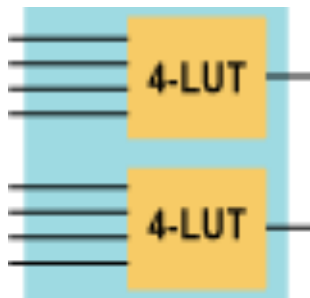


 required



ALM Normal Mode

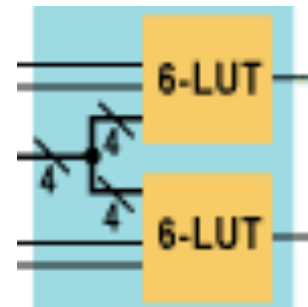
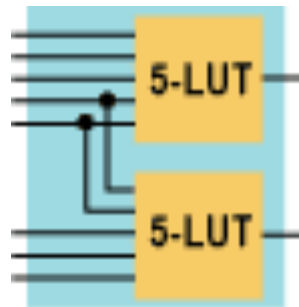
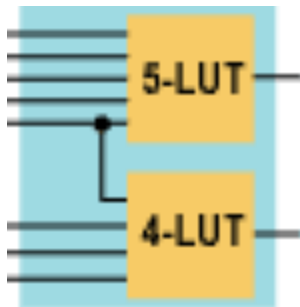
- One ALM can implement the following functions without any restrictions
 - Any two 4-input functions
 - One 5-input function and one 3-input function
 - One 6-input function

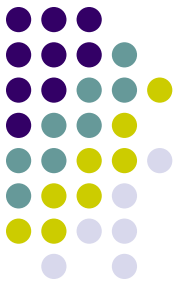




ALM Normal Mode

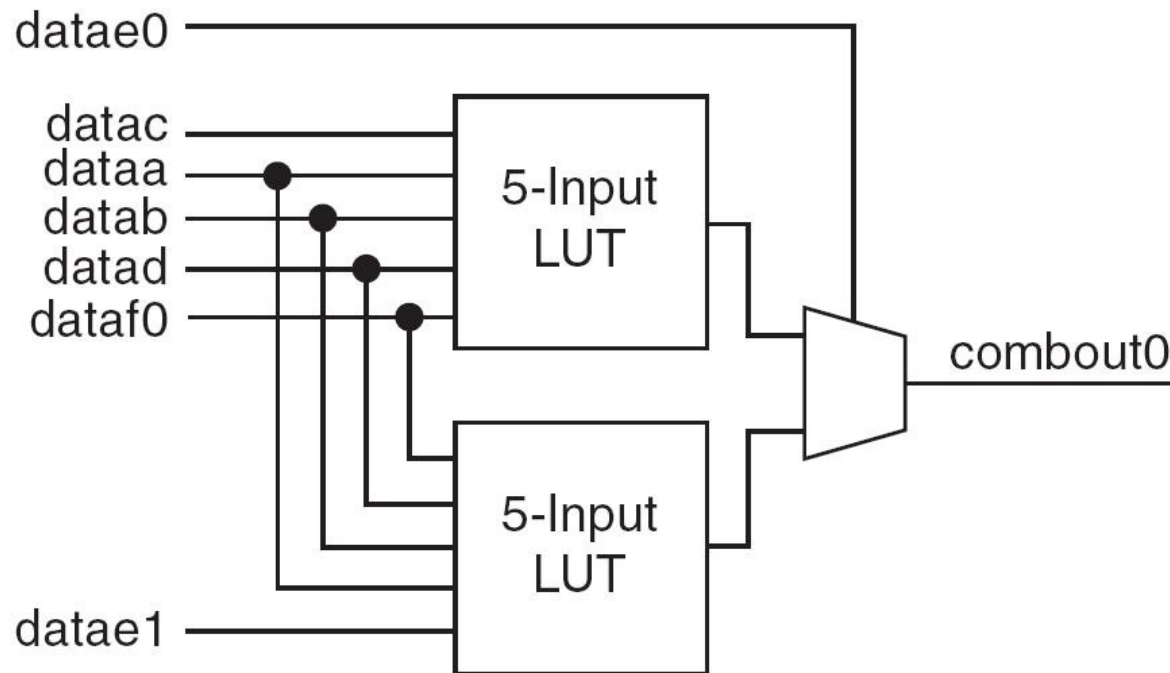
- One ALM can implement the following functions with sharing inputs
 - One 5-input function and one 4-input function that sharing one inputs
 - Two 5-input functions that sharing two inputs
 - Two 6-input functions that sharing 4 inputs





Extended LUT mode

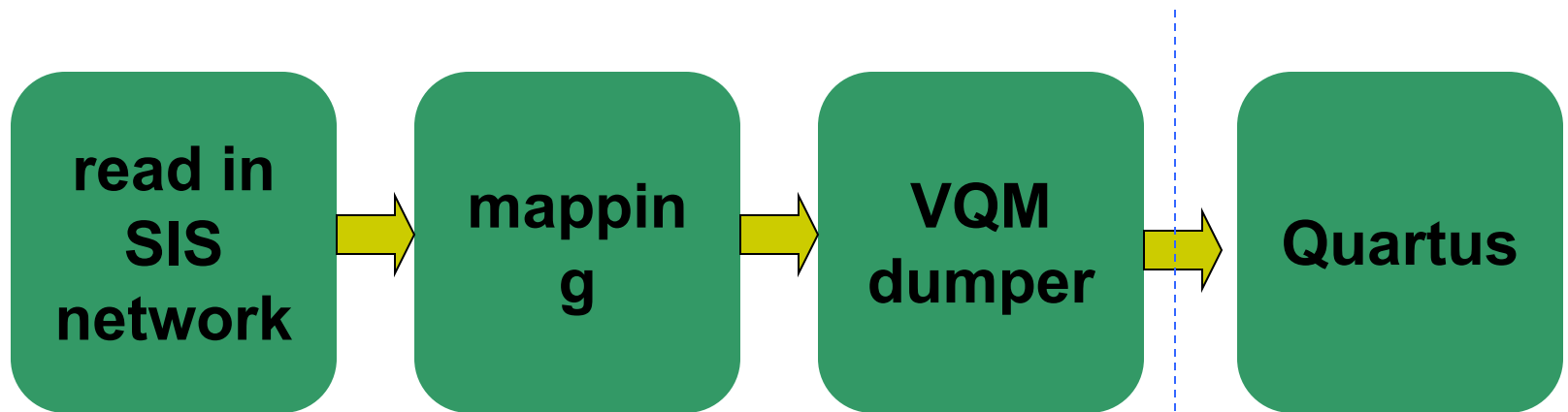
- One ALM can implement one 7-input function if it can fit into the following diagram

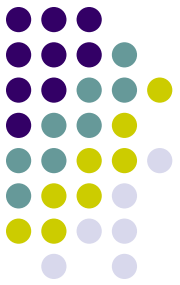




QUIP flow

- Verify the quality of the designs through state-of-art tools
- Quartus can take the QUIP interface to read in mapped circuits





Additional Notes

- Extra data fields in SIS data structure
 - One ALM may contains two LUTs
 - Pairing is needed to record the two LUTs in one ALM
 - The “undef1” field in SIS can be used for such purpose
 - typedef struct {
 int label;

} proj_aux_t;
 - API: PROJ_PAIR_ID(n)
 - Remember to allocate the memory for the structure
 - Example code in
/u/class/CS258G/project_template/myproj/com_myproj.c