

ECE 587 – Hardware/Software Co-Design

Lecture 04 State-Based Models II

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

January 27, 2025

Reading Assignment

- ▶ This lecture: 3.1.2
- ▶ Next lecture: 3.1.1

Control-Data Flow Graph

Finite State Machine with Data

Sequential Programs

- ▶ The most intuitive model to specify functionality.
- ▶ Sequential programs are actually state-based.
 - ▶ Variables and execution flow together determine the state.
 - ▶ (If we assume bounded dynamic memory allocations and recursions)
- ▶ Can we translate sequential programs directly into hardware implementations?
 - ▶ Known as behavioral synthesis or high-level synthesis.

Elements of Sequential Programs

- ▶ Variables
 - ▶ An abstraction of memory
 - ▶ Associated with types to explain the meaning of bits
- ▶ Execution flow: imperative
 - ▶ Statements specify what to do.
 - ▶ Statements are ordered for execution.
 - ▶ Control constructs (if, for, while, etc.) may change the flow but fundamentally they are all variants of branches.
- ▶ Functions and object-oriented programming (OOP) provide additional abstractions to ensure conciseness and correctness.
 - ▶ Essentially they are still executed sequentially.

Reason with Sequential Programs

- ▶ In its textual form, it's hard to reason with sequential programs.
 - ▶ Either by designers or automated tools.
- ▶ Compiling to certain target instruction set
 - ▶ While a program in assembly language can be easily manipulated by automated tools, too many implementation details are included, e.g. number of registers.
 - ▶ We need some intermediate representations that describe the functionality independent of the target instruction set.
- ▶ Control-Data Flow Graph (CDFG)
 - ▶ A specialized type of flow chart.

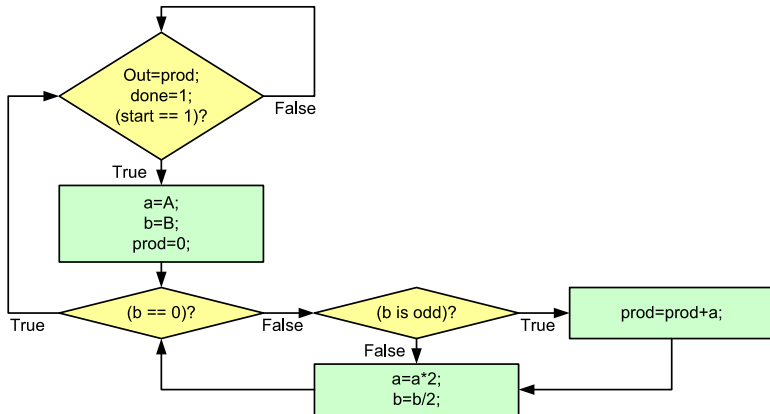
Control-Data Flow Graph (CDFG)

- ▶ Two kinds of blocks representing computations
 - ▶ Basic block (BB): rectangular shape
 - ▶ Decision block: diamond shape
- ▶ Blocks are connected by (directed) arcs representing the steps in your program.
 - ▶ Each basic block should lead to exactly one block.
 - ▶ Each decision block can lead to multiple blocks depending on the outcome of the computation in the block.
- ▶ To generate the CDFG,
 - ▶ The program is decomposed into sets of statements without branches, and branches themselves.
 - ▶ Some branch may involve complex computations. In such case, we would assume the computations are moved to before the actual branch.
 - ▶ Each set of statements is represented by a BB.
 - ▶ Each branch is represented by a decision block.

Example: Sequential Multiplication

- ▶ Input
 - ▶ A, B : two 32-bit binary number
 - ▶ $start$: 1-bit, indicate A and B will be available for a new job
- ▶ Output
 - ▶ Out : one 32-bit binary number for the product
 - ▶ $done$: 1-bit, indicate Out is ready from the previous job and the multiplier is idle for the current cycle
- ▶ Timing sequence
 - ▶ The multiplier will start a job if it is idle and $start = 1$. It will no longer be idle the next cycle.
 - ▶ A and B will be available when $start = 1$ and in the next few clock cycles.
 - ▶ The multiplier will indicate the product is ready at Out by turning $done$ to 1. It will go idle the next cycle.
 - ▶ This is the usual way to interface with hardware/chip if you don't know when the computation will finish.

CDFG for Sequential Multiplication



Control-Data Flow Graph

Finite State Machine with Data

Finite State Machine with Data (FSMD)

- ▶ An extension of FSM that allows to include additional variables into the system
 - ▶ The functions f and h now not only depend on these variables but can also update them.
 - ▶ RTL operations can be included for convenience.
- ▶ For many systems, FSMD helps to separate states that are data from states that are control signals, making the overall system easy to understand.
- ▶ Similar to FSM, FSMD can be directly translated to hardware implementations.
 - ▶ Controller+Datapath

General FSMMD Architecture

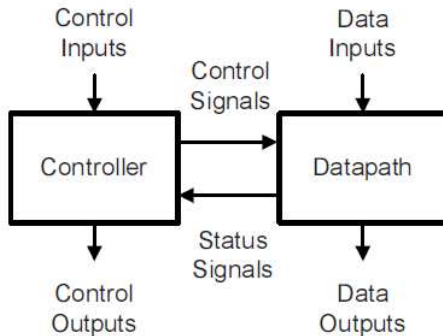


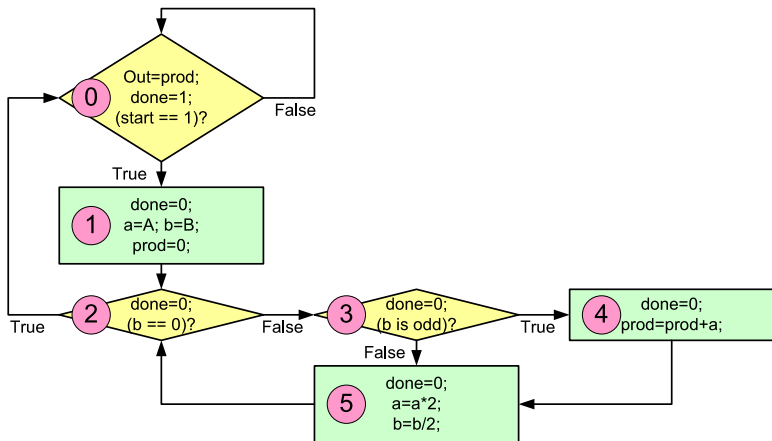
FIGURE 6.2 High-level block diagram

(Gajski et al., 2009)

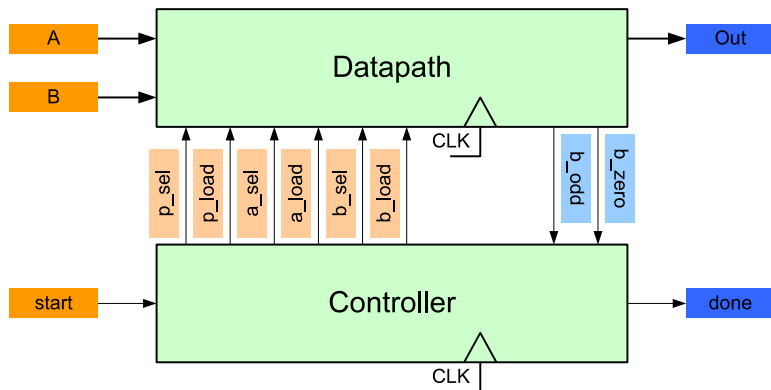
From CDFG to FSM

- ▶ One can map from CDFG to FSM by representing each block with a state and each arc with a state transition.
- ▶ Essentially, your design is still a FSM where the state bits are decomposed into two groups.
 - ▶ Variables: these bits are handled by the RTL operations
 - ▶ Controller state: these bits represent the state of the controller.

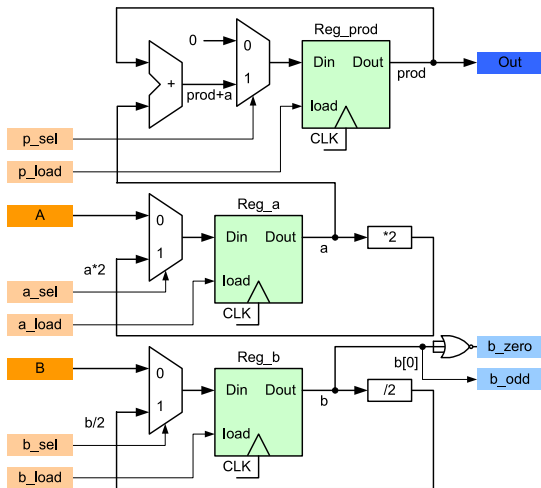
FSMD for Sequential Multiplication



Multiplier Architecture



Datapath Implementation



- ▶ Implement all RTL operations.
 - ▶ Use mux's to route operands/results.
 - ▶ Decide all control signals later in controller design.

Controller Implementation: Transition Equation

- ▶ There are 6 states: denote by S , 3 bits.

- ▶ State transitions

$S(t)$	start	b_zero	b_odd	$S(t + 1)$
0	0	X	X	0
0	1	X	X	1
1	X	X	X	2
2	X	0	X	3
2	X	1	X	0
3	X	X	1	4
3	X	X	0	5
4	X	X	X	5
5	X	X	X	2
6	X	X	X	0
7	X	X	X	0

- ▶ Pay attention to the state 6 and 7: we have to reset them

Controller Implementation: Output Equation

$S(t)$	done	p_load	p_sel	a_load	a_sel	b_load	b_sel
0	1	0	X	0	X	0	X
1	0	1	0	1	0	1	0
2	0	0	X	0	X	0	X
3	0	0	X	0	X	0	X
4	0	1	1	0	X	0	X
5	0	0	X	1	1	1	1
6	0	X	X	X	X	X	X
7	0	X	X	X	X	X	X

- ▶ For the state 6 and 7, we use $done=0$ to indicate the multiplier is not ready yet.
- ▶ Since $a_load = b_load$ and $a_sel = b_sel$, we can share them to save more cost.

Practical Considerations

- ▶ Tools could further optimize a CDFG by combining blocks or simulating the FSMD in a symbolic way to unroll loops, forming a larger BB that may contain better parallelism.
- ▶ Multiple states are necessary for large BBs in order to reduce clock periods.
- ▶ Datapath components may be reused and/or pipelined.
- ▶ These are the concerns of high-level synthesis that we will discuss later in the semester.

Summary and Discussions

- ▶ Sequential programs are modeled by CDFG, which can be translated into FSMD, leading to hardware implementations.
- ▶ Popularity of sequential programs comes from
 - ▶ Fully ordered semantics are easy to reason with.
 - ▶ Only a small portion of the state bits are changed per statement, leading to efficient implementations on controller+datapath+memory architectures.
 - ▶ Complexity can be reduced at even higher abstraction levels like object-oriented programming.