SynCore v1.1：可組合核心融合的研究起源

v1.1 / 2025-11-01 / Neo.K（許筌崴） / 歷史母稿／公開概念技術論文／可證偽研究提案

https://thisoneisneok.com/html/papers/ccp-17.html

SECTION

從單執行緒資源集中到可重構微架構的可驗證設計

英文題名： SynCore v1.1: Research Origins of Composable Core Fusion

文件編號： EML-SYNCORE-ORIGIN-2026-v1.1

作者： Neo.K（許筌崴）

協作修訂： Aletheia

機構： 一言諾科技有限公司（EveMissLab）

原始版本： 2025 年 11 月《SynCore 神核融合引擎：量子態邏輯與單核至尊的架構革命》

修訂日期： 2026 年 7 月 30 日

文件性質： 歷史母稿／公開概念技術論文／可證偽研究提案

證據等級： E0（尚未完成專用模擬器、RTL、FPGA 或流片驗證）

版本地位： 本文保存 SynCore 的「核心融合」起源；現行統一主架構為《SynCore v2.1：可重構融合—流動運算架構》

授權： 公開發布前由作者確認最終授權條款

SECTION

摘要

本文重構 SynCore 最初的研究問題：當一個晶片擁有多個物理核心，而關鍵工作負載仍受單執行緒依賴鏈、分支、記憶體延遲或軟體遺產限制時，能否讓部分原本獨立的核心資源，暫時組成一個較大的邏輯執行域，以提高單執行緒性能或降低完成相同工作的能耗？

此問題不是全新的。Core Fusion、Composable Lightweight Processors、Core Federation、MorphCore、clustered microarchitecture 與 thread-level speculation 等研究，早已探索過可組合核心、可變粒度處理器或單執行緒跨多執行後端的可能性。因此，SynCore v1.1 不再宣稱首次提出「多核融合成大核」，也不把幾個核心的 ALU、L1 快取和功率預算相加，直接等同於數倍單核性能。

本文保留原稿最有價值的核心命題：

> 並行與串列不是兩種互斥硬體，而是同一運算基底在不同工作負載下的資源組織方式；若微架構在設計時即納入可組合性，則一組物理執行 tile 可以在獨立吞吐模式與融合單執行緒模式之間切換。

新版提出四級融合模型：獨立核心模式、共享前端模式、叢集執行模式與完整融合模式；建立融合群組、分散式指令視窗、banked physical register file、operand network、load/store ordering、全域提交、檢查點／暖狀態緩衝與失效安全控制器。原稿中的「疊加態、儲存態、坍塌態」被重新解釋為預備、保存、融合三種工程狀態，不再借用量子疊加或波函數坍縮作為硬體證明。

本文同時給出性能上界、重組成本、熱功率約束、軟體介面、比較基線、模擬與 FPGA 路線，以及六項可證偽命題。SynCore v1.1 的技術地位因此被明確限定為：它是 SynCore v2.1 中 Fusion Cluster 的歷史與理論基礎，而不是另一套獨立產品，也不是已被實驗證實的超級單核晶片。

關鍵詞： 核心融合、可組合處理器、單執行緒性能、指令級並行、可重構微架構、叢集執行、精確提交、檢查點、動態異質性

SECTION

0. 修訂聲明與版本譜系

0.1 為什麼保留這篇歷史母稿

原始 SynCore v1.0 首次建立了本系列的一條重要產品主線：

> 多個獨立核心
> longleftrightarrow
> 較大的邏輯執行域.

它後來被 DEO、O-Chip、DPCPS、DryCore、SDMCA 與 SynCore v2.1 吸收。然而，若只保留 v2.1，SynCore 從何處開始、最初試圖解決什麼問題，以及哪些概念後來被修正，將不再清楚。

本文因此不與 v2.1 競爭，而負責三項工作：

- 保存「多核為單執行緒服務」的原始問題意識；
- 將原稿的物理與工程錯誤改造成可驗證的融合微架構；
- 明確說明哪些元件已被移交給後續系列。

0.2 正式版本關係

> SynCore v1.0 原始概念
> →
> SynCore v1.1 核心融合起源篇
> →
> SynCore v2.1 統一主架構.

其中：

- v1.1 只研究通用數位核心的可組合融合；
- v2.1 進一步整合數位資料流陣列、振盪器加速器、驗證與提交引擎；
- 本文任何新工程結論，均不得覆蓋 v2.1 的最新系統定義。

0.3 術語遷移

原稿術語 v1.1 正式術語 處置

神核模式 Fusion Mode／融合模式 保留暱稱，正式文件用融合模式

Core Mesh Binding Fusion Fabric／融合織網 重構

Unistream Execution Engine Fusion Execution Domain／融合執行域 重構

Thread Monarch Controller Fusion Mode Controller／融合模式控制器 去人格化

Q-Storage Checkpoint and Warm-State Buffer／檢查點與暖狀態緩衝 去量子化

Thermo-Balancer Matrix　DEO＋DPCPS＋DryCore 介面　移交後續系列

疊加／儲存／坍塌　Armed／Checkpointed／Fused　改為工程狀態

錐形光刻製造 CPU　先進封裝／hybrid bonding／UCIe 類互連　移除未成立製程敘事

SECTION

1. 研究問題：多核心時代仍存在的單執行緒瓶頸

1.1 核心數量與單執行緒完成時間是不同問題

對一個具有 N 條動態指令的單執行緒，其依賴關係可表示為有向無環圖：

> cal-G_I=(V_I,E_I).

其中節點代表動態指令，邊代表資料、控制或記憶體次序依賴。若臨界路徑長度為：

> C(cal-G_I),

則即使提供無限多個執行單元，完成時間仍不能低於此因果鏈所要求的時間。

增加核心數量主要增加的是執行緒級並行能力：

> P_(TLP)uparrow,

而單執行緒能否變快，取決於硬體能否揭露：

- 指令級並行；
- 記憶體級並行；
- 分支級推測；
- value／address speculation；
- helper execution；
- 更大的指令與資料工作集合。

因此，原稿所說「其他核心閒置」只是資源現象，不是性能證明。真正問題是：

> 能否在不使通訊、時序、功耗與控制複雜度失控的前提下，把一組獨立核心的部分微架構資源，重組為一個較大的單執行緒執行域？

1.2 串列不等於完全沒有並行

「單執行緒」只是程式對外呈現一條精確指令序列，不代表每條動態指令都必須逐一完成後才能開始下一條。亂序執行、推測執行、預取與記憶體級並行早已在單執行緒內利用局部獨立性。

SynCore 的合法目標不是把真實串列依賴變成並行，而是擴大硬體可觀察和可利用的窗口：

> Fusion Benefit
> ≈
> f(
> W_(front),
> W_(window),
> W_(issue),
> M_(MLP),
> B_(cache),
> L_(fabric)
>).

1.3 單執行緒瓶頸的不同類型

融合不會對所有工作負載有效。至少需要區分：

- 前端受限：指令快取、取指、解碼或分支限制；
- 後端受限：執行單元或排程窗口不足；
- 延遲受限：單條長依賴鏈或高延遲運算；
- 記憶體受限：cache miss、TLB miss、DRAM 延遲；
- 頻寬受限：共享快取、記憶體或互連飽和；
- 同步受限：其實是多執行緒鎖與臨界區問題；

- 軟體／ISA 受限：動態翻譯、例外、序列化指令或自修改程式碼。

只有第 1、2 類以及部分第 4 類，可能直接受益於更寬、更大的融合執行域。第 3 類若缺乏可推測或可預取空間，增加執行單元幾乎無效。

SECTION

2. 既有研究與 SynCore 的非首創性

2.1 Core Fusion

Core Fusion 探索由多個獨立小核心動態形成較大的亂序處理器。其關鍵問題包括：單執行緒如何由多個前端共同取指、指令如何 steering、跨核心 producer–consumer 如何喚醒、load/store forwarding 如何保持正確，以及多個 reorder buffer 如何精確提交。

這條研究已證明「獨立核心融合」不是一句資源相加，而是必須重新處理整個處理器管線。

2.2 Composable Lightweight Processors

Composable Lightweight Processors 與 TFlex 以分散式執行結構、顯式資料圖和可組合 tile，探索從多個窄核心形成較寬邏輯處理器。其優點是結構分散、可形成較大動態性能範圍；代價則包括 operand network、控制網路、映射與跨 tile 通訊。

2.3 MorphCore 與反向變形

MorphCore 採取相反方向：以高性能亂序核心為基底，在執行緒級並行充足時轉成高執行緒數的順序 SMT。此研究指出，從小核心融合成大核心可能引入額外 pipeline latency、cache migration 與切換成本；因此「先有小核，再融合」不是唯一策略。

2.4 SynCore 能保留的差異性

SynCore v1.1 不以「第一個融合核心」作為貢獻，而以以下研究組合作為自身定位：

- 以分級融合取代單一全開／全關模式；
- 把重組成本、熱功率承諾和精確回退列為一等設計變數；
- 將融合群組的控制介面與 O-Chip、DEO、DPCPS、DryCore 統一；

- 保留檢查點、暖狀態與可逆切換，而非只比較穩態 IPC；
- 將核心融合明確定位為 SynCore v2.1 的一個執行域。

SECTION

3. 設計原則

原則一：可組合性必須在製造前設計

SynCore 不能把任意現成核心在執行時以軟體命令「接起來」。可融合核心必須共享預定義的：

- 時脈與重置域關係；
- operand transport；
- 全域或分散式 rename／tag 空間；
- load/store ordering；
- 精確提交協議；
- cache ownership；
- checkpoint 與 replay；
- power／thermal control hooks。

原則二：融合粒度必須有限

融合群組規模增加時，跨群組導線、喚醒、選擇、旁路、提交與 cache bank 通訊成本會上升。最初原型應限制為：

> $G \in \{2,4\}$,

而不是直接宣稱 8、16、32 核可形成單一超核心。

原則三：融合只揭露既有並行度

若工作負載的有效 ILP 為 P_(ILP)，即使發射寬度增加到 W_F，可利用寬度仍受：

> W_(use)
> ≤
> min(W_F,P_(ILP),P_(front),P_(mem)).

原則四：模式切換必須可逆且失效安全

融合失敗、預測失誤、熱餘裕不足或性能退化時，系統必須可退回獨立模式。任何融合控制器都不能繞過：

- 精確例外；
- ECC／parity；
- watchdog；
- 熱關機；
- 電壓護欄；
- 記憶體保護；
- 作業系統搶占。

原則五：控制不必依賴大型 AI

模式選擇可先使用：

- 硬體計數器規則；
- EWMA；
- decision tree；
- lookup table；
- contextual bandit；
- model predictive control。

只有當簡單控制器明顯不足，才加入較複雜模型。

SECTION

4. 四級融合模型

SynCore v1.1 不再只有「普通多核／神核」二態，而採四級模式：

4.1 F0：Independent Mode

每個 Scalar Tile 獨立執行：

- 自己的前端；
- 自己的 rename／issue／commit；
- 自己的 L1；
- 共享或分區 LLC；
- 適合多程式與高 TLP 工作負載。

4.2 F1：Front-End Cooperative Mode

多個 tile 仍各自提交，但共享部分前端資訊：

- branch history；
- instruction predecode；
- shared instruction cache bank；
- coordinated prefetch；
- helper thread 或 runahead。

此模式不形成一個完整大核心，重組成本最低。

4.3 F2：Clustered Execution Mode

一條執行緒的指令可被映射到多個執行 backend，但保留分區式結構：

- 分散式 instruction window；
- banked physical register file；
- operand network；
- clustered functional units；
- 全域 tag／dependency protocol；
- 統一或分層 commit。

這是 v1.1 的主要研究模式。

4.4 F3：Full Fusion Mode

多個 tile 邏輯上組成一個較大的亂序執行域：

- 統一前端或嚴格同步的分散式前端；
- 較大的全域指令視窗；
- 擴展的 load/store window；
- 統一精確提交；
- 跨 tile speculation recovery。

F3 不一定比 F2 更快。若中央 steering、global wakeup 或 bypass latency 過高，F3 可能反而降低頻率或增加關鍵迴路延遲。

4.5 模式不是性能等級

> F0<F1<F2<F3

只代表耦合程度，不代表性能必然單調增加。系統應根據工作負載與成本選擇：

> F^(*)
> =
> argmin_(F∈{F0,F1,F2,F3})

SECTION

5. 頂層架構

5.1 Fusion Group

一個融合群組定義為：

> cal-F_g
> =
> (
> cal-T_g,
> cal-N_g,
> cal-R_g,
> cal-M_g,
> cal-C_g,
> cal-S_g
>),

其中：

- cal-T_g：參與融合的 tile 集合；
- cal-N_g：operand／control network；
- cal-R_g：register／rename 資源；
- cal-M_g：memory ordering 與 cache 資源；
- cal-C_g：commit、checkpoint、recovery；
- cal-S_g：安全、功率與熱狀態。

5.2 Scalar Tile

每個 tile 至少包含：

- 取指與解碼前端；
- branch predictor 或 predictor slice；
- rename／dispatch；
- local instruction window；
- ALU／FPU／vector units；
- banked PRF；
- L1 I／D cache；
- TLB；
- local ROB／commit metadata；
- Fusion Fabric interface。

Tile 應能在 F0 完整獨立運作，避免融合邏輯失效時整顆晶片不可用。

5.3 Fusion Fabric

Fusion Fabric 不是一般 NoC 的別名。它需要支援低延遲的：

- instruction assignment；
- operand delivery；
- wakeup notification；
- branch resolution；
- load/store replay；
- commit coordination；
- checkpoint broadcast；
- mode transition。

對 producer i 與 consumer j，跨 tile 依賴延遲可表示為：

> $$L_{(i\to j)} = L_{(tag)} + L_{(route)} + L_{(queue)} + L_{(deliver)}.$$

若此延遲高於把 producer 與 consumer 放在同一 tile 的收益，steering 策略必須優先共置依賴鏈。

5.4 Front-End Organization

三種候選：

A. 單一共享前端

優點：控制簡單、分支狀態統一。

缺點：前端可能成為單點瓶頸，且佈線集中。

B. 鎖步分散式前端

各 tile 取同一條指令流的不同區段，需共享：

- global history；
- return address stack；
- taken branch communication；
- instruction striping metadata。

C. 階層式前端

一個 lead frontend 負責方向和邊界，其餘 frontend 提供 fetch／decode bandwidth。v1.1 建議先以此作為原型。

5.5 Distributed Instruction Window

每個 tile 維持本地 window，但使用全域 tag。指令 u 的就緒條件：

> ready(u,t)
> =
> bigwedge_(v∈pred(u))
> valid_(v→ u)(t).

steering 目標同時最小化：

> J_(steer)
> =
> α Q⊠
> +
> β D_(dep)
> +
> γ P⊠
> +
> δ T⊠,

其中：

- Q⊠：目標 tile 佇列壓力；
- D_(dep)：跨 tile 依賴距離；
- P⊠：功率增量；
- T⊠：熱風險。

5.6 Banked Physical Register File

原稿的「所有核心共享一個巨大暫存器池」會造成多埠、導線與時序爆炸。新版採 banked PRF：

> cal-R
> =
> bigcup_(k=1)^(G)

每個 physical register 具有 home bank。跨 bank 讀取透過 operand network 傳送，並使用：

- locality-aware renaming；
- replication for high fan-out values；
- producer–consumer co-location；
- limited bypass domains。

5.7 Load／Store Ordering

融合模式最難的部分之一是記憶體次序。必須支援：

- 全域 load/store sequence number；
- store-to-load forwarding；
- memory dependence prediction；
- violation detection；
- replay；
- precise exception；
- fence／atomic semantics。

對 load l，其提交前必須確認：

> $\forall\ s<l,$
>
> $\neg \mathrm{alias}(s,l)$
> $\vee$
> $\mathrm{forwarded}(s,l)$
> $\vee$
> $\mathrm{resolved}(s,l).$

5.8 Global Commit and Recovery

完整融合模式需要全域邏輯順序：

> I□<I□<...<I□.

但物理 ROB 可以分散。可採：

- global sequence number；
- local ROB slices；
- pre-commit watermark；
- distributed completion bitmap；
- centralized architectural state checkpoint。

commit 條件：

> commit(I□)
> =
> done(I□)
> ∧
> bigwedge_(j<k)
> committable(I□)
> ∧
> ¬exception_(≤ k).

SECTION

6. 三種工程狀態：去量子化的三態模型

原稿的三態邏輯具有直覺價值，但不應稱為量子態。新版改成：

6.1 Armed State／預備態

Tile 尚未加入融合執行，但已完成：

- 時脈／電壓域準備；

- Fusion Fabric 連線自檢；
- 必要 code／data 預取；
- cache way／bank 預留；
- checkpoint 空間配置。

預備態的目標是降低進入融合模式的冷啟動成本。

6.2 Checkpointed State／保存態

原執行緒或背景任務的架構狀態被保存，以便 tile 資源暫時釋出。保存內容可分級：

- C0：architectural registers＋PC；
- C1：C0＋TLB／predictor hints；
- C2：C1＋selected cache lines；
- C3：C2＋microarchitectural replay metadata。

並非所有內部流水線狀態都值得保存。保存越完整，切換成本與硬體面積越高。

6.3 Fused State／融合態

tile 已加入 Fusion Group，共同執行一條邏輯執行緒。此狀態必須具備：

- 統一精確狀態；
- mode ownership；
- power／thermal certificate；
- watchdog；
- rollback checkpoint；
- OS preemption route。

6.4 狀態轉移

> Independent
> →
> Armed
> →
> Checkpointed
> →
> Fused.

退出則反向進行，必要時：

> Fused
> →
> Rollback
> →
> Independent.

這些是有限狀態機，不涉及量子疊加、相干性或測量坍縮。

SECTION

7. 檢查點與暖狀態緩衝

7.1 重新定義 Q-Storage

原稿的 Q-Storage 改名為：

> Checkpoint and Warm-State Buffer, CWB

> 檢查點與暖狀態緩衝

它可以由：

- 片上 SRAM；
- LLC 保留區；

- 3D stacked SRAM／DRAM；
- coherent memory；
- persistent memory（遠期）；

  組成，但不同介質對應不同恢復時間。

7.2 不保存任意完整快取

若完整保存 S bytes 狀態，通道頻寬為 B，最低資料搬移時間為：

$$T_{save} \geq (S)/(B) + T_{quiesce} + T_{metadata}.$$

因此「完整快取與流水線微秒級凍結」不能普遍成立。CWB 應使用：

- selective state capture；
- dirty-line prioritization；
- compressed metadata；
- lazy cache warming；
- predictive prefetch。

7.3 暖狀態不是正確性要求

cache、predictor、TLB 等暖狀態只影響性能，不是架構正確性。恢復時可：

- 先恢復 architectural state；
- 立即正確執行；

- 再以背景方式恢復性能狀態。

  這使回退更簡單。

SECTION

8. Fusion Mode Controller

8.1 控制器輸入

控制器觀察：

> $$
> x(t)
> =
> (
> IPC,
> MPKI,
> BMR,
> MLP,
> ROB_(occ),
> IQ_(occ),
> T,
> P,
> Q_(run),
> C_(switch)
>).
> $$

其中包括：

- 指令吞吐量；
- cache miss；
- branch misprediction；
- 記憶體級並行；

- ROB／issue queue 壓力；
- 溫度與功率；
- runnable thread 數；
- 切換估計成本。

8.2 模式收益判斷

融合只在以下條件成立時啟動：

> widehatG_(fusion)
> >
> C_(transition)
> +
> C_(opportunity)
> +
> C_(power)
> +
> M_(safety).

其中 opportunity cost 表示被借用 tile 無法服務其他執行緒的損失。

8.3 遲滯與最短駐留時間

避免模式震盪：

> T_(res)
> ≥
> T_(min)(F□,F□).

控制器使用不同進入與退出門檻：

> θ_(enter)
> >
> θ_(exit).

8.4 作業系統仍保有政策權

Fusion Mode Controller 提供機制與建議，但 OS／hypervisor 決定：

- 哪個執行緒可獲得融合群組；
- 是否允許借用其他 tile；
- 優先級與配額；
- 即時任務與安全域；
- 何時搶占或強制退出。

SECTION

9. 性能與成本模型

9.1 單執行緒時間下界

對融合群組大小 G：

$$
\begin{aligned}
&T_G \\
&\geq \\
&\max(\\
&C(\text{cal-}G_I), \\
&(N)/(W_{(\text{eff})}(G)), \\
&T_{(\text{front})}(G), \\
&T_{(\text{mem})}(G) \\
&) \\
&+ \\
&T_{(\text{comm})}(G) \\
&+ \\
&T_{(\text{control})}(G).
\end{aligned}
$$

其中：

$$
W_{(eff)}(G)
\neq
G \cdot W\boxtimes
$$

一般只會次線性增加。

9.2 融合加速比

$$
S_G
=
(T\boxtimes)/(T_G).
$$

其上界受 Amdahl 類限制。若可被融合加速的比例為 p，該部分加速為 s_G：

$$
S_G
\leq
(1)/((1-p)+(p)/(s_G)).
$$

因此，原稿所稱 4 核必然帶來 4–6 倍性能不成立。

9.3 淨收益

$$
G_{(net)}
=
(T\boxtimes)/(T_{(transition)}
+
T_{(run,fused)}
+
T_{(exit)}).
$$

短任務即使融合穩態較快，也可能被切換成本抵銷。

9.4 功率與能量

融合群組功率：

$$
P_F = P_{tiles} + P_{fabric} + P_{global} + P_{memory} + P_{cooling,share}.
$$

完成任務能量：

$$
E_F = \int_{t□}^{t□} P_F(t)\,dt.
$$

更高瞬時功率可能因縮短時間而降低總能量，也可能因互連與控制開銷而增加；必須實測。

9.5 系統機會成本

若 G 個 tile 被一條執行緒占用，系統吞吐損失可表示為：

$$
C_{opp} = \Sigma_{k \in cal\text{-}T_g} U□^{alternative} - U_g^{fusion}.
$$

因此，融合模式最適合：

- runnable threads 少；

- 前台延遲敏感；
- 其他 tile 本來就閒置；
- 或單執行緒完成時間的價值高於背景吞吐量。

SECTION

10. 功率、熱與可靠度介面

10.1 不再把功率預算直接相加

每個 tile 的額定功率不能無條件集中到單執行緒。融合前需取得：

> cal-C_(power)
> =
> (P_(max),I_(max),V_(range),τ_(max)),

以及：

> cal-C_(thermal)
> =
> (T_(max),Δ T_(max),B_(th),τ_(cool)).

10.2 系列分工

- DEO：選擇融合模式可使用的運作包絡；
- DPCPS：提供電流、電壓與瞬態供電承諾；
- DryCore：提供熱阻、冷卻與安全邊界；
- SDMCA：提供模組位置與系統拓撲；
- O-Chip：根據工作負載意圖決定是否請求融合；
- SynCore Fusion Group：在證書範圍內執行。

10.3 熱輪換的限制

原稿提出「核心輪換讓熱核心休息」。在融合模式中，若依賴與狀態分布在多 tile，任意輪換會造成資料與狀態搬移。新版只允許：

- execution unit gating；
- bank-level throttling；
- steering bias；
- spare tile substitution；
- mode downgrade。

而不是把正在執行的單執行緒每幾毫秒任意搬到另一組核心。

10.4 相變冷卻不是必要條件

液氮不屬於通用處理器產品路線。SynCore 必須先在一般封裝、冷板或兩相冷卻條件下成立。極端冷卻僅可作為實驗邊界，不得用來掩蓋架構本身的功耗問題。

SECTION

11. 軟體與 ISA 合約

11.1 透明模式

未修改應用可由硬體與 OS 自動選擇 F0–F2。透明模式不得改變：

- ISA 可見結果；
- memory model；
- exception order；
- debug semantics；
- timing-independent correctness。

11.2 提示模式

應用或 runtime 可提交：

> cal-H
> =
> (
> q_(latency),
> q_(throughput),
> τ_(phase),
> p_(priority),
> r_(preempt),
> ε_(energy)
>).

例如：

- 下一階段為延遲敏感單執行緒；
- 預計持續 20 ms；
- 允許借用 3 個 tile；
- 可被高優先級即時任務搶占。

11.3 編譯器協同

編譯器可提供：

- basic-block boundaries；
- dependency hints；
- code layout；
- likely long-latency loads；
- fusion-friendly regions；

- no-fuse regions；
- helper-thread candidates。

  但 v1.1 不要求新 ISA 才能開始模擬。

11.4 與 SynIR 的關係

SynCore v2.1 的 SynIR 可以標記某段程式偏好 Fusion Cluster，但 v1.1 不處理數位資料流與振盪器映射。本文只定義：

> SynIR Region
> →
> Fusion Eligibility
> →
> F0／F1／F2／F3.

SECTION

12. 典型工作負載的重新評估

12.1 舊遊戲與模擬器

原稿以特定遊戲宣稱數倍幀率提升。新版只保留它們作為候選基準，原因是：

- 可能存在單一主執行緒；
- 動態翻譯器可受益於大指令窗口和 cache；
- 分支與 indirect branch 密集；
- 幀時間具有延遲門檻。

  但性能可能受：

- 遊戲引擎鎖；
- 作業系統計時；

- GPU／driver；
- 記憶體；
- 模擬器 JIT；
- 相容性同步；

  限制。不得預先宣稱 2–6 倍。

12.2 即時音訊

低延遲音訊圖常受最長依賴鏈而非總 CPU 利用率限制。融合模式可能提高單一效果器鏈的 headroom，但其可行性要求：

- 模式切換不可造成 dropout；
- p99／p999 延遲穩定；
- 熱降頻可預測；
- OS 搶占與中斷可控。

12.3 編譯、資料庫與服務

這些工作負載常同時具有多執行緒與序列階段。Fusion Mode 應只在：

- link／serial pass；
- query coordinator；
- garbage collection critical section；
- transaction commit；

  等明確瓶頸區段啟用，而不是整個應用永久融合。

12.4 科學計算

分子動力學、氣候與 CFD 多數已有高度平行化路徑。不能因存在時間積分順序，就推論整個工作負載是單執行緒。SynCore 應以真實 profiling 找出：

- serial fraction；
- irregular kernels；
- reduction bottlenecks；
- control-heavy setup；
- sparse／pointer-heavy phases。

SECTION

13. 驗證方法

13.1 比較基線

至少比較：

- B0：單一窄核心；
- B1：同面積大亂序核心；
- B2：G 個獨立核心；
- B3：異構 big／little＋執行緒遷移；
- B4：Core Fusion 類基線；
- B5：MorphCore 類基線；
- S1：SynCore F1；
- S2：SynCore F2；
- S3：SynCore F3。

比較必須滿足相同或明確標示的：

- 製程節點；
- 面積；

- 頻率；
- cache 容量；
- 記憶體系統；
- 功率限制；
- 編譯器；
- workload input。

13.2 工作負載集合

- SPEC CPU 類單執行緒基準；
- branch-heavy；
- pointer chasing；
- graph traversal；
- emulator／DBT microbenchmarks；
- audio DAG；
- legacy game logic synthetic trace；
- server serial phase；
- multiprogrammed mixes；
- high-TLP parallel suite。

13.3 指標

- IPC；
- 完成時間；
- p50／p95／p99 latency；

- branch MPKI；
- cache／TLB MPKI；
- MLP；
- Fusion Fabric traffic；
- transition latency；
- rollback rate；
- energy／task；
- peak power；
- temperature；
- system throughput；
- fairness；
- area；
- critical path／maximum frequency。

13.4 消融實驗

依序加入：

- 共享前端；
- 分散式 window；
- banked PRF；
- operand replication；
- enlarged load/store window；
- CWB；

- predictive mode control；
- power／thermal certificate。

以辨識真正收益來源。

SECTION

14. 實作路線

V0：形式與 trace 模型

- instruction DAG；
- ILP／MLP 上界；
- transition cost；
- thermal／power budget；
- workload phase detection。

V1：gem5 或同級週期模擬

先實作：

- 2／4 tile fusion；
- F0／F1／F2；
- distributed window；
- operand network；
- global commit；
- CWB 的簡化模型。

V2：RTL microcluster

使用 RISC-V 或開放前端，建立：

- 2 個可獨立 tile；
- 共享／協作前端；
- banked register file；
- clustered ALU；
- 精確 commit；
- mode switch。

V3：FPGA 原型

FPGA 主要驗證：

- 功能正確性；
- mode transition；
- checkpoint／rollback；
- OS／runtime 介面；
- fault injection。

FPGA 頻率和導線特性不能用來預測最終 ASIC IPC。

V4：實體設計與時序評估

進行：

- synthesis；
- place and route；
- timing closure；
- power analysis；

- area overhead；
- thermal map；
- Fusion Fabric congestion。

V5：小型測試晶片

只有在 V1–V4 證明：

> $\Delta J_{(end\text{-}to\text{-}end)}>0$

且收益不是來自不公平面積或 cache 優勢時，才進入測試晶片。

SECTION

15. 證據制度

E0：概念與形式模型

本文目前狀態。

E1：軟體／trace 模型

公開：

- trace；
- 成本模型；
- workload phase；
- 模式策略。

E2：週期模擬

公開：

- simulator configuration；

- baseline；
- area／power model；
- raw results；
- confidence intervals。

E3：RTL／FPGA

公開：

- RTL 或可審查摘要；
- synthesis report；
- transition test；
- fault injection；
- correctness suite。

E4：測試晶片

公開：

- die／package；
- frequency；
- power；
- thermal；
- silicon bugs；
- reproducible benchmark。

E5：跨平台與第三方重現

由外部團隊重現核心結論。

SECTION

16. 可證偽命題

命題 H1：有限融合群組可提高部分單執行緒工作負載

在等面積、等功率或明確功率限制下，F2／F3 對至少一個預先註冊的工作負載集合，能顯著降低完成時間。

否證條件： 相對同面積大核與 big／little 基線，沒有穩定收益。

命題 H2：分級融合優於二態融合

F1／F2 能在部分工作負載上，以較低重組成本接近 F3 收益。

否證條件： F1／F2 始終被 F0 或 F3 支配。

命題 H3：跨 tile 依賴局部化是主要因素

locality-aware steering 顯著優於只做負載平衡的 steering。

否證條件： 兩者差異不顯著，或 locality 策略造成更差不平衡。

命題 H4：暖狀態緩衝可降低短階段融合成本

CWB 能降低 repeated phase 的進入／退出開銷。

否證條件： 保存與恢復能耗超過重新暖機收益。

命題 H5：預測控制必須勝過簡單遲滯控制

較複雜控制器只有在端到端收益與穩定性上顯著優於規則控制器時才值得採用。

否證條件： 簡單規則達到相同或更佳結果。

命題 H6：融合價值取決於整機而非 IPC

即使 IPC 上升，若系統吞吐、能耗、熱或公平性惡化，則該融合模式不合格。

否證條件： 以 IPC 單一指標便可穩定預測所有端到端收益。

SECTION

17. 主要風險與失敗模式

風險 原因 可能結果 緩解

steering 瓶頸 中央化分派延遲 頻率下降 階層式、分散式、區域化

operand network 過熱 跨 tile 值傳送 能耗上升 共置、複製、窄域 bypass

global commit 複雜 精確順序 提交停滯 ROB slices、watermark

LSQ 擴展困難 alias 與 replay 時序／面積爆炸 partition、predictor、限制群組

模式切換過慢 checkpoint／cache 短任務無收益 F1／F2、中間模式、遲滯

失去 TLP 吞吐 tile 被單執行緒占用 系統性能下降 OS 配額、opportunity cost

熱點 融合集中活動 降頻 DEO／DryCore 證書

控制器誤判 phase 不穩定 震盪、回退 calibration、simple fallback

除錯困難 分散式推測 驗證成本高 deterministic mode、trace、formal properties

面積不公平 額外融合結構 假性能收益 等面積比較

SECTION

18. 與 SynCore v2.1 的正式接口

SynCore v1.1 在 v2.1 中對應：

> Scalar Tiles
> +
> Fusion Cluster
> +
> Verification／Commit.

不包含：

- Digital Flow Array；

- Oscillator Compute Tile；
- Kuramoto 相位運算；
- 跨晶片 CXL 融合；
- 系統級 O-Chip 編排的完整規格。

  執行流程：

> 應用／SynIR 提示
> →
> O-Chip／OS 決策
> →
> DEO／DPCPS／DryCore 證書
> →
> Fusion Mode Controller
> →
> F0／F1／F2／F3
> →
> 精確提交或回退.

因此，v1.1 的成果應直接回填 v2.1 的 Fusion Cluster 章節，而不是形成第二條產品線。

SECTION

19. 設計決策總表

原始主張	v1.1 判定
多核可以為單執行緒服務	保留，但需專用可組合微架構
4 核資源相加等於 4 倍以上單核	取消
所有 L1／暫存器直接合併	改為 banked、分區與明確協議
同時執行所有分支可消除誤判	限定；只在成本可控時研究多路徑推測
量子三態是硬體新邏輯	取消，改成工程狀態機

Q-Storage 可保存完整流水線並微秒恢復　改為分級 checkpoint／warm state

TMC 必須是 AI 君主控制器　取消，改為可驗證模式控制器

塔形與煙囪效應是必要硬體　取消，移交封裝與 SDMCA

錐形光刻是核心融合使能製程　取消

液氮／相變冷卻支撐神核　非產品必要條件

老遊戲必然提升數倍　改為候選基準

SynCore v1 是獨立現行主架構　取消，改為 v2.1 的起源篇

SECTION

20. 結論：從「單核至尊」到「動態資源粒度」

SynCore 最初以「神核」這個強烈名稱提出一個直覺：既然一顆晶片已經擁有大量核心，為何不能在單執行緒需要時，把部分資源集中起來？這個直覺值得保留，但不能由「資源存在」直接推導「資源可以無成本合併」。

真正的核心融合必須面對：

- 前端共同取指；
- 分支狀態；
- 指令 steering；
- 跨 tile 依賴；
- register／operand transport；
- load/store ordering；
- 精確提交；
- cache ownership；
- 模式轉換；
- 功率、熱與系統機會成本。

因此，SynCore v1.1 的成熟命題不是：

> 多核可以瞬間坍縮成一顆無敵單核。

而是：

> 處理器的核心粒度不必永久固定。若可組合性在微架構、互連、狀態、作業系統與安全控制中被共同設計，物理 tile 可以在獨立吞吐與融合單執行緒之間形成有限、可逆、可量測的動態異質性。

這一命題仍然具有研究價值，但它的成敗只能由公平基線、端到端性能、能耗、熱、面積、切換成本與可重現性決定。

SynCore v1.1 到此不再向外擴張。其後續工程工作，統一歸入 SynCore v2.1 的 Fusion Cluster 驗證路線。

SECTION

參考文獻

- E. Ipek, M. Kirman, N. Kirman, and J. F. Martínez, “Core Fusion: Accommodating Software Diversity in Chip Multiprocessors,” Proceedings of ISCA, 2007. DOI: 10.1145/1273440.1250686.
- C. Kim et al., “Composable Lightweight Processors,” Proceedings of MICRO, 2007.
- Khubaib et al., “MorphCore: An Energy-Efficient Microarchitecture for High Performance ILP and High Throughput TLP,” Proceedings of MICRO, 2012. DOI: 10.1109/MICRO.2012.36.
- M. S. S. B. Robatmili et al., “Scaling Power and Performance via Processor Composability,” University of Texas at Austin Technical Report TR-10-14, 2010.
- E. Ipek, M. Kirman, N. Kirman, and J. F. Martínez, “Retrospective: Core Fusion,” 2023.
- M. D. Hill and M. R. Marty, “Amdahl’s Law in the Multicore Era,” Computer, 2008. DOI: 10.1109/MC.2008.209.
- Neo.K and Aletheia, “SynCore v2.1: A Reconfigurable Fusion–Flow Computing Architecture,” EveMissLab, 2026.

SECTION

附錄 A：最低公開資料包

若進入 E2 以上，最低公開：

- 完整 simulator configuration；
- workload 與輸入；
- compiler flags；
- microarchitecture table；
- area／power assumptions；
- transition model；
- raw benchmark output；
- failure cases；
- random seeds；
- scripts；
- SHA-256；
- 與 B0–B5 的公平性說明。

SECTION

附錄 B：第一個最小實驗

第一個可執行研究節點：

- 建立 2-tile baseline；
- 實作 F0、F1、F2；
- 不實作 F3；
- 使用 8–12 個微基準；
- 測量 transition cost；

- 加入等面積大核；
- 公開所有負結果。

成功標準不是平均 IPC 最大，而是至少找到一個清楚的工作負載區域，使：

> $\Delta T<0$,
>
> $\Delta E\leq 0$,
>
> ΔA可接受,
>
> Δ Correctness=0.

SECTION

附錄 C：術語使用規則

「神核」可以作為歷史名稱、產品暱稱或傳播語言；正式技術文獻中應使用：

- Fusion Group；
- Fusion Cluster；
- Fusion Mode；
- Fusion Execution Domain；
- Fusion Mode Controller。

「量子態」「坍塌」「波函數」除非真的涉及量子硬體，不再作為 SynCore 數位微架構的物理術語。