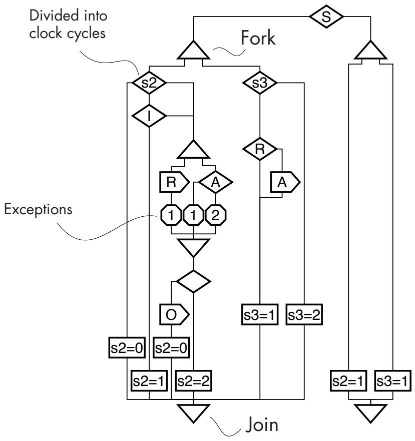
HIGH-LEVEL SYNTHESIS FROM THE SYNCHRONOUS LANGUAGE ESTEREL

Stephen A. Edwards
Columbia University, New York
sedwards@cs.columbia.edu http://www.cs.columbia.edu/~sedwards

I. Circuit Synthesis from the Program Dependence Graph

```
every R do
  loop
    await A;
    emit B;
    present C then
      emit D
    end;
    pause
  end
||
  loop
    present B then
      emit C
    end;
    pause
  end
end
```



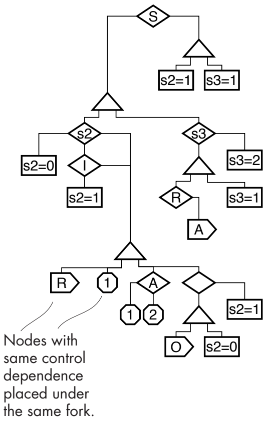
Divided into clock cycles

Fork

Exceptions

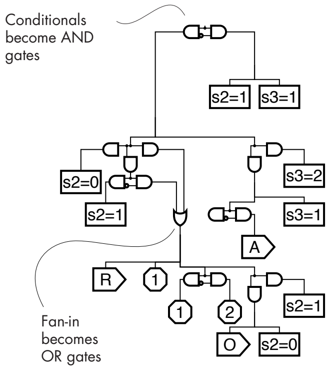
Join

Concurrent Control-Flow Graph



Nodes with same control dependence placed under the same fork.

Program Dependence Graph



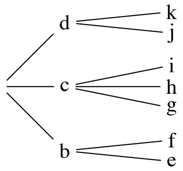
Conditionals become AND gates

Fan-in becomes OR gates

Generated Control Circuit

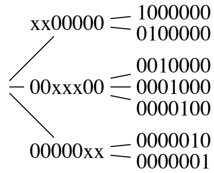
II. High-Level State Assignment

```
abort
  pause; emit e;
  pause; emit f
when b;
abort
  pause; emit g;
  pause; emit h;
  pause; emit i
when c;
abort
  pause; emit j;
  pause; emit k
when d
```

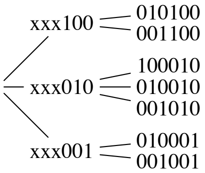


State Tree taken directly from the source

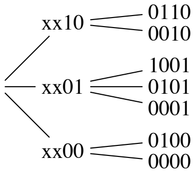
Berry's Encoding: Each leaf has its own latch.



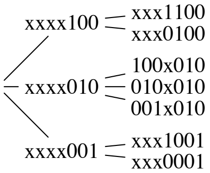
Possible encoding: 1-hot encoding at each branch point.



Encoding from my software compiler: log n encoding at branching points.

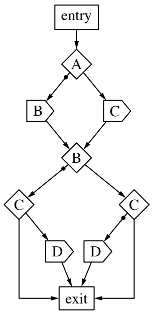


Possible encoding: redundant encoding at middle subtree to reduce communication

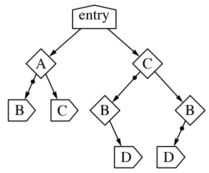


III. Don't-Care Extraction from Control-flow Information

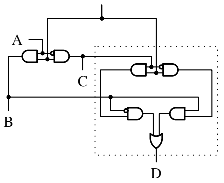
```
present A then
  emit B
else
  emit C
end;
present C then
  present B else
    emit D
  end
else
  present B then
    emit D
  end
end
```



Concurrent Control-Flow Graph



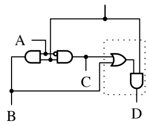
Program Dependence Graph



Karnaugh Map for output D:

		B	B	
	0	0	0	0
entry	0	1	0	1
			C	C

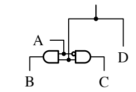
Generated Control Circuit



B and C are mutually exclusive: Logic can be simplified

		B	B	
	0	0	X	0
entry	0	1	X	1
			C	C

Simplified Control Circuit



entry implies B or C: Logic simplified further

		B	B	
	0	0	X	0
entry	X	1	X	1
			C	C

Optimal Control Circuit