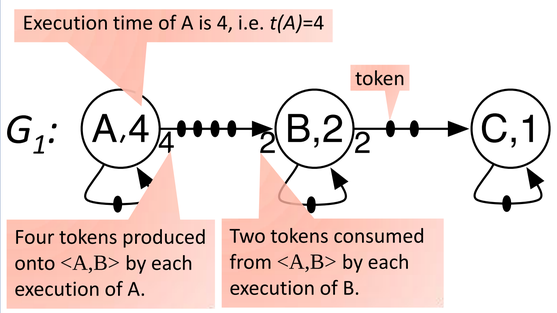
ISCAS 中国科学院软件研究所学术年会’2018 暨计算机科学国家重点实验室开放周

学术论文

基于内存约束的同步数据流吞吐量分析框架

朱雪阳
zxy@ios.ac.cn
13693211481

1 Motivation



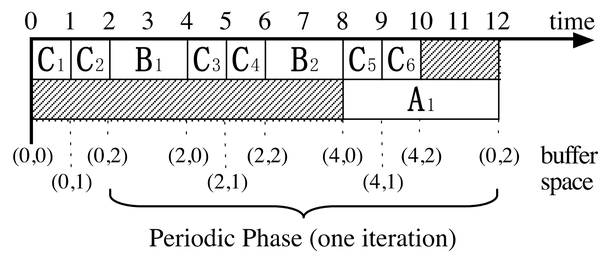
Synchronous data flow graphs (SDFGs)
- ✓ Nodes (actors) model the computations
- ✓ Edges (FIFO channels) model data dependency between actors
- ✓ E.g. an iteration of G_1 includes one firing of actor A, two firings of B and four of C

Iteration period (IP)
- ✓ the average computation time per iteration
- ✓ the reciprocal of the throughput.

Different memory abstractions may lead to different achievable IPs

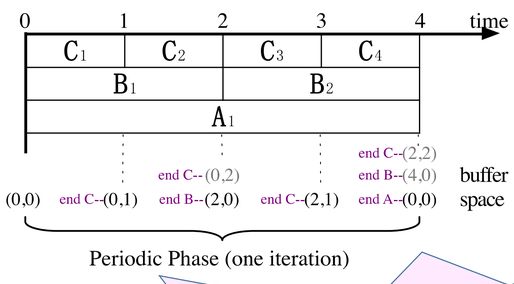
Schedules of *G1* with buffer bound 4 on edge <A,B> and 2 on <B,C>

A schedule of *G1* with IP = 1/thr. = 10



Assume that for a firing of an actor,
- ✓ the buffer space needed by the produced tokens is claimed at the start of the firing
- ✓ the buffer space of consumed tokens is released at the end of the firing.

A schedule of *G1* with IP = 1/thr. = 4



Assume that for a firing of an actor,
- ✓ the buffer space needed by the produced tokens is claimed at the end of the firing
- ✓ the buffer space of consumed tokens is released at the end of the firing.

To match up to the implementation, we present a unified framework for throughput analysis of SDFGs to deal with different memory abstractions.

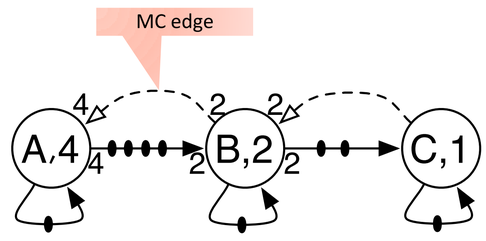
2 Problem & Solution(1)

Memory abstraction *xy*
- ✓ x=1, release space at the start of a firing; x=0, at the end of a firing;
- ✓ y=1, claim space at the start of a firing; y=0, at the end of a firing.

A memory constraint of SDFG G = (V,E) is a vector MC(E), in which MC(e)>d(e).
$IP_{xy}(G,MC)$ is the minimal achievable IP of SDFG G = (V,E) under memory constraint MC(E) based on the memory abstraction *xy*. Then $1/IP_{xy}(G,MC)$ is the maximal achievable throughput.

Given an SDFG G and a memory constraint MC, $IP_{xy}(G,MC)$ = ?

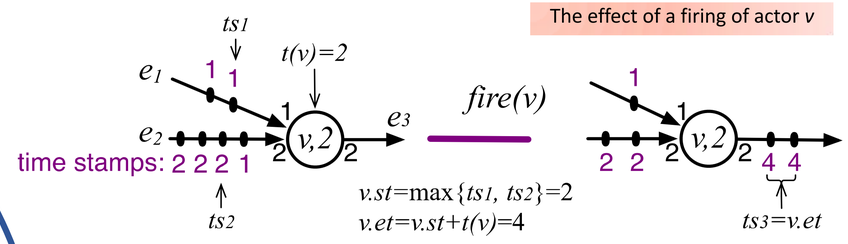
Memory-constrained SDFG (MC SDFG)
- ✓ The bound on buffer of an edge <u,v> of an SDFG can be modeled by adding edge <v,u> with tokens to model available storage space
- ✓ Edge <v,u> is called a MC edge



An self-timed execution (STE) analysis method is used in our solutions

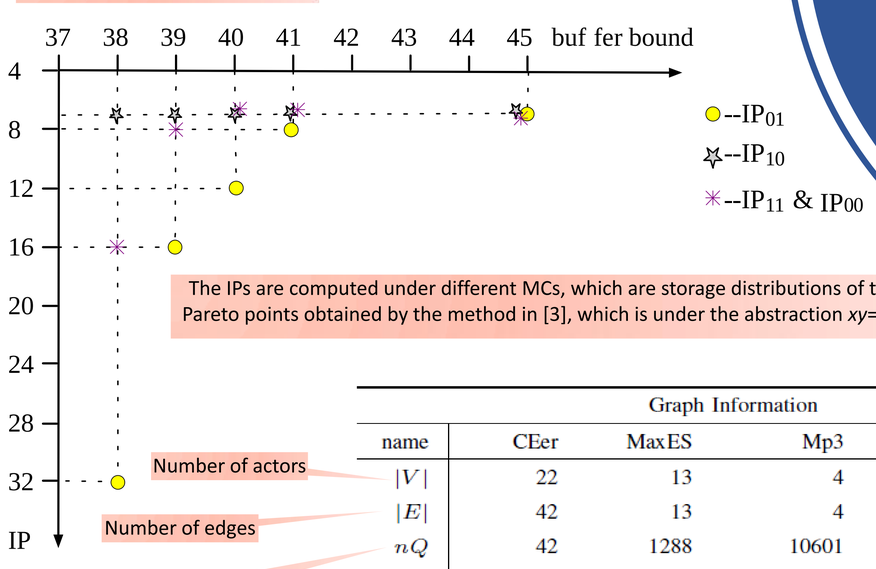
Instead of using a clock in the operational semantics of SDFGs as traditional STE analysis methods [3, 4], we use time stamps on tokens to model the time progress. Each token is tagged with a time stamp to indicate the time when it is produced.

The effect of a firing of actor *v*



v.st, the start time of the firing of actor v
v.et, the end time of the firing of actor v
t(v), the execution time of a firing of actor v

The IP_{xy}s of SDFG of a modem [1]



The IPs are computed under different MCs, which are storage distributions of the Pareto points obtained by the method in [3], which is under the abstraction *xy*=01.

Number of actors
Number of edges
Number of firings in one iteration
memory constraint

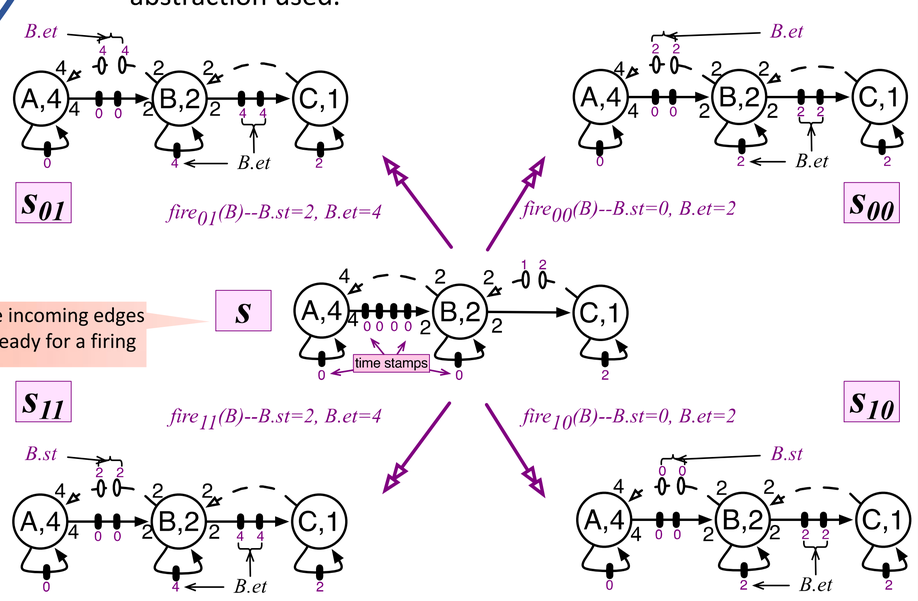
Graph Information					
name	CEer	MaxES	Mp3	SaRate	Satellite
$\|V\|$	22	13	4	6	22
$\|E\|$	42	13	4	5	26
nQ	42	1288	10601	612	4515
MC	73	1065	1980	32	1542
With Auto-concurrency (IP/imp. compare with IP_{01})					
IP_{01}	47128	9216	352584	1029	576
IP_{00}	47128/0%	8192/11%	117914/67%	392/62%	336/42%
IP_{11}	47128/0%	8192/11%	236160/33%	735/29%	336/42%
IP_{10}	47128/0%	683.7/93%	116424/67%	0/100%	96/83%
Without Auto-concurrency (IP/imp. compare with IP_{01})					
IP_{01}	47128	9216	352584	1088	1320
IP_{00}	47128/0%	8447/8%	120000/66%	960/12%	1056/20%
IP_{11}	47128/0%	8192/11%	236160/33%	960/12%	1056/20%
IP_{10}	47128/0%	8192/11%	120000/66%	960/12%	1056/20%

Most cases finished in several milliseconds; the largest execution time is 30ms.

The proposed analysis framework is helpful for implementation according to schedules based on different abstractions.
We have extended the proposed technique to CSDF graphs, but omit them here because of space limitations.

- ✓ The time stamps of tokens produced on E are not affected by the abstractions. They are always *B.et*, although the values of *B.et* may differ with respect to claim abstraction *y*.
- ✓ The time stamps of tokens on MC edges are decided by the abstraction used.

Firing rules $fire_{xy}$ are defined for different abstractions



- ✓ When the release time of consumed tokens is set to the end of the firing, i.e. *x* = 0, these time stamps are tagged with *B.et*, e.g. states s_{01} and s_{00}; otherwise, they are tagged with *B.st*, e.g. states s_{11} and s_{10}.
- ✓ Abstraction *y* affects the start time of a firing. When the required space is claimed at the start of the firing, i.e. *y* = 1, *B.st* is set to be the largest time stamp of the required tokens for the firing, e.g. $fire_{01}$ and $fire_{11}$. Otherwise, the largest time stamp of the required tokens for the firing can be seen as the end time of the firing, i.e. *B.et*, then *B.st* = *B.et* − *t(B)*, e.g. $fire_{00}$ and $fire_{10}$.

Similar to the methods in [3, 4], the $IP_{xy}(G,MC)$ can be computed via the STE according to firing rules $fire_{xy}$.

3 Solution(2)

4 Experimental Results

Abstract:
Streaming applications are often modeled with Synchronous data flow graphs (SDFGs). A proper analysis of the models is helpful to predict the performance of a system. In this paper, we focus on the throughput analysis of memory-constrained SDFGs (MC SDFGs), which needs to choose a memory abstraction that decides when the space of consumed data is released and when the required space is claimed. Different memory abstractions may lead to different achievable throughputs. The existing techniques, however, consider only a certain abstraction. If a model is implemented according to other abstractions, the analysis result may not truly evaluate the performance of the system. In this paper, we present a unified framework for throughput analysis of MC SDFGs for difference abstractions, aiming to provide evaluations matching up to the corresponding implementations.

References

[1] SS Bhattacharyya, PK Murthy, and EA Lee. 1996. Software synthesis from dataflow graphs. Vol. 360. Springer.
[2] EA Lee and DG Messerschmitt. 1987. Static scheduling of synchronous data flow programs for digital signal processing. IEEE Trans. on Computers 36, 1, 24–35.
[3] S Stuijk, M Geilen, and T Basten. 2008. Throughput-buffering trade-off exploration for cyclo-static and synchronous dataflow graphs. IEEE Trans. on Computers 57, 10, 1331–1345.
[4] XY Zhu, M Geilen, T Basten, and S Stuijk. 2014. Memory-Constrained Static Rate-Optimal Scheduling of Synchronous Dataflow Graphs via Retiming. In Proc. of DATE 2014. 1–6.

Xue-Yang Zhu. A Unified Framework for Throughput Analysis of Streaming Applications under Memory Constraints. In Proc. Of the 22nd International Conference on Engineering of Complex Computer Systems (ICECCS 2017). Fukuoka, Japan, 6-8 Nov., 2017. Pp.128-137.
Xue-Yang Zhu. A Unified Framework for Throughput Analysis of Synchronous Data Flow Graphs under Memory Constraints: Work-in-Progress. In Proc. of CODES/ISSS’17 Companion (part of ESWEEK 2017), Seoul, Republic of Korea, October 15–20, 2017. Article No. 2.