Work-in-Progress: Equivalence of Transformations of Synchronous Data Flow Graphs

Xue-Yang Zhu
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
zxy@ios.ac.cn

***Abstract*—Synchronous data flow graphs (SDFGs) are widely used to model digital signal processing (DSP) algorithms and streaming applications. Any valid SDFG can be converted to an equivalent homogenous SDFG (HSDFG). Retiming and unfolding are important graph transformation techniques for performance optimization of SDFGs. In this paper, we review the relationship among the three transformation techniques and reveal the equivalence of the combinations of them. The equivalence makes it possible to analyze and optimize SDFGs directly, without carrying out the explicit conversion procedures which are usually time and space-consuming.**

I. Introduction

The *synchronous dataflow graphs* (SDFGs) [1] are widely used to represent DSP algorithms and streaming applications, of which performance is usually important. Each node (also called actor) in an SDFG represents a computation and each edge models a FIFO channel. The sample rates of actors of an SDFG may differ. *Homogenous synchronous dataflow graphs* (HSDFGs) are a special type of SDFGs. All sample rates of actors of an HSDFG are one.

Algorithms modeled with SDFGs are usually nonterminating and repetitive. Execution of all the computations of an SDFG for the required number of times is referred to as an *iteration*. An iteration of an SDFG may include more than one execution, or *firing*, of an actor, because of its multi-rate nature. An HSDFG includes exactly one firing of an actor in an iteration. How fast an SDFG could be scheduled is limited by its *iteration period* (IP) – the minimal achievable computation time per iteration of the SDFG.

Retiming [2] and unfolding [3] are important graph transformation techniques for performance optimization of data flow graphs. Both of them are originally applied to reduce the IP of HSDFGs. Retiming reduces the IP of a graph by redistributing its initial tokens. Unfolding optimizes a graph by scheduling several iterations of the graph. Retiming explores intra-iteration parallelism, while unfolding explores inter-iteration parallelism. Combining these two techniques may further optimize an HSDFG. It seems that first unfolding then retiming an HSDFG may obtain a finer results than procedure on the reverse order. Chao et. al [4] show that, contrary to intuition, the order of retiming and unfolding is immaterial for the IP. The result makes it possible to retime an unfolded HSDFG, which needs several times of space of the original HSDFG, without explicitly unfolding the HSDFG. A valid SDFG can always be converted to an equivalent HSDFG [5] and hence its performance can be analyzed or optimized with existing techniques for HSDFGs. However, the conversion procedures are time and space-consuming. Zhu et.al [6] prove the equivalence of retiming on SDFGs and on their equivalent HSDFGs. The result makes it possible to retime an SDFG without explicitly converting it to its equivalent HSDFGs.

In this paper, we review the relationship among the three transformations and reveal the equivalence of the combinations of them. The result makes it possible to retime and unfold an SDFG without explicitly converting it to its equivalent HSDFG and without explicitly unfolding the HSDFG.

II. Preliminaries

An SDFG is a finite directed graph $G = (V, E)$. V is the set of actors, modeling the computations of a system. Actor v is weighted with its computation time $t(v)$, a nonnegative integer. E is the set of directed edges, modeling interconnections between computations. The source actor and sink actor of e are denoted by $src(e)$ and $snk(e)$, respectively. Edge e is weighted with three properties, $d(e)$, $p(e)$ and $c(e)$. The *delay* $d(e)$ is the number of initial tokens on e, the *production rate* $p(e)$ is the number of tokens produced onto e by each firing of $src(e)$, and the *consumption rate* $c(e)$ is the number of tokens consumed from e by each firing of $snk(e)$. The set of incoming edges to actor v is denoted by $InE(v)$, and the set of outgoing edges from v by $OutE(v)$. If $p(e) = c(e) = 1$ for each $e \in E$, G is a *homogenous SDFG* (HSDFG).

SDFG $G = (V, E)$ is *sample rate consistent* [1] if and only if there exists a positive integer vector $q(V)$ satisfying *balance equations*, $q(src(e)) \times p(e) = q(snk(e)) \times c(e)$ for all $e \in E$. The smallest such q is called the *repetition vector*. We use q to represent the repetition vector directly. The repetition vector of G_1 (Fig. 1 (a)) is $q = [2, 2, 1]$, corresponding to actors A, B and C, for example. An *iteration* of an SDFG is a firing sequence in which each actor v occurs exactly $q(v)$ times. The average computation time per iteration is called the *iteration period* (IP).

It is always possible to convert a sample rate consistent SDFG to an equivalent HSDFG [5], which captures the dependencies among firings in an iteration of the SDFG. An equivalent HSDFG of G_1, $H(G_1)$, is shown in Fig. 1 (b). We denote the transformation by *H-transformation*.

This work was supported in part by the National Natural Science Foundation of China (Nos. 61572478, 61472406 and 61472474).

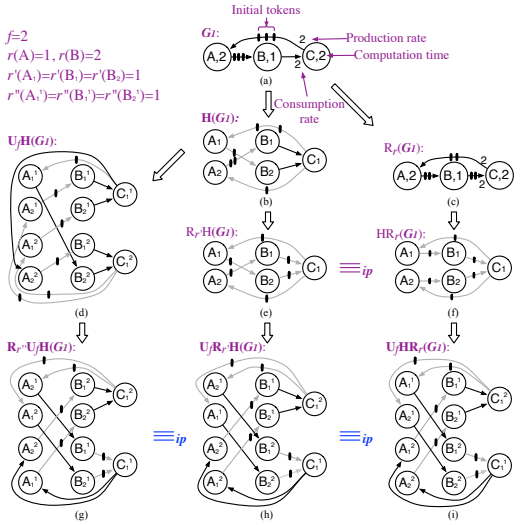

Fig. 1. (a) The SDFG G_1; (b)The equivalent HSDFG of G_1; (c) A retimed SDFG of G_1; (d) An unfloded graph of HSDFG $H(G_1)$; (e) A $R_{r'}H$-transformation of G_1; (f) A HR_r-transformation of G_1; (g) A $R_{r''}U_fH$-transformation of G_1; (h) A $U_fR_{r'}H$-transformation of G_1; (i) A U_fHR_r-transformation of G_1. The production/consumption rates are omitted when they are 1.

Retiming is a graph transformation that redistributes the graph's delays while remains the functionality of the new graph unchanged. Retiming an actor once means firing the actor once. The firing moves tokens on the incoming edges of the actor to its outgoing edges. A retimed SDFG of G_1 with retiming function $r = [1, 2, 0]$, which means retiming A once and retiming B twice, is shown in Fig. 1 (c). We denote the transformation with retiming function r by R_r-*transformation*.

A graph unfolded with an *unfolding factor* f describes f consecutive iterations of its original graph. Unfolding may reduce the IP of a graph at the cost of increasing the problem space by the unfolding factor. The HSDFG $H(G_1)$ unfolded with unfolding factor $f = 2$ is shown in Fig. 1 (d), in which nodes like A_1^2 denotes the 1st firing of actor A in the 2nd iteration of SDFG G_1. The unfolding of an SDFG is defined by the unfolding of its equivalent HSDFG. We denote the transformation with unfolding factor f by U_f-*transformation*.

The combinations of the transformations are also transformations. For simplicity, we use $U_fHR_r(G)$ to express the combinations of the transformations $U_f(H(R_r(G)))$, for example.

III. The Equivalence of Transformations of SDFGs

All the three transformations of SDFGs preserve their functionalities [5], [2], [3]. We consider the iteration period equivalence, denoted by $\equiv_{ip}$, in this paper.

Definition 1. *(IP equivalence) Let* $\mathbf{T_1}$ *and* $\mathbf{T_2}$ *be two transformations. If for any SDFG G, the IP of* $\mathbf{T_1(G)}$ *equals the IP of* $\mathbf{T_2(G)}$*, then* $\mathbf{T_1} \equiv_{ip} \mathbf{T_2}$.

Chao et. al [4] show that the order of retiming and unfolding is immaterial for the IP of an HSDFG. We restate the result as the following lemma.

Lemma 1. *[4] On HSDFGs,* $U_fR_{r'} \equiv_{ip} R_{r''}U_f$*, where*

$$r''(v^j) = \begin{cases} 1 + \lfloor \frac{r'(v)}{f} \rfloor, & \text{if } j \le r'(v)\%f; \\ \lfloor \frac{r'(v)}{f} \rfloor, & \text{otherwise.} \end{cases}$$

Zhu et.al [6] prove the equivalence of retiming on SDFGs and on their equivalent HSDFGs. We restate the result as the following lemma.

Lemma 2. *[6] Given SDFG G,* $HR_r(G)$ *and* $R_{r'}H(G)$ *are equivalent under isomorphism, where*

$$r'(v_i) = \begin{cases} 1 + \lfloor \frac{r(v)}{q(v)} \rfloor, & \text{if } i \le r(v)\%q(v); \\ \lfloor \frac{r(v)}{q(v)} \rfloor, & \text{otherwise.} \end{cases}$$

For example, $R_{r'}H(G_1)$ in Fig. 1 (e) is equivalent to $HR_r(G_1)$ in Fig. 1 (f) under the isomorphic function $\phi : R_{r'}H(G_1) \to HR_r(G_1)$ which is defined by $\phi(A_1) = A_2$, $\phi(A_2) = A_1$ and $\phi(v) = v$ for other actor v. The conclusion that $HR_r \equiv_{ip} R_{r'}H$ on SDFGs can be drawn directly by Lemma 2.

Theorem 1. *On SDFGs,* $R_{r''}U_fH \equiv_{ip} U_fR_{r'}H \equiv_{ip} U_fHR_r$ *with* r' *and* r'' *defined as that in above lemmas.*

Proof. (outline) For any SDFG G, $H(G)$ is an HSDFG. By Lemma 1, for any legal retiming r' and unfolding factor f on $H(G)$, the IP of $U_fR_{r'}(H(G))$ equals the IP of $R_{r''}U_f(H(G))$. Therefore, we have that $R_{r''}U_fH \equiv_{ip} U_fR_{r'}H$.

For any legal retiming r on G, by Lemma 2, $HR_r(G)$ and $R_{r'}H(G)$ are equivalent. Suppose the isomorphic function $\phi : R_{r'}H(G) \to HR_r(G)$ is defined by $\phi(u) = v$ for each $u \in R_{r'}H(G)$. $U_fHR_r(G)$ and $U_fR_{r'}H(G)$ are equivalent under the isomorphic function $\phi' : U_fR_{r'}H(G) \to U_fHR_r(G)$ defined by $\phi'(u^j) = v^j$ for each $u^j \in U_fR_{r'}H(G)$ and $1 \le j \le f$. Therefore, we have that $U_fR_{r'}H \equiv_{ip} U_fHR_r$. □

Examples of the three combinations of transformations, $R_{r''}U_fH(G_1)$, $U_fR_{r'}H(G_1)$ and $U_fHR_r(G_1)$, are shown in Fig. 1 (g), (h) and (i), respectively.

Let's see an application of the equivalence. Consider the problem: given an SDFG and a retiming r and an unfolding factor f on it, if the SDFG can be retimed and unfolded to achieve a given IP? Instead of retiming on the unfolded HSDFG like Fig. 1 (g), according to the equivalence, we can retiming G directly, with the extension of the IP computation method in [6], which computes the IP of an SDFG directly without converting it to its HSDFG. The extended method computes the IP of the unfolded graph directly on the SDFG with neither explicit H-transformation nor explicit U_f-transformation. The details are omitted here for the space limitation.

References

[1] E. Lee and D. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Trans. on Comput*, vol. 36, no. 1, pp. 24–35, 1987.

[2] C. Leiserson and J. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.

[3] K. Parhi and D. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Trans. Comput on Computers*, vol. 40, no. 2, pp. 178–195, 1991.

[4] L. Chao and E. Sha, "Scheduling Data-Flow Graphs via Retiming and Unfolding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 12, pp. 1259–1267, 1997.

[5] S. Sriram and S. S. Bhattacharyya, *Embedded multiprocessors: scheduling and synchronization*. CRC Press, 2009.

[6] X.-Y. Zhu, T. Basten, M. Geilen, and S. Stuijk, "Efficient retiming of multirate DSP algorithms," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 6, pp. 831–844, 2012.